Praktikum 3

Embed Database Spring

1. Pengantar

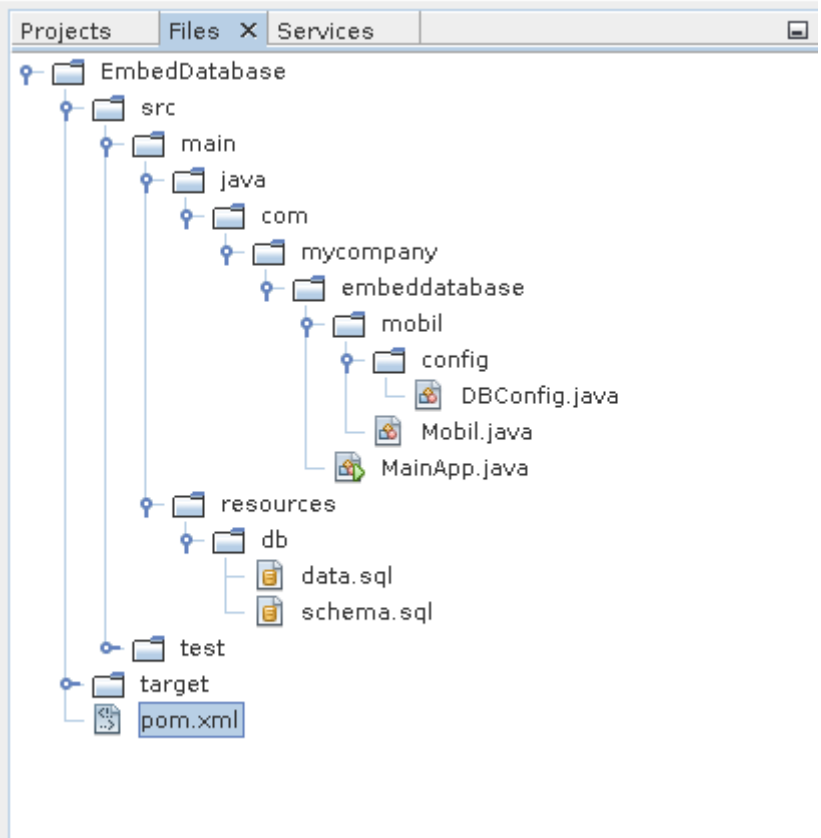
- Embed Database merupakan salah satu teknologi database internal yang mengizinkan aplikasi untuk mengakses database secara internal. Spring menyediakan fitur pengakses database baik secara internal maupun eksternal
- Praktikum ini mengajarkan bagaimana melakukan pengaksesan database secara internal dengan menggunakan Spring Framework. Praktikum ini menggunakan database internal HSQL sebagai objek praktikum, selain itu masih ada beberapa database lainnya yang telah disediakan oleh Spring.
- Mahasiswa diharapkan dapat memahami penggunaan database internal dengan menggunakan Spring Framework. Selain itu mahasiswa dapat menggunakan database internal tersebut untuk menyimpan data dengan mudah dan efektif.

2. Pembahasan

Praktikum ini dimulai dengan pembuatan projek pertama dengan menggunakan NetBeans. Kemudian membuat koneksi database internal dengan membuat memanggil library jdbc yang telah dipersiapkan menggunakan pom.xml. Dengan itu mahasiswa dapat melanjutkan projek dengan membuat kode konfigurasi dan aplikasi utama.

1. Buatlah projek Maven dengan menggunakan NetBeans.

2. Setelah projek siap, buatlah folder serta file dengan struktur (**pastikan tata letak file sudah benar, jika salah maka program dipastikan gagal berjalan**) sebagai berikut:



3. Bukalah pom.xml dan tambahkan kode dengan tulisan tebal berikut ini untuk mengunduh dependency dari DB.

```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <spring-version>5.1.3.RELEASE</spring-version>
</properties>

<dependencies>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <version>1.4.197</version>
    </dependency>

    <dependency>
        <groupId>ch.qos.logback</groupId>
        <artifactId>logback-classic</artifactId>
        <version>1.2.3</version>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>${spring-version}</version>
```

```
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>${spring-version}</version>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jdbc</artifactId>
      <version>${spring-version}</version>
    </dependency>

  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>exec-maven-plugin</artifactId>
        <version>1.6.0</version>
        <configuration>
          <mainClass>com.mycompany.embeddatabase.MainApp</mainClass>
        </configuration>
      </plugin>
    </plugins>
  </build>
```

