**Working with SpringDataJPA:**

Steps to implement

1. Create a project using Spring Initializer and add the dependencies (Spring Data JPA, MySQL Driver, Spring Web) and download it and import it into the Eclipse Workspace.
2. Create a subpackage “com.spring.data.jpa.csd408springdatajpa.entity” and create a class Product.java and annotate the class with “@Entity” and add data members as the columns name in the database and generate the getters and setters.

**(Note the table name in the database and the class name should be same)**

**Product.java**

**package** com.spring.data.jpa.csd24sd1234springdatajpa1.data;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.Id;

@Entity

**public** **class** Product {

@Id

**private** **int** product\_id;

**private** String product\_name;

**private** **int** price;

**public** **int** getProduct\_id() {

**return** product\_id;

}

**public** **void** setProduct\_id(**int** product\_id) {

**this**.product\_id = product\_id;

}

**public** String getProduct\_name() {

**return** product\_name;

}

**public** **void** setProduct\_name(String product\_name) {

**this**.product\_name = product\_name;

}

**public** **int** getPrice() {

**return** price;

}

**public** **void** setPrice(**int** price) {

**this**.price = price;

}

@Override

**public** String toString() {

**return** "Product [product\_id=" + product\_id + ", product\_name=" + product\_name + ", price=" + price + "]";

}

}

1. Create another subpackage “com.spring.data.jpa.csd408springdatajpa.repo” and create an interface “ProductRepository.java” that extends the “CrudRepository” interface .

**ProductRepository.java**

**package** com.spring.data.jpa.csd24sd1234springdatajpa1.data;

**import** org.springframework.data.repository.CrudRepository;

**import** com.spring.data.jpa.csd24sd1234springdatajpa1.data.Product;

**public** **interface** ProductRepository **extends** CrudRepository<Product, Integer>{

}

1. Update the Applicaton.properties files in the “src/main/resources”, add the datasources(name,url,username,password) of the database.

**Application.properties:**

spring.application.name=csd24sd1234-springdatajpa1

spring.datasource.name=mydb

spring.datasource.url=jdbc:mysql://localhost:3306/db

spring.datasource.username=root

spring.datasource.password=moupali123

1. Open the “Csd408SpringdatajpaApplicationTests” and implement a method “saveProduct()” and annotate it with “@Test” and run as junit test.

**package** com.spring.data.jpa.csd24sd1234springdatajpa1.data;

**import** java.util.Optional;

**import** org.junit.jupiter.api.Test;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.boot.test.context.SpringBootTest;

**import** org.springframework.context.ApplicationContext;

**import** com.spring.data.jpa.csd24sd1234springdatajpa1.data.Product;

**import** com.spring.data.jpa.csd24sd1234springdatajpa1.repo.ProductRepository;

@SpringBootTest

**class** Csd24sd1234Springdatajpa1ApplicationTests {

@Autowired

ApplicationContext context;

@Test

**void** saveProduct() {

ProductRepository repo = context.getBean(ProductRepository.**class**);

// To insert a record

Product product = **new** Product();

product.setProduct\_id(1004);

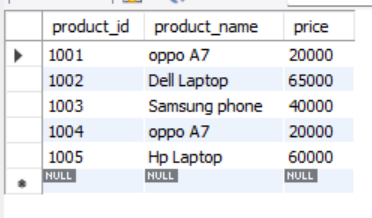
product.setProduct\_name("oppo A7");

product.setPrice(20000);

repo.save(product);

