

LAPORAN AKHIR PRAKTIKUM

Mata Praktikum : RPL 2
Kelas : 4IA06
Praktikum ke- : 5
Tanggal : 12 November 2024
Materi : Framework Spring,Pembuatan Project Spring, dan Hibernate
NPM : 50421535
Nama : Fizri Rosdiansyah
Ketua Asisten :
Paraf Asisten :
Nama Asisten : Suryo Aji Widago
Jumlah Lembar :12 Lembar

LABORATORIUM TEKNIK INFORMATIKA

UNIVERSITAS GUNADARMA

2024

Rekayasa Perangkat Lunak 2

Soal

1. Jelaskan apa itu Spring Boot dan bagaimana perbedaannya dengan kode pada pertemuan sebelumnya. Apa keuntungan utama yang ditawarkan Spring Boot bagi pengembang aplikasi?
2. Jelaskan kode dan langkah-langkah program yang telah dibuat!

Jawab

1.

- **Spring Boot** adalah framework berbasis Java yang dirancang untuk mempermudah pengembangan aplikasi dengan Spring.
- Perbedaan codingannya:
dimana Codingan Sebelumnya Harus **Mengatur file konfigurasi XML** atau kode konfigurasi Java untuk mengelola dependensi, bean, dan pengaturan database.
- **Mengonfigurasi ORM dan Hibernate secara manual** dengan menyetel koneksi database, entitas, dan session factory secara eksplisit.
- **Mengelola server aplikasi secara eksternal**, di mana Anda harus menyebarkan aplikasi ke dalam server aplikasi (misalnya Tomcat) untuk menjalankannya.
- **Keuntungan Utama Spring Boot bagi Pengembang Aplikasi**
- **Waktu Pengembangan yang Lebih Cepat:** Dengan konfigurasi otomatis dan starter dependencies, Spring Boot sangat mengurangi waktu dan upaya konfigurasi.
- **Aplikasi Standalone:** Dengan server bawaan, aplikasi bisa dijalankan langsung tanpa server aplikasi eksternal, yang memudahkan pengujian dan penyebaran.
- **Production-ready Features:** Spring Boot menyediakan alat pemantauan dan manajemen untuk aplikasi produksi, seperti Spring Boot Actuator, yang memberi wawasan mendalam tentang status aplikasi.
- **Dukungan Cloud dan Microservices:** Spring Boot memiliki integrasi yang baik untuk pengembangan aplikasi berbasis microservices dan penggunaan di lingkungan cloud.

2.A.Pertemuan5_50421535

The image displays two screenshots of an IDE (IntelliJ IDEA) showing the implementation of a Spring Boot application. The top screenshot shows the initial setup of the `Pertemuan5_50421535` class, which implements `CommandLineRunner`. The bottom screenshot shows the completion of the `main` and `run` methods. The output console shows the application running, inserting a student record, displaying a menu, and selecting an option.

```
package me.fizri;

import me.fizri.controller.MahasiswaController;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Pertemuan5_50421535 implements CommandLineRunner {

    @Autowired

    private MahasiswaController mhsController;

    public static void main(String[] args) {
        SpringApplication.run(Pertemuan5_50421535.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        mhsController.tampilkanMenu();
    }
}
```

Output - Run (Pertemuan5_50421535)

```
/
Masukkan ipk:
4
Hibernate: insert into mahasiswa (ipk,nama,npm,semester) values (?,?,?,?)
Mahasiswa Berhasil ditambahkan.

Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mml_0.id,mml_0.ipk,mml_0.nama,mml_0.npm,mml_0.semester from mahasiswa mml_0
Mahasiswa(id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0')
```

```
package me.fizri;

import me.fizri.controller.MahasiswaController;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Pertemuan5_50421535 implements CommandLineRunner {

    @Autowired

    private MahasiswaController mhsController;

    public static void main(String[] args) {
        SpringApplication.run(Pertemuan5_50421535.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        mhsController.tampilkanMenu();
    }
}
```

