LAPORAN AKHIR PRAKTIKUM

Mata Praktikum	: RPL
Kelas	: 4IA14
Praktikum ke-	:
Tanggal	:
Materi	:
NPM	: 51421599
Nama	: Haziq Duha Zainul Ihsan
Ketua Asisten	: aji
Jumlah Lembar	:

LABORATORIUM TEKNIK INFORMATIKA

UNIVERSITAS GUNADARMA

2023

MahasiswaApp

- **Class ini adalah entry point** untuk aplikasi Spring Boot.
- **@SpringBootApplication**: Menandai class sebagai konfigurasi utama Spring Boot.
- **main method**: Memulai aplikasi dengan memanggil SpringApplication.run untuk mendapatkan konteks aplikasi.
- **MahasiswaController dan MahasiswaView2**: Dipanggil secara manual untuk menginisialisasi antarmuka pengguna berbasis Swing, dengan menonaktifkan mode headless (System.setProperty).
- **ApplicationRunner**: Mengizinkan penambahan logika yang dijalankan setelah aplikasi dimulai, meskipun saat ini dibiarkan kosong.

MahasiswaController

- **@Controller**: Menandakan bahwa class ini bertindak sebagai pengendali untuk logika aplikasi.
- **@Autowired**: Menyuntikkan dependensi MahasiswaService secara otomatis oleh Spring.
- Method-method di dalamnya mengelola data mahasiswa:
    - **addMahasiswa**: Menambahkan mahasiswa baru dengan menerima data berbentuk JSON (@RequestBody).
    - **getMahasiswa**: Mengambil data mahasiswa berdasarkan id (@PathVariable).
    - **updateMahasiswa**: Memperbarui data mahasiswa berdasarkan entitas yang diberikan.
    - **deleteMahasiswa**: Menghapus mahasiswa berdasarkan id.
    - **getAllMahasiswa**: Mengambil semua data mahasiswa.

ModelMahasiswa

- **@Entity dan @Table**: Menandakan class ini sebagai entitas JPA yang dipetakan ke tabel database bernama mahasiswa.
- **Kolom database**:
    - **id**: Primary key, di-generate secara otomatis (@GeneratedValue).
    - **npm**: Nomor pokok mahasiswa (string dengan panjang maksimal 8 karakter, wajib diisi).
    - **nama**: Nama mahasiswa (string dengan panjang maksimal 50 karakter, wajib diisi).
    - **semester**: Semester aktif mahasiswa (integer).
    - **ipk**: IPK mahasiswa (float).

- **Getter dan Setter**: Digunakan untuk mengakses dan memodifikasi data entitas.
- **Constructor**: Ada dua jenis, kosong (untuk ORM) dan berisi parameter untuk inisialisasi data secara manual.

ModelTabelMahasiswa

- Class ini adalah implementasi **AbstractTableModel** dari Swing untuk mengatur data mahasiswa dalam format tabel:
    - **mahasiswaList**: Menyimpan data mahasiswa.
    - **getRowCount dan getColumnCount**: Mengembalikan jumlah baris dan kolom dalam tabel.
    - **getValueAt**: Mengambil nilai di sel tertentu berdasarkan baris dan kolom.
    - **getColumnName**: Mengembalikan nama kolom.
    - **setMahasiswaList**: Memperbarui data mahasiswa dan memberi tahu tabel bahwa data telah berubah.

MahasiswaRepository

- **@Repository**: Menandakan interface ini sebagai repository untuk Spring Data JPA.
- **JpaRepository**: Memberikan metode bawaan untuk operasi CRUD (Create, Read, Update, Delete).

MahasiswaService

- **@Service**: Menandakan bahwa class ini mengandung logika bisnis aplikasi.
- **addMhs**: Menyimpan entitas mahasiswa baru ke database.
- **getMhs**: Mengambil entitas mahasiswa berdasarkan id (jika ada).
- **updateMhs**: Memperbarui entitas mahasiswa di database.
- **deleteMhs**: Menghapus entitas mahasiswa berdasarkan id.
- **getAllMahasiswa**: Mengambil semua entitas mahasiswa dari database.

Aplikasi **MahasiswaApp** memiliki logika kerja berbasis arsitektur **MVC (Model-View-Controller)** yang memisahkan tugas utama menjadi tiga bagian. Ketika aplikasi dijalankan, Spring Boot menginisialisasi semua komponen yang diperlukan, seperti **Controller**, **Service**, dan **Repository**, yang saling terhubung untuk menangani data mahasiswa. Aplikasi ini juga menggunakan antarmuka grafis berbasis Swing, MahasiswaView2, yang ditampilkan setelah aplikasi mulai berjalan.

Logika pengelolaan data mahasiswa dimulai ketika pengguna melakukan aksi, seperti menambah, mengedit, menghapus, atau melihat data mahasiswa melalui GUI. Aksi ini diteruskan ke MahasiswaController, yang bertugas sebagai pengendali utama. Misalnya, saat menambahkan data, MahasiswaController memanggil method addMahasiswa yang meneruskan permintaan tersebut ke MahasiswaService. **Service layer** bertindak sebagai jembatan antara controller dan database, di mana logika bisnis diproses. Dalam hal ini, data mahasiswa disimpan ke database menggunakan method bawaan dari MahasiswaRepository, yang berinteraksi dengan database melalui **JPA (Java Persistence API)**.

Setiap data mahasiswa direpresentasikan sebagai objek ModelMahasiswa, yang dipetakan ke tabel mahasiswa di database. Objek ini memiliki atribut seperti id, npm, nama, semester, dan ipk. Ketika pengguna ingin melihat semua data mahasiswa, logika dalam MahasiswaService mengambil seluruh data dari tabel menggunakan method findAll pada MahasiswaRepository. Data ini kemudian dikirim kembali ke MahasiswaController dan diteruskan ke GUI dalam bentuk tabel melalui model ModelTabelMahasiswa.

Dengan logika yang terstruktur ini, aplikasi memastikan bahwa setiap operasi—baik menambah, memperbarui, menghapus, maupun melihat data—dilakukan secara terorganisasi, dengan setiap lapisan (controller, service, repository) memiliki tugas spesifik untuk menjaga modularitas dan kemudahan pemeliharaan.