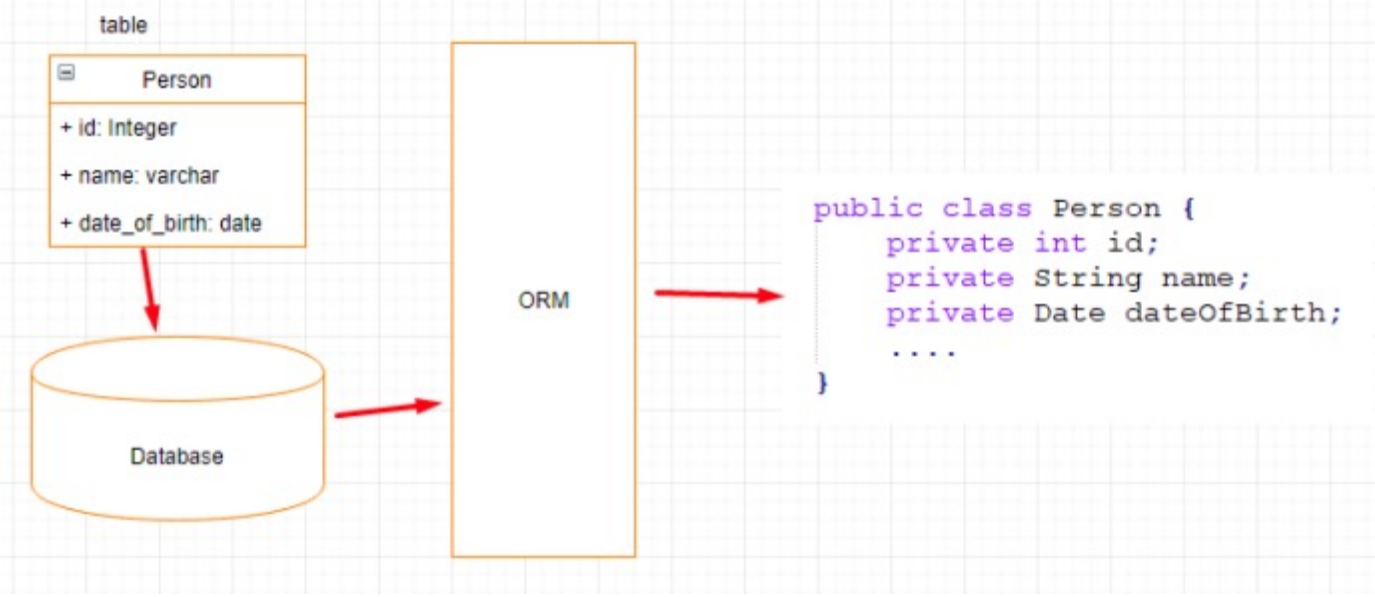
Object Relational Mapping và Spring Data JPA

Ưu nhược điểm của JDBC

- Ưu điểm của JDBC
  - Xử lý trực tiếp câu lệnh SQL
  - Hiệu năng tốt với CSDL lớn
  - Phù hợp với các ứng dụng nhỏ
  - Cú pháp đơn giản, dễ sử dụng
- Nhược điểm của JDBC
  - Phức tạp nếu sử dụng trong các dự án lớn
  - Phát sinh nhiều mã lập trình
  - Không có tính đóng gói
  - Câu truy vấn đặc thù từng loại CSDL

Tại sao cần Object Relational Mapping?

- Trong lập trình HĐT, có thể có sự không khớp giữa mô hình đối tượng và CSDL
  - Dữ liệu trong CSDL thể hiện dạng bảng
  - Dữ liệu trong hệ thống ở dạng lớp mô hình
- Object Relational Mapping (ORM) là một kỹ thuật chuyển đổi giữa các đối tượng dữ liệu trong CSDL và chương trình

