In-Class Exercise - 19/03/2025

Project Files : https://csci22082.tiiny.site

Requirements

1. Customer Entity

- **Package:** `com.example.blockbusterapiv3.model`
- **Class Name:** `Customer`
- **Your Task:**
  - How will you mark this class as a JPA entity and optionally specify a table name (e.g., "customers")?
  - Which fields should you include? Consider the following:
    - `id` (Long): How can you define it as the primary key and configure it to be auto-generated?
    - `firstName` (String): How will you map this field to JSON using a property name like `first_name`?
    - `lastName` (String): How can you map this field similarly using `last_name`?
    - `email` (String): What considerations might you have for validation or format?
    - `registeredAt` (Instant): How will you handle JSON mapping for this field (e.g., `registered_at`)?
  - Which Lombok annotations can help you automatically generate boilerplate code (like getters, setters, and constructors), and how would you apply them?

---

2. Customer Repository

- **Package:** `com.example.blockbusterapiv3.repository`
- **Interface Name:** `CustomerRepository`
- **Your Task:**
  - How can you extend Spring Data JPA’s `JpaRepository` to leverage built-in CRUD operations for the `Customer` entity?

---

3. Customer Service

- **Package:** `com.example.blockbusterapiv3.service`
- **Class Name:** `CustomerService`
- **Your Task:**
  - How will you design a service layer that interacts with the `CustomerRepository`?
  - What methods will you implement to:
    - Retrieve all customers.
    - Retrieve a single customer by ID.
    - Create a new customer.
    - Update an existing customer.

        - Delete a customer.
    - How will you handle cases when a customer is not found? Think about the use of `Optional<Customer>` or another error handling strategy.
    - Consider how dependency injection should be applied to inject the repository into the service class.

---

4. Customer Controller

- **Package:** `com.example.blockbusterapiv3.controller`
- **Class Name:** `CustomerController`
- **Your Task:**
    - How will you set up this class as a REST controller and map it to the `/api/customers` endpoint?
    - Which endpoints will you expose to:
        - Retrieve all customers (HTTP GET).
        - Retrieve a customer by ID (HTTP GET).
        - Create a new customer (HTTP POST).
        - Update an existing customer (HTTP PUT).
        - Delete a customer (HTTP DELETE).
    - What HTTP status codes would be appropriate for each operation, and how can you structure your responses to reflect success or failure (e.g., 200 OK, 404 Not Found, 204 No Content)?
    - How will you inject and use the `CustomerService` within the controller?
