

SPRING MVC

Done by Ravi Theja Kolluru

Email ID: ravi.kolluru@accolitedigital.com

Assignment

Create Spring MVC application [properly MVC architecture]

Order management application

API -> create order

API -> add Item

API -> view the items

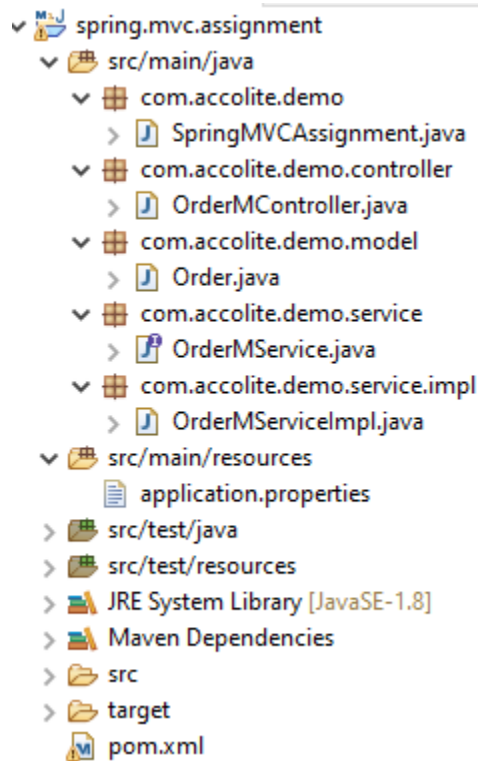
API -> delete item

API -> update item description (any field)

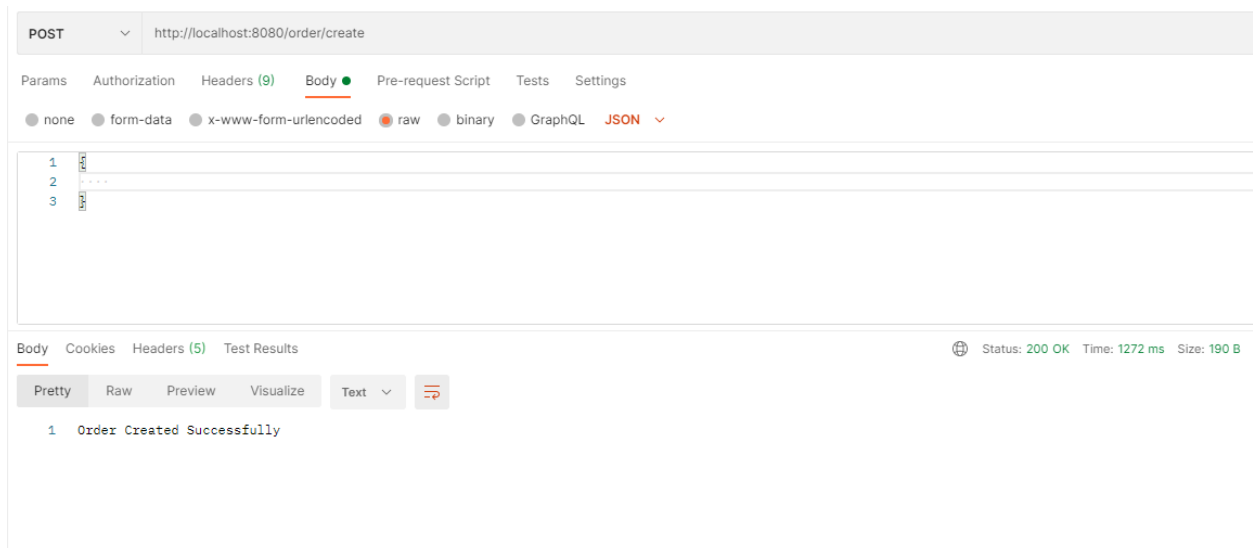
-> Dependency injection wherever applicable

-> Validations and error handling wherever applicable

ANSWER



ORDER CREATE



ORDER ADD

http://localhost:8080/order/add

POST http://localhost:8080/order/add

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "id": "1",
3   "name": "Ravi",
4   "emailId": "abc@gamil.com",
5   "phone": "5555888892",
6   "product": "Laptop",
7   "quantity": "2"
8 }
```

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 3.03 s Size: 263 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "name": "Ravi",
4   "emailId": "abc@gamil.com",
5   "phone": 5555888892,
6   "product": "Laptop",
7   "quantity": 2
8 }
```

ORDER VIEW/GET ITEMS

http://localhost:8080/order/get/1

GET http://localhost:8080/order/get/1

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "id": "1",
3   "name": "Ravi",
4   "emailId": "abc@gamil.com",
5   "phone": "5555888892",
6   "product": "Laptop",
7   "quantity": "2"
8 }
```

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 158 ms Size: 263 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "name": "Ravi",
4   "emailId": "abc@gamil.com",
5   "phone": 5555888892,
6   "product": "Laptop",
7   "quantity": 2
8 }
```

ORDER UPDATE

The screenshot displays two sequential REST client interface views for PUT requests.

First Request:

- URL: `http://localhost:8080/order/update/1`
- Method: `PUT`
- Body (JSON):

```
{  "quantity": "3"}
```
- Status: `200 OK`, Time: `31 ms`, Size: `263 B`
- Response Body (JSON):

```
{  "id": 1,  "name": "Ravi",  "emailId": "abc@gamil.com",  "phone": 5555888892,  "product": "Laptop",  "quantity": 3}
```

Second Request:

- URL: `http://localhost:8080/order/update2/1`
- Method: `PUT`
- Body (JSON):

```
{  "product": "Laptop2"}
```
- Status: `200 OK`, Time: `43 ms`, Size: `264 B`
- Response Body (JSON):

```
{  "id": 1,  "name": "Ravi",  "emailId": "abc@gamil.com",  "phone": 5555888892,  "product": "Laptop2",  "quantity": 3}
```

ORDER DELETE

The screenshot displays a REST client interface for a DELETE request. The URL bar shows `http://localhost:8080/order/delete/1`. The method is set to **DELETE**. The request body is empty. The response is displayed in the 'Body' tab, showing `Deleted Successfully` with a status of **200 OK**, a time of **126 ms**, and a size of **184 B**. The interface includes tabs for Overview, Params, Authorization, Headers (7), Body, Pre-request Script, Tests, and Settings. The 'Body' tab is active, showing the response text. The 'Send' button is visible in the top right corner.

Note: Dependency Injections included.

Validations and Error Handling

The screenshot shows a REST client interface with the following details:

- Overview:** PUT http://localhost:8080/... and POST http://localhost:8080/...
- URL:** http://localhost:8080/order/add
- Method:** POST
- Body:**

```

{
  "id": "1",
  "emailId": "abc@gamil.com",
  "phone": "5555888892",
  "product": "Laptop",
  "quantity": "2"
}

```
- Response:**

```

{
  "timestamp": "2021-01-25T06:58:48.927+00:00",
  "status": 400,
  "error": "Bad Request",
  "message": "",
  "path": "/order/add"
}

```
- Status:** 400 Bad Request, Time: 343 ms, Size: 257 B

```
codes [order.name,name]; arguments []; default message [name]]; default message [Name can't be empty]] ]
```
