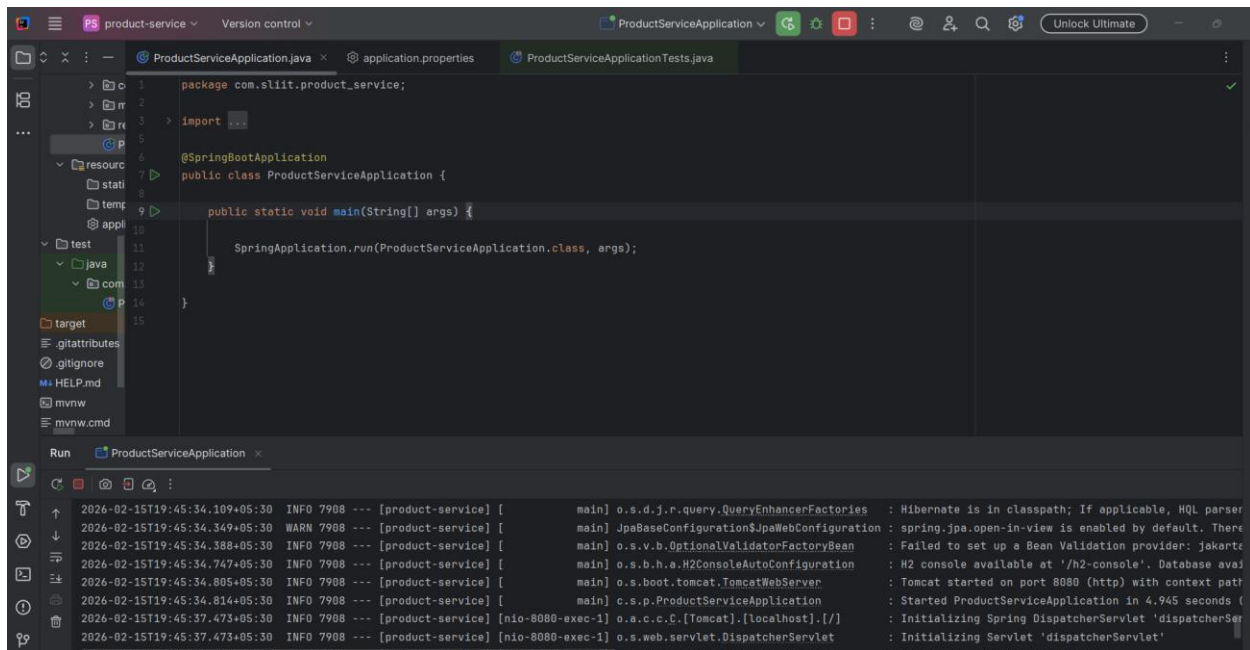


IT22341204 – K Rangana Malmi Nadee

Lab 03 - Current Trends in Software Engineering (SE4010)

Run Application in IDE

- Application was executed successfully. The console shows that the embedded Tomcat server started on port 8080 and the application started without errors.



The screenshot shows an IDE with the following components:

