

LAPORAN AKHIR PRAKTIKUM

Mata Praktikum : RPL 2

Kelas : 4IA06

Praktikum ke- : 6

Tanggal : 19 November 2024

Materi
Hibernate : Implementasi AOP dan dependency injection pada project Spring dan

NPM : 50421535

Nama : Fizri Rosdiansyah

Ketua Asisten : Gilbert Jefferson Faozato Mendrofa

Paraf Asisten :

Nama Asisten :

Jumlah Lembar : 11 Lembar

LABORATORIUM TEKNIK INFORMATIKA

UNIVERSITAS GUNADARMA

2024

Rekayasa Perangkat Lunak 2

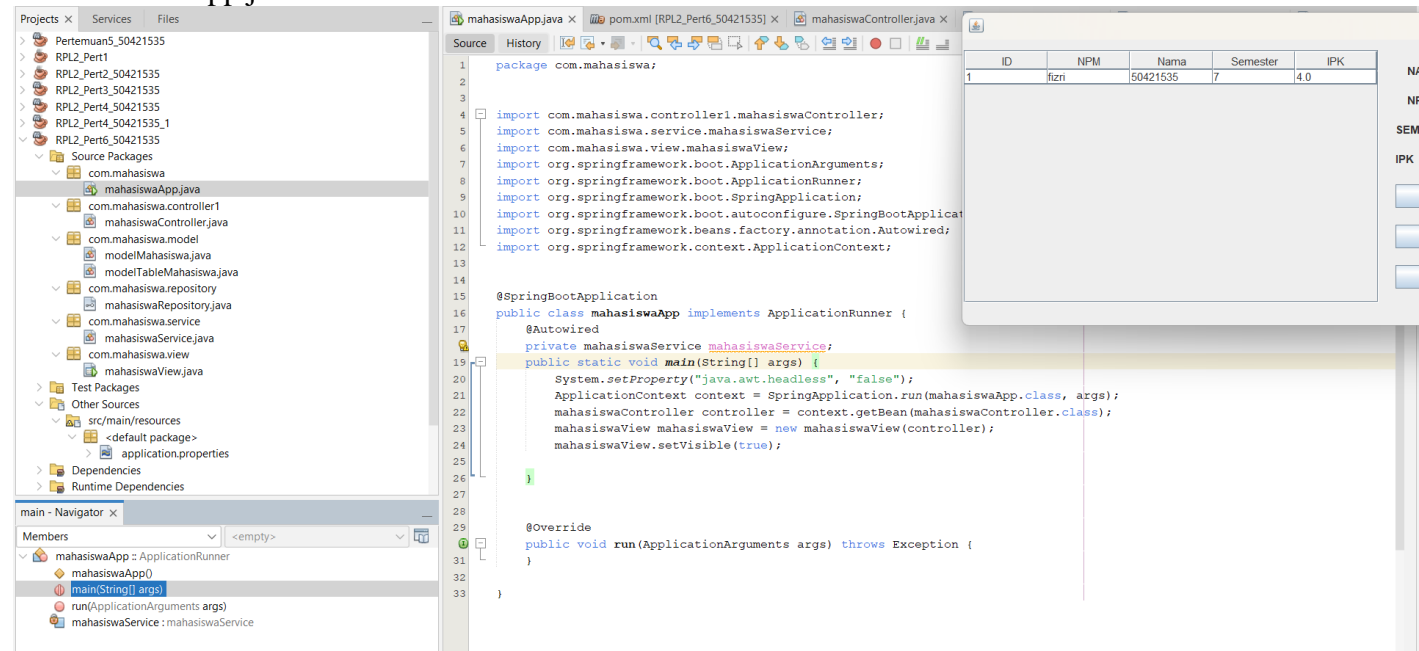
Soal

1. Jelaskan satu per satu Codingan kalian dari hasil screenshot Activity!

Jawab

1

A.mahasiswaApp.java



Paket dan Komponen MVC:

- **Controller (mahasiswaController):** Mengatur alur logika aplikasi dan berkomunikasi antara tampilan (View) dan layanan (Service).
- **Service (mahasiswaService):** Menyimpan logika bisnis utama, seperti pengolahan data mahasiswa.
- **View (mahasiswaView):** Menangani tampilan antarmuka pengguna (UI).

