

République Tunisienne Ministère
De l'Enseignement Supérieur
Et de la Recherche Scientifique

Université De Sfax

École Nationale d'Électronique
Et des Télécommunications de Sfax



Atelier 3 : SPRING BOOT

Réalisé par : Wissem Karous

Remis à : Mme.Kallel Emna

1/CRÉATION D'UNE ASSOCIATION ONETOMANY ENTRE DEUX ENTITÉS

```

1 package com.gt.produits.entities;
2
3 import com.fasterxml.jackson.annotation.JsonIgnore;
4 import jakarta.persistence.*;
5 import lombok.AllArgsConstructor;
6 import lombok.Data;
7 import lombok.NoArgsConstructor;
8
9 import java.util.List;
10 @Data 12 usages
11 @NoArgsConstructor
12 @AllArgsConstructor
13 @Entity
14 public class Categorie {
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private Long idCat;
18     private String nomCat;
19     private String descriptionCat;
20
21     @JsonIgnore
22     @OneToMany(mappedBy = "categorie")
23     private List<Produit> produits;
24 }
25
```

2/

```

1 package com.gt.produits.entities;
2
3 import jakarta.persistence.*;
4
5 import java.util.Date;
6 @Entity
7 public class Produit {
8     @Id
9     @GeneratedValue(strategy = GenerationType.IDENTITY)
10     private Long idProduit;
11     private String nomProduit; 4 usages
12     private Double prixProduit; 4 usages
13     private Date dateProduits ; 4 usages
14
15     @Contract(pure = true)
16     public Produit() { super(); }
17     @Contract(pure = true)
18     public Produit(String nomProduit, Double prixProduit, Date dateProduits) {
19         super();
20         this.nomProduit = nomProduit;
21         this.prixProduit = prixProduit;
22         this.dateProduits = dateProduits;
23     }
24
25     @ManyToOne 2 usages
26     private Categorie categorie;

```

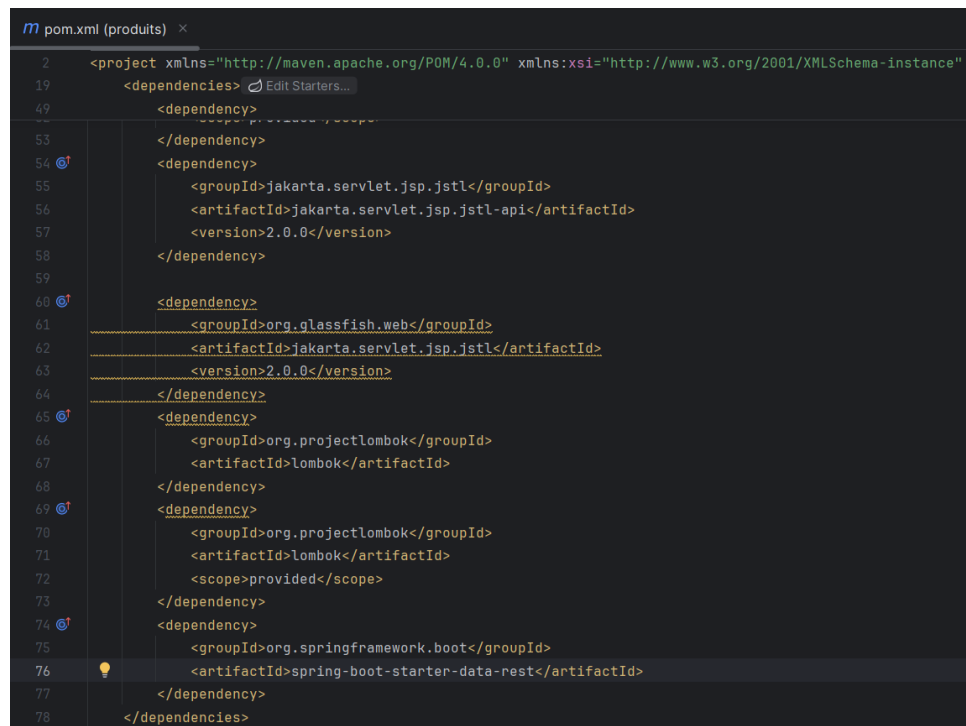
3/ TABLE CATÉGORIE EST CRÉÉ AVEC Succès.

