Creating a Spring Boot application and writing code in IntelliJ is a common task for Java developers. Here is a step-by-step procedure for creating the "ExProductService" application using the provided code.

---

## Step 1: Create a New Spring Boot Project

1. Open IntelliJ IDEA.
2. Click on **File > New > Project...**.
3. In the left-hand panel, select **Spring Initializr**.
4. Configure the project:
    - **Project SDK**: Select **Java 21**.
    - **Type**: Choose **Maven**.
    - **Group**: com.learning
    - **Artifact**: ExProductService
    - **Name**: ExProductService
5. Click **Next**.
6. Select the required dependencies by searching for them:
    - Spring Web
    - Spring Data JPA
    - MySQL Driver
    - Spring Boot DevTools (optional but recommended for development)
7. Click **Create**. IntelliJ will generate the project structure and download the dependencies.

---

## Step 2: Configure the Database

1. **Start MySQL Server**: Ensure your MySQL database server is running.
2. **Create the Database**: Open a MySQL client and create a new database named demodb2.

   SQL

   ```
   CREATE DATABASE demodb2;
   ```

3. **Update application.properties**: Navigate to src/main/resources/ and open the application.properties file. Copy and paste the content from your provided file. Make sure to update the spring.datasource.username and spring.datasource.password fields to match your MySQL credentials.

   Properties

   ```
   spring.application.name=ExProductService

   server.port=8081

   #Database connectivity for Datasource creation in Spring Data JPA
   spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
   ```

```
spring.datasource.url=jdbc:mysql://localhost:3306/demodb2
spring.datasource.username=root
spring.datasource.password=

#logging.level.root=debug

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

The spring.jpa.hibernate.ddl-auto=update property will automatically create the product table based on your Product entity when the application runs for the first time.

---

## Step 3: Create the Java Classes

1. **Create the Model Package**: Right-click on com.learning.ExProductService and select **New > Package**. Name it Model.
2. **Create Product.java**: Inside the Model package, create a new Java class named Product.java. Copy and paste the provided code into this file. This class defines the Product entity, which maps to the product table in your database. Refer the below file:

   https://github.com/kannanmano6cfs/SBAug2025/blob/main/ExProductService/src/main/java/com/learning/exproductservice/Model/Product.java

3. **Create the Repository Package**: Right-click on com.learning.ExProductService and select **New > Package**. Name it Repository.
4. **Create ProductRepository.java**: Inside the Repository package, create a new Java **Interface** named ProductRepository.java. Copy and paste the provided code. This interface extends JpaRepository, giving you access to built-in CRUD operations.

   https://github.com/kannanmano6cfs/SBAug2025/blob/main/ExProductService/src/main/java/com/learning/exproductservice/Repository/ProductRepository.java

5. **Create the Controller Package**: Right-click on com.learning.ExProductService and select **New > Package**. Name it Controller.
6. **Create ProductController.java**: Inside the Controller package, create a new Java class named ProductController.java. Copy and paste the provided code. This class handles incoming REST requests and uses the ProductRepository to interact with the database.

   https://github.com/kannanmano6cfs/SBAug2025/blob/main/ExProductService/src/main/java/com/learning/exproductservice/Controller/ProductController.java

---

## Step 4: Run the Application

1. Navigate to the main application class, which is usually named ExProductServiceApplication.java. It should be located in the root package (com.learning.ExProductService).
2. Right-click on the file and select **Run 'ExProductServiceApplication.main()'**.

3. The application will start on port 8081. You can check the console output to confirm it has started successfully.
4. Open a web browser or API testing tool (like Postman) and test the endpoints, for example, http://localhost:8081/ should display "Welcome to the Product Service".
5. Practice all these
    1. http://localhost:8081/products
    2. http://localhost:8081/new/v1
    3. http://localhost:8081/new/v2

       Add this for body as json

```
{
    "prdname": "Samsung s54",
    "prdprice": 1000.0,
    "prddesc": "Samsung advanced smart mobile device"
}
```

    4. http://localhost:8081/product/v3?name=Sam