Output - Run (Pertemuan5_50421535)

```
/
Masukkan ipk:
4
Hibernate: insert into mahasiswa (ipk,nama,npm,semester) values (?,?,?,?)
Mahasiswa Berhasil ditambahkan.

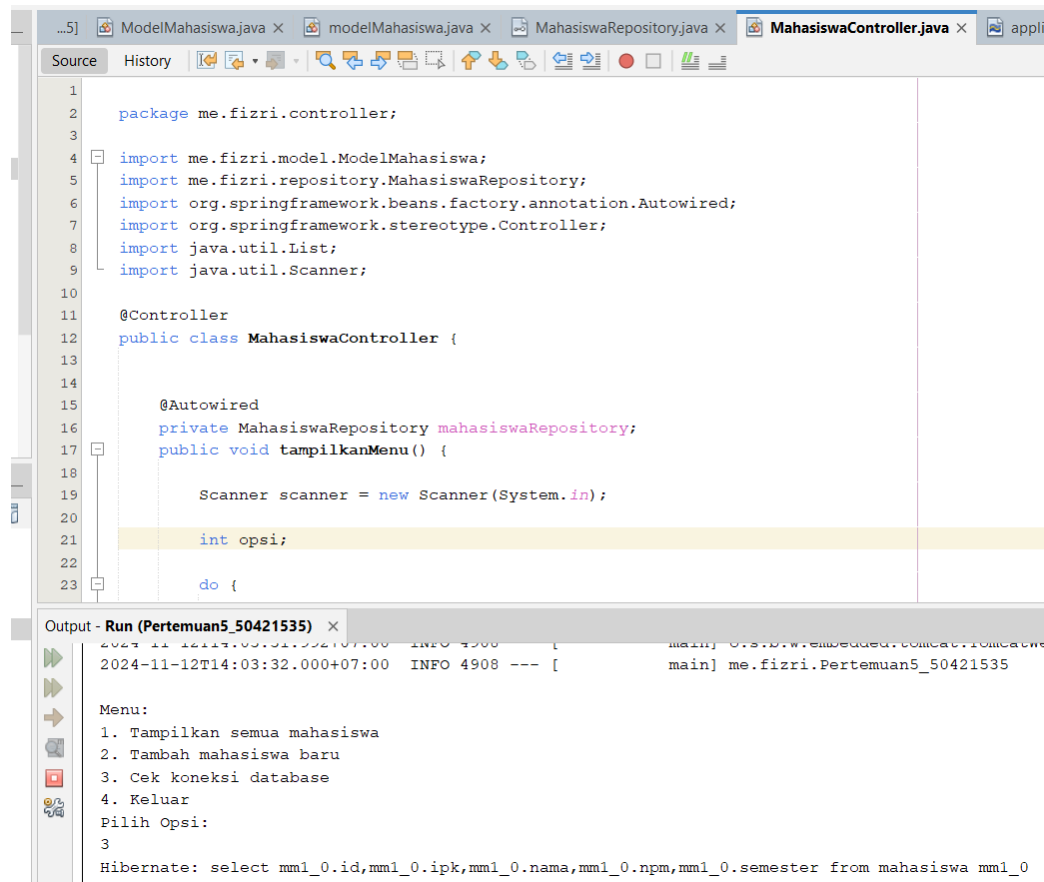
Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mml_0.id,mml_0.ipk,mml_0.nama,mml_0.npm,mml_0.semester from mahasiswa mml_0
Mahasiswa(id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0')
```

1. **Inisialisasi Aplikasi Spring Boot:** Kelas Pertemuan5_50421535 adalah titik masuk aplikasi Spring Boot, dan menjalankan aplikasi dengan konfigurasi otomatis.
2. **Dependency Injection:** Menggunakan anotasi `@Autowired`, Spring secara otomatis menyediakan (meng-*inject*) objek `MahasiswaController` ke dalam kelas ini, sehingga dapat digunakan tanpa perlu membuatnya secara manual.
3. **Eksekusi Logika Saat Startup:** Kelas ini mengimplementasikan `CommandLineRunner`, yang memungkinkan metode `run()` dijalankan segera setelah aplikasi dimulai. Di dalam metode ini, `mhsController.tampilkanMenu()` dipanggil, yang kemungkinan besar memicu tampilan menu untuk pengguna.

Intinya:

Kode ini adalah aplikasi Spring Boot sederhana yang, pada saat dijalankan, memulai aplikasi, menginisialisasi `MahasiswaController`, dan menampilkan menu dengan memanggil `tampilkanMenu()`.

