บทที่ 4

Database Integration with Spring Data JPA & Hibernate

หัวข้อที่จะเรียนรู้

- พื้นฐาน Relational Database
- แนวคิด ORM (Object Relational Mapping)
- Spring Data JPA & Hibernate
- Entity, Repository, Service
- CRUD Operation กับฐานข้อมูล

Relational Database

- เก็บข้อมูลในรูปแบบ **ตาราง (Table)**
- ใช้ **SQL (Structured Query Language)**
- ความสัมพันธ์ (Relationships)
  - One-to-One
  - One-to-Many
  - Many-to-Many

ORM (Object Relational Mapping)

- แนวคิด: แปลง **Object ↔️ Table**
- ไม่ต้องเขียน SQL ดิบตลอดเวลา
- Hibernate = ORM Framework ที่ Spring Boot ใช้
- ทำงานผ่าน **JPA (Java Persistence API)**

Spring Data JPA

- เป็น abstraction บน JPA/Hibernate
- ลด boilerplate code
- ใช้ **Repository Interface** สำหรับ CRUD
- มี Query Method แบบอัตโนมัติ เช่น
  - `findByEmail(String email)`
  - `findByRole(String role)`

Entity Class

```
@Entity
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String email;
    private String role;
}
```

Repository Interface

```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    List<User> findByRole(String role);
}
```

Service Layer

```
@Service
public class UserService {
    @Autowired
    private UserRepository repo;

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

    public User createUser(User u) {
        return repo.save(u);
    }
}
```

Controller Layer

```
@RestController
@RequestMapping("/users")
public class UserController {
    @Autowired
    private UserService service;

    @GetMapping
    public List<User> getUsers() {
        return service.getAllUsers();
    }

    @PostMapping
    public User create(@RequestBody User u) {
        return service.createUser(u);
    }
}
```

Spring Boot Configuration

application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/mydb
spring.datasource.username=root
spring.datasource.password=1234
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Lab (วันนี้)

1. ติดตั้ง MySQL / PostgreSQL
2. สร้างตาราง `users` (id, name, email, role)
3. เขียน Spring Boot Entity + Repository + Controller
4. ทดสอบ CRUD ผ่าน Postman

Assignment

- สร้าง API `/users` ที่รองรับ:
  - GET `/users` → คืนผู้ใช้ทั้งหมด
  - GET `/users/{id}` → คืนผู้ใช้ตาม id
  - POST `/users` → เพิ่มผู้ใช้ใหม่
  - PUT `/users/{id}` → แก้ไขข้อมูลผู้ใช้
  - DELETE `/users/{id}` → ลบผู้ใช้
- เพิ่ม Query Method `findByRole(String role)`
  - GET `/users?role=admin` → คืนผู้ใช้เฉพาะ role ที่เลือก