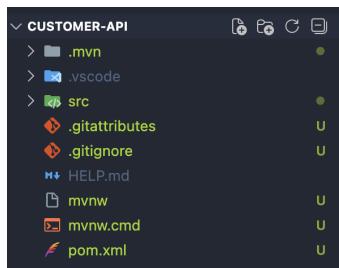


Lab 8

Github site & link

<https://github.com/alexis0704/web-dev-practice-lab/tree/main/Lab8/customer-api>

Task 1.1: Create Spring Boot Project



Configuration:

Spring Boot: 4.0.0

Group: com.example

Artifact: customer-api

Java: 17

Task 1.2, 1.3, 2.1, 2.2, 2.3, 3.1, 3.2, 3.3, 4.1, 4.2: Database Setup, Create Customer Entity, Create Request DTO, Create Response DTO, Create Error Response DTO, Create Repository, Create Service Interface, Implement Service, Create Basic REST Controller, Add Exception Handling

I simply pasted in the given code in the guide.

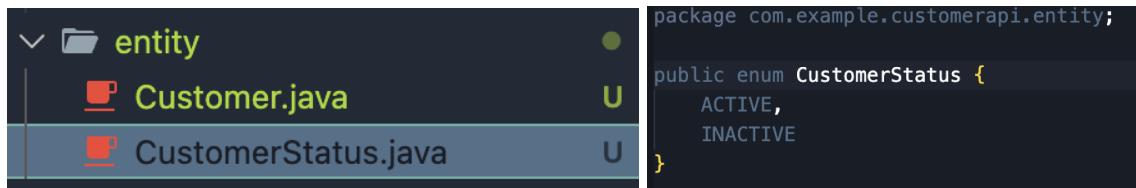
However, there's a bug as follows

```
private CustomerResponseDTO convertToResponseDTO(Customer customer) {
    CustomerResponseDTO dto = new CustomerResponseDTO();
    dto.setId(customer.getId());
    dto.setCustomerCode(customer.getCustomerCode());
    dto.setFullName(customer.getFullName());
    dto.setEmail(customer.getEmail());
    dto.setPhone(customer.getPhone());
    dto.setAddress(customer.getAddress());
    dto.setStatus(customer.getStatus().toString());  Enum.toString() is defined in an inaccessible class or interface
    dto.setCreatedAt(customer.getCreatedAt());
    return dto;
}
```

```
// Enum for status
enum CustomerStatus {
    ACTIVE,
    INACTIVE
}
```

The service couldn't use the method of the CustomerStatus enum, defined in the Customer.java entity file. Here, I think we can either modify the Customer's getStatus() method to implement toString(), or we can create a new CustomerStatus enum class.

I go with the second approach, because it improves maintainability and readability. Also, it avoids modifying the entity's basic method unnecessarily.



```
package com.example.customerapi.entity;

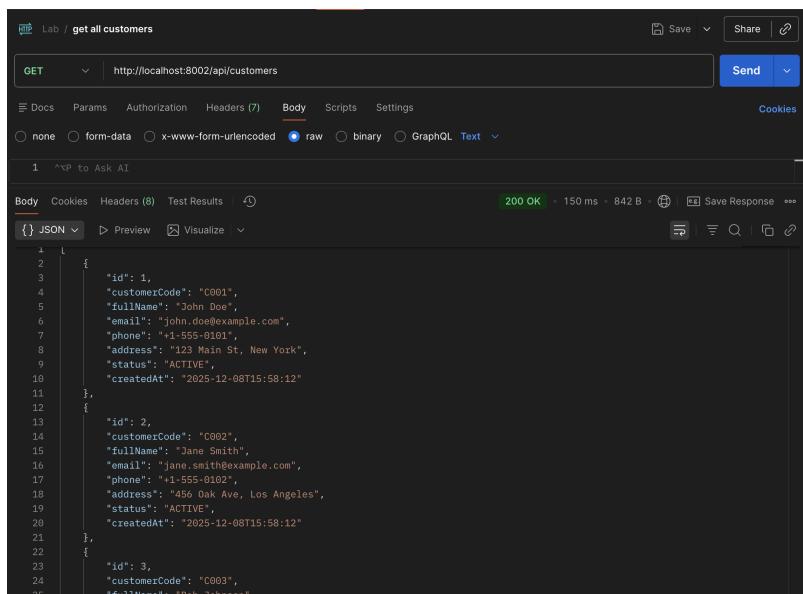
public enum CustomerStatus {
    ACTIVE,
    INACTIVE
}
```

After moving CustomerStatus to its own public enum class, the method shows no bug anymore:

```
private CustomerResponseDTO convertToResponseDTO(Customer customer) {
    CustomerResponseDTO dto = new CustomerResponseDTO();
    dto.setId(customer.getId());
    dto.setCustomerCode(customer.getCustomerCode());
    dto.setFullName(customer.getFullName());
    dto.setEmail(customer.getEmail());
    dto.setPhone(customer.getPhone());
    dto.setAddress(customer.getAddress());
    dto.setStatus(customer.getStatus().toString());
    dto.setCreatedAt(customer.getCreatedAt());
    return dto;
}
```