B. MahasiswaController



```
1 package me.fizri.controller;
2
3
4 import me.fizri.model.ModelMahasiswa;
5 import me.fizri.repository.MahasiswaRepository;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Controller;
8 import java.util.List;
9 import java.util.Scanner;
10
11 @Controller
12 public class MahasiswaController {
13
14
15     @Autowired
16     private MahasiswaRepository mahasiswaRepository;
17     public void tampilkanMenu() {
18
19         Scanner scanner = new Scanner(System.in);
20
21         int opsi;
22
23         do {
24
25             Menu:
26             1. Tampilkan semua mahasiswa
27             2. Tambah mahasiswa baru
28             3. Cek koneksi database
29             4. Keluar
30             Pilih Opsi:
31             3
32             Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
```

Output - Run (Pertemuan5_50421535) x

```
2024-11-12T14:03:32.000+07:00 INFO 4908 [main] O.S.B.W.EmbeddedTomcatTomcatw
2024-11-12T14:03:32.000+07:00 INFO 4908 --- [main] me.fizri.Pertemuan5_50421535

Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
3
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
```

```
...5] ModelMahasiswa.java x modelMahasiswa.java x MahasiswaRepository.java x MahasiswaController.java x a
Source History
25     System.out.println("\nMenu: ");
26     System.out.println("1. Tampilkan semua mahasiswa");
27     System.out.println("2. Tambah mahasiswa baru");
28     System.out.println("3. Cek koneksi database");
29     System.out.println("4. Keluar");
30     System.out.println("Pilih Opsi: ");
31     opsi = scanner.nextInt();
32     scanner.nextLine();
33     switch (opsi) {
34         case 1:
35             tampilkanSemuaMahasiswa();
36             break;
37         case 2:
38             tambahMahasiswa(scanner);
39             break;
40         case 3:
41             cekKoneksi();
42             break;
43         case 4:
44             System.out.println("Keluar dari program");
45             break;
46     }
47
```

Output - Run (Pertemuan5_50421535) x

```
2. Tampilkan semua mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_
Mahasiswa{id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0'}
```

```
...5] ModelMahasiswa.java x modelMahasiswa.java x MahasiswaRepository.java x MahasiswaController.java x applic
Source History
48     default:
49         System.out.println("Opsi tidak valid, coba ");
50     }
51     while (opsi != 4);
52 }
53
54 private void tampilkanSemuaMahasiswa () {
55
56     List<ModelMahasiswa> mahasiswaList = mahasiswaRepository.findAll();
57
58     if (mahasiswaList.isEmpty()) {
59         System.out.println("Tidak ada data mahasiswa.");
60     } else {
61         mahasiswaList.forEach (mahasiswa -> System.out.println (mahasiswa));
62     }
63 }
64
65
66
67
68
69
70
```

Output - Run (Pertemuan5_50421535) x

```
2. Tampilkan semua mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
Mahasiswa{id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0'}
```

```
...5] ModelMahasiswa.java X modelMahasiswa.java X MahasiswaRepository.java X MahasiswaController.java X application.properties X
Source History
69
70 private void tambahMahasiswa (Scanner scanner) {
71
72     System.out.println("Masukkan NPM: ");
73     String npm = scanner.nextLine();
74     System.out.println("Masukkan Nama: ");
75     String nama = scanner.nextLine();
76     System.out.println("Masukkan semester: ");
77     int semester = scanner.nextInt();
78     System.out.println("Masukkan ipk: ");
79     float ipk = scanner.nextFloat();
80     ModelMahasiswa mahasiswa = new ModelMahasiswa (0, npm, nama, semester, ipk);
81     mahasiswaRepository.save(mahasiswa);
82     System.out.println("Mahasiswa Berhasil ditambahkan.");
83 }
84
85 private void cekKoneksi () {
86
87     try {
88
89         mahasiswaRepository.findAll();
90
91         System.out.println("Koneksi ke database berhasil");
92
93     } catch (Exception e) {
94
95         System.out.println("Gagal terhubung ke database.");
96
97     }
98
99 }
100 }
```

```
Output - Run (Pertemuan5_50421535) X
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
Mahasiswa{id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0'}
```

```
Output - Run (Pertemuan5_50421535) X
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
Mahasiswa{id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0'}
```

