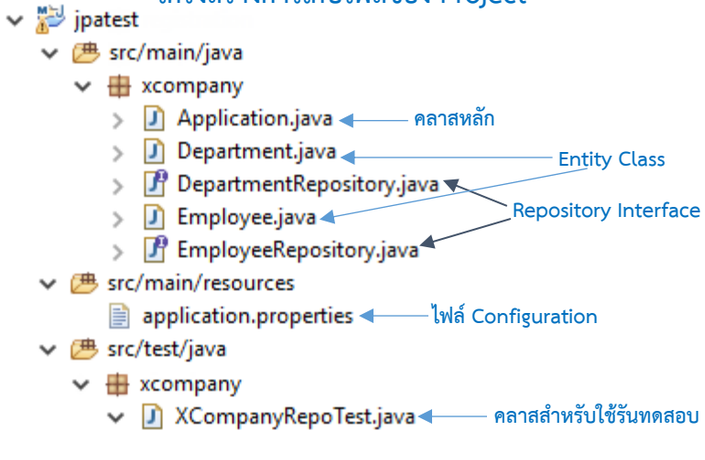
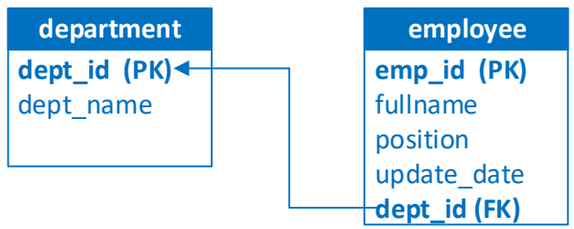
โครงสร้างการเก็บไฟล์ของ Project



โครงสร้างฐานข้อมูล



ขั้นตอนการสร้างส่วนติดต่อฐานข้อมูลด้วย Spring Data JPA

(By...ธีระยุทธ ทองเครือ)

1. กำหนด dependency ที่จะใช้ใน pom.xml
2. สร้างไฟล์ Configuration เพื่อกำหนดค่าเกี่ยวกับการติดต่อฐานข้อมูล
3. สร้าง Entity Class ของแต่ละตาราง
   - ใช้ annotation ระบุคีย์หลัก @Id หากจะกำหนดเป็น Auto Increment ให้เพิ่ม @GeneratedValue(strategy=GenerationType.AUTO)
   - หากมีความสัมพันธ์ ให้อธิบายด้วย เช่น @OneToMany, @ManyToOne
   - ชื่อฟิลด์ที่อยู่ในรูปแบบ Camel case เช่น deptName จะถูกแปลงเป็น dept_name ในฐานข้อมูลอัตโนมัติ
4. สร้าง Repository Interface ของแต่ละ Entity - หากมีการดึงข้อมูลตามเงื่อนไขให้ประกาศเมธอดเพิ่มตามความต้องการ ส่วนเมธอดพื้นฐานไม่จำเป็นต้องประกาศ
5. สร้าง Class หลักของ Spring Boot Application - คลาสที่มีเมธอด main( )
6. สร้าง Class ทดสอบการทำงาน - สร้างเมธอดแยกแต่ละ Test case โดยระบุ @Test ไว้เหนือเมธอด

❶ pom.xml

```xml
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.2.RELEASE</version>
</parent>

<dependencies>
    <!-- ชุด Starter สำหรับ JPA -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <!-- ฐานข้อมูลที่ใช้ -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>
    <!-- ไว้ทดสอบ (UnitTest) -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
    </dependency>
</dependencies>
```

❷ Configuration

application.properties

```
spring.datasource.url = jdbc:mysql://localhost/mydb?characterEncoding=utf-8
spring.datasource.username = root
spring.datasource.password =

spring.jpa.show-sql = true
spring.jpa.hibernate.ddl-auto = update
```

ชื่อฐานข้อมูล

❸ Entity Class

Department.java

```java
@Entity
public class Department {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Integer deptId;
    private String deptName;

    @OneToMany(fetch = FetchType.EAGER, mappedBy = "department")
    private List<Employee> employee;

    // Getter & Setter Method
}
```

Employee.java

```java
@Entity                    // คลาสที่ใช้ import ส่วนใหญ่อยู่ใน javax.persistence.*
public class Employee {
        @Id
        @GeneratedValue(strategy=GenerationType.AUTO)
        private Integer empId;     // ไม่ใช้ควร primitive type เช่น int, float
        private String fullname;
        private String position;
        private Date updateDate;  // ใช้แพ็กเกจ java.util.Date

        @ManyToOne
        @JoinColumn(name="deptId")
        private Department department;

        public Employee(String fullname, String position, Department department,
                        Date updateDate) {
                this.fullname = fullname;
                this.position = position;
                this.department = department;
                this.updateDate = updateDate;
        }

        // ถ้ามี Constructor อื่น จะต้องใส่ Constructor ว่างด้วย
        public Employee() { }

        // Getter & Setter Method
}
```

❹ Repository Interface

DepartmentRepository.java

```java
public interface DepartmentRepository
                extends  CrudRepository<Department, Integer> {
}
```

ชื่อ Entity Class — ชนิดของคีย์หลัก

EmployeeRepository.java

```java
public interface EmployeeRepository
                        extends CrudRepository<Employee, Integer> {

        List<Employee> findByPositionContains(String name);
}
```

เมธอดที่กำหนดเพิ่มเติม

❺ Main Class

Application.java

```java
@SpringBootApplication
public class Application {
        public static void main(String[] args) {
                SpringApplication.run(Application.class, args);
        }
}
```

❻ Test Class

XCompanyRepoTest.java

```java
@RunWith(SpringJUnit4ClassRunner.class)
@SpringBootTest(classes = Application.class)
public class XCompanyRepoTest {

    @Autowired          // สร้าง (Initialize) และผูก (Wiring)  Object ให้อัตโนมัติช่วง runtime
    private DepartmentRepository deptRepo;

    @Autowired
    private EmployeeRepository empRepo;

    @Before @Test      // เมื่อต้องการให้ทดสอบก่อนให้ใส่ @Before
    public void เพิ่มข้อมูลใหม่() {
        Department dept = new Department();
        dept.setDeptName("ฝ่ายพัฒนาระบบงาน");
        Department addedDept = deptRepo.save(dept);
        empRepo.save(new Employee("ฟอร์ดจัง", "Back-end Developer", addedDept, new Date()));
        empRepo.save(new Employee("มอสจัง", "Front-end Developer", addedDept, new Date()));

        dept = new Department();
        dept.setDeptName("ฝ่ายซ่อมบำรุง");
        addedDept = deptRepo.save(dept);
        empRepo.save(new Employee("แอปเปิ้ลจัง", "แม่บ้าน", addedDept, new Date()));
        empRepo.save(new Employee("ช่าจัง", "ช่างแอร์", addedDept, new Date()));
        empRepo.save(new Employee("ผักกาด", "ผู้ช่วยช่างแอร์", addedDept, new Date()));
    }

    @Test
    public void แสดงพนักงานทั้งหมด() {
        for (Employee employee : empRepo.findAll()) {
            System.out.println(employee.getFullname() + ", " + employee.getPosition());
        }
    }

    @Test
    public void ค้นพนักงานตามตำแหน่ง() {
        List<Employee> list = empRepo.findByPositionContains("แอร์");
        for (int i=0; i<list.size(); i++) {
            Employee emp = list.get(i);
            System.out.println(emp.getFullname() + ", " + emp.getPosition());
        }
    }
}
```