LANGKAH-LANGKAH MEMBUAT SPRING BOOT + OTENTIKASI JASON WEB TOKEN (JWT) DAN DATABASE MENGGUNAKAN VS CODE

1. Ikuti modul LANGKAH-LANGKAH MEMBUAT SPRING BOOT + OTENTIKASI JASON WEB TOKEN (JWT) MENGGUNAKAN VS CODE sebelumnya.

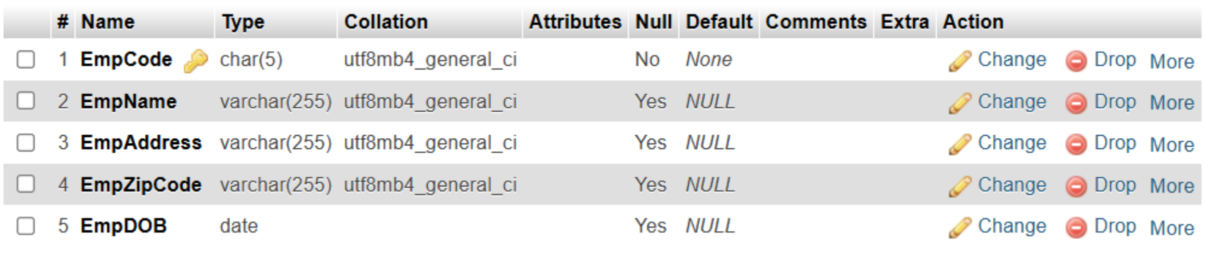
2. Update konfigurasi MySQL Connection dalam application.properties

Dalam src/main/resources/application.properties:

```
spring.application.name=jwtdb
# --- Konfigurasi Server ---
server.port=8080

# --- Konfigurasi MySQL ---
spring.datasource.url=jdbc:mysql://localhost:3306/dbmarket?useSSL=false&ser
verTimezone=UTC
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# --- Konfigurasi JPA/Hibernate ---
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

spring.jpa.hibernate.naming.physical-
strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
spring.jpa.hibernate.naming.implicit-
strategy=org.hibernate.boot.model.naming.ImplicitNamingStrategyLegacyJpaImp
l
```

3. Pastikan memiliki Database (Misal:DBMarket) dan Tabel (Misal:employee)

Contoh struktur tabel:



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	EmpCode	char(5)	utf8mb4_general_ci		No	*None*			Change Drop More
2	EmpName	varchar(255)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
3	EmpAddress	varchar(255)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
4	EmpZipCode	varchar(255)	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
5	EmpDOB	date			Yes	*NULL*			Change Drop More

4. Buatlah package baru bernama entity.

Package digunakan agar pengembang fokus ke database dan JPA yang berada dalam package entity, sedangkan model biarkan fokus pada request / response. Pada contoh ini, kita akan memanfaatkan library lombok. Lombok ini akan meng-generate kode di-compile time, jadi kita tidak perlu nulis boilerplate seperti :

- getter
- setter
- constructor
- toString
- equals & hashCode

Tanpa mengubah behaviour runtime.
Sekarang, buatlah kelas Employee dalam package tersebut

```
package com.rmn.entity;

import jakarta.persistence.*;
import lombok.*;

import java.time.LocalDate;

@Entity
@Table(name = "employee")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Employee {

    @Id
    @Column(name = "EmpCode", length = 5)
    private String empCode;

    @Column(name = "EmpName")
    private String empName;

    @Column(name = "EmpAddress")
    private String empAddress;

    @Column(name = "EmpZipCode")
    private String empZipCode;

    @Column(name = "EmpDOB")
    private LocalDate empDob;
}
```

5. Membuat Repository

Dalam Spring Framework, Repository adalah komponen yang digunakan untuk tujuan utama yaitu mengakses data (persistance) dan mengabstraksi logika interaksi dengan database. Sekarang buatlah folder baru bernama repository dalam package com.rmn dan kelas EmployeeRepository.java lalu masukkan kode berikut:

```
package com.rmn.repository;

import com.rmn.entity.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, String>
{

    List<Employee> findByEmpNameContainingIgnoreCase(String empName);
    List<Employee> findByEmpZipCode(String empZipCode);

    List<Employee> findByEmpNameContainingIgnoreCaseAndEmpZipCode(
            String empName, String empZipCode);
}
```
Spring Data JPA akan auto-generate SQL tanpa query manual.