- **Anotasi @Controller:**
- Kelas ini adalah bagian dari lapisan *controller* dalam Spring dan diberi anotasi @Controller, sehingga Spring mengenalinya sebagai komponen *controller*.
- **Dependency Injection dengan @Autowired:**
- MahasiswaRepository di-*inject* ke dalam kelas ini menggunakan anotasi @Autowired, memungkinkan kelas ini untuk berinteraksi dengan database melalui MahasiswaRepository tanpa perlu membuat objeknya secara manual.
- **Metode tampilkanMenu:**
- Metode ini menampilkan menu utama bagi pengguna untuk melakukan berbagai tindakan terkait data mahasiswa. Menggunakan *loop* do-while agar menu tetap ditampilkan sampai pengguna memilih opsi "Keluar".

- **Pilihan dalam Menu:**
- **Opsi 1 - tampilkanSemuaMahasiswa:** Menampilkan semua data mahasiswa yang ada dalam database. Jika data kosong, akan mencetak pesan "Tidak ada data mahasiswa."
- **Opsi 2 - tambahMahasiswa:** Menambahkan data mahasiswa baru ke database. Mengambil input dari pengguna untuk detail mahasiswa seperti NPM, nama, semester, dan IPK, lalu menyimpannya melalui mahasiswaRepository.
- **Opsi 3 - cekKoneksi:** Memeriksa koneksi ke database dengan mencoba mengakses data. Jika koneksi berhasil, mencetak pesan keberhasilan; jika gagal, mencetak pesan kegagalan.
- **Opsi 4 - Keluar:** Menghentikan program.
- **Metode tampilkanSemuaMahasiswa:**
- Mengambil data semua mahasiswa dari database menggunakan mahasiswaRepository.findAll(), lalu menampilkannya satu per satu.
- **Metode tambahMahasiswa:**
- Mengambil input dari pengguna untuk membuat objek ModelMahasiswa baru, lalu menyimpan objek tersebut ke database.
- **Metode cekKoneksi:**
- Memeriksa koneksi ke database dengan mencoba menjalankan findAll() dari mahasiswaRepository. Menampilkan pesan berhasil atau gagal sesuai dengan hasil koneksi.

C.ModelMahasiswa

```

1 package me.fizri.model;
2
3 import jakarta.persistence.Column;
4 import jakarta.persistence.Entity;
5 import jakarta.persistence.GeneratedValue;
6 import jakarta.persistence.GenerationType;
7 import jakarta.persistence.Id;
8 import jakarta.persistence.Table;
9
10 @Entity
11 @Table(name = "mahasiswa")
12 public class ModelMahasiswa {
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     @Column(name = "id")
16     private int id;
17
18     @Column(name = "npm", nullable = false, length = 8)
19     private String npm;
20
21     @Column(name = "nama", nullable = false, length = 50)
22     private String nama;
23 }

```

Output - Run (Pertemuan5_50421535)

```

2. tampilkan mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
Mahasiswa{id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0'}

```

```
...5] ModelMahasiswa.java x modelMahasiswa.java x MahasiswaRepository.java x MahasiswaController.java x appli
Source History
21 @Column(name = "nama", nullable = false, length = 50)
22 private String nama;
23
24 @Column(name = "semester")
25 private int semester;
26
27 @Column(name = "ipk")
28 private float ipk;
29
30 public ModelMahasiswa() {
31
32 }
33
34 public ModelMahasiswa(int id, String npm, String nama, int semester, float ipk) {
35     this.id = id;
36     this.npm = npm;
37     this.nama = nama;
38     this.semester = semester;
39     this.ipk = ipk;
40 }
41
42 public int getId() {
43     return id;
44 }
```

Output - Run (Pertemuan5_50421535) x

```
2. Tampilkan mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
Mahasiswa{id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0'}
```

```
...5] ModelMahasiswa.java x modelMahasiswa.java x MahasiswaRepository.java x MahasiswaController.java x appli
Source History
41
42 public int getId() {
43     return id;
44 }
45
46 public void setId(int id) {
47     this.id = id;
48 }
49
50 public String getNpm() {
51     return npm;
52 }
53
54 public void setNpm(String npm) {
55     this.npm = npm;
56 }
57
58 public String getNama() {
59     return nama;
60 }
61
62 public void setNama(String nama) {
63     this.nama = nama;
64 }
```

