LAPORAN HASIL PRAKTIKUM 6

PEMROGRAMAN BERORIENTASI OBJEK LANJUT

“Spring Boot-2”

Dosen Pengampu :

Sri Hartati Wijono, M.Kom.

Oleh

Nama : Maria Gresia Plena Br Purba

NIM : 235314094

Kelas : DP

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS SANATA DHARMA

YOGYAKARTA

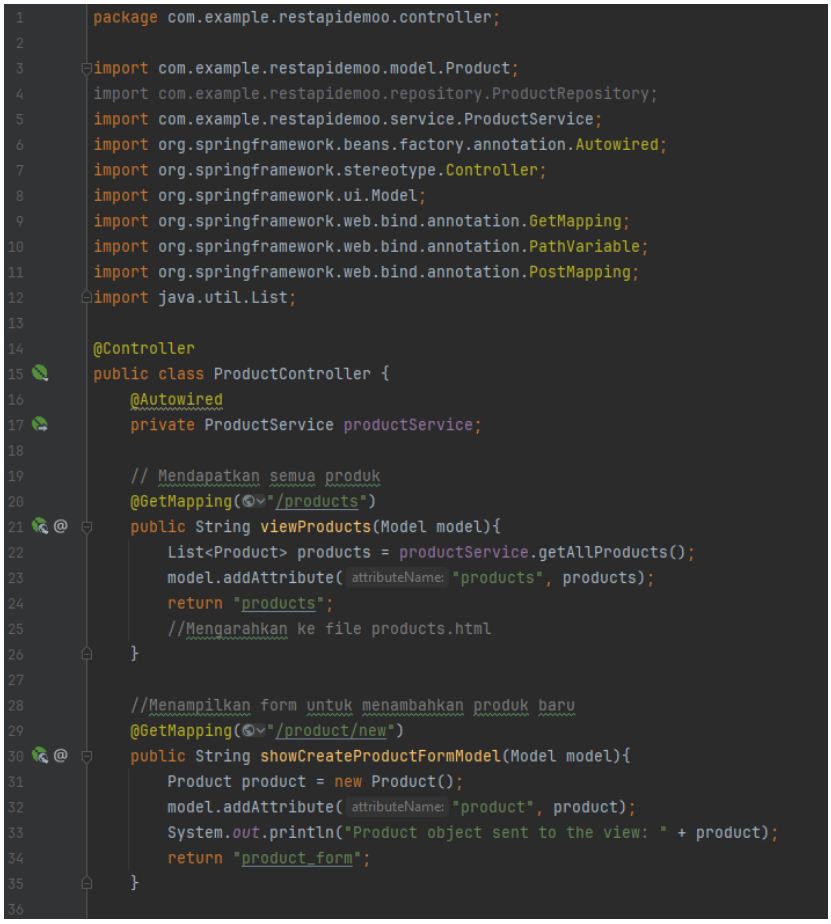
2024

A. TUJUAN

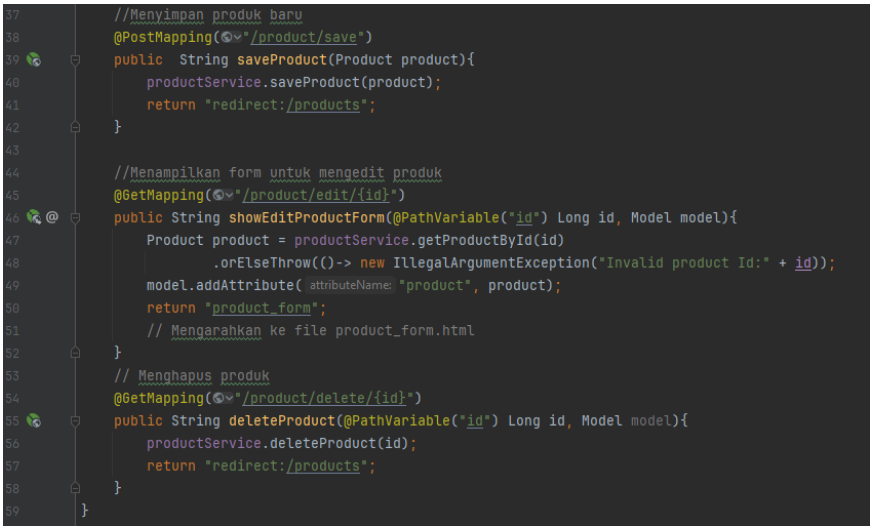
1. Memahami cara membuat aplikasi Spring Boot REST API.
2. Memahami konsep CRUD (Create, Read, Update, Delete).
3. Menggunakan H2 Database untuk menyimpan data produk.
4. Menggunakan Spring Data JPA untuk operasi CRUD.

