

# Thao tác CSDL với Spring Data

# Thao tác CSDL thông qua Spring JDBC

- Lớp JdbcTemplate:
  - Cung cấp phương tiện để thực hiện thao tác với CSDL thuận tiện hơn.
- Phương pháp truyền thống: Tạo Connection, Statment, Query, Update ...
- Sử dụng JdbcTemplate:

```
private JdbcTemplate jdbc;  
  
@Override  
public Ingredient findOne(String id) {  
    return jdbc.queryForObject(  
        "select id, name, type from Ingredient where id=?",  
        this::mapRowToIngredient, id);  
}  
  
private Ingredient mapRowToIngredient(ResultSet rs, int rowNum)  
    throws SQLException {  
    return new Ingredient(  
        rs.getString("id"),  
        rs.getString("name"),  
        Ingredient.Type.valueOf(rs.getString("type")));  
}
```

## Điều chỉnh các lớp mô hình

- CSDL có trường khoá -> Các lớp mô hình bổ sung thuộc tính Id (nếu chưa có).
- Bổ sung thuộc tính createdAt và placedAt cho các lớp Taco và Order.

```
@Data
public class Taco {
    private Long id;
    private Date createdAt;
    ...
}
```

```
@Data
public class Order {
    private Long id;
    private Date placedAt;
    ...
}
```

## Làm việc với JdbcTemplate

- Bổ sung thêm thư viện vào dự án (file pom.xml)

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-jdbc</artifactId>
```

```
</dependency>
```

- Thiết lập cấu hình CSDL để sử dụng: Sử dụng CSDL MySQL.

```
<dependency>
```

```
    <groupId>mysql</groupId>
```

```
    <artifactId>mysql-connector-java</artifactId>
```

```
    <scope>runtime</scope>
```

```
</dependency>
```

# Định nghĩa các Repositories

- Repository thực hiện các công việc:
  - Truy vấn lấy tất cả các thành phần từ CSDL vào 1 tập hợp các đối tượng Ingredient.
  - Truy vấn lấy 1 thành phần duy nhất theo Id.
  - Lưu thông tin 1 thành phần vào CSDL.

```
package tacos.data;
```

```
import tacos.Ingredient;
```

```
public interface IngredientRepository {  
    Iterable<Ingredient> findAll();  
    Ingredient findById(String id);  
    Ingredient save(Ingredient ingredient);  
}
```

## Lớp thực thi giao diện

- Sử dụng JdbcTemplate để thao tác CSDL:

```
package tacos.data;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.stereotype.Repository;
import tacos.Ingredient;

@Repository
public class JdbcIngredientRepository
    implements IngredientRepository {
    private JdbcTemplate jdbc;
    @Autowired
    public JdbcIngredientRepository(JdbcTemplate jdbc) {
        this.jdbc = jdbc;
    }
    ...
}
```

## Lớp thực thi giao diện

- Chú giải `@Repository`: Đánh dấu lớp nhằm báo cho Spring container biết và khởi tạo nó như 1 bean trong hệ thống.
- Khi Spring tạo bean `IngredientRepository`, Spring sẽ tạo và gắn đối tượng `JdbcTemplate` vào nó bằng chú giải `@Autowired` cho hàm dựng.
- Tiếp tục thực thi các phương thức `findAll()` và `findById()` để truy vấn dữ liệu.

# Lớp thực thi giao diện

```
@Override
public Iterable<Ingredient> findAll() {
    return jdbc.query("select id, name, type from Ingredient",
        this::mapRowToIngredient);
}

@Override
public Ingredient findById(String id) {
    return jdbc.queryForObject(
        "select id, name, type from Ingredient where id=?",
        this::mapRowToIngredient, id);
}