- File Explorer:** Shows the project structure with folders like 'resources', 'static', 'temp', 'app', 'test', 'java', 'com', 'target', '.gitattributes', '.gitignore', 'HELP.md', 'mvnw', and 'mvnw.cmd'.
- Editor:** Displays the code for `ProductServiceApplication.java`. The code is as follows:

```
package com.slit.product_service;

import ...

@SpringBootApplication
public class ProductServiceApplication {

    public static void main(String[] args) {

        SpringApplication.run(ProductServiceApplication.class, args);

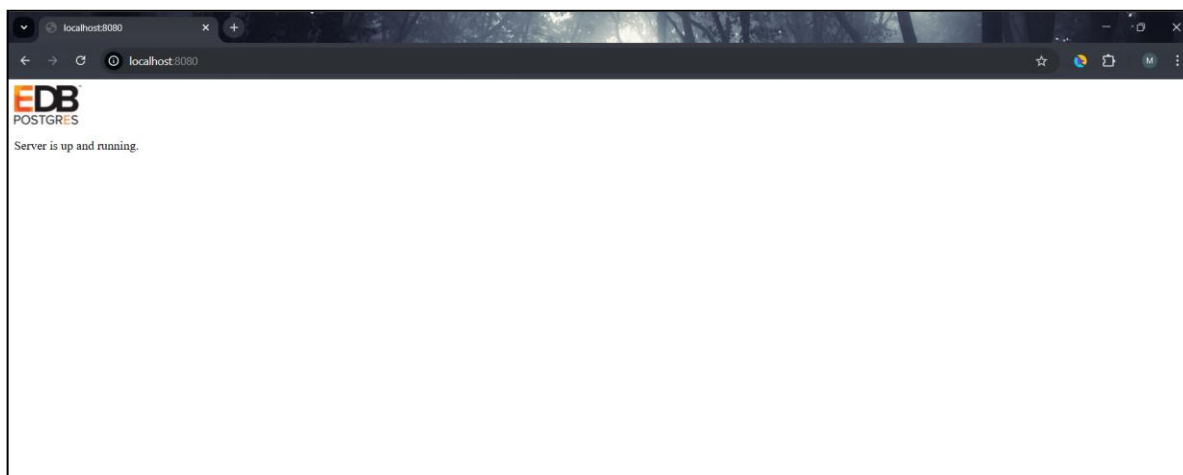
    }

}
```
- Run Console:** Shows the output of the application. The output includes the following lines:

```
2026-02-15T19:45:34.109+05:30 INFO 7908 --- [product-service] [main] o.s.d.j.r.query.QueryEnhancerFactories : Hibernate is in classpath; If applicable, HQL parser
2026-02-15T19:45:34.349+05:30 WARN 7908 --- [product-service] [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. There
2026-02-15T19:45:34.388+05:30 INFO 7908 --- [product-service] [main] o.s.v.b.OptionalValidatorFactoryBean : Failed to set up a Bean Validation provider: jakarte
2026-02-15T19:45:34.747+05:30 INFO 7908 --- [product-service] [main] o.s.b.h.a.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database avail
2026-02-15T19:45:34.805+05:30 INFO 7908 --- [product-service] [main] o.s.boot.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path
2026-02-15T19:45:34.814+05:30 INFO 7908 --- [product-service] [main] c.s.p.ProductServiceApplication : Started ProductServiceApplication in 4.945 seconds (
2026-02-15T19:45:37.473+05:30 INFO 7908 --- [product-service] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherSer
2026-02-15T19:45:37.473+05:30 INFO 7908 --- [product-service] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2026-02-15T19:45:37.473+05:30 INFO 7908 --- [product-service] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Servlet initialization complete; 2.0 seconds
```

Open browser and visit :

URL → <http://localhost:8080>



Access H2 Console

- The H2 in-memory database console was accessed successfully.

Open browser and visit :

URL → <http://localhost:8080/h2-console>

localhost:8080/h2-console/login.jsp?sessionId=746f6948057009c1bdc8e1d56ab7a771

English Preferences Tools Help

Login

Saved Settings: Generic H2 (Embedded)

Setting Name: Generic H2 (Embedded) Save Remove

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:file:D:/Slit/New_Folder/Y4_Sem_2/SE4010_CTS

User Name: sa

Password:

Connect Test Connection

H2 Product Table

localhost:8080/h2-console/login.do?sessionId=06d8a829c9b9b85b8600cc0778b8c14f

Auto commit Max rows: 1000 Auto complete Off Auto select On

jdbc:h2:file:D:/Slit/New_Folder/Y4_Sem_2/SE4010_CTS Run Run Selected Auto complete Clear SQL statement:

Select * From Product

Insert Into Product (NAME, PRICE) Values('Laptop', 200000);
Insert Into Product (NAME, PRICE) Values('Smartphone', 80000);
Insert Into Product (NAME, PRICE) Values('Tablet', 40000);
Insert Into Product (NAME, PRICE) Values('Smartwatch', 20000);
Insert Into Product (NAME, PRICE) Values('Headphones', 15000);
Insert Into Product (NAME, PRICE) Values('Mouse', 3000);

