

Lab Week 05 | BITP 3123 Distributed Application Development

# Development of Web Service Consumer Application

By

Emaliana Kasmuri

FTMK, UTeM

## Table of Contents

<b>1</b>	<b>Learning Outcomes .....</b>	<b>1</b>
<b>2</b>	<b>Software Tools and Resources for the Exercise .....</b>	<b>1</b>
<b>3</b>	<b>An Overview of the Case Study .....</b>	<b>1</b>
<b>4</b>	<b>Adding a New Controller Class.....</b>	<b>2</b>
4.1	Create a New Controller Class.....	2
4.2	Create a Method to Consume a Web Service .....	2
4.3	Request /ordertype/list from Postman.....	4
4.4	Validating the Request .....	4
<b>5</b>	<b>Add Thymeleaf and Other Dependency .....</b>	<b>5</b>
<b>6</b>	<b>Import the Front-End Files.....</b>	<b>6</b>
6.1	Import the CSS File .....	6
6.2	Import Header and Footer.....	6
6.3	Import HTML Files .....	6
<b>7</b>	<b>Displaying a List of Order Types .....</b>	<b>6</b>
7.1	New Definition of getOrderTypes.....	6
7.2	Display ordertypes.html .....	7
<b>8</b>	<b>Adding New Order Type .....</b>	<b>8</b>
8.1	Define updateOrderType ( ) Method .....	8
8.2	Implement updateOrderType( ) Method .....	10
8.3	Define getOrderType( ) Method .....	11
8.4	Implement getOrderType( ) method .....	12
<b>9</b>	<b>Attaching The Mapping of getOrderType( ) to the Front End .....</b>	<b>12</b>
9.1	Attaching the Mapping to Add New Order Type Link .....	12
9.2	Adding a Form to Add a New OrderType .....	13
9.3	Display ordertypeinfo.html.....	13
<b>10</b>	<b>Add New Order Type.....</b>	<b>14</b>
<b>11</b>	<b>Edit an Order Type .....</b>	<b>15</b>
11.1	Attaching getOrderType( ) Mapping to Edit Order Type Link .....	15
11.2	Edit an OrderType from ordertypeinfo.html.....	15
<b>12</b>	<b>Delete an Order Type .....</b>	<b>17</b>

12.1	Define deleteOrderType() Method.....	17
12.2	Implement deleteOrderType() .....	18
12.3	Attaching deleteOrderType ( ) Mapping to Delete Order Type Link .....	18
12.4	Display Link to Delete an Order Type .....	19
12.5	Delete an OrderType from a Link .....	20
<b>13</b>	<b>Exercise 1: Consume Web Service for REST Product Type .....</b>	<b>20</b>

## **1 Learning Outcomes**

At the end of this lab exercise, the student should be able to: -

1. Implement the consumption of REST web service using a controller class.
2. Implement the front using Thymeleaf framework.
3. Attach the mapping of the controller class to the front end.
4. Execute integration testing for each CRUD operation from the front end.
5. Implement full-stack CRUD via REST web service consumption.

## **2 Software Tools and Resources for the Exercise**

This lab exercise requires the following software tools to implement the case study.

1. Eclipse
2. Web service provider implemented from Week 04
3. Web browser
4. Internet connection

## **3 An Overview of the Case Study**

Tba

## 4 Adding a New Controller Class

### 4.1 Create a New Controller Class

1. Create the class shown in Figure 4.1. This class manage the request from the front end and consume REST web service to manage the CRUD operation.

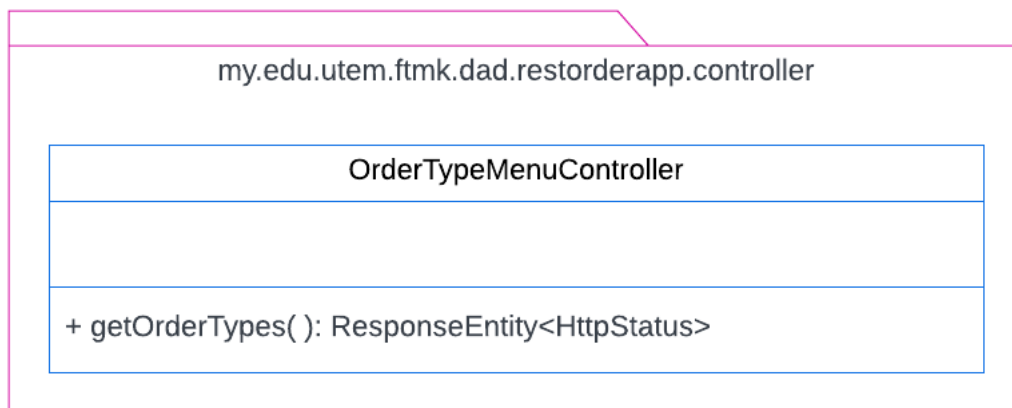


Figure 4.1: Class OrderTypeMenuController in UML Model

2. Annotate the class with `@Controller`.
3. Import `@Controller` from `org.springframework.stereotype`.
4. Describe the class in a block of comment.
5. Save the class.