private Ingredient mapRowToIngredient(ResultSet rs, int rowNum)
    throws SQLException {
    return new Ingredient(
        rs.getString("id"),
        rs.getString("name"),
        Ingredient.Type.valueOf(rs.getString("type")));
}
```



## Lớp thực thi giao diện

- findAll() findById() làm việc với JdbcTemplate theo cách tương tự.
- findAll():
  - Trả về 1 danh sách đối tượng.
  - Sử dụng phương thức query() của JdbcTemplate.
  - Chấp nhận câu truy vấn SQL và hàm thực thi RowMapper của Spring để chuyển đổi các hàng trong CSDL sang đối tượng.
  - Có thể nhận tham số, nhưng chưa cần trong ví dụ này.
- findById:
  - Trả về 1 đối tượng.
  - Sử dụng queryForObject() thay vì query().
  - Tham số là câu lệnh truy vấn, hàm thực thi RowMapper, Id của đối tượng cần tìm (thay thế vị trí ? trong truy vấn)
- Trong cả 2 trường hợp, hàm thực thi RowMapper đều là hàm mapRowToIngredient().

## Lớp thực thi giao diện

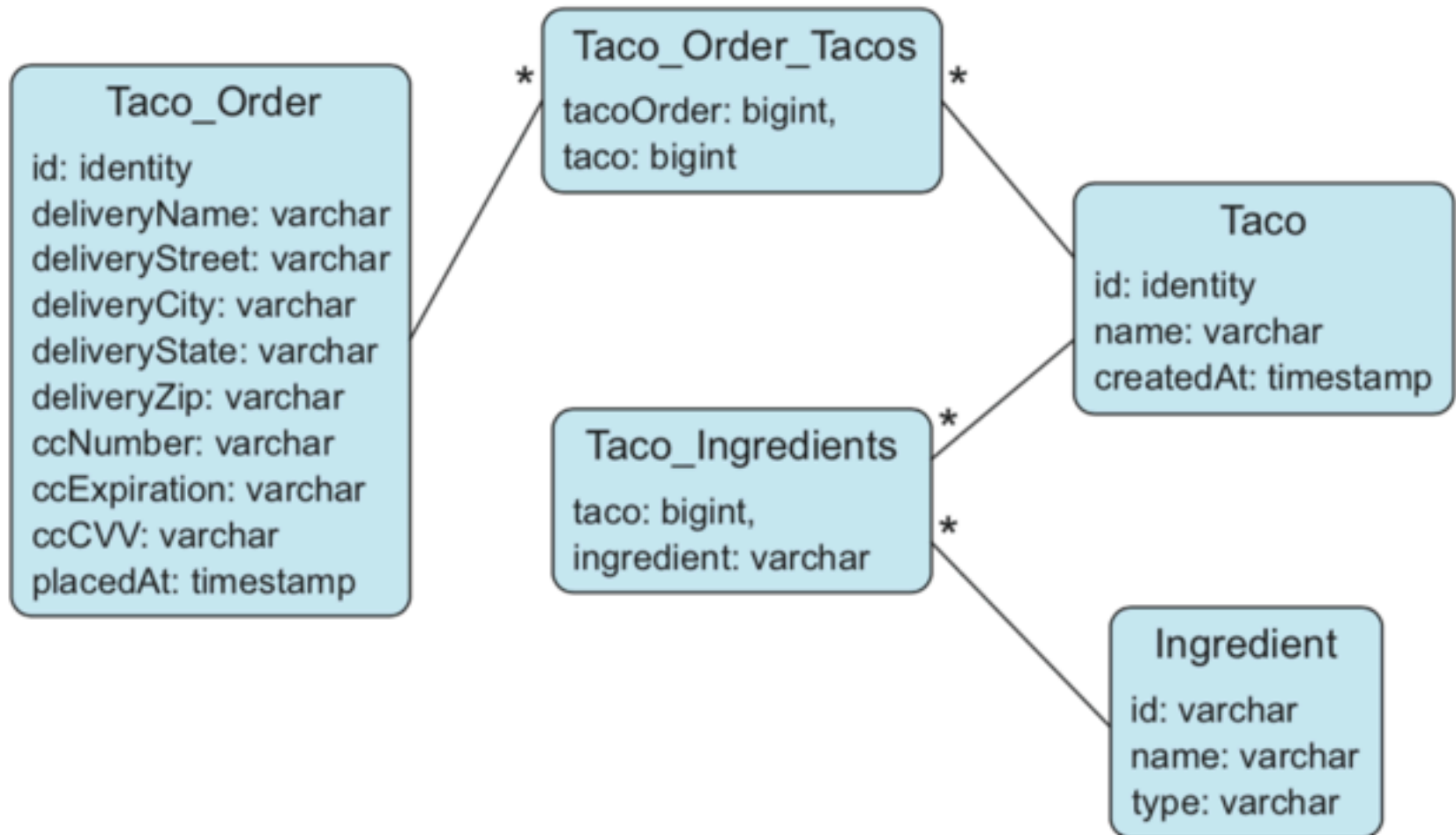
- Để cập nhật dữ liệu, sử dụng phương thức update() của JdbcTemplate.
  - Chỉ cần 1 câu lệnh + các tham số.
  - Việc cập nhật dữ liệu cho các bảng có khoá ngoại sẽ phức tạp hơn. Sẽ tìm hiểu khi thực hiện bằng Spring JPA.