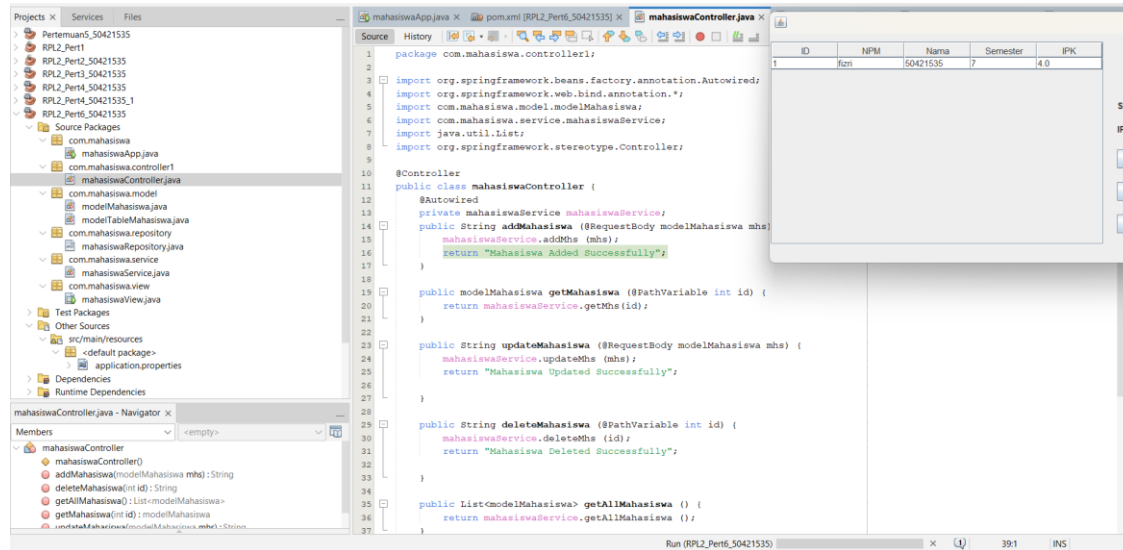
Main Class (mahasiswaApp):

- Menandai aplikasi sebagai **Spring Boot** dengan **@SpringBootApplication**.
- **main:** Memulai aplikasi, menjalankan konteks Spring, dan menampilkan GUI menggunakan **mahasiswaView** yang terhubung ke **mahasiswaController**.
- **run:** Disiapkan untuk logika tambahan saat aplikasi dijalankan, tapi saat ini masih kosong.

Fitur GUI:

- GUI ditampilkan melalui **mahasiswaView.setVisible(true)**, memungkinkan pengguna untuk mengelola data mahasiswa melalui antarmuka grafis.

B.mahasiswaController



Controller sebagai Penghubung:

- Kelas **mahasiswaController** berperan sebagai penghubung antara **View** (tampilan) dan **Service** (logika bisnis).

Dependency Injection:

- @Autowired mahasiswaService**: Controller menggunakan layanan **mahasiswaService** untuk memproses data mahasiswa.