6. Membuat Service Baru

Saat ini kita sudah memiliki package service yang digunakan untuk mengimplementasikan logika bisnis (Business Logic) dan mengkoordinasikan alur kerja aplikasi. Langkah selanjutnya adalah menambahkan service baru bernama EmployeeService.java lalu masukkan kode berikut:
```
package com.rmn.service;

import com.rmn.entity.Employee;
import com.rmn.repository.EmployeeRepository;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class EmployeeService {

    private final EmployeeRepository repo;

    public EmployeeService(EmployeeRepository repo) {
        this.repo = repo;
    }

    public Employee save(Employee emp) {
        return repo.save(emp);
    }

    public List<Employee> findAll() {
        return repo.findAll();
    }

    public Employee findByCode(String code) {
        return repo.findById(code).orElseThrow();
    }

    public void delete(String code) {
        repo.deleteById(code);
    }

    public List<Employee> search(String name, String zip) {
        if (name != null && zip != null) {
           return repo.findByEmpNameContainingIgnoreCaseAndEmpZipCode(name,
zip);
        }
        if (name != null) {
            return repo.findByEmpNameContainingIgnoreCase(name);
        }
        if (zip != null) {
            return repo.findByEmpZipCode(zip);
        }
        return repo.findAll();
    }
}
```

7. Membuat Controller REST API

Ini adalah komponen yang bertugas menerima permintaan HTTP dari klien dan mengirimkan respons kembali. *Controller* digunakan untuk mengekspos *endpoint* REST.
Buat kelas EmployeeController ke dalam package controller berikut:

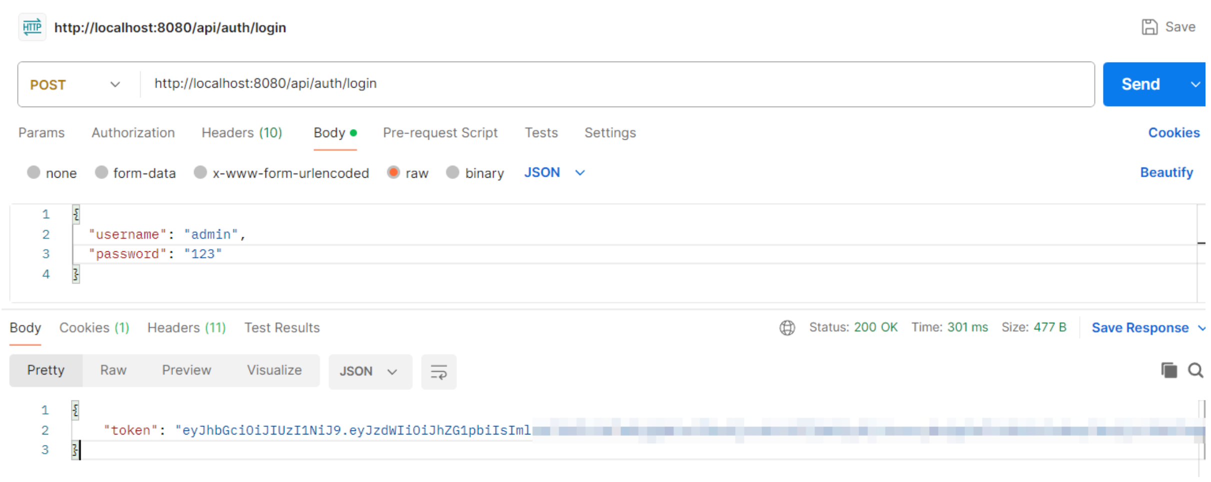
```
package com.rmn.controller;

import com.rmn.entity.Employee;
import com.rmn.service.EmployeeService;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/employees")
public class EmployeeController {

    private final EmployeeService service;

    public EmployeeController(EmployeeService service) {
        this.service = service;
    }

    @PostMapping
    public Employee create(@RequestBody Employee emp) {
        return service.save(emp);
    }

    @GetMapping
    public List<Employee> list(
            @RequestParam(required = false) String name,
            @RequestParam(required = false) String zipcode) {
        return service.search(name, zipcode);
    }

    @GetMapping("/{code}")
    public Employee detail(@PathVariable String code) {
        return service.findByCode(code);
    }

    @PutMapping("/{code}")
    public Employee update(@PathVariable String code,
                           @RequestBody Employee emp) {
        emp.setEmpCode(code);
        return service.save(emp);
    }

    @DeleteMapping("/{code}")
    public void delete(@PathVariable String code) {
        service.delete(code);
    }
}
```