}

**OUTPUT:**



//To fetch the record

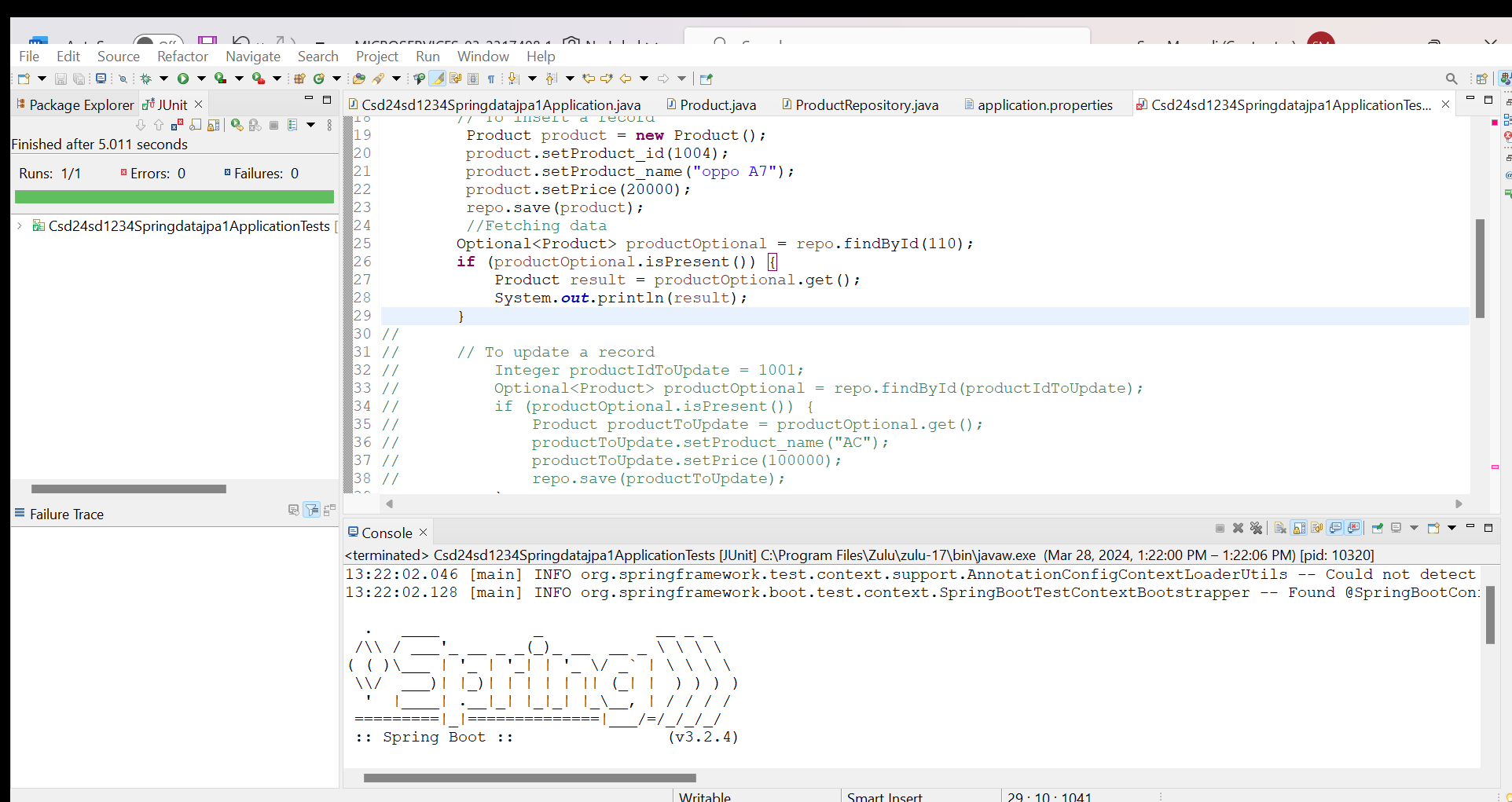
Optional<Product> productOptional = repo.findById(1002);

