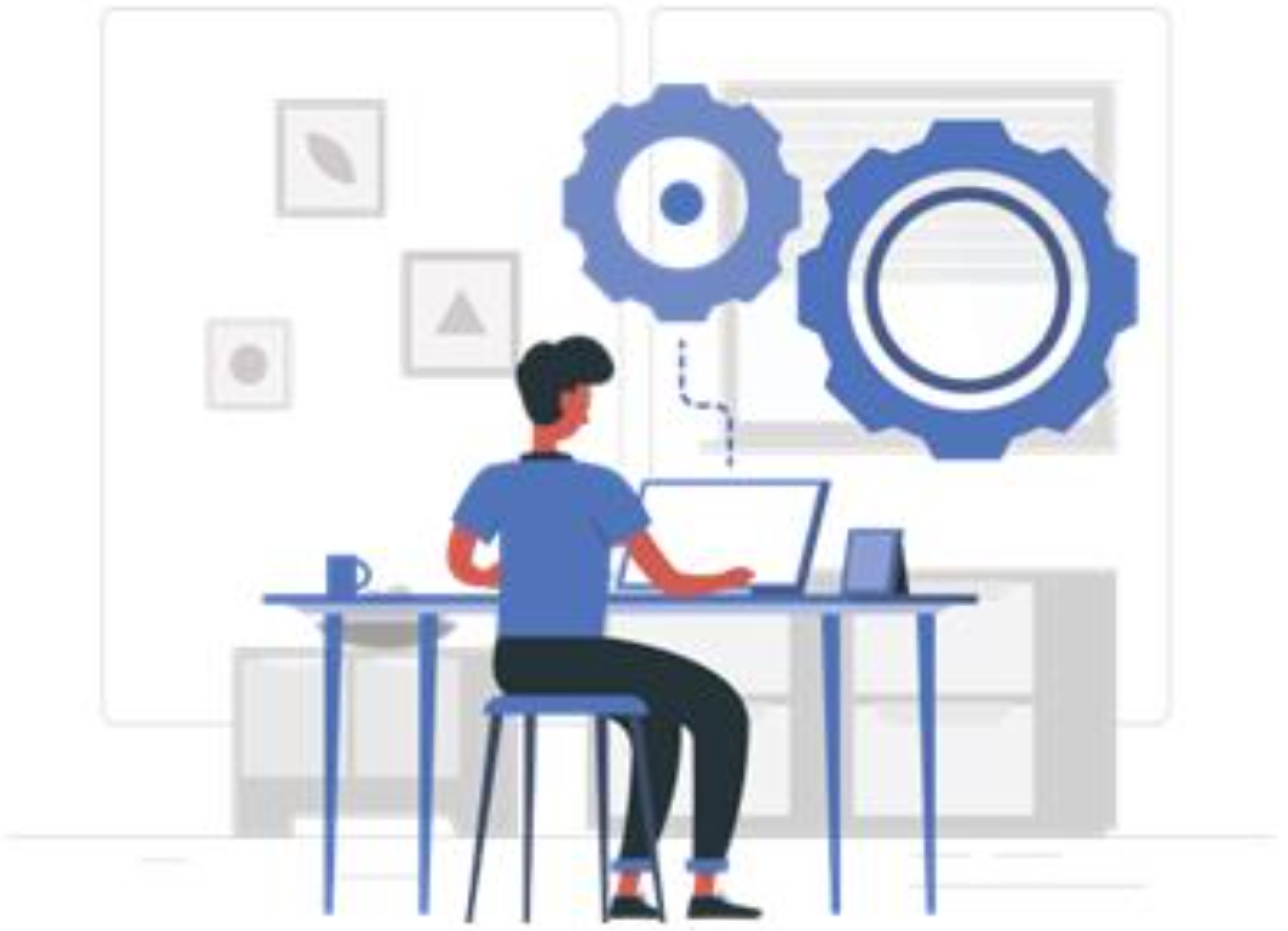


Practice

Manage Semi-structured and Unstructured Data and Handle Exceptions Within a RESTful Service by Using Mongo Repository

Exercise

- Practice 1: Customer Service

Implementation Environmet

- Refer to the documentation below before starting the challenge.
    - Spring Data Mongo
- If MongoDB does not start automatically, follow these steps in Windows:
    - ***Goto -> Control Panel -> Administrative Tools -> Services -> double click -> search for MongoDB Server(MongoDB) -> right click and start service***