Testing all endpoints with Postman:

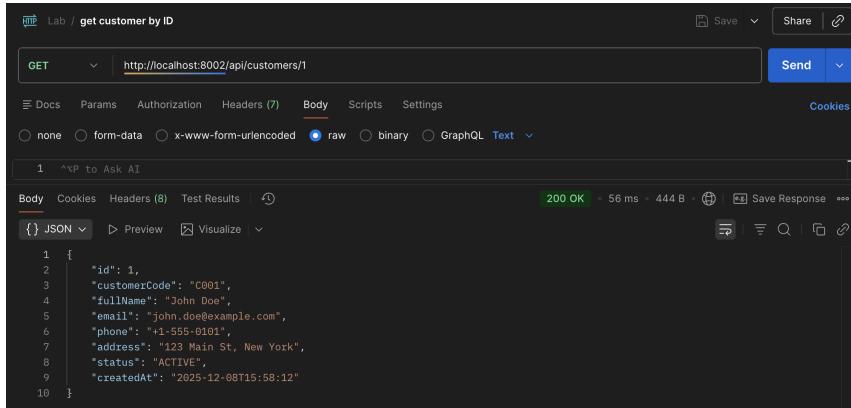
Get all customers



id	customerCode	fullName	email	phone	address	status	createdAt
1	C001	John Doe	john.doe@example.com	+1-555-0101	123 Main St, New York	ACTIVE	2025-12-08T15:58:12
2	C002	Jane Smith	jane.smith@example.com	+1-555-0102	456 Oak Ave, Los Angeles	ACTIVE	2025-12-08T15:58:12
3	C003	Bob Johnson	bob.johnson@example.com	+1-555-0103	789 Elm St, Chicago	ACTIVE	2025-12-08T15:58:12

```
[  
  {  
    "id": 3,  
    "customerCode": "C003",  
    "fullName": "Bob Johnson",  
    "email": "bob.johnson@example.com",  
    "phone": "+1-555-0103",  
    "address": "789 Pine Rd, Chicago",  
    "status": "ACTIVE",  
    "createdAt": "2025-12-08T15:58:12"  
  }  
]
```

Get customer by ID



HTTP Lab / get customer by ID

GET <http://localhost:8002/api/customers/1> Send

Docs Params Authorization Headers (7) Body Scripts Settings Cookies

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

1 ^xP to Ask AI

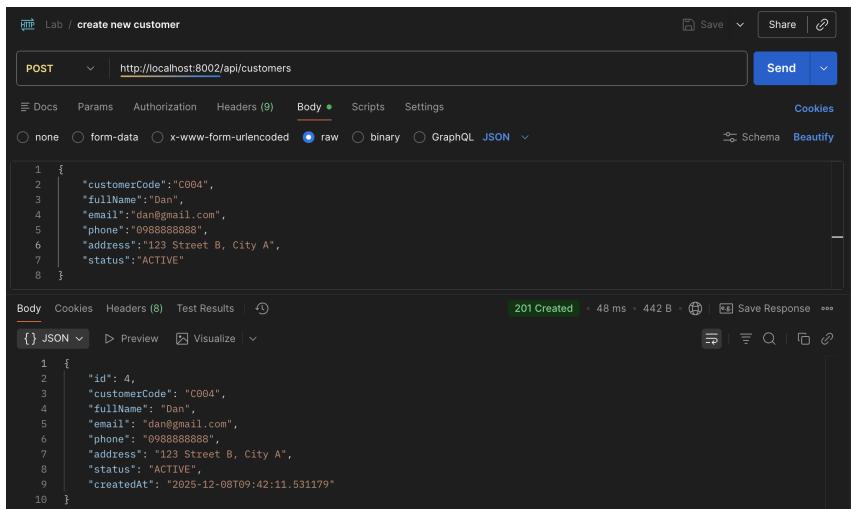
Body Cookies Headers (8) Test Results

200 OK 56 ms 444 B Save Response

{ } JSON Preview Visualize

```
[  
  {  
    "id": 1,  
    "customerCode": "C001",  
    "fullName": "John Doe",  
    "email": "john.doe@example.com",  
    "phone": "+1-555-0101",  
    "address": "123 Main St, New York",  
    "status": "ACTIVE",  
    "createdAt": "2025-12-08T15:58:12"  
  }  
]
```

Create new customer



HTTP Lab / create new customer

POST <http://localhost:8002/api/customers> Send

Docs Params Authorization Headers (9) Body Scripts Settings Cookies

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

1 {
 "customerCode": "C004",
 "fullName": "Dan",
 "email": "dan@gmail.com",
 "phone": "0988888888",
 "address": "123 Street B, City A",
 "status": "ACTIVE"
}
2
3
4
5
6
7
8
9
10

Body Cookies Headers (8) Test Results

201 Created 48 ms 442 B Save Response

{ } JSON Preview Visualize

```
[  
  {  
    "id": 4,  
    "customerCode": "C004",  
    "fullName": "Dan",  
    "email": "dan@gmail.com",  
    "phone": "0988888888",  
    "address": "123 Street B, City A",  
    "status": "ACTIVE",  
    "createdAt": "2025-12-08T09:42:11.531179"  
  }  
]
```