B. PRAKTIKUM

1. Product Controller



```java
package com.example.restapidemoo.controller;

import com.example.restapidemoo.model.Product;
import com.example.restapidemoo.repository.ProductRepository;
import com.example.restapidemoo.service.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import java.util.List;

@Controller
public class ProductController {
    @Autowired
    private ProductService productService;

    // Mendapatkan semua produk
    @GetMapping("/products")
    public String viewProducts(Model model){
        List<Product> products = productService.getAllProducts();
        model.addAttribute( attributeName: "products", products);
        return "products";
        //Mengarahkan ke file products.html
    }

    //Menampilkan form untuk menambahkan produk baru
    @GetMapping("/product/new")
    public String showCreateProductFormModel(Model model){
        Product product = new Product();
        model.addAttribute( attributeName: "product", product);
        System.out.println("Product object sent to the view: " + product);
        return "product_form";
    }

}
```



```
    //Menyimpan produk baru
    @PostMapping("/product/save")
    public  String saveProduct(Product product){
        productService.saveProduct(product);
        return "redirect:/products";
    }

    //Menampilkan form untuk mengedit produk
    @GetMapping("/product/edit/{id}")
    public String showEditProductForm(@PathVariable("id") Long id, Model model){
        Product product = productService.getProductById(id)
                .orElseThrow(()-> new IllegalArgumentException("Invalid product Id:" + id));
        model.addAttribute( attributeName: "product", product);
        return "product_form";
        // Mengarahkan ke file product_form.html
    }
    // Menghapus produk
    @GetMapping("/product/delete/{id}")
    public String deleteProduct(@PathVariable("id") Long id, Model model){
        productService.deleteProduct(id);
        return "redirect:/products";
    }
}
```

Penjelasan :

Method viewProducts berfungsi untuk mendapatkan semua produk dari productService menggunakan method getAllPorducts() yang akan disimpan ke dalam variabel “products” dan menambahkannya sebagai attribute model dengan menggunakan addAttribute() dan disimpan dengan nama “products”. Kemudian, mengembalikan tampilann yang diarahkan ke file products.html. Pada method showCreateProductFormModel() membuat objek yang bernama “product” dan menambahkan attribute dengan menggunakan addAttribute() pada model. Kemudian, mencetak pesan yang menunjukkan produk apa yang dikirim dan mengembalikan tampilan yang diarahkan ke file product_form.html. Method saveProduct() berfungsi untuk menyimpan produk baru ke productService dengan memanggil method saveProduct(). Kemudian, halaman akan dialihkan ke /products. Method showEditProductForm() berfungsi untuk menampilkan form untuk mengedit produk. Mendapatkan id dengan menggunakan getProductById dari productService yang akan disimpan ke dalam “product”. Jika tidak ditemukan, maka akan menampilkan pesan yang menunjukkan bahwa id yang dicari tidak valid. Kemudian, menambahkan attribute baru dengan menggunakan addAttribute pada model yang akan diarahkan ke file products_form.html. Method productService() berfungsi untuk menghapus produk berdasarkan id dengan menggunakan method deleteProduct() pada productService. Kemudian, halaman akan dialihkan ke /products.