SELECT * FROM PRODUCT WHERE ID = 3;

DELETE FROM PRODUCT WHERE ID = 6;

Select * From Product;

ID	NAME	PRICE
1	Laptop	200000.0
2	Smartphone	80000.0
3	Tablet	40000.0
4	Smartwatch	20000.0
5	Headphones	15000.0

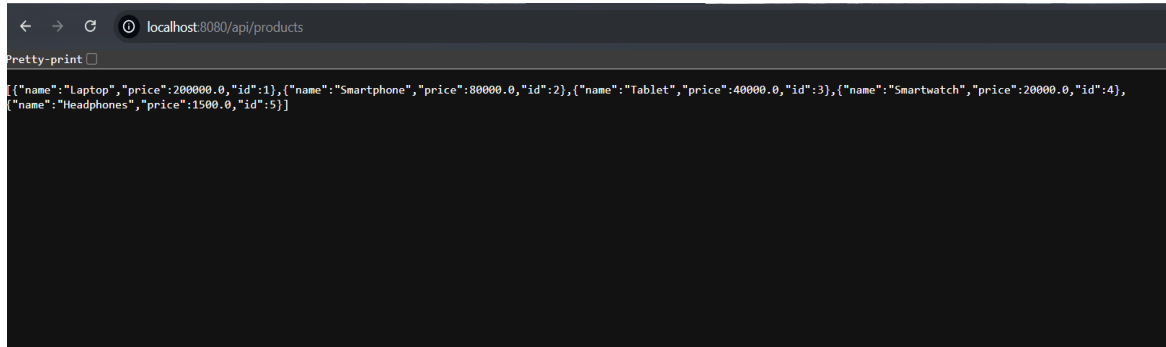
(5 rows, 1 ms)

Edit

This endpoint returns all products in JSON format, confirming that the GET API works correctly.

Open browser and visit :