Output - Run (Pertemuan5_50421535) x

```
2. Tampilkan mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
Mahasiswa{id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0'}
```



```
...5] ModelMahasiswa.java x modelMahasiswa.java x MahasiswaRepository.java x MahasiswaController.java x app
Source History
66 public int getSemester() {
67     return semester;
68 }
69
70 public void setSemester(int semester) {
71     this.semester = semester;
72 }
73
74 public float getIpk() {
75     return ipk;
76 }
77
78 public void setIpk(float ipk) {
79     this.ipk = ipk;
80 }
81
82 @Override
83
84 public String toString() {
85
86     return "Mahasiswa{" +
87
88     "id=" + id +
89
90     ", npm=" + npm + '\n' +
91
92     ", nama=" + nama + '\n' +
93
94     ", semester=" + semester + '\n' +
95
96     ", ipk=" + ipk + '\n' +
97
98     '}' +
99
100 }
101 }
```

Output - Run (Pertemuan5_50421535) x

```
2. Tampilkan mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
Mahasiswa{id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0'}
```

```
Output - Run (Pertemuan5_50421535) x
2. Tampilkan mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
Mahasiswa{id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0'}
```

1. Anotasi @Entity dan @Table:

- Anotasi @Entity menunjukkan bahwa kelas ini adalah entitas yang akan dipetakan ke dalam tabel database.
- @Table(name = "mahasiswa") menunjukkan bahwa entitas ini dipetakan ke tabel bernama mahasiswa di database.

2. Atribut (Fields) dan Anotasi Kolom:

- Kelas ini memiliki beberapa atribut yang mewakili kolom dalam tabel mahasiswa:
 - **id:** Kolom ID sebagai *primary key*, dengan anotasi @Id dan *auto increment*

- (@GeneratedValue(strategy = GenerationType.IDENTITY)).
 - **npm:** Kolom NPM, sebuah String yang dibatasi hingga 8 karakter dan tidak boleh bernilai null.
 - **nama:** Kolom Nama, sebuah String yang dibatasi hingga 50 karakter dan tidak boleh bernilai null.
 - **semester:** Kolom Semester, disimpan sebagai tipe int.
 - **ipk:** Kolom IPK, disimpan sebagai tipe float.
 - Setiap kolom diberi anotasi @Column untuk mengatur nama kolom dan beberapa atribut lainnya, seperti nullable dan length.
- 3. **Constructor:**
 - **Constructor tanpa parameter:** Diperlukan oleh Hibernate untuk membuat instance entitas secara otomatis.
 - **Constructor dengan parameter:** Mempermudah pembuatan objek ModelMahasiswa dengan semua atribut yang diperlukan.
- 4. **Getter dan Setter:**
 - Setiap atribut memiliki metode get dan set untuk memungkinkan akses dan modifikasi nilai dari luar kelas.
- 5. **Metode toString:**
 - Metode toString memberikan representasi String dari objek ModelMahasiswa, mencantumkan semua atribut. Ini berguna untuk debugging atau ketika ingin menampilkan data mahasiswa dengan format yang mudah dibaca.

Intinya:

Kelas ModelMahasiswa adalah model entitas yang memetakan data mahasiswa ke tabel mahasiswa dalam database. Kelas ini memiliki atribut yang sesuai dengan kolom tabel, serta metode untuk mengakses dan memodifikasi data tersebut.

D.MahasiswaRepository.java

```

1 package me.fizri.repository;
2 import me.fizri.model.ModelMahasiswa;
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 @Repository
7
8 public interface MahasiswaRepository extends JpaRepository<ModelMahasiswa, Long> {
9
10
11 }
12

```

Output - Run (Pertemuan5_50421535) x

```

1. Tambah mahasiswa baru
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
Mahasiswa{id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0'}

```

1. Anotasi @Repository:

- Menandai interface ini sebagai komponen Spring untuk interaksi dengan database. Dengan anotasi ini, Spring akan mengenali MahasiswaRepository sebagai komponen Repository yang menangani operasi CRUD pada entitas ModelMahasiswa.

2. Pewarisan dari JpaRepository:

- Interface MahasiswaRepository mewarisi dari JpaRepository, yang merupakan interface bawaan Spring Data JPA.
- JpaRepository<ModelMahasiswa, Long> berarti MahasiswaRepository dikhususkan untuk menangani entitas ModelMahasiswa dengan *primary key* bertipe Long.