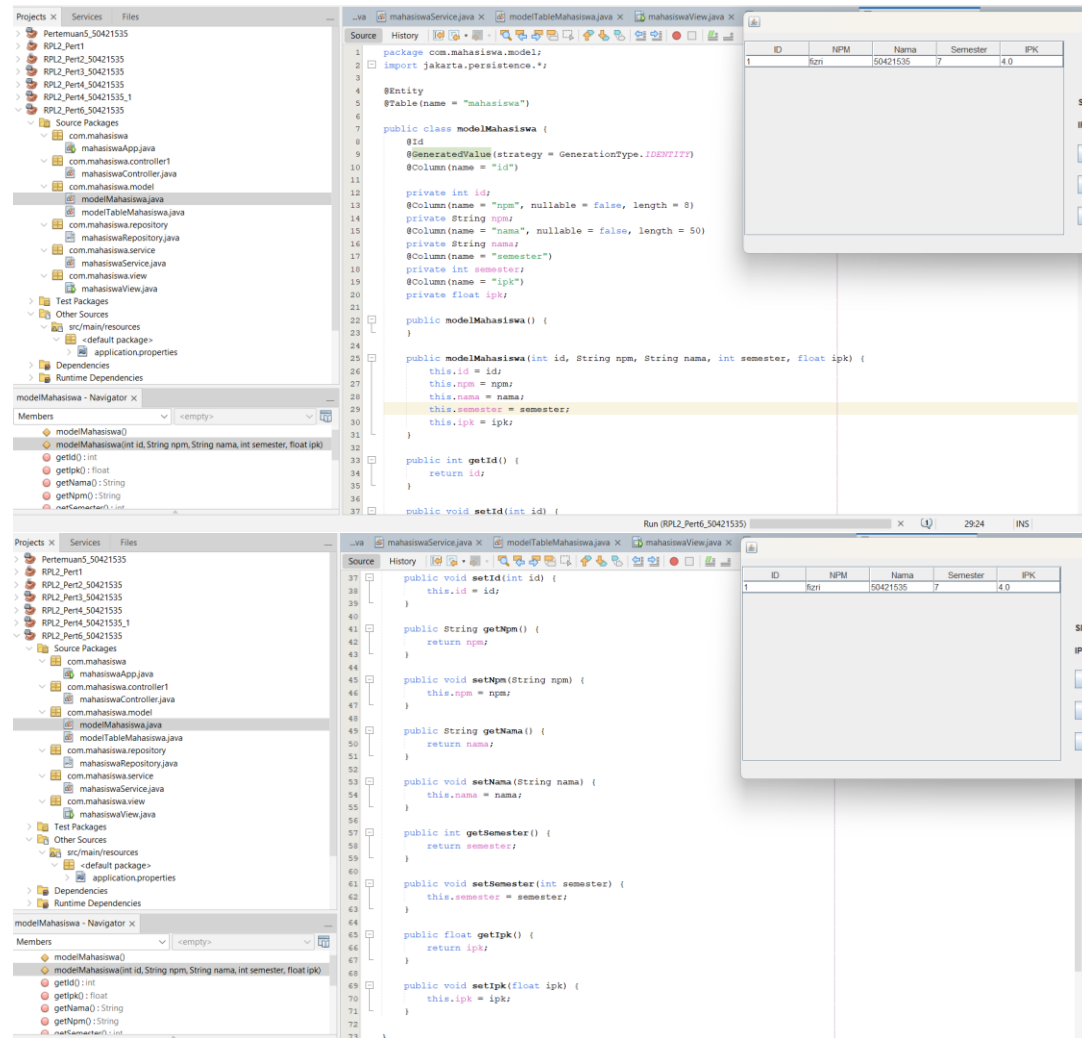
Fungsi CRUD (Create, Read, Update, Delete):

- addMahasiswa**: Menambahkan data mahasiswa baru ke dalam sistem.
- getMahasiswa**: Mengambil data mahasiswa berdasarkan **ID**.
- updateMahasiswa**: Memperbarui data mahasiswa yang ada.
- deleteMahasiswa**: Menghapus data mahasiswa berdasarkan **ID**.
- getAllMahasiswa**: Mengambil daftar seluruh data mahasiswa.

Konsep:

- Setiap fungsi memanggil metode dari **mahasiswaService**, yang menangani logika pengelolaan data.
- Fungsi mengembalikan respons berupa pesan atau data mahasiswa.

C.modelMahasiswa



Entity dan Table Mapping:

- **@Entity:** Menandai kelas sebagai entitas yang akan dipetakan ke tabel di database.
- **@Table(name = "mahasiswa"):** Menentukan nama tabel di database untuk entitas ini.

Atribut dan Kolom Database:

- **@Id:** Menandai kolom **id** sebagai primary key.
- **@GeneratedValue(strategy = GenerationType.IDENTITY):** Mengatur agar nilai **id** di-generate otomatis oleh database.
- **@Column:** Memetakan atribut kelas ke kolom database, dengan opsi seperti nama kolom, panjang maksimal, dan nullable.

Atribut yang dipetakan:

- **id:** ID unik untuk setiap mahasiswa (primary key).
- **npm:** Nomor pokok mahasiswa, panjang maksimal 8 karakter.
- **nama:** Nama mahasiswa, panjang maksimal 50 karakter.
- **semester**
- **ipk**