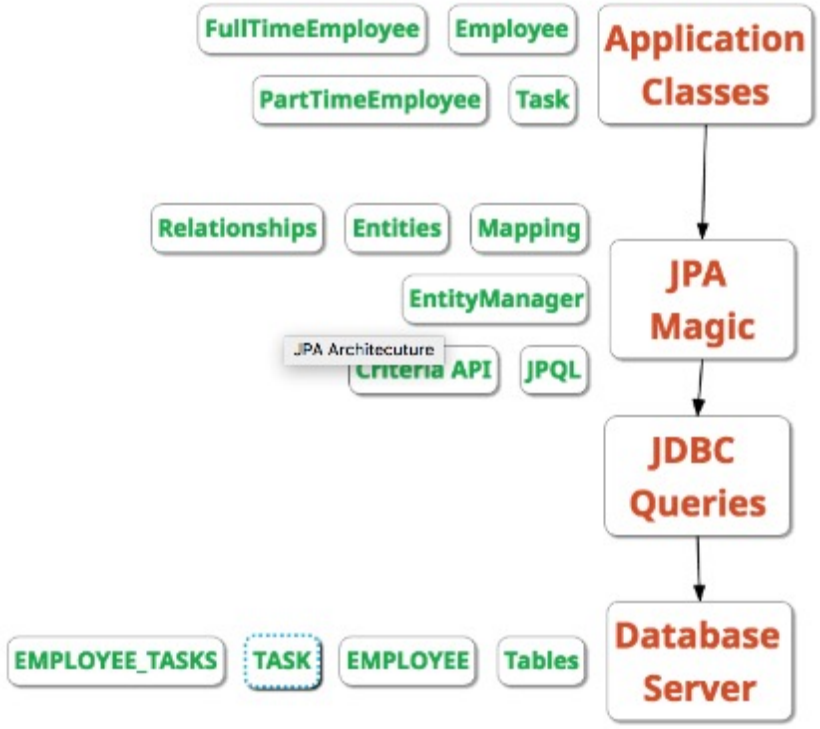


Ưu nhược điểm của ORM

- Ưu điểm của ORM so với JDBC:
  - Ít mã chương trình hơn
  - Ẩn bớt các chi tiết của truy vấn SQL
  - Cơ bản vẫn dựa trên JDBC
  - Các thực thể dựa trên dữ liệu nghiệp vụ thay vì cấu trúc CSDL
  - Giúp phát triển ứng dụng nhanh hơn
- Nhược điểm của ORM so với JDBC:
  - Khả năng truy vấn hạn chế (nhiều khi vẫn phải dùng native SQL)
  - Khó tối ưu câu lệnh SQL (do ORM tự sinh ra)
- Một số Java ORM framework phổ biến :
  - Hibernate
  - EclipseLink
  - Spring DAO ...

Java Persistence API (JPA)

- Là đặc tả ORM API của Java, mô tả:
  - Cách định nghĩa các thực thể (Entity)
  - Cách ánh xạ các thuộc tính
  - Cách ánh xạ các quan hệ CSDL
- Các framework như Hibernate ... là 1 thực thi JPA



Thao tác với CSDL thông qua Spring JPA

- Spring hỗ trợ việc ứng dụng ORM thông qua thành phần Spring Data JPA .

- Spring Data JPA có khả năng tự động tạo các lớp repositories dựa trên các giao diện đặc tả repositories.

- Thêm thư viện vào file pom/xml: Lưu ý thư viện này mặc định bổ sung Hibernate như là bản thực thi JPA trong Spring. Nếu muốn có thể bỏ Hibernate và dùng thư viện khác (tuy nhiên có thể phải thay đổi cách chú giải).

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Bổ sung chú giải JPA cho các lớp mô hình:

- Lớp Ingredient:

```
package tacos;
import javax.persistence.Entity;
import javax.persistence.Id;
import lombok.NoArgsConstructor;
import lombok.RequiredArgsConstructor;
import lombok.AccessLevel;
…
@NoArgsConstructor(access=AccessLevel.PRIVATE, force=true)
@Entity
public class Ingredient {
  @Id
  private final String id;
  private final String name;
  @Enumerated(EnumType.STRING)
  private final Type type;
  public static enum Type {
    WRAP, PROTEIN, VEGGIES, CHEESE, SAUCE
  }
}
```

Bổ sung chú giải JPA cho các lớp mô hình:

- Chú giải @Entity: Đánh dấu lớp thực thể (trùng tên bảng trong CSDL)
- Chú giải @Id: Đánh dấu thuộc tính khoá
- Chú giải @Enumerated đánh dấu kiểu Enum là String
- JPA yêu cầu lớp thực thể có hàm dựng không tham số. Do vậy cần dùng chú giải NoArgsConstructor của Lombok (force=true để khởi tạo giá trị null cho các biến final).