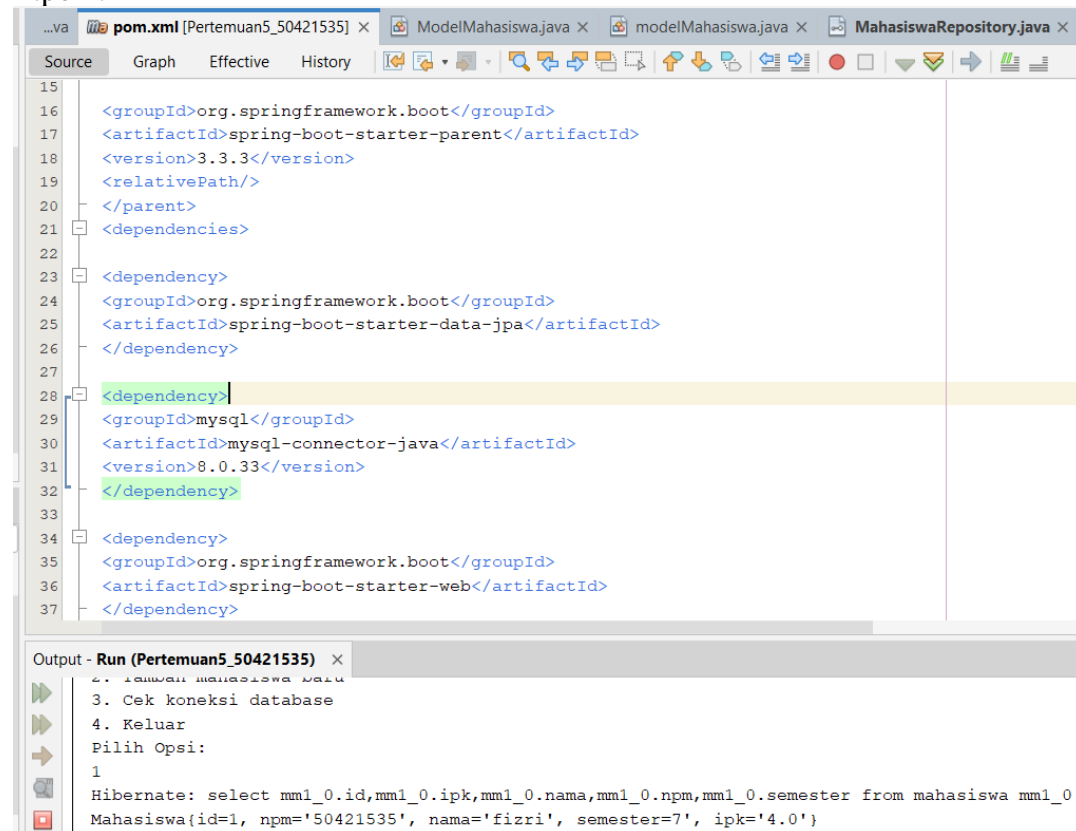
3. Operasi CRUD Otomatis:

- Dengan mewarisi JpaRepository, interface ini secara otomatis mendapatkan berbagai metode standar untuk operasi basis data, seperti:
 - findAll() - Mengambil semua data entitas.
 - save() - Menyimpan atau memperbarui entitas ke database.
 - findById() - Mengambil entitas berdasarkan ID.
 - deleteById() - Menghapus entitas berdasarkan ID.
- Hal ini berarti Anda tidak perlu menulis kode tambahan untuk operasi CRUD dasar.

Intinya:

MahasiswaRepository adalah interface yang memfasilitasi interaksi dengan database untuk entitas ModelMahasiswa. Dengan menggunakan JpaRepository, Spring secara otomatis menyediakan berbagai metode CRUD standar, sehingga menghemat waktu dalam pembuatan query manual.