LE CHAMP CATEGORIE_ID_CAT EST AJOUTÉ AVEC SUCCÈS DANS LA TABLE PRODUIT.

```
27
28     public Categorie getCategorie() { no usages
29         return categorie;
30     }
31     public void setCategorie(Categorie categorie) { no usages
32         this.categorie = categorie;
33     }
34
35     public Long getIdProduit() { 1 usage
36
37         return idProduit;
38     }
39     public void setIdProduit(Long idProduit) { no usages
40         this.idProduit = idProduit;
41     }
42
43     public String getNomProduit() { no usages
44
45         return nomProduit;
46     }
47     public void setNomProduit(String nomProduit) { no usages
48
49         this.nomProduit = nomProduit;
50     }
51
52     public Double getPrixProduit() { no usages
53
54         return prixProduit;
55     }
```

```
56     public void setPrixProduit(Double prixProduit) { 1 usage
57
58         this.prixProduit = prixProduit;
59     }
60
61     public Date getDateProduits() { no usages
62
63         return dateProduits;
64     }
65     public void setDateProduits(Date dateProduits) { 1 usage
66
67         this.dateProduits = dateProduits;
68     }
69
70     @Override
71     public String toString() {
72         return "Produit{" +
73             "idProduit=" + idProduit +
74             ", nomProduit=" + nomProduit + '\n' +
75             "prixProduit=" + prixProduit +
76             ", dateProduits=" + dateProduits +
77             '}' ;
78     }
79 }
```

2/ UTILISATION DE LOMBOK



```
m pom.xml (produits) x
2   <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
19   <dependencies>
49     <dependency>
53     </dependency>
54     <dependency>
55       <groupId>jakarta.servlet.jsp.jstl</groupId>
56       <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
57       <version>2.0.0</version>
58     </dependency>
59
60     <dependency>
61       <groupId>org.glassfish.web</groupId>
62       <artifactId>jakarta.servlet.jsp.jstl</artifactId>
63       <version>2.0.0</version>
64     </dependency>
65     <dependency>
66       <groupId>org.projectlombok</groupId>
67       <artifactId>lombok</artifactId>
68     </dependency>
69     <dependency>
70       <groupId>org.projectlombok</groupId>
71       <artifactId>lombok</artifactId>
72       <scope>provided</scope>
73     </dependency>
74     <dependency>
75       <groupId>org.springframework.boot</groupId>
76       <artifactId>spring-boot-starter-data-rest</artifactId>
77     </dependency>
78   </dependencies>
```

MODIFICATION DE LA CLASSE CATEGORIE

```

1 package com.gt.produits.entities;
2
3 import com.fasterxml.jackson.annotation.JsonIgnore;
4 import jakarta.persistence.*;
5 import lombok.AllArgsConstructor;
6 import lombok.Data;
7 import lombok.NoArgsConstructor;
8
9 import java.util.List;
10 @Data 12 usages
11 @NoArgsConstructor
12 @AllArgsConstructor
13 @Entity
14 public class Categorie {
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private Long idCat;
18     private String nomCat;
19     private String descriptionCat;
20
21     @JsonIgnore
22     @OneToMany(mappedBy = "categorie")
23     private List<Produit> produits;
24 }
25
```

INTERROGATION DES ENTITÉS EN FOURNISSANT UN ATTRIBUT NON CLÉ,

```

9 public class ProduitServiceImpl implements ProduitService {
10     @Autowired
11     ProduitsRepository produitRepository;
12     @Override 2 usages
13     public Produit saveProduit(Produit p) {
14         return produitRepository.save(p);
15     }
16     @Override 1 usage
17     public Produit updateProduit(Produit p) { return produitRepository.save(p); }
18     @Override no usages
19     public void deleteProduit(Produit p) { produitRepository.delete(p); }
20     @Override 1 usage
21     public void deleteProduitById(Long id) { produitRepository.deleteById(id); }
22     @Override 1 usage
23     public Produit getProduit(Long id) {
24         return produitRepository.findById(id).get();
25     }
26     @Override 2 usages
27     public List<Produit> getAllProduits() {
28         return produitRepository.findAll();
29     }
30     @Override no usages
31     public List<Produit> findByNameProduit(String nom) {
32         return produitRepository.findByNameProduit(nom);
33     }
34     @Override no usages
35     public List<Produit> findByNamePrix(String nom, Double prix) {
36         return produitRepository.findByNamePrix(nom, prix);
37     }
38     @Override no usages
39     public List<Produit> findByCategorie(Categorie categorie) {
40         return produitRepository.findByCategorie(categorie);
41     }
42 }