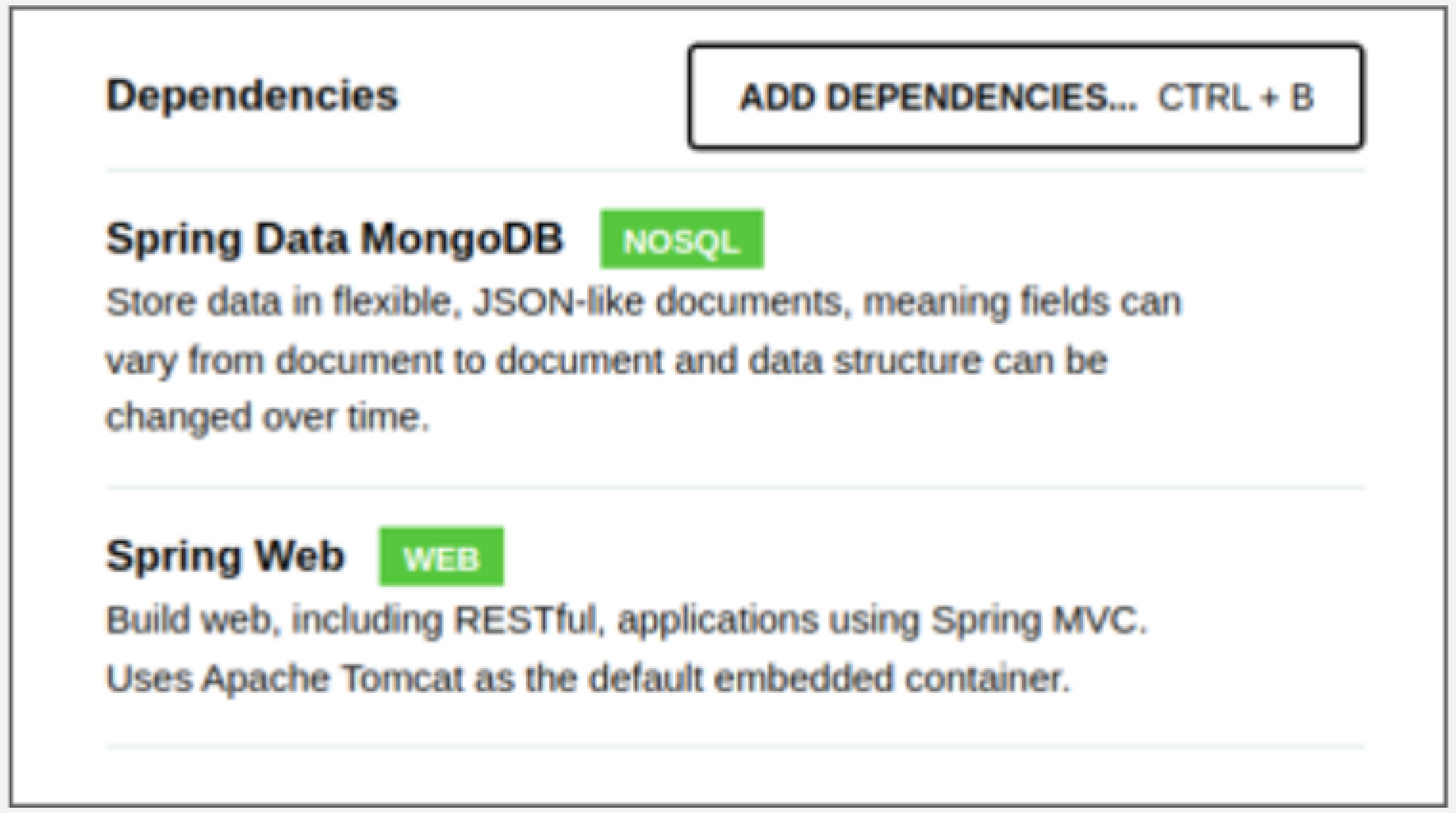
PRACTICE

Practice 1: Customer Service

An electronics store maintains sales records of its products and customers.
The store management wants to extract information on the purchase of specific products, such as purchase details, etc.
Build a REST API that will help store, save, and retrieve all the information.

Instructions for the Practice

- Create a Spring Boot application from the Spring Initializr.
- Add the necessary dependencies in pom.xml.
- Download the project into your local machine.
- Extract the zip file.
- Export the project in your local IDE.