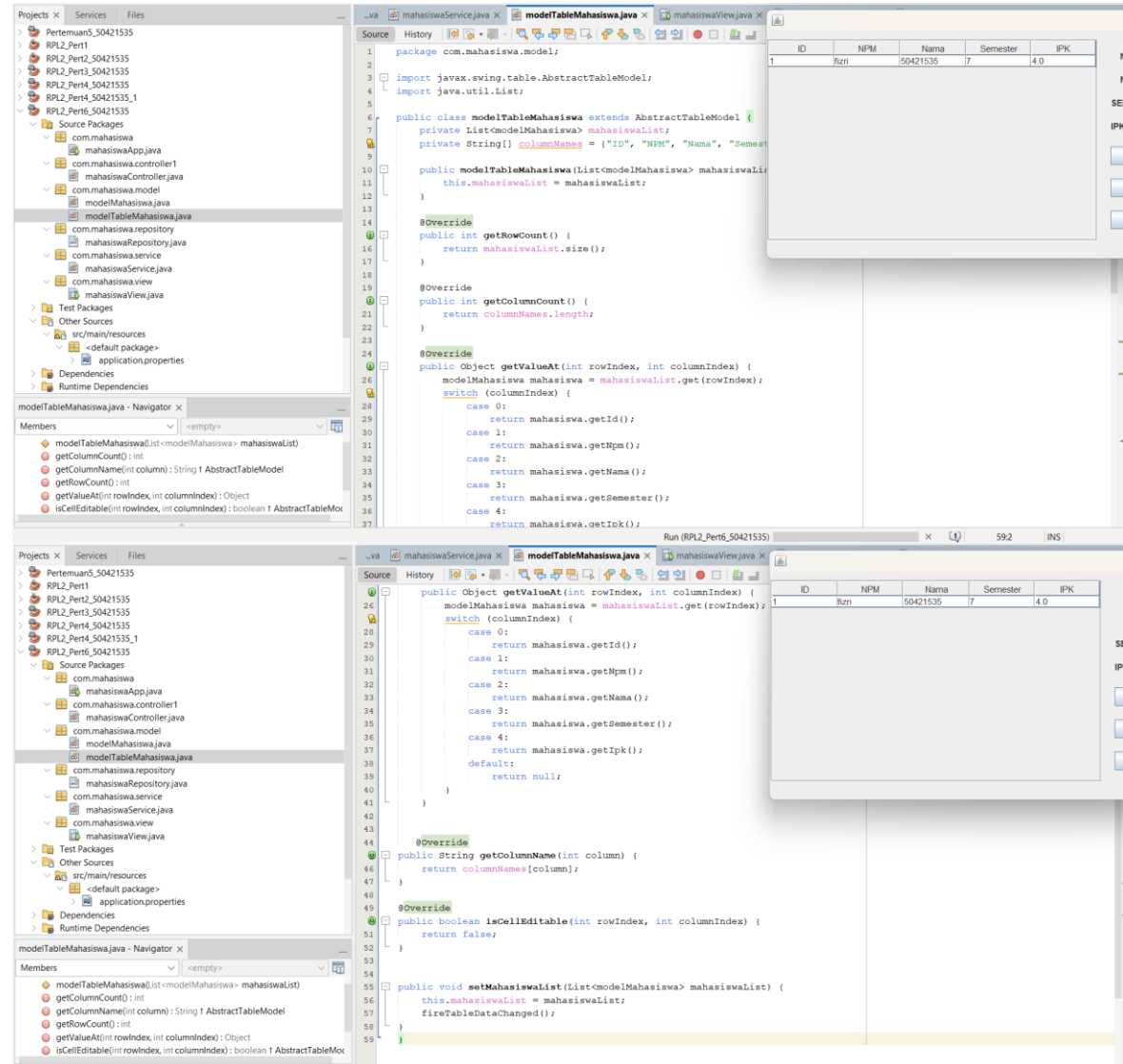
Konstruktor:

- **Default Constructor:** Diperlukan oleh JPA untuk proses instansiasi entitas.
- **Parameterized Constructor:** Mempermudah pembuatan objek dengan data awal.

Getter dan Setter:

- Fungsi **getter** dan **setter** untuk setiap atribut memungkinkan akses dan manipulasi data entitas ini.

D.modelTableMahasiswa



Atribut :

- **mahasiswaList**: Menyimpan daftar objek **modelMahasiswa** yang akan ditampilkan di tabel.
- **columnNames**: Menyimpan nama kolom yang akan ditampilkan di header tabel.