2. Product

```java
package com.example.restapidemoo.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

14 usages
@Entity //memberi tahu JPA bahwa kelas ini adalah entitas yang akan dipetakan ke tabel dalam database
public class Product {
    @Id // Menandakan bahwa id adalah primary key
    @GeneratedValue(strategy = GenerationType.IDENTITY) //Menetapkan strategi untuk menghasilkan nilai ID secara otomatis
    private Long id;
    6 usages
    private String name;
    6 usages
    private double price;

    5 usages
    private String description;

    public Product() {
        this.name = name;
        this.description = description;
        this.price = price;
    }

    no usages
    public Product(Long id, String name, double price) {
        this.name = name;
        this.id = id;
        this.price = price;
    }
    no usages
    public Product(Long id, String name, String description, double price) {
        this.name = name;
        this.id = id;
        this.description = description;
        this.price = price;
    }
```

```java
    public Long getId() { return id; }

    no usages
    public void setId(Long id) { this.id = id; }

    no usages
    public String getName() { return name; }

    no usages
    public void setName(String name) { this.name = name; }

    no usages
    public double getPrice() { return price; }

    no usages
    public void setPrice(double price) { this.price = price; }

    no usages
    public String getDescription() { return description; }

    no usages
    public void setDescription(String description) { this.description = description; }
}
```

Penjelasan :

Mendeklarasikan atribut bernama id bertipe Long, name dan description bertipe String, dan price bertipe double yang masing-masing bersifat private. Terdapat 3 constructor yang di mana satu default, satu dengan tiga parameter, dan satu dengan empat parameter. Menggunakan getter setter yang berfungsi untuk mendapatkan nilai dan mengubah nilai dari atribut.

3. ProductRepository

```java
package com.example.restapidemoo.repository;

import com.example.restapidemoo.model.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

4 usages
@Repository
public interface ProductRepository extends JpaRepository<Product, Long>{

}
```

Penjelasan :

Interface ini mewarisi JpaRepository yang memiliki method method yang dapat digunakan, seperti findAll(), save(), findById(), dan deleteById().

4. ProductService

```java
package com.example.restapidemoo.service;

import com.example.restapidemoo.model.Product;
import com.example.restapidemoo.repository.ProductRepository;
import com.example.restapidemoo.model.Product;
import com.example.restapidemoo.repository.ProductRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
import java.util.Optional;

2 usages
@Service
public class ProductService {
    @Autowired
    private ProductRepository productRepository;

    1 usage
    public List<Product> getAllProducts() { return productRepository.findAll(); }

    1 usage
    public Optional<Product> getProductById(Long id) { return productRepository.findById(id); }

    1 usage
    public Product saveProduct(Product product) { return productRepository.save(product); }

    1 usage
    public void deleteProduct(Long id) { productRepository.deleteById(id); }
}
```

Penjelasan :

Method getAllProducts berfungsi untuk mendapatkan semua produk dengan memanggil method findAll() pada productRepository. Method getProductById berfungsi untuk mendapatkan data berdasarkan id dengan menggunakan method findById() pada productRepository. Method saveProduct() akan berfungsi untuk menyimpan produk baru dengan menggunakan save() pada productRepository. Kemudian, method deleteProduct() berfungsi untuk menghapus data dengan menggunakan deleteById() pada productRepository.

5. RestApiDemooApplication

```java
package com.example.restapidemoo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class RestApiDemooApplication {

    public static void main(String[] args) {
        SpringApplication.run(RestApiDemooApplication.class, args);
    }
}
```

Penjelasan :

Pada class ini terdapat method main() yang akan memanggil SpringApplication.run() untuk menjalankan keseluruhan aplikasi.

6. product_form.html

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Product Form</title>
</head>
<body>
<form th:action="@{/product/save}" method="post">
    <input type="hidden"
           th:field="*{product.id}">
    <div>
        <label>Name:</label>
        <input type="text"
               th:field="*{product.name}" placeholder="Product Name" required>
    </div>
    <div>
        <label>Description:</label>
        <input type="text"
               th:field="*{product.description}" placeholder="Product Description" required>
    </div>
    <div>
        <label>Price:</label>
        <input type="number"
               th:field="*{product.price}" placeholder="Product Price" step="0.01" required>
    </div>
    <div>
        <button type="submit">Save</button>
    </div>
</form>
</body>
</html>
```

Penjelasan :

Form ini menggunakan atribut th:field dari Thymeleaf. Form ini berfungsi untuk meminta input pada product name, product description, dan product price kepada user. Inputan tersebut akan disimpan dengan menggunakn method “post” dengan type “hidden” yang memiliki field “product.id”. Product name dan product description memiliki type “text”, sedangkan price memiliki type “number”. Terdapat button dengan nama “submit” untuk mengirim dan menyimpan data.

7. products.html