Update Customer

The screenshot shows a Postman request to update a customer. The method is PUT, the URL is `http://localhost:8002/api/customers/1`, and the body contains the following JSON:

```
1 {
2     "customerCode": "C111",
3     "fullName": "JohnJohnJohn",
4     "email": "john@gmail.com",
5     "phone": "0999999999",
6     "address": "123 Street A, City B",
7     "status": "ACTIVE"
8 }
```

The response status is 200 OK, with a response time of 109 ms and a size of 440 B. The response body is:

```
1 {
2     "id": 1,
3     "customerCode": "C001",
4     "fullName": "JohnJohnJohn",
5     "email": "john@gmail.com",
6     "phone": "0999999999",
7     "address": "123 Street A, City B",
8     "status": "ACTIVE",
9     "createdAt": "2025-12-08T15:58:12"
10 }
```

Delete customer

The screenshot shows a Postman request to delete a customer. The method is DELETE, the URL is `http://localhost:8002/api/customers/4`, and the body is empty.

The response status is 200 OK, with a response time of 64 ms and a size of 296 B. The response body is:

```
1 {
2     "message": "Customer deleted successfully"
3 }
```

Search customer

The screenshot shows the Postman interface with the following details:

- URL:** `http://localhost:8002/api/customers/search?keyword=john`
- Method:** GET
- Query Params:**

Key	Value	Description
keyword	john	
- Response Status:** 200 OK
- Response Body (JSON):**

```

1  [
2    {
3      "id": 1,
4      "customerCode": "C001",
5      "fullName": "JohnJohnJohn",
6      "email": "john@gmail.com",
7      "phone": "0999999999",
8      "address": "123 Street A, City B",
9      "status": "ACTIVE",
10     "createdAt": "2025-12-08T15:58:12"
11   },
12   {
13     "id": 3,
14     "customerCode": "C003",
15     "fullName": "Bob Johnson",
16     "email": "bob.johnson@example.com",
17     "phone": "+1-555-0103",
18     "address": "789 Pine Rd, Chicago",
19     "status": "ACTIVE",
20     "createdAt": "2025-12-08T15:58:12"
21   }
]

```

Get customer by status

There's a type mismatch bug here for enum.

The screenshot shows the Postman interface with the following details:

- URL:** `http://localhost:8002/api/customers/status/ACTIVE`
- Method:** GET
- Body:** raw
- Response Status:** 500 Internal Server Error
- Response Body (Text):**

```

1  {
2    "status": 500,
3    "error": "Internal Server Error",
4    "message": "Argument [ACTIVE] of type [java.lang.String] did not match parameter type [com.example.customerapi.entity.CustomerStatus (n/a)]",
5    "path": "/api/customers/status/ACTIVE",
6    "details": null,
7    "timestamp": "2025-12-08T09:58:30.058134"
8  }

```

This is because the entity's attribute is `CustomerStatus`, but the whole flow from the Controller, to Service and to Repository throw in `String` status.

To fix the bug, I make the Service class convert the string to the enum type and make the Repository class save the enum:

- Old repository method

```
List<Customer> findByStatus(String status);
```

- New repository method

```
List<Customer> findByStatus(CustomerStatus status);
```

- Old service method

```
@Override  
public List<CustomerResponseDTO> getCustomersByStatus(String status) {  
    return customerRepository.findByStatus(status).stream().map(this::convertToResponseDTO)  
        .collect(Collectors.toList());  
}
```

- New service method

```
@Override  
public List<CustomerResponseDTO> getCustomersByStatus(String status) {  
    CustomerStatus enumStatus = CustomerStatus.valueOf(status.toUpperCase());  
    return customerRepository.findByStatus(enumStatus)  
        .stream()  
        .map(this::convertToResponseDTO)  
        .collect(Collectors.toList());  
}
```

Result of the bug resolve:

The screenshot shows the Postman interface with the following details:

- URL:** http://localhost:8002/api/customers/status/ACTIVE
- Method:** GET
- Body:** Raw JSON response (selected)
- Headers:** (7)
- Response Status:** 200 OK
- Response Time:** 91 ms
- Response Size:** 838 B
- Response Content:** (JSON array of three customer objects)

```
[{"id": 1, "customerCode": "C001", "fullName": "JohnJohnJohn", "email": "john@gmail.com", "phone": "0999999999", "address": "123 Street A, City B", "status": "ACTIVE", "createdAt": "2025-12-08T15:58:12"}, {"id": 2, "customerCode": "C002", "fullName": "Jane Smith", "email": "jane.smith@example.com", "phone": "+1-555-0102", "address": "456 Oak Ave, Los Angeles", "status": "ACTIVE", "createdAt": "2025-12-08T15:58:12"}, {"id": 3, "customerCode": "C003", "fullName": "Bob Johnson", "email": "bob.johnson@example.com"}]
```