### 4.2 Create a Method to Consume a Web Service

1. Define the method shown in Figure 4.1. This method consume a GET web service and display a list of record on the console.
2. Annotate the method with `@GetMapping("/ordertype/list")`.
3. Import the annotation from `org.springframework.web.bind.annotation`. The initial definition shall be similar as shown in Figure 4.2.

```

@GetMapping("/ordertype/list")
public ResponseEntity<String> getOrderTypes() {

}

```

Figure 4.2: Initial definition of getOrderTypes( ) method

4. Add the implementation shown in Figure 4.3 in the getOrderTypes( ) body of method.

```

// The URI for GET order types
String uri = "http://localhost:8080/orderapp/api/ordertypes";

// Get a list order types from the web service
RestTemplate restTemplate = new RestTemplate();
ResponseEntity<OrderType[]> response =
    restTemplate.getForEntity(uri, OrderType[].class);

// Parse JSON data to array of object
OrderType orderTypes[] = response.getBody();

// Generate message
System.out.println(this.getClass().getSimpleName());
System.out.println("Total records: " + orderTypes.length + "\n");

// Display the records
for (OrderType orderType:orderTypes) {

    System.out.print(orderType.getOrderTypeId() + "-");
    System.out.print(orderType.getCode() + "-");
    System.out.println(orderType.getName());
}

// For Postman status
String message = "Check out the message in the console";
return new ResponseEntity<>(message, HttpStatus.OK);

```

Figure 4.3: Implementation of getOrderTypes( ) method

5. Import RestTemplate from org.springframework.web.client.
6. Import other classes from the correct package.
7. Add comment to describe the method.
8. Fix errors, if any.
9. Save the class.

### 4.3 Request /ordertype/list from Postman

1. Open Postman
2. Create a new collection named **REST OrderType Consume** in the same workspace as previous lab,
3. Create a new request under the collection in Step 2.
4. Paste <http://localhost:8080/orderapp/ordertype/list> in the request address bar.
5. Select the method to **GET**.
6. Then click **Send**.
7. The response should as shown in Figure 4.4: Response from a web service to list order type.

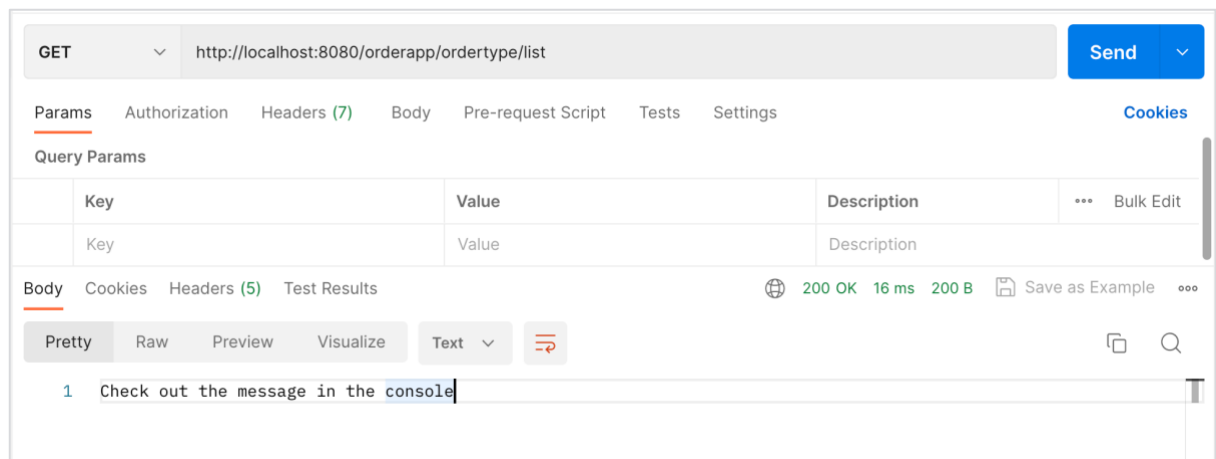
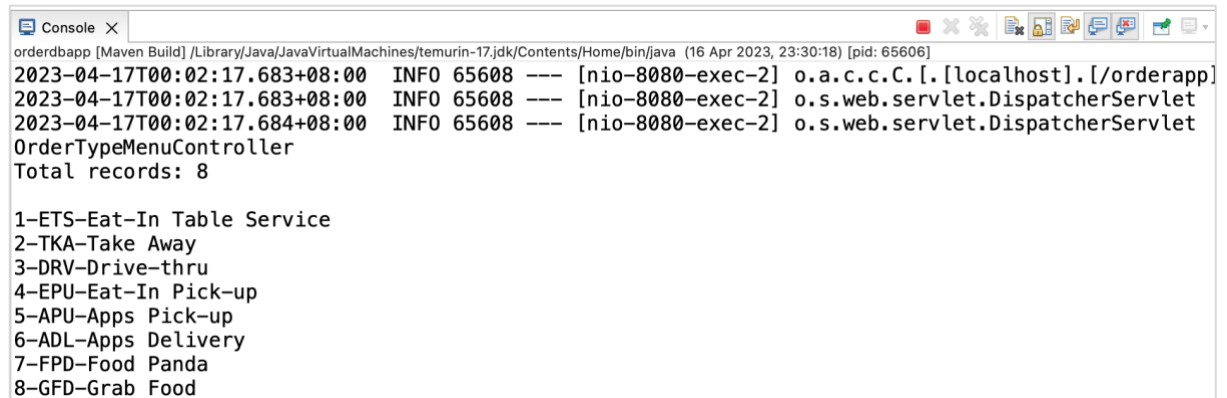


