

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

```
1 package com.example.restapidemoo.controller;
2
3 import com.example.restapidemoo.model.Product;
4 import com.example.restapidemoo.repository.ProductRepository;
5 import com.example.restapidemoo.service.ProductService;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Controller;
8 import org.springframework.ui.Model;
9 import org.springframework.web.bind.annotation.GetMapping;
10 import org.springframework.web.bind.annotation.PathVariable;
11 import org.springframework.web.bind.annotation.PostMapping;
12 import java.util.List;
13
14 @Controller
15 public class ProductController {
16     @Autowired
17     private ProductService productService;
18
19     // Mendapatkan semua produk
20     @GetMapping("/products")
21     public String viewProducts(Model model){
22         List<Product> products = productService.getAllProducts();
23         model.addAttribute("products", products);
24         return "products";
25         //Mengarahkan ke file products.html
26     }
27
28     //Menampilkan form untuk menambahkan produk baru
29     @GetMapping("/product/new")
30     public String showCreateProductFormModel(Model model){
31         Product product = new Product();
32         model.addAttribute("product", product);
33         System.out.println("Product object sent to the view: " + product);
34         return "product_form";
35     }
36 }
```

```

37 //Menyimpan produk baru
38 @PostMapping("/product/save")
39 public String saveProduct(Product product){
40     productService.saveProduct(product);
41     return "redirect:/products";
42 }
43
44 //Menampilkan form untuk mengedit produk
45 @GetMapping("/product/edit/{id}")
46 public String showEditProductForm(@PathVariable("id") Long id, Model model){
47     Product product = productService.getProductById(id)
48         .orElseThrow(() -> new IllegalArgumentException("Invalid product Id: " + id));
49     model.addAttribute("product", product);
50     return "product_form";
51     // Mengarahkan ke file product_form.html
52 }
53
54 // Menghapus produk
55 @GetMapping("/product/delete/{id}")
56 public String deleteProduct(@PathVariable("id") Long id, Model model){
57     productService.deleteProduct(id);
58     return "redirect:/products";
59 }

```

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

```
1 package com.example.restapidemo.model;
2
3 import jakarta.persistence.Entity;
4 import jakarta.persistence.GeneratedValue;
5 import jakarta.persistence.GenerationType;
6 import jakarta.persistence.Id;
7
8 //14 usages
9 @Entity //memberi tahu JPA bahwa kelas ini adalah entitas yang akan dipetakan ke tabel dalam database
10 public class Product {
11     @Id // Menandakan bahwa id adalah primary key
12     @GeneratedValue(strategy = GenerationType.IDENTITY) //Menetapkan strategi untuk menghasilkan nilai ID secara otomatis
13     private Long id;
14     //6 usages
15     private String name;
16     //6 usages
17     private double price;
18
19     //5 usages
20     private String description;
21
22     public Product() {
23         this.name = name;
24         this.description = description;
25         this.price = price;
26     }
27
28     //no usages
29     public Product(Long id, String name, double price) {
30         this.name = name;
31         this.id = id;
32         this.price = price;
33     }
34
35     //no usages
36     public Product(Long id, String name, String description, double price) {
37         this.name = name;
38         this.id = id;
39         this.description = description;
40         this.price = price;
41     }
42
43     //no usages
44     public Long getId() { return id; }
45
46     //no usages
47     public void setId(Long id) { this.id = id; }
48
49     //no usages
50     public String getName() { return name; }
51
52     //no usages
53     public void setName(String name) { this.name = name; }
54
55     //no usages
56     public double getPrice() { return price; }
57
58     //no usages
59     public void setPrice(double price) { this.price = price; }
60
61     //no usages
62     public String getDescription() { return description; }
63
64     //no usages
65     public void setDescription(String description) { this.description = description; }
66 }
```