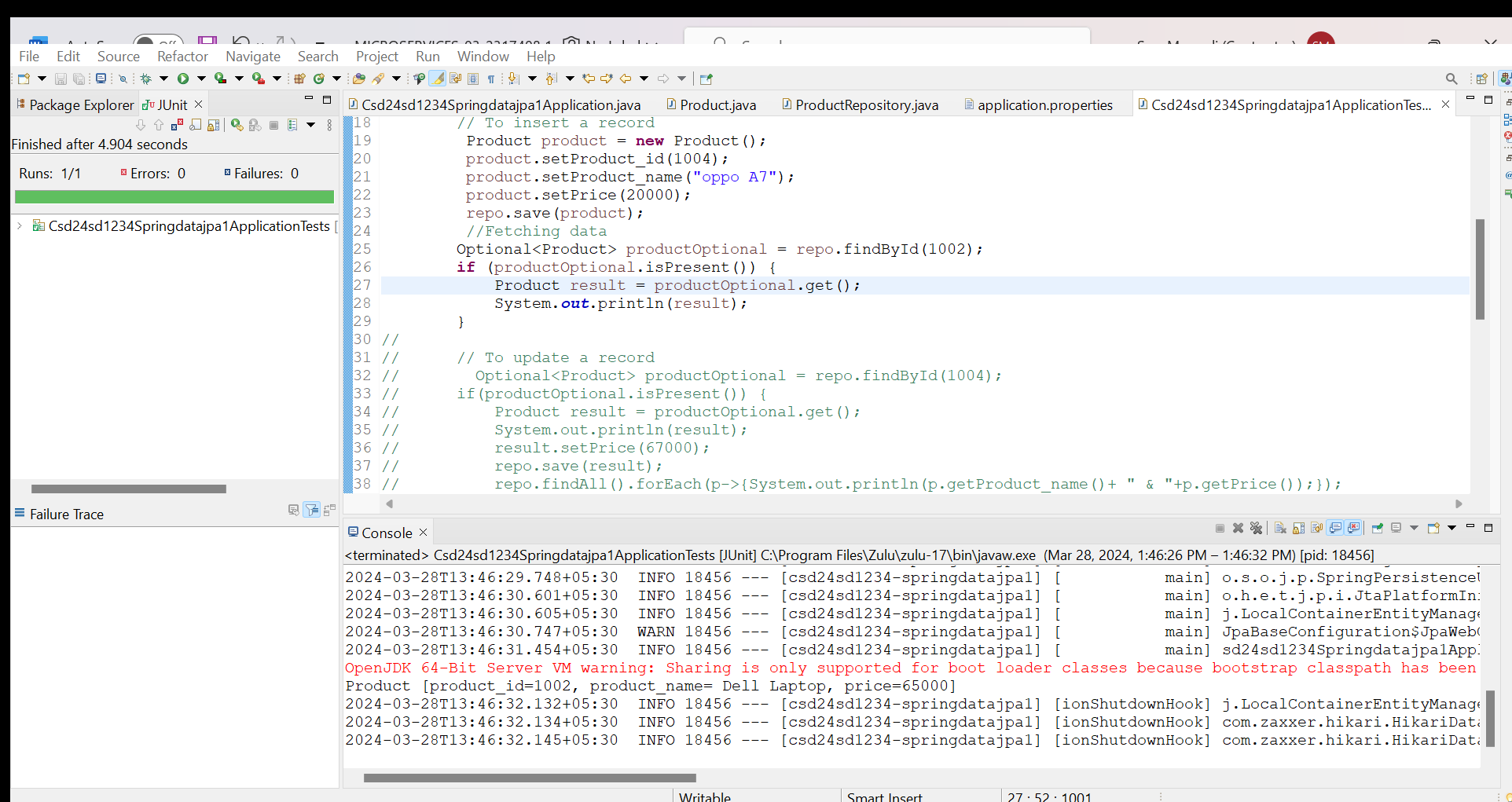
**if** (productOptional.isPresent()) {

Product result = productOptional.get();