Figure 4.4: Response from a web service to list order type

8. Save the request with an appropriate name.

### 4.4 Validating the Request

1. Switch to **Eclipse**.
2. Click the **Console tab** at the bottom of the Eclipse. The output should be similar as shown in Figure 4.5.



```
Console X
orderdbapp [Maven Build] /Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java (16 Apr 2023, 23:30:18) [pid: 65606]
2023-04-17T00:02:17.683+08:00 INFO 65608 --- [nio-8080-exec-2] o.a.c.c.C. [localhost] [/orderapp]
2023-04-17T00:02:17.683+08:00 INFO 65608 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet
2023-04-17T00:02:17.684+08:00 INFO 65608 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet
OrderTypeMenuController
Total records: 8

1-ETS-Eat-In Table Service
2-TKA-Take Away
3-DRV-Drive-thru
4-EPU-Eat-In Pick-up
5-APU-Apps Pick-up
6-ADL-Apps Delivery
7-FPD-Food Panda
8-GFD-Grab Food
```

Figure 4.5: Sample of output from the request

## 5 Add Thymeleaf and Other Dependency

1. Open **pom.xml**.
2. Locate the end of dependencies block, `</dependencies>`.
3. Add the dependencies show in Figure 5.1 before the of `</dependencies>`.

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

<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>bootstrap</artifactId>
  <version>4.6.2</version>
</dependency>

<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>jquery</artifactId>
  <version>3.6.1</version>
</dependency>

<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>webjars-locator-core</artifactId>
</dependency>
```

Figure 5.1: The front end dependencies for the project

4. Save **pom.xml**.
5. Update the project.
6. Build the project.



## 6 Import the Front-End Files

### 6.1 Import the CSS File

1. Download **style.css** from ulearn.
2. Open **Eclipse**.
3. Create a folder named **css** in **src/main/resources/static**.
4. Import **style.css** (that was downloaded in step 1) into the folder created in step 3.

### 6.2 Import Header and Footer

1. Download **header.html** and **footer.html** from ulearn.
2. Create a folder named **fragments** in **src/main/resources/templates**.
3. Import the files into **fragments** folder that was created in step 2.

### 6.3 Import HTML Files

1. Download **ordertypes.html** from ulearn.
2. Import the file in step 1 into **src/main/resources/templates**.

## 7 Displaying a List of Order Types

### 7.1 New Definition of getOrderTypes

1. Amend method `getOrderTypes ( )` with the new definition as shown in Figure 7.1.

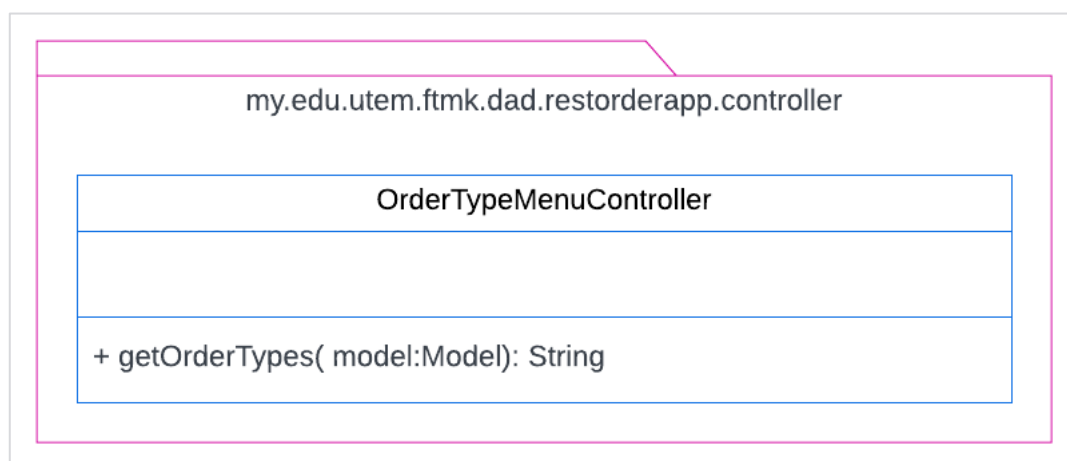


Figure 7.1: New design of `getOrderTypes ( )` method

2. Import `Model` from `org.springframework.ui`.

3. Locate the code that parse JSON data into an array object in the method.
4. Remove all the codes after it. The implementation of `getOrderTypes ( )` method should be similar as shown in Figure 7.2.