8. Menjalankan Aplikasi

Pada VS Code:

- Gunakan terminal:
  ```
  ./mvnw spring-boot:run
  ```

Check API menggunakan Postman:



http://localhost:8080/api/auth/login

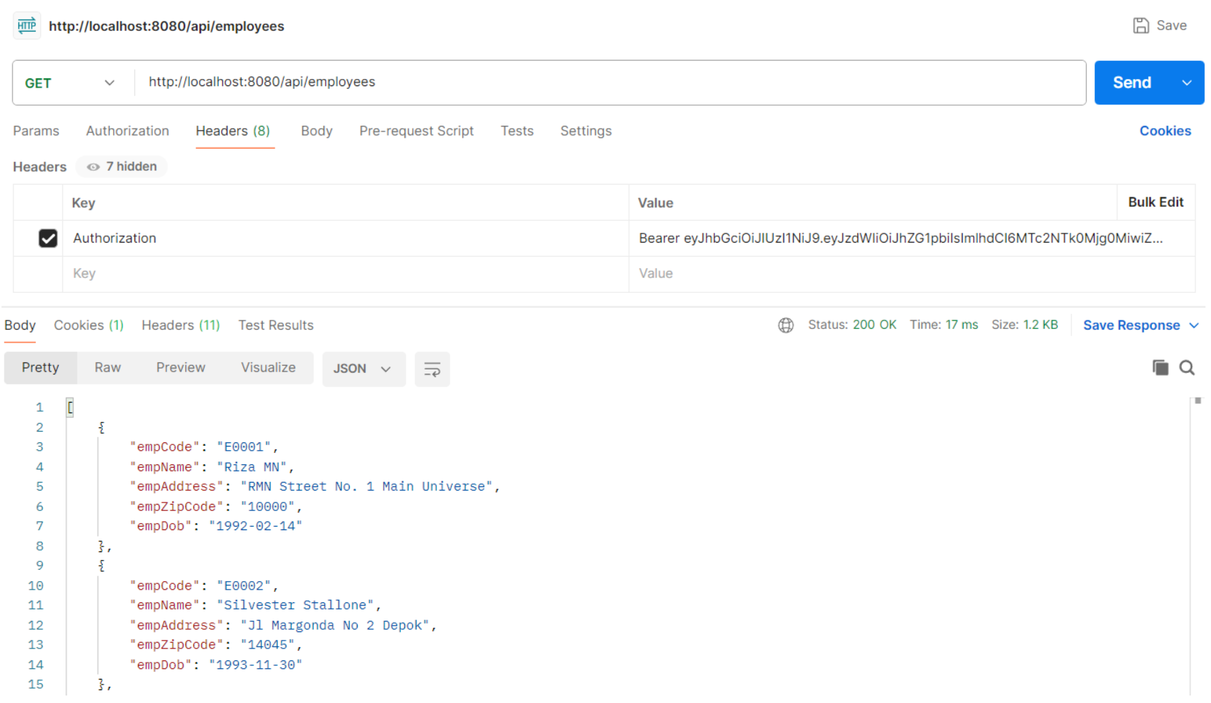
POST http://localhost:8080/api/auth/login

```
{
  "username": "admin",
  "password": "123"
}
```