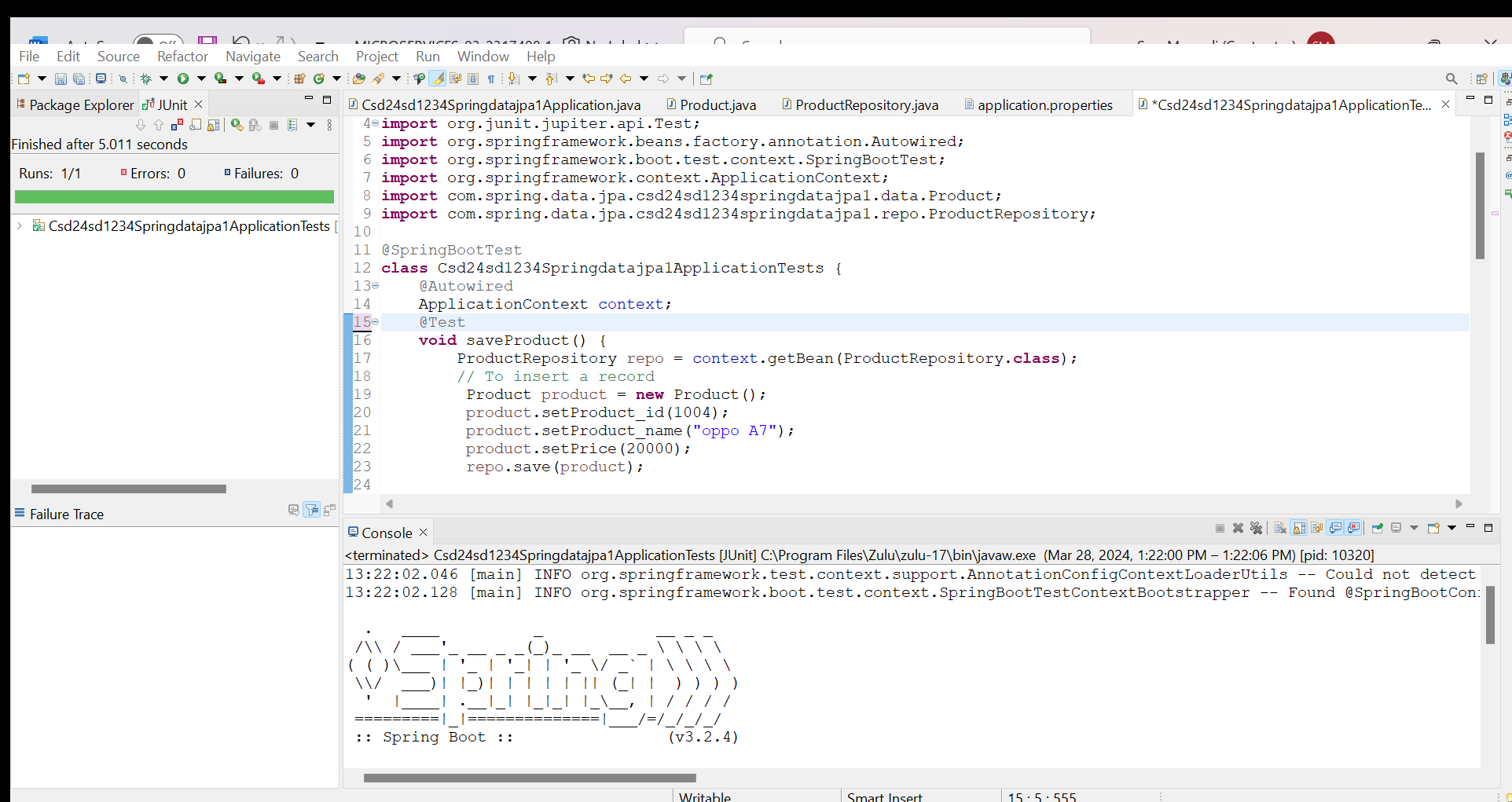
System.***out***.println(result);

}





1. To find a record by id update the . Csd408SpringdatajpaApplicationTests.java file with the below code and see the result.



7.To update a existing record update the Csd408SpringdatajpaApplicationTests.java file with the following code

// To update a record

Optional<Product> productOptional = repo.findById(1004);

**if**(productOptional.isPresent()) {

Product result = productOptional.get();

System.***out***.println(result);