```
@GetMapping("/ordertype/list")
public String getOrderTypes(Model model) {

    // The URI for GET order types
    String uri = "http://localhost:8080/orderapp/api/ordertypes";

    // Get a list order types from the web service
    RestTemplate restTemplate = new RestTemplate();
    ResponseEntity<OrderType[]> response =
        restTemplate.getForEntity(uri, OrderType[].class);

    // Parse JSON data to array of object
    OrderType orderTypes[] = response.getBody();

}
```

Figure 7.2: `getOrderTypes ( )` method after removing console-based code

5. Add the codes shown in Figure 7.3 before the end of the method.

```
// Parse an array to a list object
List<OrderType> orderTypeList = Arrays.asList(orderTypes);

// Attach list to model as attribute
model.addAttribute("orderTypes",orderTypeList);

return "ordertypes";
```

Figure 7.3: An additional new implementation for `getOrderTypes ( )` method

6. Import `List` and `Arrays` from `java.util`.
7. Fix errors, if any.
8. Save the class.

## 7.2 Display `ordertypes.html`

1. Open a browser.
2. Paste <http://localhost:8080/orderapp/ordertype/list> in the address bar.  
The browser should display a list of record as shown in Figure 7.4.

Master Information   Order Types   Add New Order Type		
List of Order Types		
Id	Code	Name
1	ETS	Eat-In Table Service
2	TKA	Take Away
3	DRV	Drive-thru
4	EPU	Eat-In Pick-up
5	APU	Apps Pick-up
6	ADL	Apps Delivery
7	FPD	Food Panda
8	GFD	Grab Food
Developed at FTMK, UTeM		

Figure 7.4: A list of record from the consumption of a GET web service

## 8 Adding New Order Type

### 8.1 Define updateOrderType ( ) Method

1. Open `OrderTypeMenuController.java`.
2. Define attribute `defaultURI` as shown in Figure 8.1.

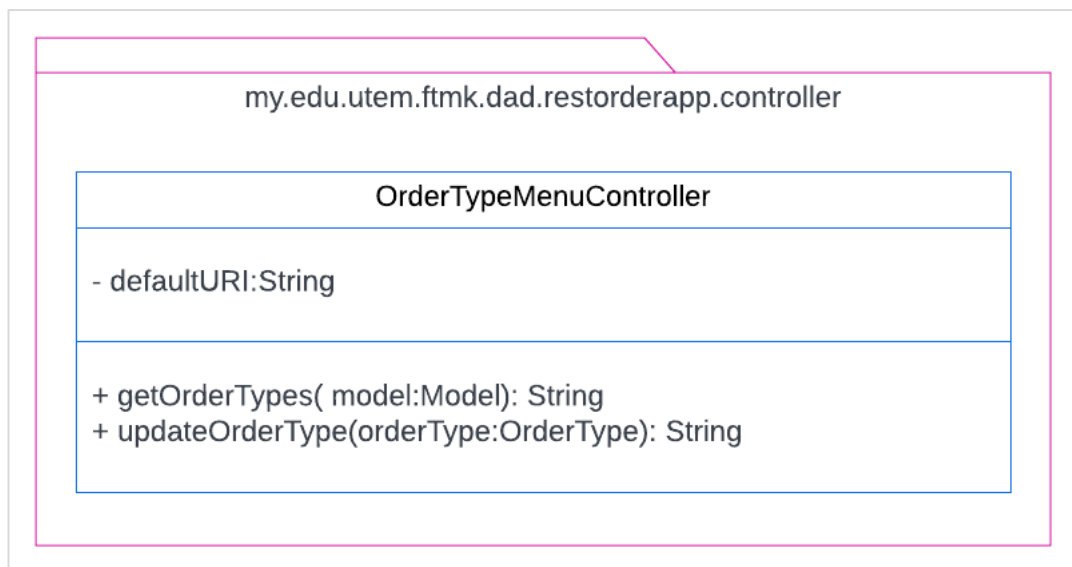


Figure 8.1: A revised model of `OrderTypeMenuController` class

3. Define `updateOrderType ( )` method shows in Figure 8.1. This method shall add or edit an order type.
4. Annotate the `OrderType` parameter with `@ModelAttribute`.

5. Import the annotation from  
`org.springframework.web.bind.annotation.`
6. Annotate the method with `@RequestMapping("/ordertype/save")`.
7. Import the annotation from  
`org.springframework.web.bind.annotation.`
8. Describe the method in a block of comment.
9. Save the class.
10. The initial definition of the method shall look similar as shown in Figure 8.2.