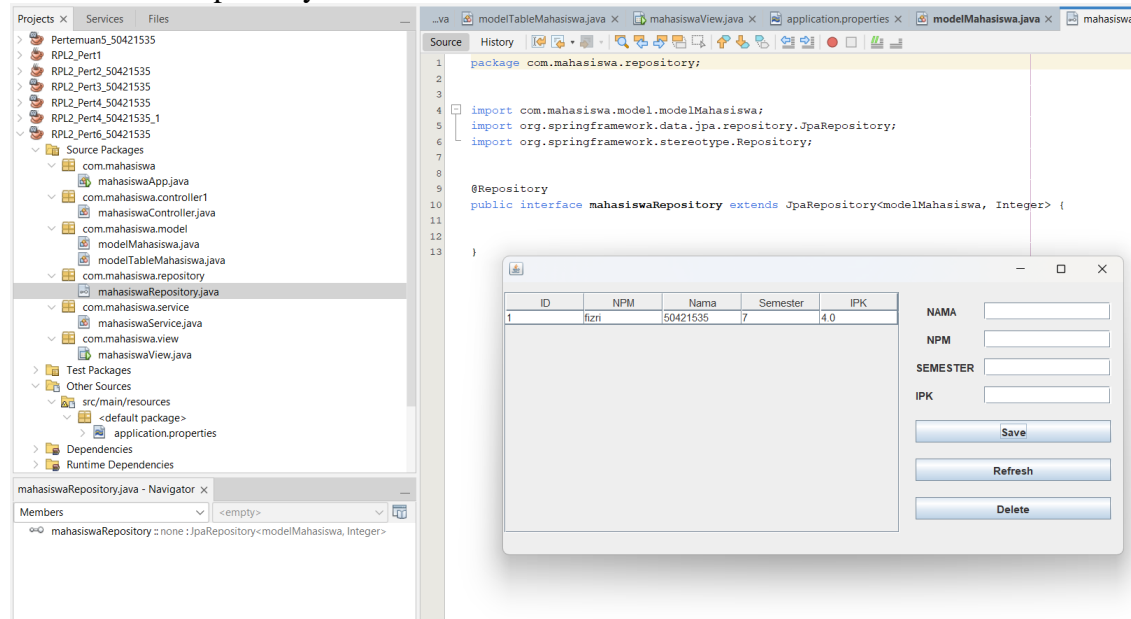
Metode Override :

- **getRowCount**: Mengembalikan jumlah baris berdasarkan ukuran **mahasiswaList**.
- **getColumnCount**: Mengembalikan jumlah kolom berdasarkan panjang array **columnNames**.
- **getValueAt**: Mengembalikan data di baris dan kolom tertentu berdasarkan atribut dari objek **modelMahasiswa**.
- **getColumnName**: Mengembalikan nama kolom berdasarkan indeks.
- **isCellEditable**: Mengatur apakah sel di tabel bisa diedit (diatur **false** sehingga tidak bisa diedit).

Metode setMahasiswaList:

- Memperbarui data di tabel dengan daftar mahasiswa baru.
- **fireTableDataChanged()**: Memberitahu tabel bahwa data telah berubah, sehingga tampilan diperbarui.

E.mahasiswaRepository



Annotation @Repository:

- Menandai bahwa **mahasiswaRepository** adalah komponen **Repository** di Spring. Komponen ini bertanggung jawab untuk operasi data, seperti menyimpan, mengambil, memperbarui, atau menghapus data di database.

Pewarisan dari JpaRepository:

- JpaRepository<modelMahasiswa, Integer>:**
 - modelMahasiswa:** Jenis entitas yang akan dikelola oleh repository.
 - Integer:** Tipe data dari primary key entitas (**id** pada **modelMahasiswa**).
- Dengan mewarisi **JpaRepository**, repository ini otomatis memiliki fungsi CRUD bawaan, seperti:
 - save():** Menyimpan atau memperbarui data.
 - findById():** Mengambil data berdasarkan ID.
 - findAll():** Mengambil semua data.
 - deleteById():** Menghapus data berdasarkan ID.

F.mahasiswaService

```
1 package com.mahasiswa.service;
2
3 import com.mahasiswa.model.modelMahasiswa;
4 import com.mahasiswa.repository.mahasiswaRepository;
5 import java.util.List;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Service;
8
9
10 @Service
11 public class mahasiswaService {
12     @Autowired
13     private mahasiswaRepository repository;
14
15     public void addMhs(modelMahasiswa mhs) {
16         repository.save(mhs);
17     }
18
19     public modelMahasiswa getMhs(int id) {
20         return repository.findById(id).orElse(null);
21     }
22
23     public void updateMhs(modelMahasiswa mhs) {
24         repository.save(mhs);
25     }
26
27     public void deleteMhs(int id) {
28         repository.deleteById(id);
29     }
30
31     public List<modelMahasiswa> getAllMahasiswa() {
32         return repository.findAll();
33     }
34 }
```

ID	NPM	Nama	Semester	IPK
1	fizri	50421535	7	4.0

Dependency Injection:

- **@Autowired mahasiswaRepository:** Menggunakan repository untuk berinteraksi dengan database.