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

```
1 package com.example.restapidemoo.repository;
2
3 import com.example.restapidemoo.model.Product;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository
8 public interface ProductRepository extends JpaRepository<Product, Long>{
9
10 }
```

Penjelasan :

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

4. ProductService

```
1 package com.example.restapidemoo.service;
2
3 import com.example.restapidemoo.model.Product;
4 import com.example.restapidemoo.repository.ProductRepository;
5 import com.example.restapidemoo.model.Product;
6 import com.example.restapidemoo.repository.ProductRepository;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.stereotype.Service;
9 import java.util.List;
10 import java.util.Optional;
11
12 @Service
13 public class ProductService {
14     @Autowired
15     private ProductRepository productRepository;
16
17     public List<Product> getAllProducts() { return productRepository.findAll(); }
18
19     public Optional<Product> getProductById(Long id) { return productRepository.findById(id); }
20
21     public Product saveProduct(Product product) { return productRepository.save(product); }
22
23     public void deleteProduct(Long id) { productRepository.deleteById(id); }
24 }
```

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

```
1 package com.example.restapidemoo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class RestApiDemooApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(RestApiDemooApplication.class, args);
11     }
12 }
```

Penjelasan :

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

6. product_form.html

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <title>Product Form</title>
5 </head>
6 <body>
7 <form th:action="@{/product/save}" method="post">
8 <input type="hidden"
9 th:field="*{product.id}">
10 <div>
11 <label>Name:</label>
12 <input type="text"
13 th:field="*{product.name}" placeholder="Product Name" required>
14 </div>
15 <div>
16 <label>Description:</label>
17 <input type="text"
18 th:field="*{product.description}" placeholder="Product Description" required>
19 </div>
20 <div>
21 <label>Price:</label>
22 <input type="number"
23 th:field="*{product.price}" placeholder="Product Price" step="0.01" required>
24 </div>
25 <div>
26 <button type="submit">Save</button>
27 </div>
28 </form>
29 </body>
30 </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 menggunakan 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

```
3 <head>
4   <title>Product List</title>
5 </head>
6 <body>
7   <h1>Product List</h1>
8   <a href="/product/new">Add New Product</a>
9   <table border="1">
10    <thead>
11      <tr>
12        <th>ID</th>
13        <th>Name</th>
14        <th>Description</th>
15        <th>Price</th>
16        <th>Actions</th>
17      </tr>
18    </thead>
19    <tbody>
20      <tr th:each="product : ${products}">
21        <td th:text="${product.id}">ID</td>
22        <td th:text="${product.name}">Product Name</td>
23        <td th:text="${product.description}">Product Description</td>
24        <td th:text="${product.price}">Product Price</td>
25        <td>
26          <a th:href="@{/product/edit/{id}(id=${product.id})}">Edit</a>
27          <a th:href="@{/product/delete/{id}(id=${product.id})}"
28             onclick="return confirm('Are you sure you want to delete this product?');">Delete</a>
29        </td>
30      </tr>
31    </tbody>
32  </table>
33 </body>
34 </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