```
/**
 * This method will update or add an order type.
 *
 * @param orderType
 * @return
 */
@RequestMapping("/ordertype/save")
public String updateOrderType (@ModelAttribute OrderType orderType) {

}
```

Figure 8.2: An initial definition of `updateOrderType()` method

## 8.2 Implement updateOrderType( ) Method

1. Implement the body of updateOrderType( ) method using the code shown in Figure 8.3.

```
// Create a new RestTemplate
RestTemplate restTemplate = new RestTemplate();

// Create request body
HttpEntity<OrderType> request = new HttpEntity<OrderType>(orderType);

String orderTypeResponse = "";

if (orderType.getOrderTypeId() > 0) {
    // This block update an new order type and

    // Send request as PUT
    restTemplate.put(defaultURI, request, OrderType.class);
} else {
    // This block add a new order type

    // send request as POST
    orderTypeResponse = restTemplate.postForObject(
        defaultURI, request, String.class);
}

System.out.println(orderTypeResponse);

// Redirect request to display a list of order type
return "redirect:/ordertype/list";
```

Figure 8.3: An implementation of updateOrderType() method

2. Import HttpEntity from org.springframework.http.
3. Save the class.

### 8.3 Define getOrderType( ) Method

1. Define getOrderType ( ) method shows in Figure 8.4. This method gets an order type.

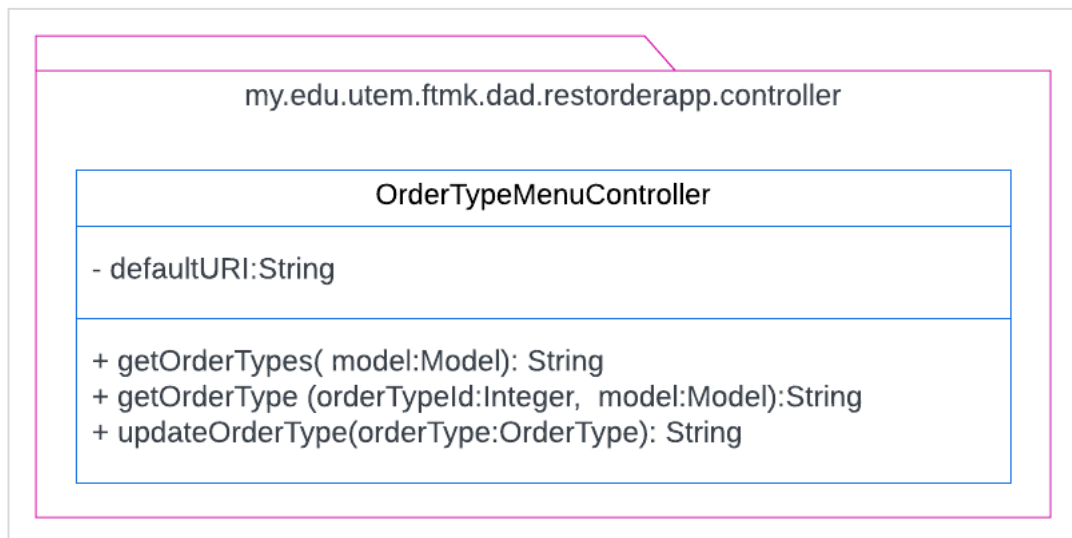


Figure 8.4: A revised model of OrderTypeMenuController with getOrderType( ) method

2. Annotate the `orderId` parameter with `@PathVariable`.
3. Import the annotation from `org.springframework.web.bind.annotation`.
4. Annotate the method with `@GetMapping("/ordertype/{orderId}").`
5. Describe the method in a block of comment. The initial definition of the method should be similar as shown in Figure 8.5.