URL → <http://localhost:8080/api/products>



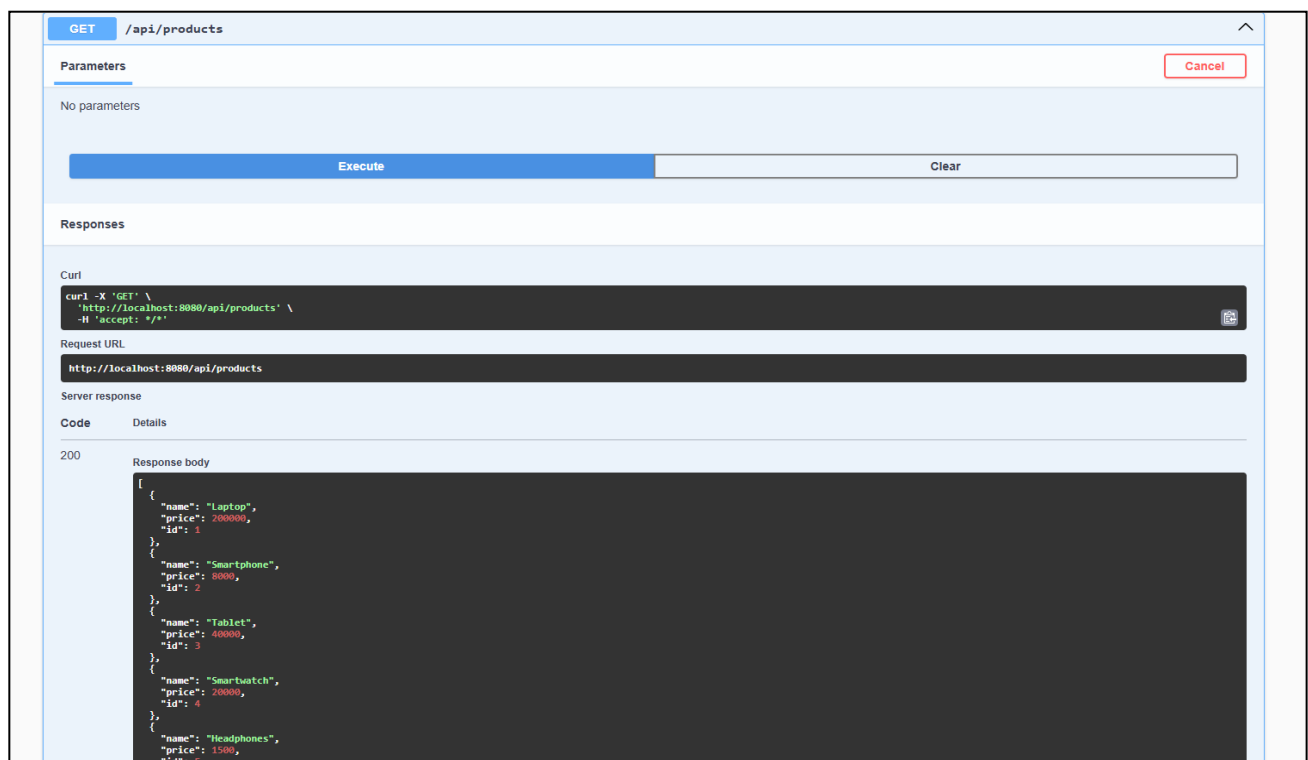
Swagger UI – API List

- Swagger UI displays all available REST endpoints and allows interactive testing

Open browser and visit :

URL → <http://localhost:8080/swagger-ui.html>

GET /products – Retrieve all products



POST /products – Create new product

POST

/api/products

Cancel

Reset

Parameters

No parameters

Request body required

application/json

```
{
  "name": "Monitor",
  "price": 75000
}
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8080/api/products' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "Monitor",
    "price": 75000
  }'
```

Request URL

http://localhost:8080/api/products

Server response

Code	Details
200	<div>Response body</div> <div><pre>{ "name": "Monitor", "price": 75000, "id": 17 }</pre></div> <div>Response headers</div> <div><pre>connection: keep-alive content-type: application/json date: Sun, 15 Feb 2026 15:02:08 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre></div>

Download

GET /products/{id} – Retrieve product by ID

The screenshot shows a REST client interface with a blue header bar. The method is **GET** and the URL is `/api/products/{id}`. A **Cancel** button is in the top right. Below the header, there's a **Parameters** section with a table:

Name	Description
id * required Integer(\$int64) (path)	1

Below the parameters is an **Execute** button and a **Clear** button. The **Responses** section is expanded, showing the following details:

- Curl:**

```
curl -X 'GET' \
  'http://localhost:8080/api/products/1' \
  -H 'accept: */*'
```
- Request URL:** `http://localhost:8080/api/products/1`
- Server response:**
 - Code:** 200
 - Response body:**

```
{
  "name": "Laptop",
  "price": 200000,
  "id": 1
}
```
 - Response headers:**

```
connection: keep-alive
content-type: application/json
date: Sun, 15 Feb 2026 15:04:19 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

DELETE /products/{id} – Delete product

The screenshot shows a REST client interface with a red header bar. The method is **DELETE** and the URL is `/api/products/{id}`. A **Cancel** button is in the top right. Below the header, there's a **Parameters** section with a table:

Name	Description
id * required Integer(\$int64) (path)	5

Below the parameters is an **Execute** button and a **Clear** button. The **Responses** section is expanded, showing the following details:

- Curl:**

```
curl -X 'DELETE' \
  'http://localhost:8080/api/products/5' \
  -H 'accept: */*'
```
- Request URL:** `http://localhost:8080/api/products/5`
- Server response:**
 - Code:** 200
 - Response headers:**

```
connection: keep-alive
content-length: 0
date: Sun, 15 Feb 2026 15:07:09 GMT
keep-alive: timeout=60
```

At the bottom, there's a **Responses** table:

Code	Description	Links
200	OK	No links