Fungsi CRUD:

- **addMhs(modelMahasiswa mhs):**
 - Menambahkan data mahasiswa ke database menggunakan **repository.save()**.
- **getMhs(int id):**
 - Mengambil data mahasiswa berdasarkan ID menggunakan **repository.findById()**.
 - Jika ID tidak ditemukan, mengembalikan **null**.
- **updateMhs(modelMahasiswa mhs):**
 - Memperbarui data mahasiswa yang sudah ada (juga menggunakan **repository.save()**, karena fungsi ini dapat digunakan untuk insert dan update).
- **deleteMhs(int id):**
 - Menghapus data mahasiswa berdasarkan ID menggunakan **repository.deleteById()**.
- **getAllMahasiswa():**
 - Mengambil semua data mahasiswa dari database menggunakan **repository.findAll()**.

G.mahasiswaView

The screenshot displays the development environment for a Java Swing application named 'G.mahasiswaView'. The interface is divided into several panels:

- Project Explorer:** Shows the project structure, including source packages, test packages, and other sources. The 'mahasiswaView' package is highlighted.
- Source Editor:** Displays the code for the 'mahasiswaView' class. The code includes imports for the controller, model, and Swing components, and defines the 'mahasiswaView' class and its methods.
- Design View:** Shows a visual representation of the user interface. It features a table with columns 'Title 1', 'Title 2', 'Title 3', and 'Title 4'. To the right of the table are input fields for 'NAMA', 'NPM', 'SEMESTER', and 'IPK', along with 'Save', 'Refresh', and 'Delete' buttons.
- Palette:** Lists various Swing components and containers, such as 'Panel', 'Split Pane', 'Tabbed Pane', 'Scroll Pane', 'Desktop Pane', 'Layered Pane', 'Swing Containers', 'Swing Controls', 'Swing Menus', 'Swing Windows', 'Swing Fillers', 'AWT', and 'Beans'.
- Run Console:** Shows the output of the application, including a table of student data and a warning message.

The code in the Source Editor is as follows:

```
package com.mahasiswa.view;

import com.mahasiswa.controller1.mahasiswaController;
import com.mahasiswa.model.modelMahasiswa;
import com.mahasiswa.model.modelTableMahasiswa;

import java.util.List;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class mahasiswaView extends javax.swing.JFrame {
    private mahasiswaController controller;

    public mahasiswaView(mahasiswaController controller) {
        this.controller = controller;
        initComponents();
        loadMahasiswaTable();
    }

    private mahasiswaView() {
        throw new UnsupportedOperationException("Not Supported Yet.");
    }

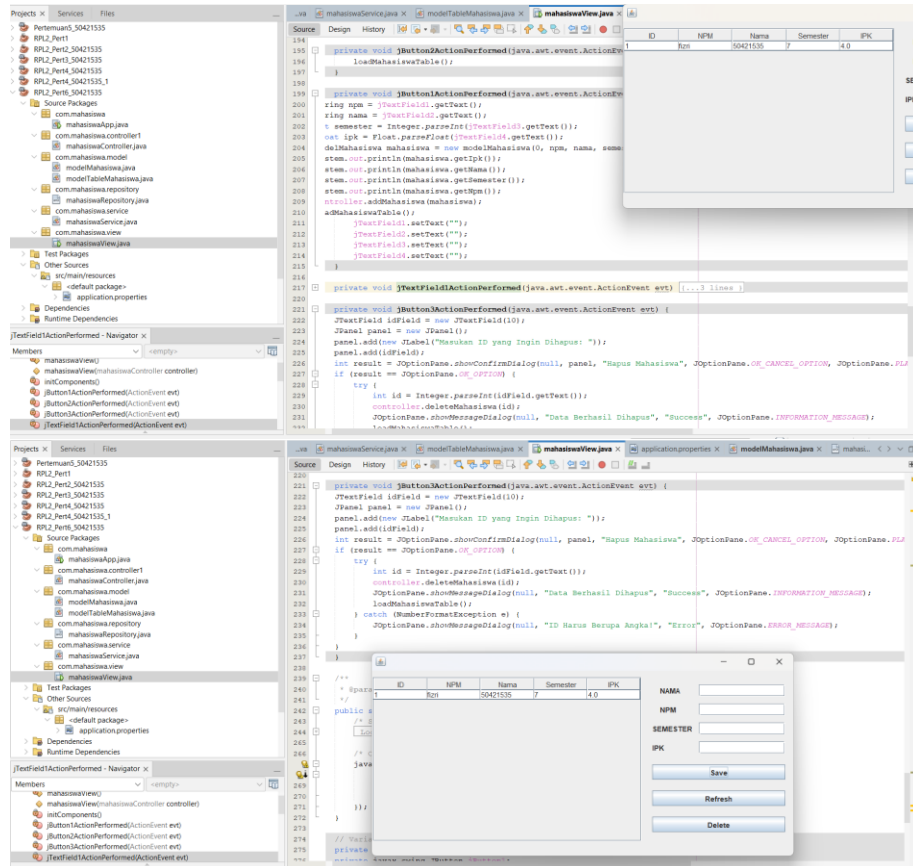
    public void loadMahasiswaTable() {
        List<modelMahasiswa> listMahasiswa = controller.getAllMahasiswa();
        modelTableMahasiswa tableModel = new modelTableMahasiswa(listMahasiswa);
        dataTable.setModel(tableModel);
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
}
```