Status: 200 OK Time: 301 ms Size: 477 B

```
{
    "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbiIsIml[redacted]
}
```

Otentikasi API



http://localhost:8080/api/employees

GET http://localhost:8080/api/employees

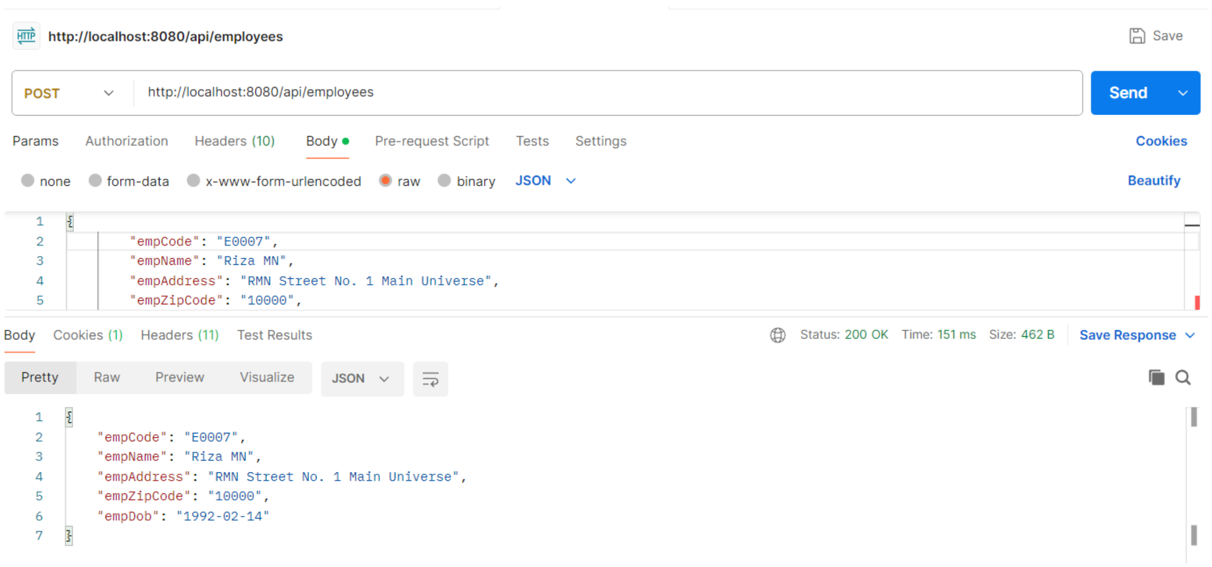
Headers (8)

	Key	Value
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbiIsImlhdCI6MTc2NTk0Mjg0MiwiZ...
	Key	Value

Status: 200 OK Time: 17 ms Size: 1.2 KB

```
[
    {
        "empCode": "E0001",
        "empName": "Riza MN",
        "empAddress": "RMN Street No. 1 Main Universe",
        "empZipCode": "10000",
        "empDob": "1992-02-14"
    },
    {
        "empCode": "E0002",
        "empName": "Silvester Stallone",
        "empAddress": "Jl Margonda No 2 Depok",
        "empZipCode": "14045",
        "empDob": "1993-11-30"
    },
```