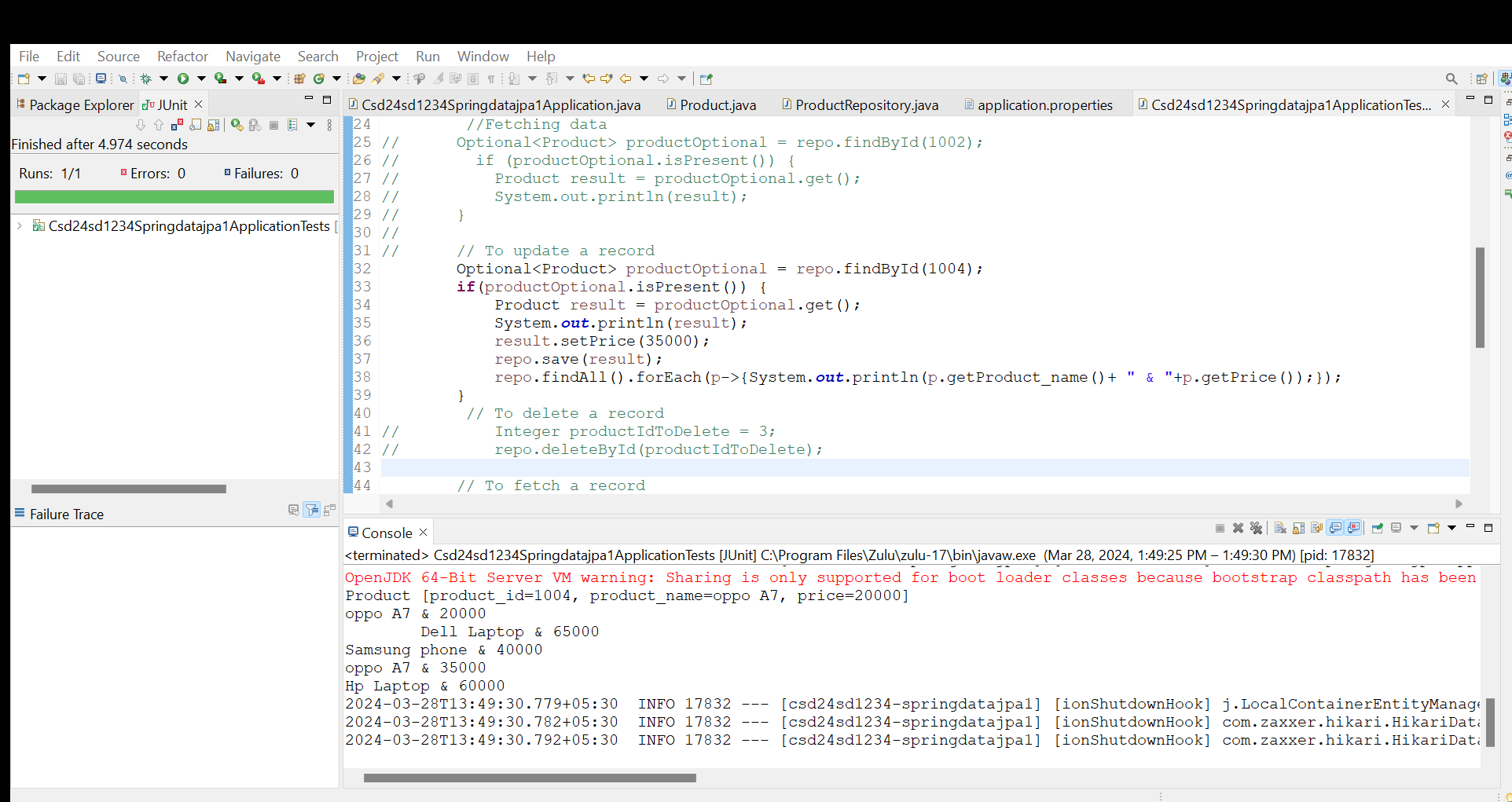
result.setPrice(35000);

repo.save(result);

repo.findAll().forEach(p->{System.***out***.println(p.getProduct\_name()+ " & "+p.getPrice());});

}

OUTPUT:-



1. To delete a record update the following code in the Csd408SpringdatajpaApplicationTests.java file and run as junit test.

//To delete a record

Optional<Product> productOptional = repo.findById(1001);

**if**(productOptional.isPresent()) {

Product result = productOptional.get();

repo.delete(result);

repo.findAll().forEach(p->{System.***out***.println(p.getProduct\_name()+ " & "+p.getPrice());});

}

OUTPUT:-

A screenshot of a computer

Description automatically generated

1. Create a subpackage in the “src/main/java” “com.spring.data.jpa.csd408springdatajpa.controller” and ceate a class “ProductsController” and annotate it with “@RestController”.
2. Create object “ProductRepository repository” and annotate it with “@Autowired”.
3. Implement the “GET,PUT,POST,DELETE” methods as shown below and run the Application.java file and check the respective output in the Postman tool.
4. ProductController.java

**package** com.spring.data.jpa.csd24sd1234springdatajpa1.data;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.DeleteMapping;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.PutMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.spring.data.jpa.csd24sd1234springdatajpa1.data.Product;

**import** com.spring.data.jpa.csd24sd1234springdatajpa1.repo.ProductRepository;

@RestController

@RequestMapping("/product")