```
/**
 * This method gets an order type
 *
 * @param orderId
 * @param model
 * @return
 */
@GetMapping("/ordertype/{orderId}")
public String getOrderType (@PathVariable Integer orderId, Model model) {

}
```

Figure 8.5: Initial definition of getOrderType() method

6. Save the class.

## 8.4 Implement getOrderType( ) method

1. Implement the body of getOrderType ( ) method using the code shown in Figure 8.6.

```
String pageTitle = "New Order Type";
OrderType orderType = new OrderType();

// This block get an order type to be updated
if (orderId > 0) {

    // Generate new URI and append orderId to it
    String uri = defaultURI + "/" + orderId;

    // Get an order type from the web service
    RestTemplate restTemplate = new RestTemplate();
    orderType = restTemplate.getForObject(uri, OrderType.class);

    // Give a new title to the page
    pageTitle = "Edit Order Type";
}

// Attach value to pass to front end
model.addAttribute("orderType", orderType);
model.addAttribute("pageTitle", pageTitle);

return "ordertypeinfo";
```

Figure 8.6: The Java code for implementation of getOrderType( ) method

2. Save the class.

## 9 Attaching The Mapping of getOrderType( ) to the Front End

### 9.1 Attaching the Mapping to Add New Order Type Link

1. Open **header.html**.
2. Find a comment that contains [A menu link to add a new order type](#).
3. Remove the comment block after it. The **html** code should be similar as shown in Figure X.

```

<!-- A menu link to add a new order type -->
<li class="nav-item">
  <a class="nav-link" th:href="@{/ordertype/0}">Add New Order Type</a>
</li>

```

Figure 9.1: A menu link to add a new order

4. Save the file.

## 9.2 Adding a Form to Add a New OrderType

1. Download **ordertypeinfo.html** from **ulearn**.
2. Import the file into **src/main/resources/templates**.

## 9.3 Display ordertypeinfo.html

1. Stop the execution of the project.
2. Update the project.
3. Build the project.
4. Open a browser.
5. Paste <http://localhost:8080/orderapp/ordertype/list> into the address bar.  
The browser should display a list of order type, similar to Figure 7.4.
6. Click **Add New Order Type** link at the top of the page. A page named **New Order Type**, as shown in Figure 9.2 will be displayed.

Figure 9.2: A page to add a new order type

7. Click the **Cancel** button. The browser shall display a list of order types, similar to Figure 7.4.



## 10 Add New Order Type

1. Click **Add New Order Type** link.
2. Add the data described in Table 10.1 to create a new record.

Table 10.1: A new order type data

Field	Value
Code	EPU
Name	Eat-In Pick-Up Order

3. Click the **Save** button.
4. The new record will be displayed in the list. It should look similar as shown in Figure 10.1: EPU order type is added at the end of the list Figure 10.1.

List of Order Types		
Id	Code	Name
1	ETS	Eat-In Table Service
2	TKA	Take Away
3	DRV	Drive-thru
4	EPU	Eat-In Pick-up
5	APU	Apps Pick-up
6	ADL	Apps Delivery
7	FPD	Food Panda
8	GFD	Grab Food
16	EPU	Eat-In Pick-Up Order
Developed at FTMK, UTeM		

Figure 10.1: EPU order type is added at the end of the list

5. Repeat step 1 until 3 to add the more record for the order type. Use the data describe in Table 10.2.

Table 10.2: New data for order types

Code	Name
ETO	Eat-In Table Order
ECR	Eat-In Cash Register Order
OTT	Order type test data

## 11 Edit an Order Type

### 11.1 Attaching getOrderType( ) Mapping to Edit Order Type Link

1. Open **ordertypes.html**.
2. Find `<th scope="col">Action</th>` in a comment block.
3. Remove the comment block once it is found.
4. Find a text '[A link to edit an order type](#)' in a comment block.
5. Remove the comment block after the statement in step 4. It should look similar as shown in Figure 11.1.