E.pom.xml

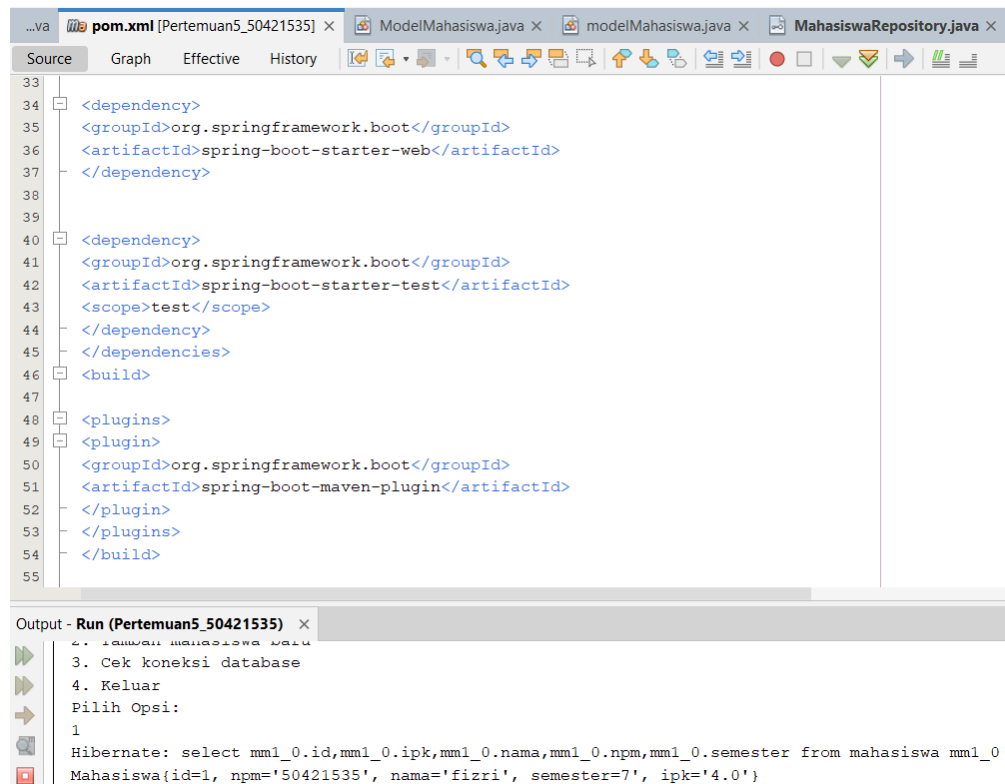


The screenshot shows an IDE with the following components:

- Source Editor:** Displays the `pom.xml` file with the following content:

```
15 <groupId>org.springframework.boot</groupId>
16 <artifactId>spring-boot-starter-parent</artifactId>
17 <version>3.3.3</version>
18 <relativePath/>
19 </parent>
20 </parent>
21 <dependencies>
22
23 <dependency>
24 <groupId>org.springframework.boot</groupId>
25 <artifactId>spring-boot-starter-data-jpa</artifactId>
26 </dependency>
27
28 <dependency>
29 <groupId>mysql</groupId>
30 <artifactId>mysql-connector-java</artifactId>
31 <version>8.0.33</version>
32 </dependency>
33
34 <dependency>
35 <groupId>org.springframework.boot</groupId>
36 <artifactId>spring-boot-starter-web</artifactId>
37 </dependency>
```
- Output Console:** Shows the output of a run command for `Pertemuan5_50421535`. The output includes:

```
2. Tampilkan mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
Mahasiswa{id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0'}
```



The screenshot shows an IDE with several tabs: pom.xml [Pertemuan5_50421535], ModelMahasiswa.java, modelMahasiswa.java, and MahasiswaRepository.java. The pom.xml file is open in the Source view, showing XML configuration for dependencies and build plugins. Below the code editor, the Output window for 'Run (Pertemuan5_50421535)' displays the execution steps and a Hibernate SQL query.