**public** **class** ProductController {

@Autowired

ProductRepository repository;

@GetMapping

**public** Iterable<Product> getProducts() {

**return** repository.findAll();

}

@PostMapping

**public** Product createRecord(@RequestBody Product product) {

**return** repository.save(product);

}

//update a record

@PutMapping(value="/{id}")

**public** Product updateRecord(@PathVariable("id") **int** id , @RequestBody Product product) {

product.setProduct\_id(id);

product.setProduct\_name(product.getProduct\_name());

product.setPrice(product.getPrice());

**return** repository.save(product);

}

//delete a record

@DeleteMapping(value="/{id}")

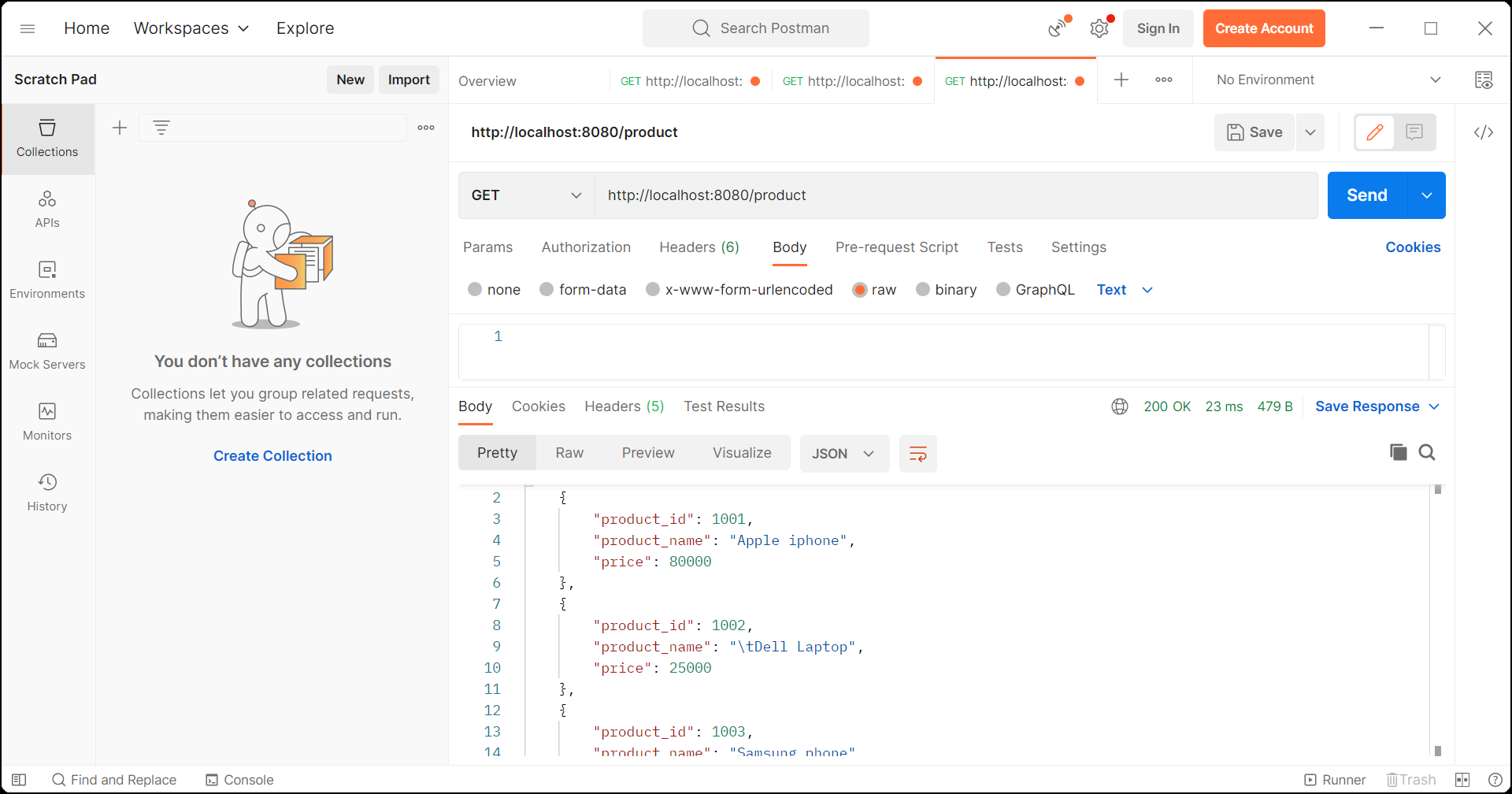
**public** **void** deleteRecord(@PathVariable("id") **int** id) {

repository.deleteById(id);