```
<head>
    <title>Product List</title>
</head>
<body>
<h1>Product List</h1>
<a href="/product/new">Add New Product</a>
<table border="1">
    <thead>
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Description</th>
        <th>Price</th>
        <th>Actions</th>
    </tr>
    </thead>
    <tbody>
    <tr th:each="product : ${products}">
        <td th:text="${product.id}">ID</td>
        <td th:text="${product.name}">Product Name</td>
        <td th:text="${product.description}">Product Description</td>
        <td th:text="${product.price}">Product Price</td>
        <td>
            <a th:href="@{/product/edit/{id}(id=${product.id})}">Edit</a>
            <a th:href="@{/product/delete/{id}(id=${product.id})}"
               onclick="return confirm('Are you sure you want to delete this product?');">Delete</a>
        </td>
    </tr>
    </tbody>
</table>
</body>
</html>
```

Penjelasan :

From ini menggunakan th:each dari Thymeleaf. From ini berfungsi untuk menampilkan daftar produk. Terdiri dari 5 kolom yang bernama “ID”, “Name”, “Description”, “Price”, dan “Actions”. Membuat pilihan untuk untuk mengedit produk berdasarkan id nya dengan menggunakan “@{/product/edit/{id}(id=${product.id})}” yang kemudian diberikan nama “Edit”. Kemudian, terdapat pilihan untuk menghapus dengan menggunakan "@{/product/delete/{id}(id=${product.id})}" yang memberikan teks konfirmasi penghapusan untuk memastikan ingin menghapus data atau tidak dan diberikan nama “Delete”.

8. application.properties

```
spring.application.name=rest api demoo

# Konfigurasi Database H2
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
#spring.h2.console.enabled=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

# Mengaktifkan Konsol H2
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console

# mengatur frontend
spring.thymeleaf.prefix=classpath:/templates/
spring.thymeleaf.suffix=.html
server.port=8081
spring.thymeleaf.cache=false

```

Penjelasan :

Database h2 disetting dengan menggunakan “jdbc:h2:mem:testdb” dengan driverClassName “org.h2.Driver”. Kemudian, memberikan data username, yaitu “sa” tanpa password. Mengaktifkan konsol h2 dengan menggunakan “spring.h2.console.enabled=true” dan memiliki path “path=/h2-console”. Lalu, mengatur frontend dengan menggunakan “spring.thymeleaf” yang di mana “.prefix=classpath:/templates/” untuk dan “.suffix=.html” untuk , menggunakan port 8081 dan “spring.thymeleaf.cache=false” untuk

9. Pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.3.4</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>org.example</groupId>
    <artifactId>rest-api-demoo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>rest api demoo</name>
    <description>rest api demoo</description>
    <url/>
    <licenses>
        <license/>
    </licenses>
    <developers>
        <developer/>
    </developers>
    <scm>
        <connection/>
        <developerConnection/>
        <tag/>
        <url/>
    </scm>
    <properties>
        <java.version>17</java.version>
        <start-class>com.example.restapidemoo.RestApiDemooApplication</start-class>
    </properties>

    <dependencies> Edit Starters...
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
```

```xml
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

Penjelasan :

Memiliki groupId org.example dan artifactId rest-api-demoo, Menggunakan java dengan versi 17. Pada dependencied terdapat spring-boot-starter-data-jpa yang berfungsi untuk mengimplementasikan JPA repositories, spring-boot-starter-thymeleaf yang berfungsi untuk mengonfigurasi template Thymeleaf secara otomatis dalam aplikasi web, spring-boot-starter-web yang berfungsi untuk menyediakan aplikasi web Spring Boot yang siap produksi, dan spring-boot-starter-test yang berfungsi untuk menyediakan alat untuk menguji. Pada build plugind terdapat spring-boot-maven-plugin yang berfungsi untuk menyediakan dukungan Spring Boot di Apache Maven

10. Output