The Run Console output shows a table of student data:

ID	NPM	Nama	Semester	IPK
1	fizri	50421535	7	4.0

A warning message is also displayed: "WARNING: Do NOT modify this code. The content of this method is always regenerated by the Form Editor."



Konstruksi View:

- **mahasiswaView(mahasiswaController controller):**
 - Konstruktor menerima **controller** sebagai parameter untuk menghubungkan View dengan logika aplikasi.
 - Memanggil **initComponents()** untuk menginisialisasi komponen GUI.
 - Memuat data mahasiswa ke tabel dengan **loadMahasiswaTable()**.

2. Fungsi Utama:

- **loadMahasiswaTable():**
 - Mengambil daftar mahasiswa dari **controller** melalui **getAllMahasiswa()**.
 - Menggunakan **modelTableMahasiswa** sebagai model untuk menampilkan data dalam tabel GUI.
 - Tabel di-update dengan data terbaru.

3. Event Listener:

- **Tombol Refresh (jButton2ActionPerformed):**
 - Memanggil **loadMahasiswaTable()** untuk merefresh data tabel.
- **Tombol Tambah (jButton1ActionPerformed):**
 - Mengambil input dari field teks (NPM, Nama, Semester, IPK).
 - Membuat objek **modelMahasiswa** baru dari data input.
 - Memanggil metode **addMahasiswa()** di **controller** untuk menambahkan data ke database.
 - Memuat ulang tabel dengan data terbaru.
 - Membersihkan field teks setelah data ditambahkan.

4. Fitur GUI:

- Komponen seperti **tabel (dataTable)**, **field input**, dan **action button** memfasilitasi pengguna untuk melihat, menambah, dan memperbarui data mahasiswa secara interaktif.
- Menggunakan **JOptionPane** untuk dialog pemberitahuan/error jika diperlukan.

H.pom.xml

The screenshot displays an IDE with the following components:

- Project Explorer (Left):** Shows the project structure for RPL2_Pert6_50421535, including source packages (com.mahasiswa, com.mahasiswa.controller, com.mahasiswa.model, com.mahasiswa.repository, com.mahasiswa.service, com.mahasiswa.view), test packages, other sources, and project files (pom.xml, nbactions.xml).
- POM model (Bottom Left):** Displays the project's metadata: Model Version: 4.0.0, GroupId: com.mahasiswa, ArtifactId: RPL2_Pert6_50421535, Packaging: jar, and Name: spring-boot-starter-parent.
- Source Code (Center):** Shows the pom.xml file with the following content:

```
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>3.3.3</version>
<relativePath/>
</parent>

<dependencies>
<!-- Hibernate + Spring Data JPA -->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<!-- MySQL Connector -->
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>8.0.33</version>
</dependency>

<!-- Spring Boot Web dependency (for MVC if needed) -->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>

<!-- Testing dependencies -->
<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>
```
- Table (Right):** A table with 5 columns: ID, NPM, Nama, Semester, and IPK. It contains one row with the following data: ID: 1, NPM: fizr, Nama: 50421535, Semester: 7, IPK: 4.0.

Parent

Menentukan spring-boot-starter-parent sebagai konfigurasi dasar. Ini menyederhanakan pengelolaan versi dan kompatibilitas pustaka yang digunakan.