Bổ sung chú giải JPA cho các lớp mô hình:

- Lớp Taco:

```
@Data
@Entity
public class Taco {
  @Id
  @GeneratedValue(strategy=GenerationType.IDENTITY)
  private Long id;
  @NotNull
  @Size(min=5, message="Name must be at least 5 characters long")
  private String name;
  private Date createdAt;
  @ManyToMany(targetEntity=Ingredient.class)
  @Size(min=1, message="You must choose at least 1 ingredient")
  private List<Ingredient> ingredients;
  @PrePersist
  void createdAt() {
    this.createdAt = new Date();
  }
}
```

Bổ sung chú giải JPA cho các lớp mô hình:

- Thiết lập mã tự động tăng với chú giải @GeneratedValue
- Để thiết lập quan hệ với lớp Ingredient, sử dụng chú giải @ManyToMany.
- Chú giải @PrePersist dùng để thiết lập giá trị cho trường createdAt trước khi lưu thông tin của đối tượng (= ngày hiện tại).

Bổ sung chú giải JPA cho các lớp mô hình:

- Lớp Order: Lưu ý chú giải @Table ấn định tên bảng khác (do CSDL không cho phép tên bảng là Order)

```
@Entity
@Table(name="Taco_Order")
public class Order implements Serializable {
  private static final long serialVersionUID = 1L;
  @Id
  @GeneratedValue(strategy=GenerationType.IDENTITY)
  private Long id;
  private Date placedAt;
  ...
  @ManyToMany(targetEntity=Taco.class)
  private List<Taco> tacos = new ArrayList<>();
  public void addDesign(Taco design){this.tacos.add(design);}
  @PrePersist
  void placedAt() {
    this.placedAt = new Date();
  }
}
```

Khai báo các lớp Repositories

- Kế thừa giao diện CrudRepository: Giao diện này khai báo các phương thức thực hiện các thao tác cơ bản như create, read, update, delete ...
- Không cần viết các lớp thực thi, vì SpringBoot sẽ tự động tạo các đối tượng thực thi cho các repositories
- Cần có 2 tham số là kiểu thực thể và kiểu của thuộc tính khoá.

```
package tacos.data;
import
org.springframework.data.repository.CrudRepository;
import tacos.Ingredient;
public interface IngredientRepository
         extends CrudRepository<Ingredient, String> {
}
```

Gắn vào các lớp Controllers

- Cuối cùng, chỉ cần gắn các repositories vào các lớp controller:

```
public class DesignTacoController {
   private final IngredientRepository ingredientRepo;
   private final TacoRepository tacoRepo;
   @Autowired
   public DesignTacoController(IngredientRepository ingredientRepo,
            TacoRepository   tacoRepo) {
      this.ingredientRepo = ingredientRepo;
      this.tacoRepo = tacoRepo;
   }
    @ModelAttribute
    public void addIngredientsToModel(Model model) {
     List<Ingredient> ingredients =
     (List<Ingredient>)ingredientRepo.findAll();
     ...
     @PostMapping
     public String processDesign(Taco taco) {
             // Save the taco design...
             tacoRepo.save(taco);
             return "redirect:/orders/current";
     }
```

Gắn vào các lớp Controllers

```
public class OrderController {
        private final OrderRepository orderRepo;
        @Autowired
        public OrderController(OrderRepository orderRepo) {
                this.orderRepo = orderRepo;
        }

        ...

        @PostMapping
        public String processOrder(Order order) {
                orderRepo.save(order);
                return "redirect:/";
        }
```

Tạo CSDL

- Tạo tự động: Sử dụng các thiết lập sau trong file application.properties, hibernate sẽ tự động tạo bảng theo tên lớp mô hình và các thuộc tính của lớp

```
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto=update
```

  - Các chế độ ddl-auto: create | update | none
- Tạo thủ công hoặc dùng script
  - Mặc định tên lớp sẽ trùng tên bảng, tên thuộc tính sẽ trùng tên trường trong CSDL. Nếu muốn khác tên cần thêm chú giải @Table(tên_bảng) hoặc @Field(tên_trường)
  - Ngoài ra có một số quy tắc khác về tự động chuyển đổi tên