Menampilkan semua data Employee



http://localhost:8080/api/employees

POST http://localhost:8080/api/employees

Body: raw JSON

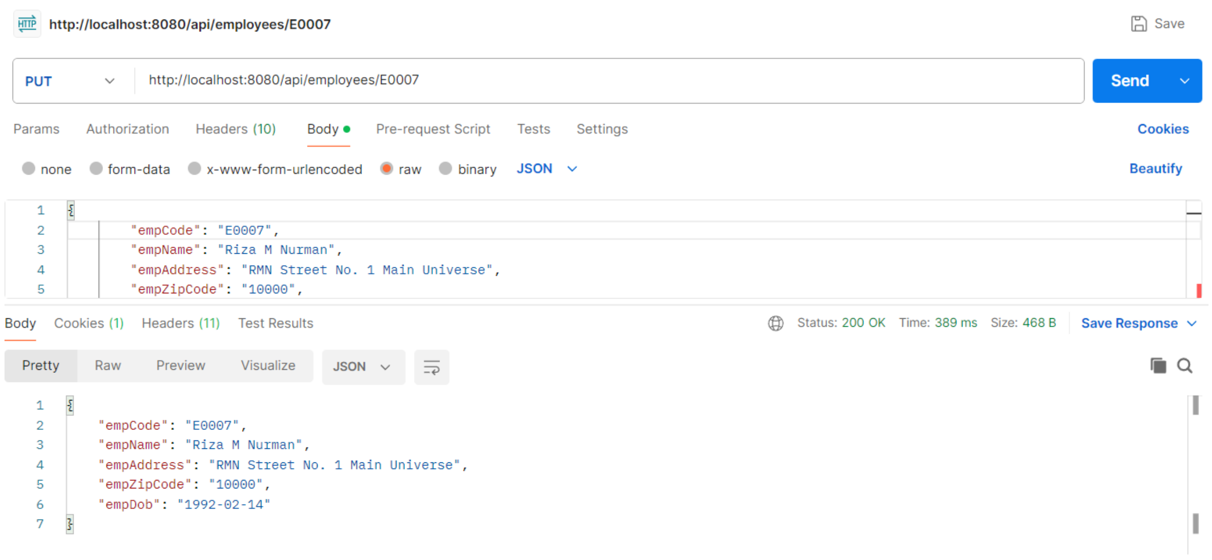
```
{
    "empCode": "E0007",
    "empName": "Riza MN",
    "empAddress": "RMN Street No. 1 Main Universe",
    "empZipCode": "10000",
```

Status: 200 OK Time: 151 ms Size: 462 B

JSON

```
{
    "empCode": "E0007",
    "empName": "Riza MN",
    "empAddress": "RMN Street No. 1 Main Universe",
    "empZipCode": "10000",
    "empDob": "1992-02-14"
}
```

Memasukkan data Employee



http://localhost:8080/api/employees/E0007

PUT http://localhost:8080/api/employees/E0007

Body: raw JSON

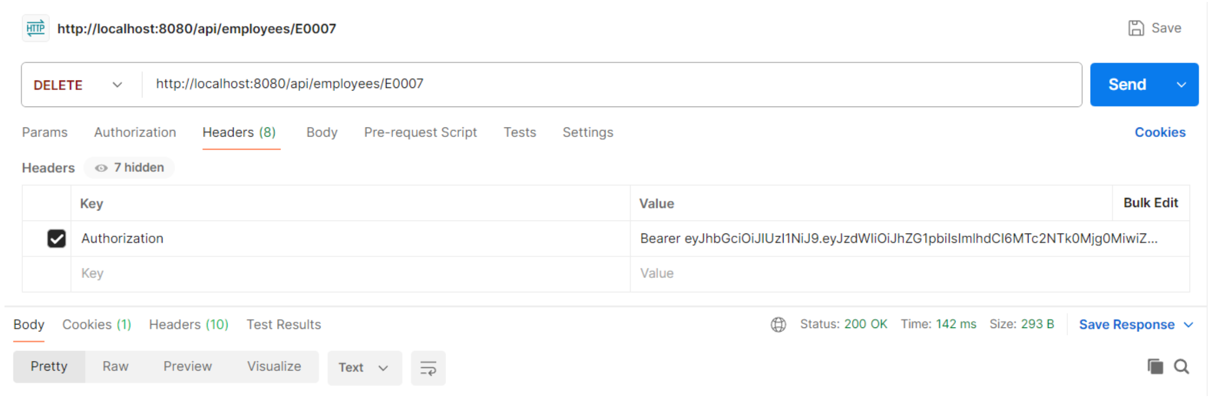
```
{
    "empCode": "E0007",
    "empName": "Riza M Nurman",
    "empAddress": "RMN Street No. 1 Main Universe",
    "empZipCode": "10000",
```