```
@Override
public Ingredient save(Ingredient ingredient) {
    jdbc.update(
        "insert into Ingredient (id, name, type) values
        (?, ?, ?)",
        ingredient.getId(),
        ingredient.getName(),
        ingredient.getType().toString());
    return ingredient;
}
```

# Gắn vào lớp TacoDesignController (sửa phần bôi đỏ)

```
@Controller
@RequestMapping("/design")
@SessionAttributes("order")
public class DesignTacoController {
    private final IngredientRepository ingredientRepo;
    @Autowired
    public DesignTacoController(IngredientRepository ingredientRepo) {
        this.ingredientRepo = ingredientRepo;
    }
    @ModelAttribute
    public void addIngredientsToModel(Model model) {
        List<Ingredient> ingredients = new ArrayList<>();
        ingredientRepo.findAll().forEach(ingredients::add);
        Type[] types = Ingredient.Type.values();
        for (Type type : types) {
            model.addAttribute(type.toString().toLowerCase(),
                filterByType(ingredients, type));
        }
    }
    ...
}
```

# Tạo các bảng trong CSDL và nạp dữ liệu ban đầu



## Script tạo bảng:

- Tạo CSDL tacocloud
- Lưu trong file schema.sql ở thư mục src/main/resource
- Spring sẽ tự động chạy script khi khởi động ứng dụng

```
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,  
    name varchar(50) not null,  
    createdAt timestamp not null,  
    PRIMARY KEY (id)  
);  
  
create table if not exists Taco_Ingredients (  
    taco int not null,  
    ingredient varchar(4) not null,  
    FOREIGN KEY (taco) REFERENCES Taco(id),  
    FOREIGN KEY (ingredient) REFERENCES Ingredient(id)  
);
```

## Script tạo bảng:

```
create table if not exists Taco_Order (  
    id int NOT NULL,  
    deliveryName varchar(50) not null,  
    deliveryStreet varchar(50) not null,  
    deliveryCity varchar(50) not null,  
    deliveryState varchar(2) not null,  
    deliveryZip varchar(10) not null,  
    ccNumber varchar(16) not null,  
    ccExpiration varchar(5) not null,  
    ccCVV varchar(3) not null,  
    placedAt timestamp not null,  
    PRIMARY KEY (id)  
);  
  
create table if not exists Taco_Order_Tacos (  
    tacoOrder int not null,  
    taco int not null,  
    FOREIGN KEY (tacoOrder) REFERENCES Taco_Order(id),  
    FOREIGN KEY (taco) REFERENCES Taco(id)  
);
```

# Script tạo dữ liệu ban đầu:

- Lưu trong file data.sql ở thư mục src/main/resources

```
delete from Taco_Order_Tacos;
delete from Taco_Ingredients;
delete from Taco;
delete from Taco_Order;
delete from Ingredient;
insert into Ingredient (id, name, type)
            values ('FLTO', 'Flour Tortilla', 'WRAP');
insert into Ingredient (id, name, type)
            values ('COTO', 'Corn Tortilla', 'WRAP');
insert into Ingredient (id, name, type)
            values ('GRBF', 'Ground Beef', 'PROTEIN');
insert into Ingredient (id, name, type)
            values ('CARN', 'Carnitas', 'PROTEIN');
insert into Ingredient (id, name, type)
            values ('TMTO', 'Diced Tomatoes', 'VEGGIES');
insert into Ingredient (id, name, type)
            values ('LETC', 'Lettuce', 'VEGGIES');
insert into Ingredient (id, name, type)
            values ('CHED', 'Cheddar', 'CHEESE');
insert into Ingredient (id, name, type)
            values ('JACK', 'Monterrey Jack', 'CHEESE');
insert into Ingredient (id, name, type)
            values ('SLSA', 'Salsa', 'SAUCE');
insert into Ingredient (id, name, type)
            values ('SRCR', 'Sour Cream', 'SAUCE');
```

# Sử dụng CSDL MySQL

- Bổ sung thư viện CSDL vào pom.xml

```
<dependency>  
    <groupId>mysql</groupId>  
    <artifactId>mysql-connector-java</artifactId>  
    <scope>runtime</scope>  
</dependency>
```

- Khai báo cấu hình Data Source trong file application.properties

```
spring.datasource.url=jdbc:mysql://localhost/tacocloud  
spring.datasource.username=.....  
spring.datasource.password=.....  
spring.datasource.initialization-mode=always
```


- SpringBoot sẽ sử dụng các thông tin DataSource để tự động tạo Connection Pool trong Tomcat



# Kết quả khi chạy với các thành phần từ CSDL

localhost

## Design your taco!



**Designate your wrap:**

- ☐ Flour Tortilla
- ☐ Corn Tortilla

**Choose your cheese:**

- ☐ Cheddar
- ☐ Monterrey Jack

**Select your sauce:**

- ☐ Salsa
- ☐ Sour Cream

**Name your taco creation:**

**Pick your protein:**

- ☐ Ground Beef
- ☐ Carnitas

**Determine your veggies:**

- ☐ Diced Tomatoes
- ☐ Lettuce

# Thêm thành phần từ form nhập

- Tạo lớp IngredientController để xử lý việc thêm mới 1 thành phần vào CSDL từ form.

```
package tacos.web;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import lombok.extern.slf4j.Slf4j;
import tacos.data.IngredientRepository;
import tacos.Ingredient;

@Slf4j
@Controller
@RequestMapping("/ingredient")
public class IngredientController {
    private final IngredientRepository ingredientRepo;

    @Autowired
    public IngredientController(IngredientRepository ingredientRepo) {
        this.ingredientRepo = ingredientRepo;
    }
}
```

# Thêm thành phần từ form nhập

```
@GetMapping("/add")
public String showAddForm(Model model) {
    model.addAttribute("ingredient", new Ingredient(null, null, null));
    return "addIngredient";
}

@PostMapping
public String addIngredient(Ingredient ingredient, Model model) {
    ingredientRepo.save(ingredient);
    model.addAttribute(ingredient);
    log.info("Ingredient saved: " + ingredient);
    return "addIngredientSuccess";
}
}
```

## Lớp IngredientController

- Đường dẫn mức lớp là /ingredient.
- Đối tượng IngredientRepository được gắn vào lớp này để thao tác tới bảng Ingredient trong CSDL.
- Phương thức showAddForm() sẽ chuyển tiếp đến trang hiển thị form nhập thành phần mới.
  - Gắn với GetMapping ở đường dẫn mức phương thức /add
  - Tạo một đối tượng Ingredient rỗng chuyển qua form
  - Trả về tên view là addIngredient

## Lớp IngredientController

- Phương thức addIngredient() được gọi khi form được submit. Gắn với method POST và không có đường dẫn mức phương thức (sẽ dùng đường dẫn mức lớp)
  - Phương thức này nhận tham số là đối tượng Ingredient được chuyển tới từ form
  - Thực hiện lưu đối tượng vào CDSL thông qua phương thức save của IngredientRepository.
  - Thực hiện lưu đối tượng ingredient vào Model và chuyển đến trang hiển thị thông tin thêm thành công

# Tạo trang view là form nhập thành phần

- Tạo trang view có tên là addIngredient.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">
  <head>
    <title>Taco Cloud</title>
    <link rel="stylesheet" th:href="@{/styles.css}" />
  </head>
  <body>
    <form method="POST" th:action="@{/ingredient}" th:object="${ingredient}">
      <h1>Add a new Ingredient!</h1>
      <h3>Input the ingredient information ...</h3>
      <table cellpadding="5" border="0">
        <tr>
          <td align="right"><label for="id">ID: </label></td>
          <td align="right"><input type="text" th:field="*{id}" /></td>
        </tr>
```

# Tạo trang view là form nhập thành phần

```
<tr>
  <td align="right"><label for="name">Name: </label></td>
  <td align="right"><input type="text" th:field="**{name}"/></td>
</tr>
<tr>
  <td align="right"><label for="type">Type: </label></td>
  <td align="left">
    <select th:field="**{type}">
      <option th:value=""WRAP"" th:text="WRAP"></option>
      <option th:value=""PROTEIN"" th:text="PROTEIN"></option>
      <option th:value=""VEGGIES"" th:text="VEGGIES"></option>
      <option th:value=""CHEESE"" th:text="CHEESE"></option>
      <option th:value=""SAUCE"" th:text="SAUCE"></option>
    </select>
  </td>
</tr>
```

# Tạo trang view là form nhập thành phần

```
<tr>
  <td></td>
  <td><br><input type="submit" value="Submit"></td>
</tr>
</table>
</form>
</body>
</html>
```

- Lưu ý: Các trường của form sẽ được gắn vào đối tượng ingredient (ẩn định trong thuộc tính th:object của form) và chuyển đi cùng với request khi form được submit.



# Tạo trang view hiển thị nhập thành công

- Tạo trang view có tên là addIngredientSuccess.html

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml"
```

```
    xmlns:th="http://www.thymeleaf.org">
```

```
<head>
```

```
    <title>Taco Cloud</title>
```

```
    <link rel="stylesheet" th:href="@{/styles.css}" />
```

```
</head>
```

```
<body>
```

```
    <h1>Add Ingredient Successfully!</h1>
```

```
    <table cellpadding="5" border="0">
```

```
    <tr>
```

```
        <td align="right"><label for="id">ID: </label></td>
```

```
        <td align="left"><label th:text="{ingredient.id}"></label></td>
```

```
    </tr>
```

# Tạo trang view hiển thị nhập thành công

```
<tr>
```

```
    <td align="right"><label for="name">Name: </label></td>
```

```
    <td align="left"><label th:text="{ingredient.name}"></label></td>
```

```
</tr>
```

```
<tr>
```

```
    <td align="right"><label for="type">Type: </label></td>
```

```
    <td align="left"><label th:text="{ingredient.type}"></label></td>
```

```
</tr>
```

```
</table>
```

```
<p>To enter another Ingredient, click on the Back <br>  
button in your browser or the Return button shown <br>  
below.</p>
```

```
<form action="/ingredient/add" method="get">
```

```
    <input type="submit" value="Return">
```

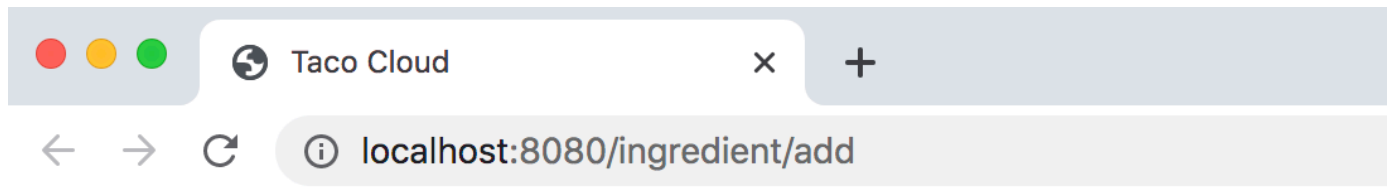
```
</form>
```

```
</body>
```

```
</html>
```

## Thực hiện nhập

- Gõ localhost:8080/ingredient/add vào trình duyệt, form nhập hiện ra:



## Add a new Ingredient!

**Input the ingredient information ...**

ID:

Name:

Type:

## Thực hiện nhập

- Nhập thông tin và bấm Submit, thành phần sẽ được lưu vào CSDL và chuyển đến trang xác nhận

← → ↻ ⓘ localhost:8080/ingredient

# Add Ingredient Successfully!

ID: abc

Name: Test

Type: PROTEIN

To enter another Ingredient, click on the Back button in your browser or the Return button shown below.

Return

## Kiểm tra kết quả

- Quay lại trang design (localhost:8080/design) để kiểm tra đã có thành phần mới (hoặc kiểm tra trong CSDL)

### **Designate your wrap:**

- ☐ Corn Tortilla
- ☐ Flour Tortilla

### **Pick your protein:**

- ☐ Test
- ☐ Carnitas
- ☐ Ground Beef