Tạo thủ công

- Thay đổi CSDL cho phù hợp quy tắc của Hibernate

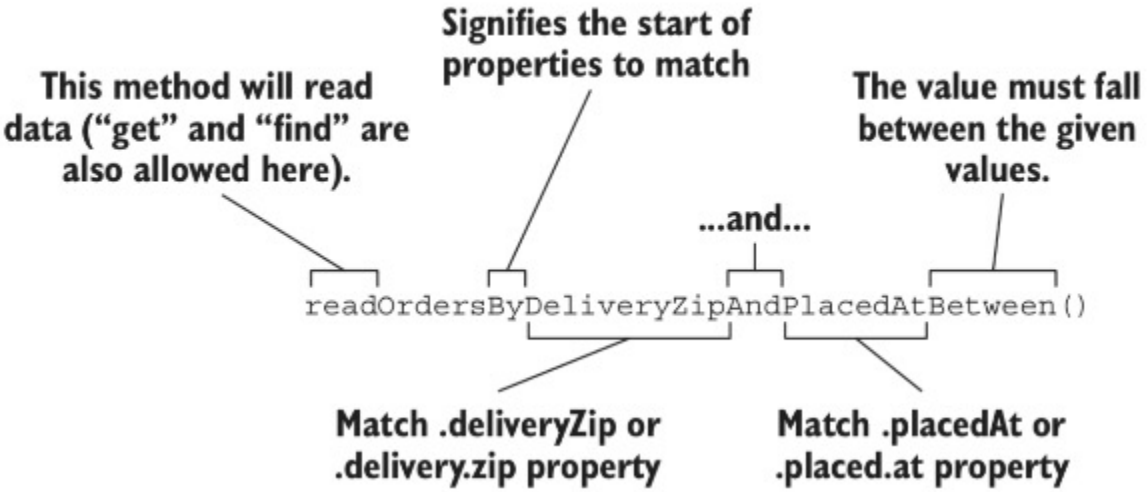
```
create table if not exists Ingredient (
  id varchar(4) not null,
  name varchar(25) not null,
  type varchar(10) not null,
  PRIMARY KEY (id)
);
create table if not exists Taco (
  id int NOT NULL AUTO_INCREMENT,
  name varchar(50) not null,
  created_at timestamp not null,
  PRIMARY KEY (id)
);
create table if not exists Taco_Ingredients (
  taco_id int not null,
  ingredients_id varchar(4) not null,
  FOREIGN KEY (taco_id) REFERENCES Taco(id),
  FOREIGN KEY (ingredients_id) REFERENCES Ingredient(id)
);
```

Tạo thủ công

```
create table if not exists Taco_Order (
    id int NOT NULL AUTO_INCREMENT,
    name varchar(50) not null,
    street varchar(50) not null,
    city varchar(50) not null,
    state varchar(2) not null,
    zip varchar(10) not null,
    cc_number varchar(16) not null,
    cc_expiration varchar(5) not null,
    cccvv varchar(3) not null,
    placed_at timestamp not null,
    PRIMARY KEY (id)
);
create table if not exists Taco_Order_Tacos (
  tacoOrder_id int not null,
  taco_id int not null,
  FOREIGN KEY (tacoOrder_id) REFERENCES Taco_Order(id),
  FOREIGN KEY (taco_id) REFERENCES Taco(id)
);
```

Cải tiến các CRUD Repository

- Có thể bổ sung thêm các phương thức cho Repository bằng cách đặt tên phương thức theo một số từ khoá. Spring sẽ tự động đoán hành động của phương thức theo tên, VD:
  - `List<Order> findByDeliveryZip(String deliveryZip);`
  - `List<Order> readOrdersByDeliveryZipAndPlacedAtBetween( String deliveryZip, Date startDate, Date endDate);`
  - `List<Order> findByDeliveryToAndDeliveryCityAllIgnoresCase( String deliveryTo, String deliveryCity);`



Cải tiến các CRUD Repository

- Ngoài ra, có thể đặt tên bất kỳ và dùng chú giải @Query(). Lưu ý mặc định dùng JPQL, muốn sử dụng SQL cho nativeQuery=true. VD:

```
@Query(value=”select * from Order o where o.deliveryCity='Seattle'”,
nativeQuery=true)
List<Order> readOrdersDeliveredInSeattle();
```

- Bên cạnh CRUDRepository có thể dùng JpaRepository để sử dụng thêm các tính năng đặc thù của JPA.

Do them yourself ...

- Bổ sung các phương thức cần thiết cho thao tác với các thực thể Ingredient, Taco, Order.
- Viết các chức năng để thêm/sửa/xoá/tìm kiếm thành phần Ingredient sử dụng JPA.
- Viết chức năng để xem các đơn hàng đã đặt và lọc theo ngày.