```

EXÉCUTION DU TESTE DE LA MÉTHODE FINDBYNOMPRODUIT :

```
ProduitsApplicationTests.java x
13 class ProduitsApplicationTests {
14 }
15
16 @Test
17 public void testfindByNomPrix()
18 {
19     List<Produit> prods = produitRepository.findByNomPrix( nom: "dell", prix: 1000.0);
20     for (Produit p : prods)
21     {
22         System.out.println(p);
23     }
24 }
25
26 @Test
27 public void testfindByCategorie()
28 {
29     Categorie cat = new Categorie();
30     cat.setIdCat(1L); //afficher les produits de catégorie 1
31     List<Produit> prods = produitRepository.findByCategorie(cat);
32     for (Produit p : prods)
33     {
34         System.out.println(p);
35     }
36 }
37
38 @Test
39 public void findByCategorieIdCat()
40 {
41     List<Produit> prods = produitRepository.findByCategorieIdCat(1L);
42     for (Produit p : prods)
43     {
44         System.out.println(p);
45     }
46 }
47 }
```

ECRIRE DES REQUÊTES @QUERY EN UTILISANT LE LANGAGE JPQL

```
ProduitsRepository.java x
1 package com.gt.produits.repos;
2
3 import java.util.List;
4
5 import com.gt.produits.entities.Categorie;
6 import com.gt.produits.entities.Produit;
7 import org.springframework.data.jpa.repository.JpaRepository;
8 import org.springframework.data.jpa.repository.Query;
9 import org.springframework.data.rest.core.annotation.RepositoryRestResource;
10
11 @RepositoryRestResource(path = "rest")
12 public interface ProduitsRepository extends JpaRepository<Produit,Long> {
13     List<Produit> findByNomProduit(String nom); 2 usages
14
15     @Query("select p from Produit p where p.nomProduit like %?1 and p.prixProduit > ?2") 2 usages
16     List<Produit> findByNomPrix (String nom, Double prix);
17
18     @Query("select p from Produit p where p.categorie = ?1") 2 usages
19     List<Produit> findByCategorie (Categorie categorie);
20
21     List<Produit> findByCategorieIdCat(Long id); 2 usages
22 }
```

phpMyAdmin - Server: localhost:3306 - Database: spring_db - Table: produit

Showing rows 0 - 6 (7 total. Query took 0.0004 seconds) [categorie_id_cat: 1... - 2...]

SELECT * FROM 'produit' ORDER BY 'produit`.`categorie_id_cat` ASC

Number of rows: 25 Filter rows: Search this table Sort by key: None

	id_produit	date_produit	nom_produit	prix_produit	categorie_id_cat
<input type="checkbox"/>	1	2023-04-15 09:42:16.259000	PC Dell	2200.5	1
<input type="checkbox"/>	7	2023-04-19 00:00:00.000000	PC lenovo	1500	1
<input type="checkbox"/>	8	2023-04-19 00:00:00.000000	pc lenovo	4000	1
<input type="checkbox"/>	14	2023-04-30 15:43:26.513000	PC Dell	2200.5	1
<input type="checkbox"/>	15	2023-04-30 15:46:40.499000	Asus	2200.4	1
<input type="checkbox"/>	5	2023-04-06 00:00:00.000000	souris	30	2
<input type="checkbox"/>	6	2023-04-05 00:00:00.000000	clavier	50	2