Status: 200 OK Time: 389 ms Size: 468 B

JSON

```
{
    "empCode": "E0007",
    "empName": "Riza M Nurman",
    "empAddress": "RMN Street No. 1 Main Universe",
    "empZipCode": "10000",
    "empDob": "1992-02-14"
}
```

Mengedit data Employee



http://localhost:8080/api/employees/E0007

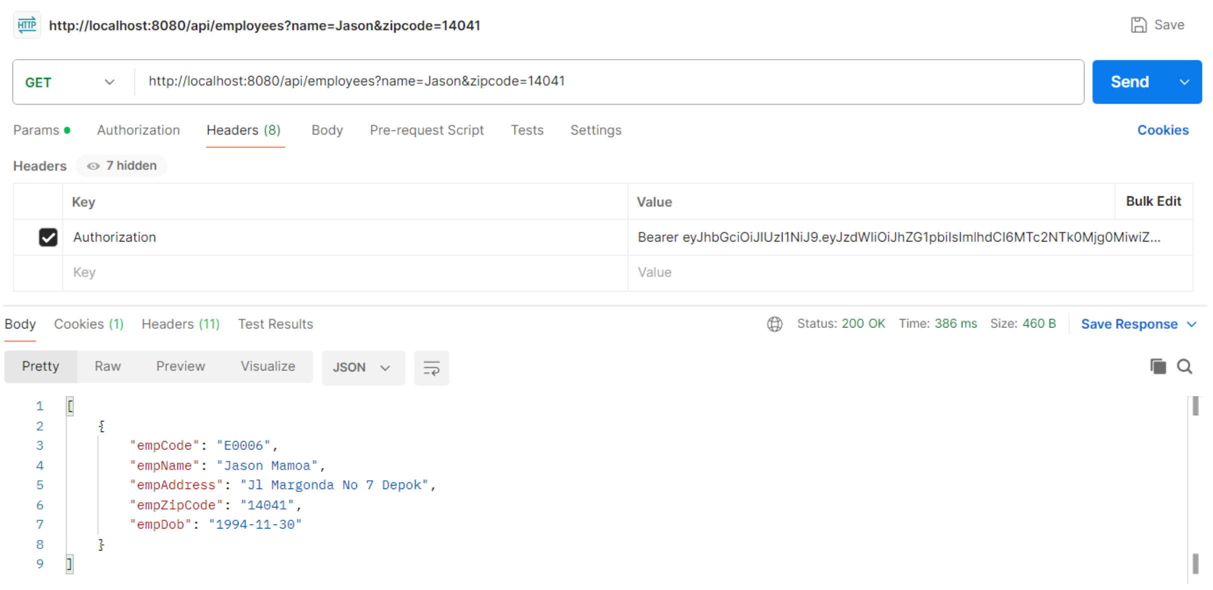
DELETE http://localhost:8080/api/employees/E0007

Headers 7 hidden

	Key	Value
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbiIsImlhdCI6MTc2NTk0Mjg0MiwiZ...
	Key	Value

Status: 200 OK Time: 142 ms Size: 293 B

Menghapus data Employee


http://localhost:8080/api/employees?name=Jason&zipcode=14041

GET http://localhost:8080/api/employees?name=Jason&zipcode=14041

	Key	Value
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbiIsImlhdCI6MTc2NTk0Mjg0MiwiZ...

Status: 200 OK Time: 386 ms Size: 460 B

```
[
    {
        "empCode": "E0006",
        "empName": "Jason Mamoa",
        "empAddress": "Jl Margonda No 7 Depok",
        "empZipCode": "14041",
        "empDob": "1994-11-30"
    }
]
```

Mencari data Employee