```
1  spring.application.name=rest api demoo
2
3  # Konfigurasi Database H2
4  spring.datasource.url=jdbc:h2:mem:testdb
5  spring.datasource.driverClassName=org.h2.Driver
6  spring.datasource.username=sa
7  spring.datasource.password=
8  #spring.h2.console.enabled=true
9  spring.jpa.hibernate.ddl-auto=update
10 spring.jpa.show-sql=true
11
12 # Mengaktifkan Konsol H2
13 spring.h2.console.enabled=true
14 spring.h2.console.path=/h2-console
15
16 # mengatur frontend
17 spring.thymeleaf.prefix=classpath:/templates/
18 spring.thymeleaf.suffix=.html
19 server.port=8081
20 spring.thymeleaf.cache=false
21
```

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

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
5      <modelVersion>4.0.0</modelVersion>
6      <parent>
7          <groupId>org.springframework.boot</groupId>
8          <artifactId>spring-boot-starter-parent</artifactId>
9          <version>3.3.4</version>
10         <relativePath/> <!-- Lookup parent from repository -->
11     </parent>
12     <groupId>org.example</groupId>
13     <artifactId>rest-api-demo0</artifactId>
14     <version>0.0.1-SNAPSHOT</version>
15     <name>rest api demo0</name>
16     <description>rest api demo0</description>
17     <url/>
18     <licenses>
19         <license/>
20     </licenses>
21     <developers>
22         <developer/>
23     </developers>
24     <scm>
25         <connection/>
26         <developerConnection/>
27         <tag/>
28         <url/>
29     </scm>
30     <properties>
31         <java.version>17</java.version>
32         <start-class>com.example.restapidemo0.RestApiDemo0Application</start-class>
33     </properties>
34
35     <dependencies> Edit Starters...
36     <dependency>
37         <groupId>org.springframework.boot</groupId>
38         <artifactId>spring-boot-starter-data-jpa</artifactId>
39     </dependency>
```

```
40
41     <dependency>
42         <groupId>org.springframework.boot</groupId>
43         <artifactId>spring-boot-starter-thymeleaf</artifactId>
44     </dependency>
45
46     <dependency>
47         <groupId>org.springframework.boot</groupId>
48         <artifactId>spring-boot-starter-web</artifactId>
49     </dependency>
50
51     <dependency>
52         <groupId>com.h2database</groupId>
53         <artifactId>h2</artifactId>
54         <scope>runtime</scope>
55     </dependency>
56
57     <dependency>
58         <groupId>org.springframework.boot</groupId>
59         <artifactId>spring-boot-starter-test</artifactId>
60         <scope>test</scope>
61     </dependency>
62 </dependencies>
63
64 <build>
65     <plugins>
66         <plugin>
67             <groupId>org.springframework.boot</groupId>
68             <artifactId>spring-boot-maven-plugin</artifactId>
69         </plugin>
70     </plugins>
71 </build>
72 </project>
```

Penjelasan :

Memiliki groupId org.example dan artifactId rest-api-demo, Menggunakan java dengan versi 17. Pada dependencies 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 plugin terdapat spring-boot-maven-plugin yang berfungsi untuk menyediakan dukungan Spring Boot di Apache Maven

10. Output

```

  ____  _
 / ___|| | | |
 \___ \| |_| |
  ___) | __| |
 |____|_|_|_|

:: Spring Boot ::      (v3.3.4)

2024-10-16T10:16:17.180+07:00 INFO 6040 --- [rest api demo] [main] c.e.r.RestApiDemoApplication : Starting RestApiDemoApplication using Java 17.0.8 with PID 6040 (D:\Kampus\semester 3\pbol\rest ap
2024-10-16T10:16:17.163+07:00 INFO 6040 --- [rest api demo] [main] c.e.r.RestApiDemoApplication : No active profile set, falling back to 1 default profile: "default"
2024-10-16T10:16:17.796+07:00 INFO 6040 --- [rest api demo] [main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2024-10-16T10:16:17.828+07:00 INFO 6040 --- [rest api demo] [main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 25 ms. Found 1 JPA repository interface.
2024-10-16T10:16:18.123+07:00 INFO 6040 --- [rest api demo] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8081 (http)
2024-10-16T10:16:18.136+07:00 INFO 6040 --- [rest api demo] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-10-16T10:16:18.137+07:00 INFO 6040 --- [rest api demo] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.30]
2024-10-16T10:16:18.194+07:00 INFO 6040 --- [rest api demo] [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2024-10-16T10:16:18.194+07:00 INFO 6040 --- [rest api demo] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 942 ms
2024-10-16T10:16:18.220+07:00 INFO 6040 --- [rest api demo] [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-10-16T10:16:18.350+07:00 INFO 6040 --- [rest api demo] [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:testdb user=SA
```