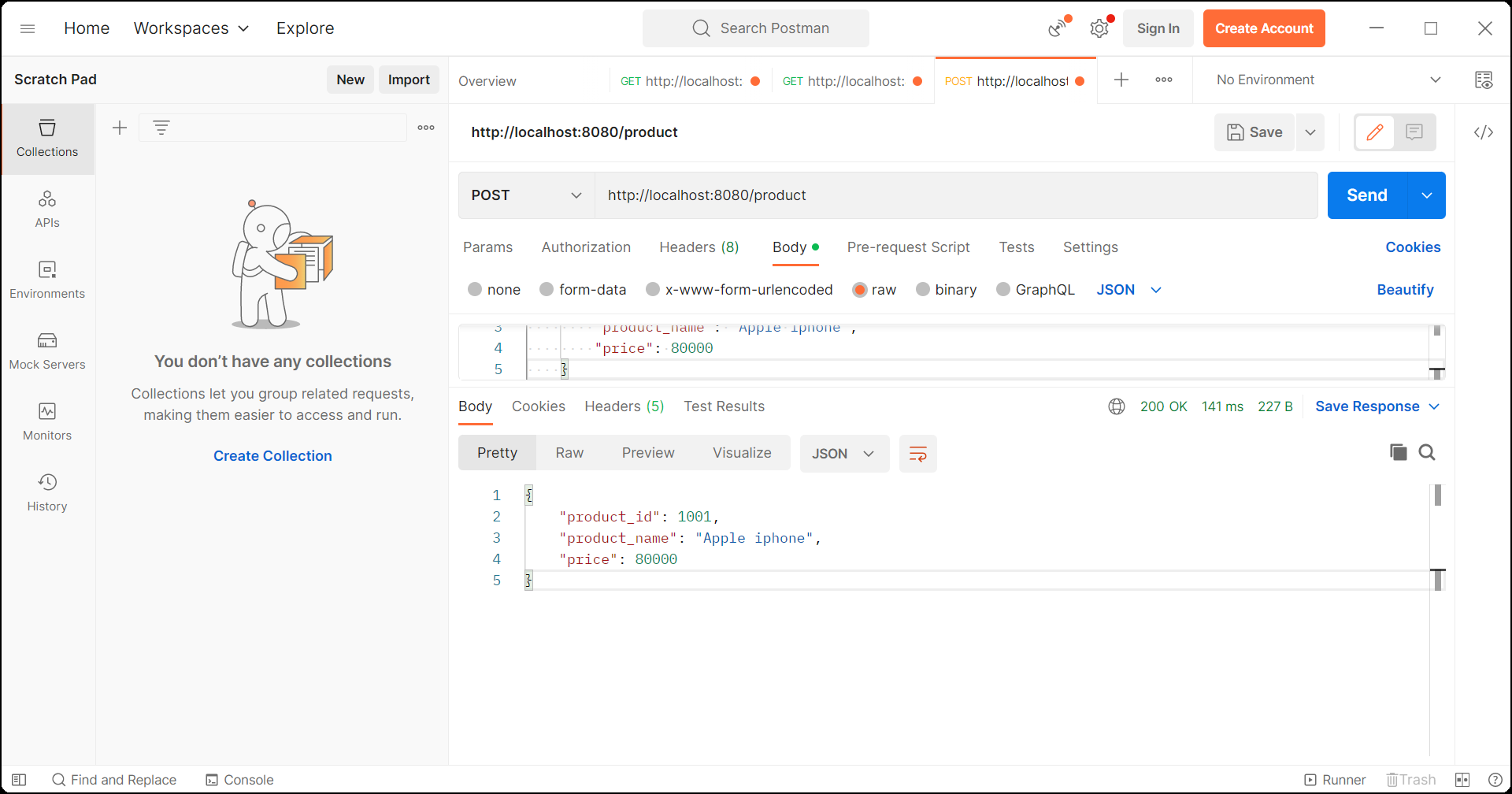
System.***out***.println("the deleted row is "+id);