4. Setelah itu buka file Mobil.java (yang sudah dibuat tadi), dan ketikkan kode berikut ini:

```
import java.util.Objects;


public class Mobil {
    private Long id;
   private String name;
   private int price;

   public Long getId() {
      return id;
   }

   public void setId(Long id) {
      this.id = id;
   }

   public String getName() {
      return name;
```

```
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getPrice() {
        return price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Mobil car = (Mobil) o;
        return price == car.price &&
                Objects.equals(id, car.id) &&
                Objects.equals(name, car.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, name, price);
    }

    @Override
    public String toString() {
        final StringBuilder sb = new StringBuilder("Car{");
        sb.append("id=").append(id);
        sb.append(", name='").append(name).append('\'');
        sb.append(", price=").append(price);
        sb.append('}');
        return sb.toString();
    }
}
```

5. Jika sudah, bukalah file DBConfig.java, dan masukkan kode berikut ini:

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.datasource.embedded.EmbeddedDatabaseBuilder;
import org.springframework.jdbc.datasource.embedded.EmbeddedDatabaseType;

import javax.sql.DataSource;
import org.springframework.jdbc.datasource.embedded.EmbeddedDatabase;
```

```
@Configuration
public class DBConfig {
    @Bean
    public DataSource dataSource() {

        EmbeddedDatabaseBuilder builder = new EmbeddedDatabaseBuilder();
        EmbeddedDatabase db = builder
                .setType(EmbeddedDatabaseType.H2) // HSQL or DERBY
                .addScript("db/schema.sql")
                .addScript("db/data.sql")
                .build();
        return db;
    }

    @Bean
    public JdbcTemplate createJdbcTeamplate() {

        JdbcTemplate template = new JdbcTemplate();
        template.setDataSource(dataSource());

        return template;
    }
}
```

6. Agar projek bisa dieksekusi, ubahlah file MainApp.java dan masukkan kode berikut ini:

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;

import com.mycompany.embeddatabase.mobil.Mobil;
import java.util.List;
import org.springframework.context.ApplicationContext;
/**
 *
 * @author maulana
 */
@ComponentScan(basePackages = "com.mycompany")
public class MainApp {
    private static final Logger logger = LoggerFactory.getLogger(MainApp.class);

    public static void main(String[] args) {
```

```
        ApplicationContext ctx = new AnnotationConfigApplicationContext(MainApp.class);
        MainApp app = ctx.getBean(MainApp.class);

        app.run();
    }

    @Autowired
    private JdbcTemplate jdbcTemplate;

    private void run() {

        String sql = "SELECT * FROM cars";

        List<Mobil> cars = jdbcTemplate.query(sql, new BeanPropertyRowMapper<>(Mobil.class));

        cars.forEach(car -> logger.info("{}", car));
    }
}
```

7. Tahap terakhir yang harus dilakukan adalah mengubah schema.sql dan data.sql seperti kode berikut ini:

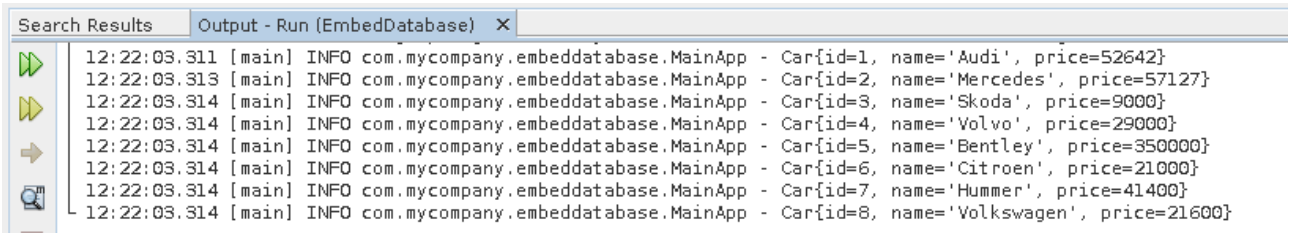
schema.sql

```
CREATE TABLE cars(id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(150), price INT);
```

data.sql

```
INSERT INTO cars(name, price) VALUES('Audi', 52642);
INSERT INTO cars(name, price) VALUES('Mercedes', 57127);
INSERT INTO cars(name, price) VALUES('Skoda', 9000);
INSERT INTO cars(name, price) VALUES('Volvo', 29000);
INSERT INTO cars(name, price) VALUES('Bentley', 350000);
INSERT INTO cars(name, price) VALUES('Citroen', 21000);
INSERT INTO cars(name, price) VALUES('Hummer', 41400);
INSERT INTO cars(name, price) VALUES('Volkswagen', 21600);
```

8. Langkah terakhir ialah Build dan Run Projek. Jika berhasil maka projek akan menampilkan data tersebut sebagai berikut ini:



3. Penugasan

- Kirimlah projek tersebut ke email **maulanahirzan@usm.ac.id** dengan subjek **TugasPraktikum1PFJ19** untuk menghindari penumpukan dari tugas-tugas mahasiswa lainnya. Jangan lupa sertakan nama dan NIM di dalam body email!
- Dikirimkan paling lambat 19 April 2019.
- Kirim projek dalam keadaan zip/rar/7z, sertakan nama dan NIM