Number of rows: 25 Filter rows: Search this table Sort by key: None

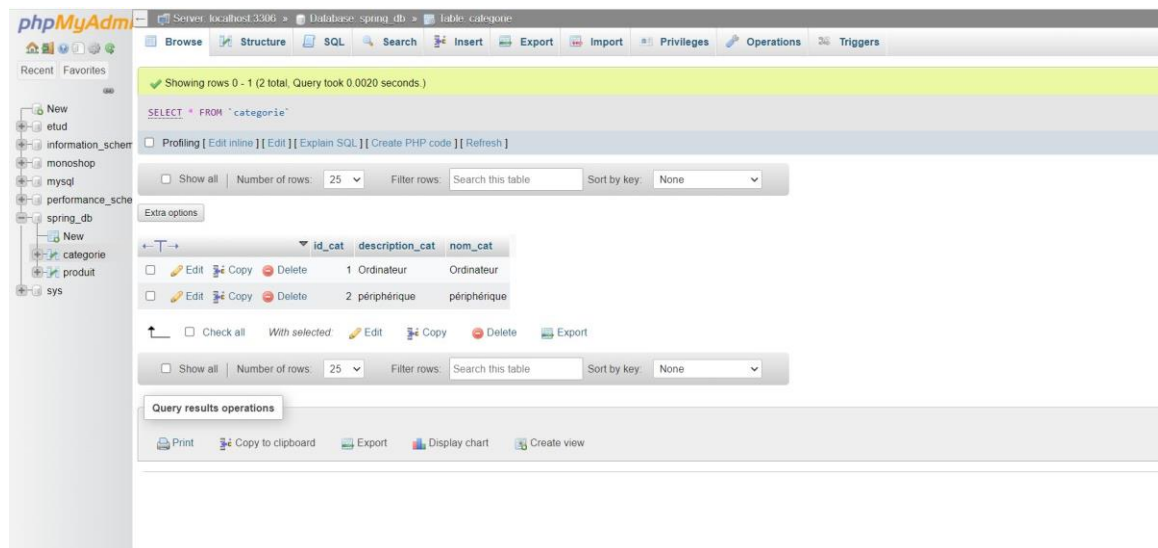
ECRIRE DES REQUÊTES @QUERY EN PASSANT DES ENTITÉS EN PARAMÈTRE

```

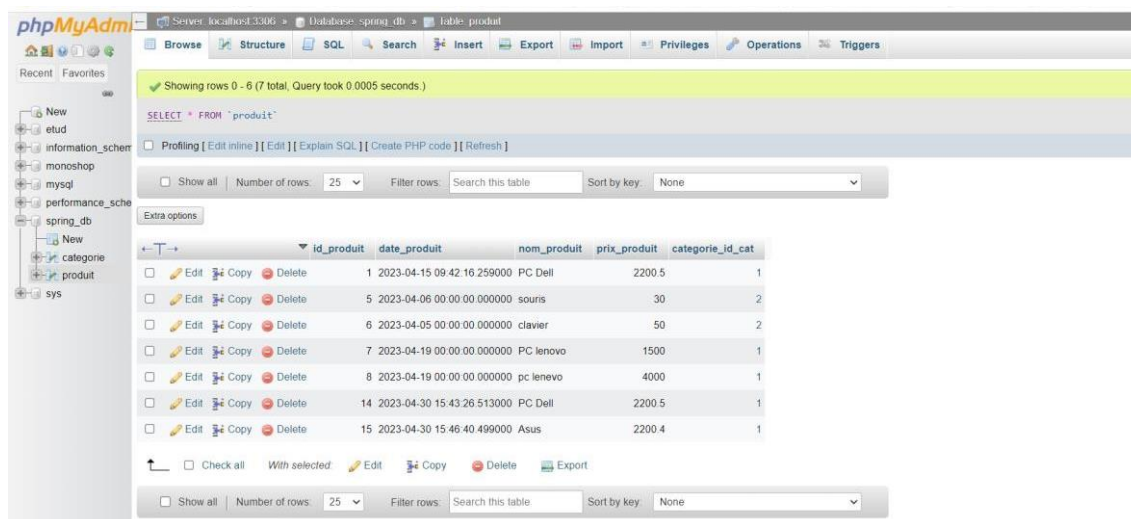
1 package com.gt.produits.service;
2
3 import java.util.List;
4
5 import com.gt.produits.entities.Categorie;
6 import com.gt.produits.entities.Produit;
7
8 public interface ProduitService {
9     Produit saveProduit(Produit p);
10    Produit updateProduit(Produit p);
11    void deleteProduit(Produit p);
12    void deleteProduitById(Long id);
13    Produit getProduit(Long id);
14
15    List<Produit> getAllProduits();
16    List<Produit> findByNomProduit(String nom);
17    List<Produit> findByNomPrix (String nom, Double prix);
18    List<Produit> findByCategorie (Categorie categorie);
19    List<Produit> findByCategorieIdCat(Long id);
20 }