```
<!-- A link to edit an order type -->
<a th:href="@{'/ordertype/' + ${orderType.orderTypeId}}"
    title="Edit this order type"
    class="fa-regular fa-pen-to-square icon-dark"></a>
    &nbsp;
```

Figure 11.1: Uncommented code to edit a product type

6. Save the file.

### 11.2 Edit an OrderType from ordertypeinfo.html

1. Click **Order Type** link from the header to the web page. It should display a list of order type. It should look similar as shown in Figure 11.2.

List of Order Types			
Id	Code	Name	Action
1	ETS	Eat-In Table Service	
2	TKA	Take Away	
3	DRV	Drive-thru	
4	EPU	Eat-In Pick-up	
5	APU	Apps Pick-up	
6	ADL	Apps Delivery	
7	FPD	Food Panda	
8	GFD	Grab Food	
16	EPU	Eat-In Pick-Up Order	
17	ECR	Eat-In Cash Register Order	
18	OTT	Order type test data	

Developed at FTMK, UTeM

Figure 11.2: Order type records with pencil icons to edit

- Click the **pencil** icon at row with code **OTT**. The browser shall display page to edit an order type, as shown in Figure 11.3.

## Edit Order Type

Code

Name

Save

Cancel

Developed at FTMK, UTeM

Figure 11.3: A page to edit an order type

- Change the value for **Name** to **Order type test data (Edited)**.

4. Click the **Save** button. The name of the updated order type shall be similar as in step 3 after the Save button is clicked.
5. Repeat steps 1 until 4 for other records. Use your creativity.

## 12 Delete an Order Type

### 12.1 Define deleteOrderType() Method

1. Define deleteOrderType() method as shown in Figure 12.1. This method will delete an order type.

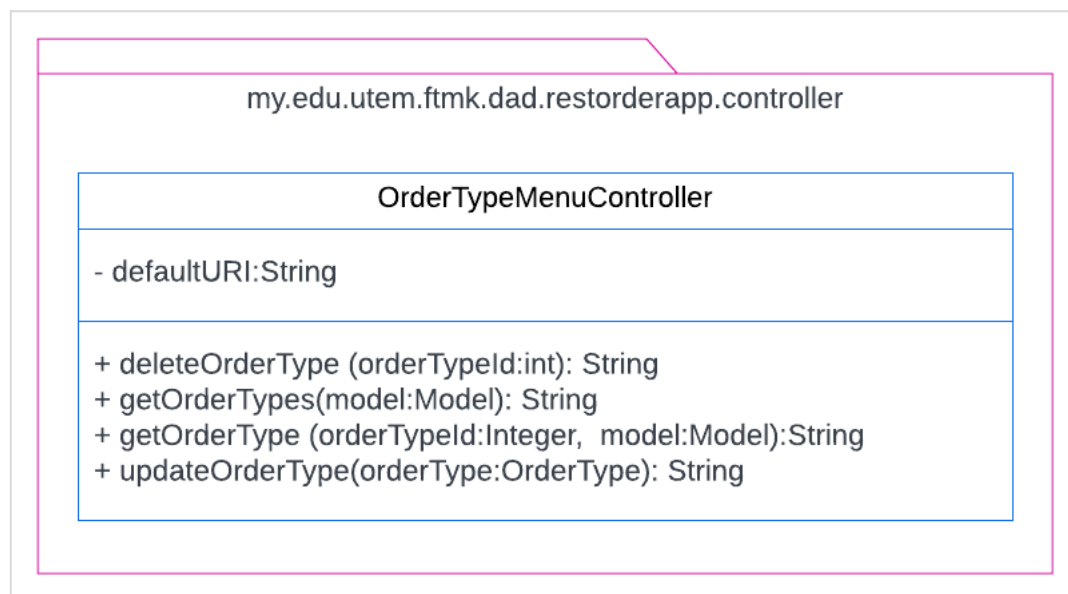


Figure 12.1: A revised model or OrderTypeMenuController class with deleteOrderType() method

2. Describe the method in a comment block.
3. Annotate the parameter in the method to `@PathVariable`.
4. Annotate the method to `@RequestMapping("/ordertype/delete/{orderTypeId}")`. The initial definition shall be similar as shown in Figure 12.2.

```

/**
 * This method deletes an order type
 *
 * @param orderId
 * @return
 */
@RequestMapping("/ordertype/delete/{orderId}")
public String deleteOrderType(@PathVariable Integer orderId) {

}

```

Figure 12.2: Initial definition of deleteOrderType() method

5. Save the class.

## 12.2 Implement deleteOrderType()

1. Add the code shown in Figure 12.3 into the body of deleteOrderType().