}}

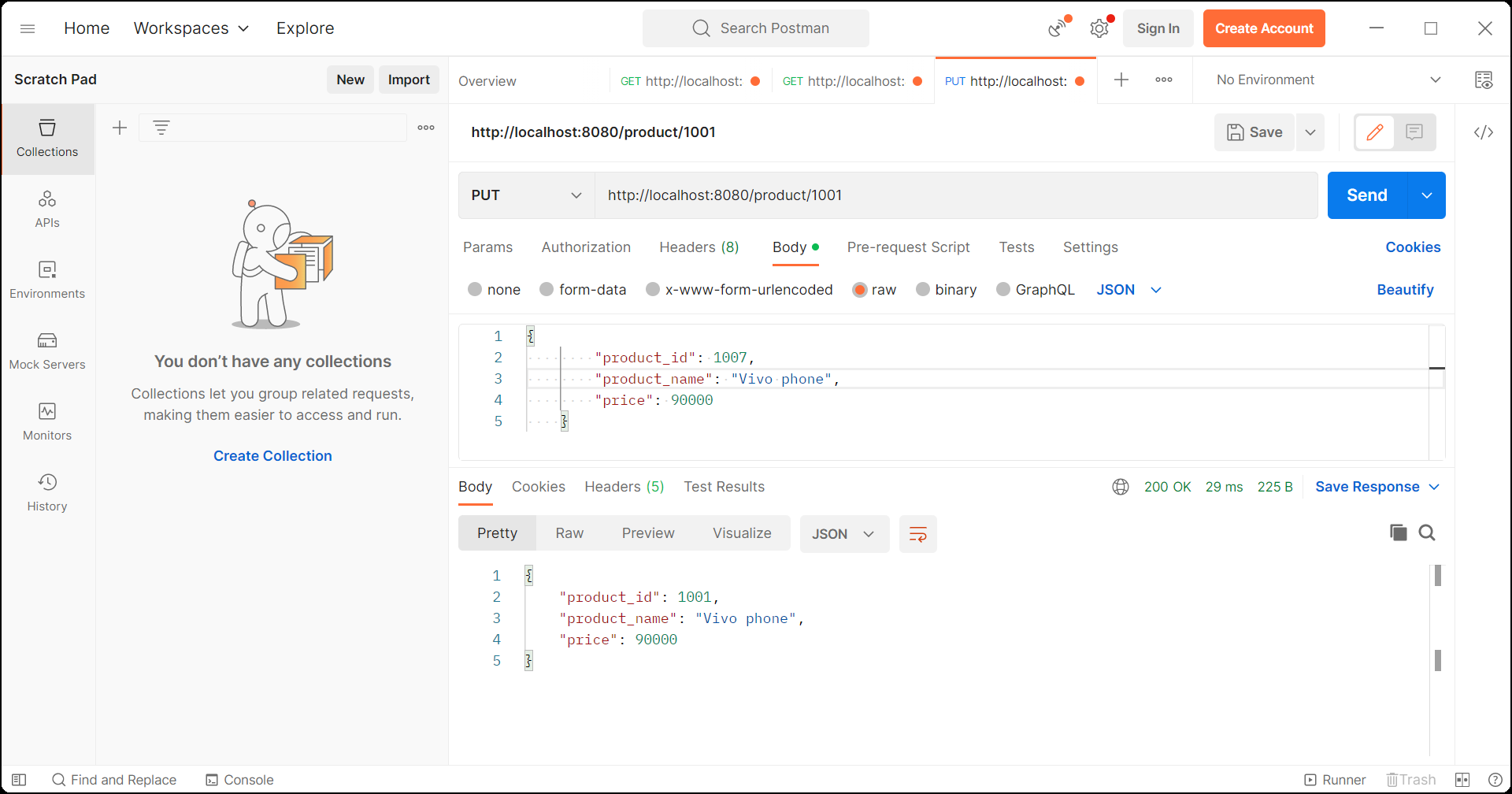
GET:



POST:



PUT:



DELETE:

