Hướng dẫn cập nhật dữ liệu cho 2 bảng quan hệ n-n có thêm trường bổ sung

- Với các bảng có quan hệ n-n nhưng bảng tách không có thêm trường bổ sung (chỉ có 2 trường khoá), cập nhập dữ liệu theo cách đã làm trong các Project trước về Spring Data JPA là cách làm hiệu quả do Spring sẽ tự động cập nhật các bảng liên kết.
- Tuy nhiên, trong trường hợp các bảng tách có trường bổ sung, cách làm đó không khả thi khi không cập nhật được trường thông tin bổ sung. Khi đó cần thực hiện như sau (ví dụ thực hiện cho trường hợp quan hệ giữa lớp Ingredient và Taco):

Bước 1: Tạo thêm lớp tách của 2 lớp Ingredient và Taco, đặt tên là TacoIngredient, đồng thời tạo một lớp đại diện cho khoá tổng hợp của nó có tên là TacoIngredientId.

Lớp TacoIngredient

```
package tacos;

import java.io.Serializable;
import java.util.List;

import javax.persistence.EmbeddedId;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.MapsId;
import javax.persistence.Table;

import lombok.*;
import tacos.Ingredient.Type;

@Getter
@Setter
@NoArgsConstructor
@Entity
@Table(name="Taco_Ingredients")
public class TacoIngredient implements Serializable{
 @EmbeddedId
 private TacoIngredientId id;
```

```
    @ManyToOne
    @MapsId("taco_id")
    @JoinColumn
    private Taco taco;

    @ManyToOne
    @MapsId("ingredient_id")
    @JoinColumn
    private Ingredient ingredient;

 public TacoIngredient(Taco taco, Ingredient ingredient) {
       this.taco = taco;
       this.ingredient = ingredient;
       this.id = new TacoIngredientId(taco.getId(), ingredient.getId());
 }
}
```

Lớp TacoIngredientId

```
package tacos;

import java.io.Serializable;
import java.util.Objects;

import javax.persistence.Embeddable;
import lombok.*;

@Embeddable
@Data
@RequiredArgsConstructor
public class TacoIngredientId implements Serializable{
 private Long taco_id;
 private String ingredient_id;

 @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof TacoIngredientId)) return false;
```

```
        TacoIngredientId that = (TacoIngredientId) o;
        return Objects.equals(taco_id, that.getTaco_id()) &&
                Objects.equals(ingredient_id, that.getIngredient_id());
    }

    @Override
    public int hashCode() {
        return Objects.hash(taco_id, ingredient_id);
    }

 public TacoIngredientId(Long taco_id, String ingredient_id) {
       super();
       this.taco_id = taco_id;
       this.ingredient_id = ingredient_id;
 }
}
```

Lưu ý:

- Lớp TacoIngredientId được đánh dấu là @Embeddable và lớp TacoIngredient có 1 thuộc tính khoá kiểu TacoIngredientId được đánh dấu là EmbeddedId
- Lớp TacoIngredient sẽ quan hệ với các lớp Taco và Ingredient theo kiểu One To Many (vì đã tách ra), nên có 2 thuộc tính kiểu Taco và Ingredient được đánh dấu OneToMany.
- Lớp TacoIngredientId phải ghi đè phương thức equals và hashCode và viết thêm 1 số Constructor.

Bước 2: Thay đổi hai lớp Taco và Ingredient như sau:

```
package tacos;

import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.util.List;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
```

```
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.OneToMany;
import javax.persistence.PrePersist;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

import org.hibernate.annotations.GenericGenerator;

import lombok.Data;
import lombok.Getter;
import lombok.Setter;
import tacos.Ingredient.Type;

@Getter
@Setter
@Entity
public class Taco {
 @Id
 @GeneratedValue(strategy= GenerationType.AUTO,generator="native")
 @GenericGenerator(name = "native",strategy = "native")
 private Long id;
 @NotNull
 @Size(min = 5, message = "Name must be at least 5 characters long")
 private String name;
 private Date createdAt;
 @OneToMany(mappedBy = "taco", cascade = CascadeType.ALL)
 private List<TacoIngredient> tacoIngredients = new
ArrayList<TacoIngredient>();

 @PrePersist
 void createdAt() {
      this.createdAt = new Date();
 }

 public void addIngredient(Ingredient ingredient) {
        TacoIngredient tacoIngredient = new TacoIngredient(this,
ingredient);
```

```
        tacoIngredients.add(tacoIngredient);
        ingredient.getTacoIngredients().add(tacoIngredient);
    }
}
```

Lớp Ingredient:

```
package tacos;

import java.util.ArrayList;
import java.util.List;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.Id;
import javax.persistence.OneToMany;

import lombok.Data;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.RequiredArgsConstructor;
import lombok.Setter;
import lombok.AccessLevel;

@Getter
@Setter
@RequiredArgsConstructor
@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
 }
 @OneToMany(mappedBy = "ingredient", cascade = CascadeType.ALL)
    private List<TacoIngredient> tacoIngredients = new
ArrayList<TacoIngredient>();
}
```

Lưu ý 2 lớp này thay đổi thuộc tính liên kết bằng chú giải @OneToMany và lớp Taco cần bổ sung phương thức addIngredient.

Ngoài ra do lỗi trong phương thức toString() của Lombok sẽ dẫn đến lặp vòng khi khởi tạo các đối tượng nên bỏ chú giải @Data thay bằng các chú giải @Setter và @Getter.

Bước 3: Bổ sung thêm lớp TacoIngredientRepository và thay đổi lớp TacoDesignController.

```
package tacos.web;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;

import org.hibernate.Session;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.Errors;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.SessionAttributes;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import lombok.extern.slf4j.Slf4j;
```

```
import tacos.Taco;
import tacos.TacoIngredient;
import tacos.data.IngredientRepository;
import tacos.data.TacoIngredientRepository;
import tacos.data.TacoRepository;
import tacos.Ingredient;
import tacos.Ingredient.Type;

@Slf4j
@Controller
@RequestMapping("/design")
public class DesignTacoController {
 IngredientRepository ingredientRepo;
 private final TacoRepository tacoRepo;
 TacoIngredientRepository tacoIngredientRepo;

 @Autowired
 public DesignTacoController(IngredientRepository ingredientRepo,
TacoRepository tacoRepo, TacoIngredientRepository tacoIngredientRepo) {
       this.ingredientRepo = ingredientRepo;
       this.tacoRepo = tacoRepo;
       this.tacoIngredientRepo = tacoIngredientRepo;
 }

 @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));
       }
 }

 @GetMapping
 public String showDesignForm(Model model) {
       model.addAttribute("taco", new Taco());
```

```
        return "design";
    }

    @PostMapping
    public String processDesign(@RequestParam("ingredients") String
ingredientIds, @RequestParam("name") String name) {
        // Save the taco design...
        Taco taco = new Taco();
        taco.setName(name);
        List<TacoIngredient> tacoIngredients = new
ArrayList<TacoIngredient>();
        for (String ingredientId : ingredientIds.split(",")) {
            Ingredient ingredient =
ingredientRepo.findById(ingredientId).get();
            taco.addIngredient(ingredient);
        }
        tacoRepo.save(taco);
        log.info("Processing design: " + taco);
        return "redirect:/orders/current";
    }

    private List<Ingredient> filterByType(List<Ingredient> ingredients, Type
type) {
        List<Ingredient> ingrList = new ArrayList<Ingredient>();
        for (Ingredient ingredient: ingredients) {
            if (ingredient.getType().equals(type))
ingrList.add(ingredient);
        }
        return ingrList;
    }
}
```

Lưu ý: trường bổ sung của bảng tách có thể được cập nhật trong phương thức @PostMapping trước khi gọi phương thức save để lưu dữ liệu.