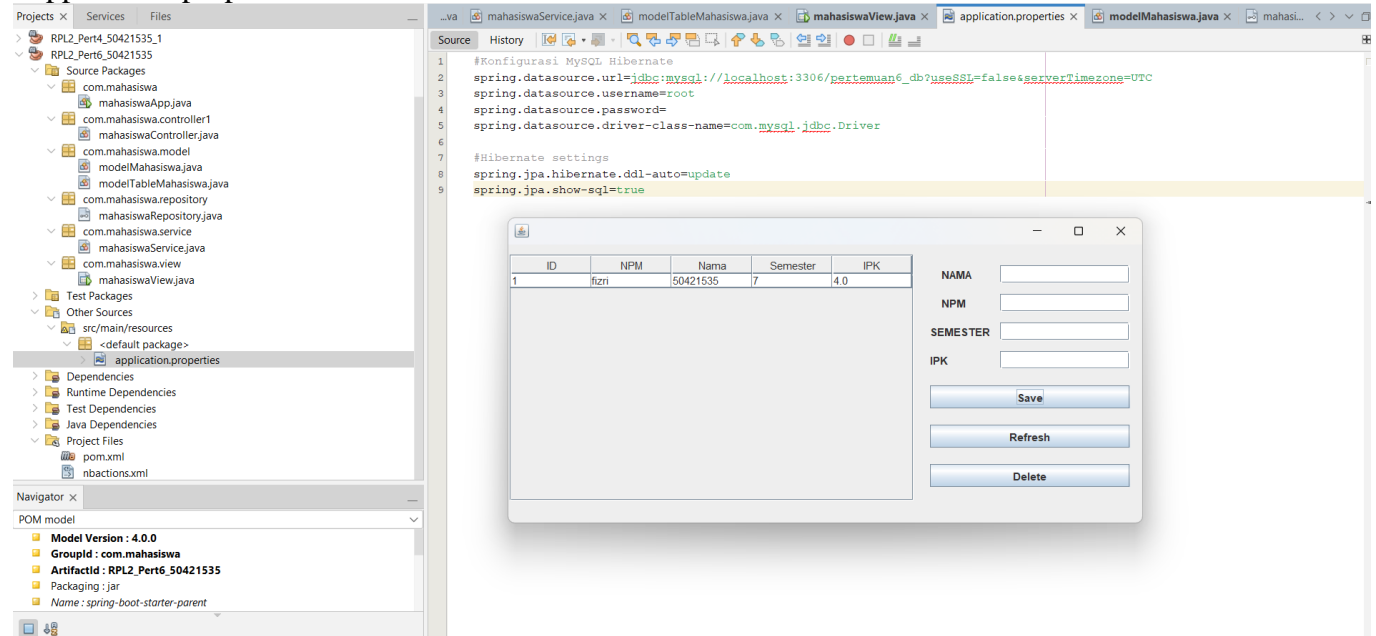
Dependencies

- **Spring Data JPA:** Mengintegrasikan Hibernate untuk ORM dan membantu dalam pengelolaan data di database.
- **MySQL Connector:** Driver JDBC untuk menghubungkan aplikasi dengan database MySQL.
- **Spring Boot Web:** Untuk membangun aplikasi berbasis web atau REST API.
- **Spring Boot Test:** Dependensi untuk menjalankan pengujian aplikasi (hanya aktif dalam *scope* pengujian).

Build Plugins

Menyertakan spring-boot-maven-plugin, yang mempermudah menjalankan dan membangun aplikasi Spring Boot dengan perintah Maven (mvn spring-boot:run, dll.).

I.application.properties



Konfigurasi Koneksi MySQL

- `spring.datasource.url`: URL koneksi ke database MySQL bernama `pertemuan6_db` di `localhost` pada port default 3306. Opsi `useSSL=false` dan `serverTimezone=UTC` digunakan untuk menghindari peringatan SSL dan memastikan zona waktu konsisten.
- `spring.datasource.username`: Nama pengguna database (dalam hal ini `root`).
- `spring.datasource.password`: Kata sandi pengguna database (kosong dalam contoh ini).
- `spring.datasource.driver-class-name`: Menentukan driver JDBC MySQL yang akan digunakan.

Pengaturan Hibernate (JPA)

- `spring.jpa.hibernate.ddl-auto`: Menentukan strategi Hibernate untuk pembuatan atau pembaruan skema database. Nilai `update` berarti Hibernate akan memperbarui skema berdasarkan perubahan entitas tanpa menghapus data yang ada.
- `spring.jpa.show-sql`: Mengaktifkan pencatatan (logging) kueri SQL yang dieksekusi Hibernate ke konsol, berguna untuk debugging.