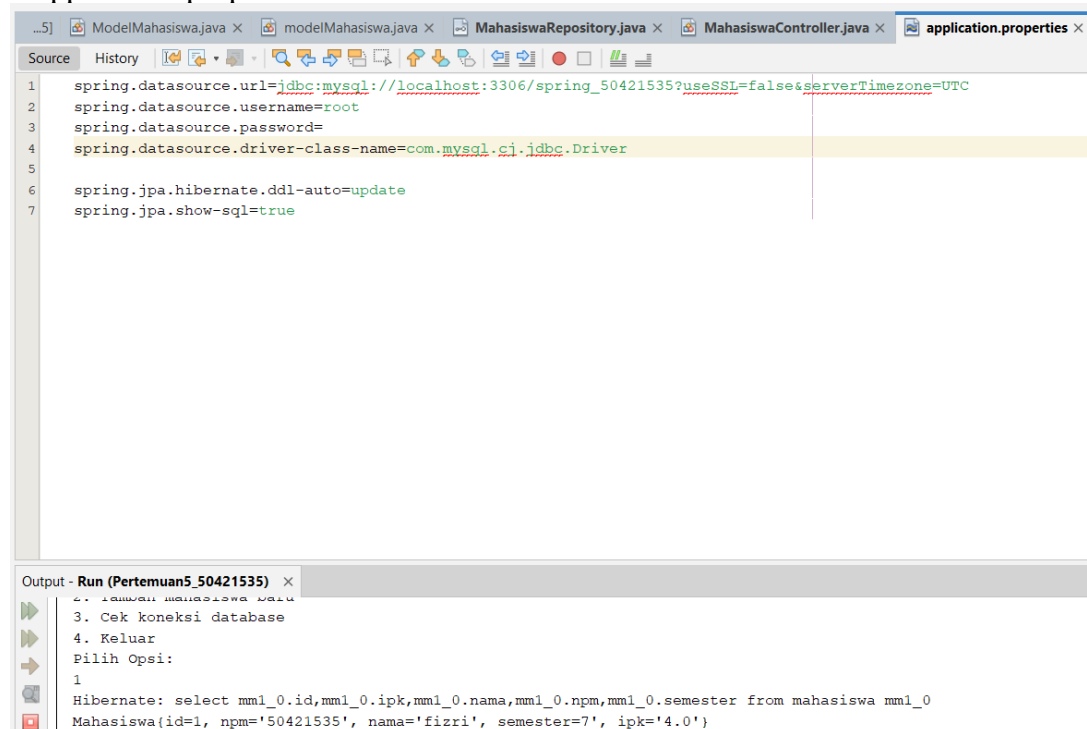
```
33 <dependency>
34 <groupId>org.springframework.boot</groupId>
35 <artifactId>spring-boot-starter-web</artifactId>
36 </dependency>
37
38
39
40 <dependency>
41 <groupId>org.springframework.boot</groupId>
42 <artifactId>spring-boot-starter-test</artifactId>
43 <scope>test</scope>
44 </dependency>
45 </dependencies>
46 <build>
47
48 <plugins>
49 <plugin>
50 <groupId>org.springframework.boot</groupId>
51 <artifactId>spring-boot-maven-plugin</artifactId>
52 </plugin>
53 </plugins>
54 </build>
55
```

Output - Run (Pertemuan5_50421535)

```
2. Masukkan mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
Mahasiswa{id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0'}
```

File pom.xml ini mengonfigurasi proyek Maven untuk mengimpor dependensi yang diperlukan untuk mengembangkan aplikasi Spring Boot dengan dukungan JPA, konektor MySQL, dan pengembangan aplikasi web. Selain itu, juga menyiapkan plugin Maven yang memungkinkan aplikasi untuk dijalankan dengan mudah dan diuji.

F.application.properties



The screenshot shows an IDE with several tabs: ModelMahasiswa.java, modelMahasiswa.java, MahasiswaRepository.java, MahasiswaController.java, and application.properties. The application.properties file is open in the Source view, showing configuration for the Spring application. Below the code editor, the Output window for 'Run (Pertemuan5_50421535)' displays the execution steps and a Hibernate SQL query.

```
1 spring.datasource.url=jdbc:mysql://localhost:3306/spring_50421535?useSSL=false&serverTimezone=UTC
2 spring.datasource.username=root
3 spring.datasource.password=
4 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5
6 spring.jpa.hibernate.ddl-auto=update
7 spring.jpa.show-sql=true
```

Output - Run (Pertemuan5_50421535)

```
2. Masukkan mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
Mahasiswa{id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0'}
```

Konfigurasi ini mengatur aplikasi Spring Boot untuk terhubung ke database MySQL dengan nama spring_50421535 di localhost, menggunakan pengguna root (dengan atau tanpa kata sandi), dan menggunakan Hibernate untuk mengelola skema database. Query SQL yang dieksekusi oleh Hibernate akan ditampilkan di log aplikasi