```

INSERTION DE 2 LIGNES DANS LA TABLE CATÉGORIE :



ASSOCIATION D'UNE CATÉGORIE A UN OU PLUSIEURS PRODUITS




```

13 class ProduitsApplicationTests {
53 }
54 @Test
55 public void testfindByNomPrix()
56 {
57     List<Produit> prods = produitRepository.findByNomPrix( nom: "dell", prix: 1000.0);
58     for (Produit p : prods)
59     {
60         System.out.println(p);
61     }
62 }
63 @Test
64 public void testfindByCategorie()
65 {
66     Categorie cat = new Categorie();
67     cat.setIdCat(1L); //afficher les produits de catégorie 1
68     List<Produit> prods = produitRepository.findByCategorie(cat);
69     for (Produit p : prods)
70     {
71         System.out.println(p);
72     }
73 }
74 @Test
75 public void findByCategorieIdCat()
76 {
77     List<Produit> prods = produitRepository.findByCategorieIdCat(1L);
78     for (Produit p : prods)
79     {
80         System.out.println(p);
81     }
82 }

```

INTERROGATION LES PRODUITS SELON L'ID DE LEUR CATÉGORIE

phpMyAdmin - Server: localhost:3306 » Database: spring_db » Table: produit

Showing rows 0 - 6 (7 total, Query took 0.0005 seconds)

SELECT * FROM 'produit'

Number of rows: 25 Filter rows: Search this table Sort by key: None

		id_produit	date_produit	nom_produit	prix_produit	categorie_id_cat
<input type="checkbox"/>	Edit Copy Delete	1	2023-04-15 09:42:16 259000	PC Dell	2200.5	1
<input type="checkbox"/>	Edit Copy Delete	5	2023-04-06 00:00:00 000000	souris	30	2
<input type="checkbox"/>	Edit Copy Delete	6	2023-04-05 00:00:00 000000	clavier	50	2
<input type="checkbox"/>	Edit Copy Delete	7	2023-04-19 00:00:00 000000	PC lenovo	1500	1
<input type="checkbox"/>	Edit Copy Delete	8	2023-04-19 00:00:00 000000	pc lenovo	4000	1
<input type="checkbox"/>	Edit Copy Delete	14	2023-04-30 15:43:26 513000	PC Dell	2200.5	1
<input type="checkbox"/>	Edit Copy Delete	15	2023-04-30 15:46:40 499000	Asus	2200.4	1

Check all With selected Edit Copy Delete Export

Number of rows: 25 Filter rows: Search this table Sort by key: None

ECRIRE DES REQUÊTES @QUERY EN PASSANT DES ENTITÉS EN PARAMÈTRE

```
© Produit.java ×
1 package com.gt.produits.entities;
2
3 import jakarta.persistence.*;
4
5 import java.util.Date;
6 @Entity
7 public class Produit {
8     @Id
9     @GeneratedValue(strategy = GenerationType.IDENTITY)
10    private Long idProduit;
11    private String nomProduit; 4 usages
12    private Double prixProduit; 4 usages
13    private Date dateProduits ; 4 usages
14
15    @Contract(pure = true)
16    public Produit() { super(); }
17    @Contract(pure = true)
18    public Produit(String nomProduit, Double prixProduit, Date dateProduits) {
19        super();
20        this.nomProduit = nomProduit;
21        this.prixProduit = prixProduit;
22        this.dateProduits = dateProduits;
23    }
24
25    @ManyToOne 2 usages
26    private Categorie categorie;
```