```

// Generate new URI, similar to the mapping in OrderTypeRestController
String uri = defaultURI + "{orderId}";

// Send a DELETE request and attach the value of orderId into URI
RestTemplate restTemplate = new RestTemplate();
restTemplate.delete(uri,
    Map.of("orderId", Integer.toString(orderId)));

return "redirect:/ordertype/list";

```

Figure 12.3: The Java implementation code for deleteOrderType() method

2. Import Map from java.util.
3. Save the class.

## 12.3 Attaching deleteOrderType ( ) Mapping to Delete Order Type Link

1. Open ordertypes.html.
2. Find [A link to delete an order type](#) in a comment block.
3. Remove the comment block after the statement in step 2. It should look similar as shown in Figure 12.4.

```

<!-- A link to delete an order type -->
<a th:href="@{/ordertype/delete/" + ${orderType.orderTypeId}"
th:orderTypeCode="${orderType.code}" id="btnDelete"
title="Delete this order type"
class="fa-regular fa-trash-can icon-dark btn-delete"></a>

```

Figure 12.4: Uncommented menu link to delete an order type

4. Save the file.

## 12.4 Display Link to Delete an Order Type

1. Stop the project execution.
2. Update the project.
3. Build the project.
4. Paste <http://localhost:8080/orderapp/ordertype/list> in a browser. A list of order type record will be displayed with bin icon, as shown in Figure 12.5 will be displayed.























List of Order Types			
Id	Code	Name	Action
1	ETS	Eat-In Table Service	 
2	TKA	Take Away	 
3	DRV	Drive-thru	 
4	EPU	Eat-In Pick-up	 
5	APU	Apps Pick-up	 
6	ADL	Apps Delivery	 
7	FPD	Food Panda	 
8	GFD	Grab Food	 
16	EPU	Eat-In Pick-Up Order	 
17	ECR	Eat-In Cash Register Order	 
18	OTT	Order type test data (Edited)	 
Developed at FTMK, UTeM			

Figure 12.5: List of order types with bin icon to delete the record

## 12.5 Delete an OrderType from a Link

1. Click the bin at the row of OTT. A modal window as shown in Figure 12.6 will be displayed.

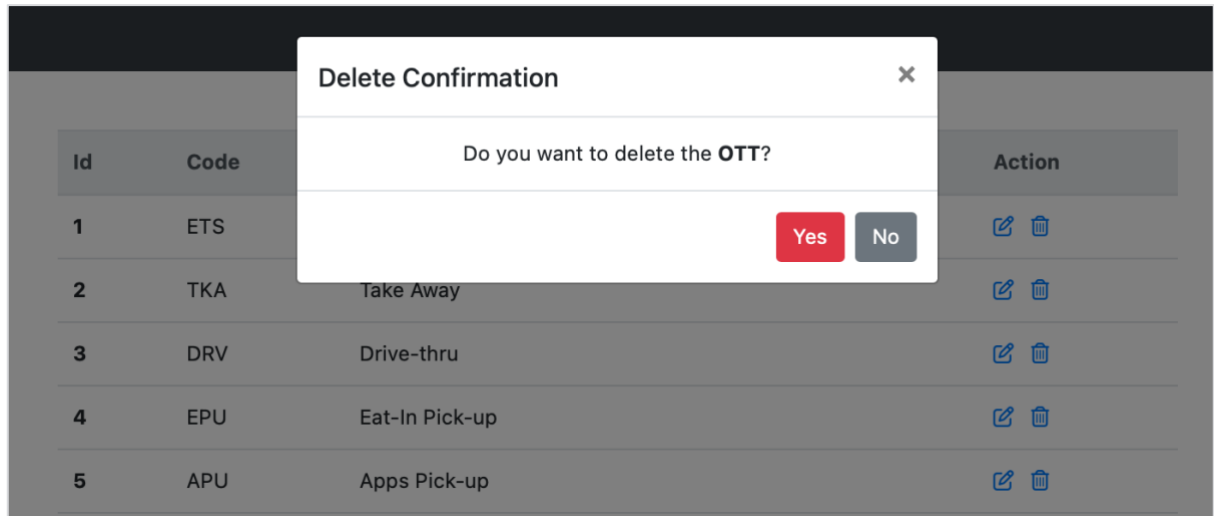


Figure 12.6: A modal window to confirm deletion of an order type

2. Click the **Yes** button. The OTT record will be deleted and disappear from the list.
3. Repeat step 1 and 2 for other records of your choice.

## 13 Exercise 1: Consume Web Service for REST Product Type

1. Create a solution to consume the REST web service for Product Type that was developed in Lab 4.
2. Push your solution in GitHub.
3. Invite emalian.kasmuri at gmail dot com to your Github.

---

*End of Document*

---