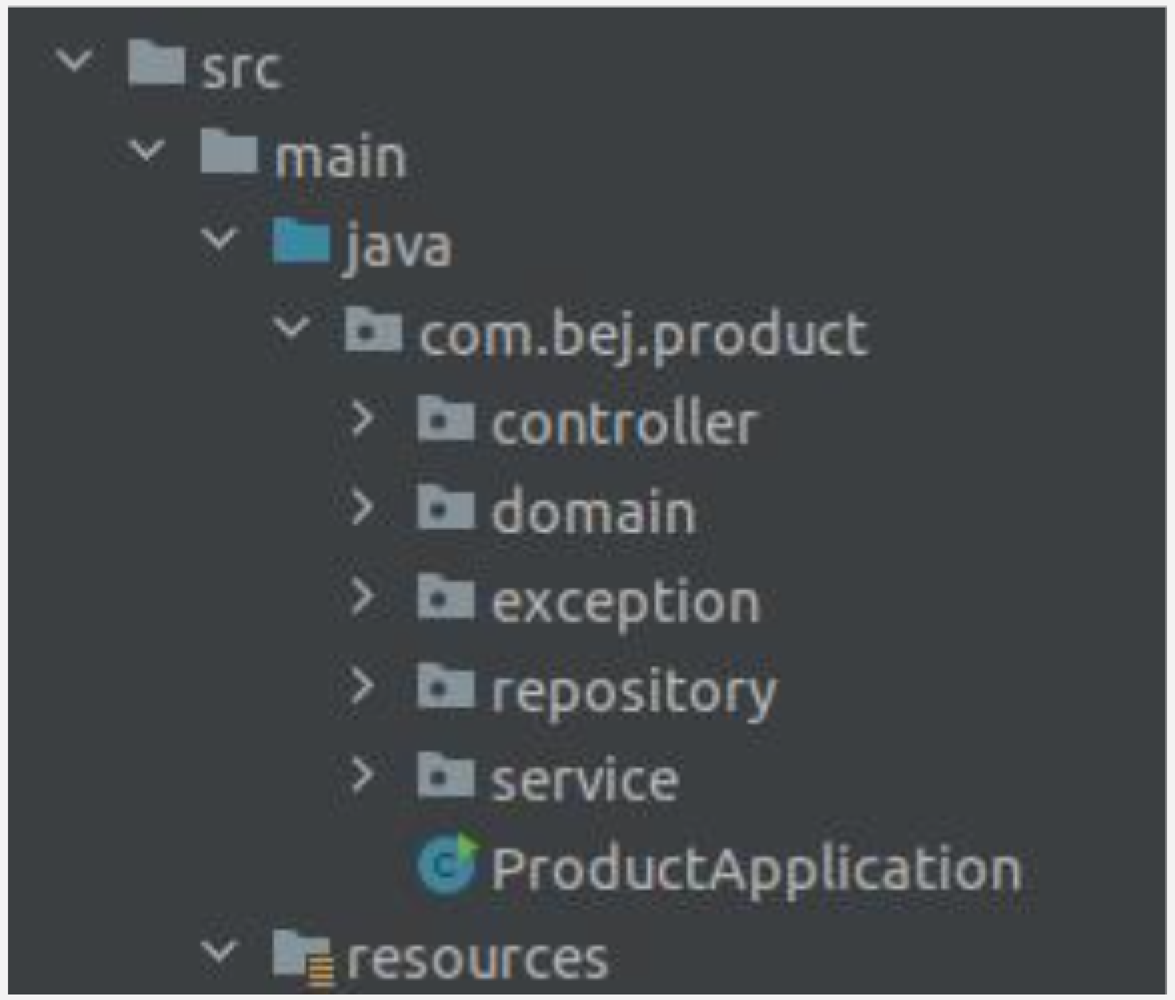


Spring Data MongoDB NOSQL
Store data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Task # 1 – Domain Classes

- Define the domain classes within the `domain` package
  - `Customer` with attributes `customerId` `customerName`, `customerPhoneNo`, `customerProduct` of type `Product`
  - `Product` with attributes `productId`, `productName`, `productDescription`
- Provide appropriate `@Document` and `@Id` annotations for the `Customer` class.
- Generate:
  - Getter and setter methods
  - Constructors – no argument and parameterized
- Override the `toString()` method.



```
src
  main
    java
      com.bej.product
        controller
        domain
        exception
        repository
        service
        ProductApplication
    resources
```

Structure of the Project

Task # 2 – Repository Layer

- Define a `CustomerRepository` interface that will inherit the `MongoRepository` inside the `repository` package.
- The `CustomerRepository` will have two parameters:
  1. The class annotated with `@Document`
  2. The datatype of the attribute annotated with `@Id` attribute
- Define a method inside the `CustomerRepository`, that will fetch all the details of customers who have bought a Samsung phone from the store.
- This method will return a `List` of `Customer` objects.
- Annotate the method with `@Query` and write the query.

Task # 3 – Service Layer

- Create the `ICustomerService` nterface and `CustomerServiceImpl` class inside the `service` package to provide the business logic for the application.
- Annotate the `CustomerServiceImpl` class with the `@Service` annotation.
- Create methods in the `ICustomerService` interface for the actions below:
    - Save a Customer and return the saved Customer object.
    - Delete a Customer and return true, if success, and false, if failure.
    - Retrieve all the Customers present in the database as a List.
    - Retrieve all the Customers who bought a Samsung phone.
- Exception handling should be done for all the methods.

Task # 3 – Service Layer (Contd.)

- The `CustomerServiceImpl` class must implement the `ICustomerService` interface and provide implementation for all its methods.
- Autowire `CustomerRepository` inside the `CustomerServiceImpl` class.
- Make calls to the appropriate methods of the `CustomerRepository` methods in the `CustomerServiceImpl` class.

Task # 4 – Controller Layer

- Create the `CustomerController` class in the Controller package
- Annotate the class with `@RestController` and `@RequestMapping` annotations.
- Autowire the `CustomerServiceImpl` class in the `CustomerController.`
- Define all the handler methods `(@PostMapping, @GetMapping, @DeleteMapping`) for handling the requests from the client.
- Call the appropriate service layer methods to process all the responses.
- Send the response back to the client.
- Set up the MongoDB database configuration details in the `application.properties` file.
- Run the boot application by using the Spring way of execution.
- Open the Postman and call all the REST API.

Submission Instructions

- Create a new repository on Git named **BEJ_C2_S2_REST_API_MONGODB_PC_1.**
- Push your code into the repository.
- There is no boilerplate for this practice.