Atelier Framework Spring-03
JPA –Partie04

Objectifs

- **JPA et Spring Data (Approche2)**
  - Créer une interface Spring Data (repository)
  - Personnaliser le repository
  - Rechercher par requête paramétrée avec Spring Data

- **Spring Data JPA** est un autre module de Spring qui facilite grandement l'utilisation de JPA.
- **Spring Data JPA** permet de **générer toutes sortes d'opérations vers la base de données**, sans que nous ayons à écrire la moindre requête, ni même la moindre implémentation DAO. Il nous suffit d'hériter de l'interface « **JpaRepository**».

1. Réaliser une copie du projet «**jpa-spring-boot**» en le renommant «**jpa-spring-data-spring-boot**»
2. Créer un nouveau package sous « **src/main/java** » nommé : «**spring.jpa.repository**».
3. Créer, dans ce package, une interface «**ProduitRepository**» qui étend une interface appelée «**JpaRepository**» fournie par le module « **Spring Data** ».

«**JpaRepository**» est une interface générique (**paramétrée**) qui prend en paramètres :
- ✓ Le type de l'entité que l'on manipule («**Produit**» dans cet exemple)
- ✓ et le type de l'identifiant de l'entité («**Long**» dans cet exemple)

```
package spring.jpa.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import spring.jpa.model.Produit;

public interface ProduitRepository extends
JpaRepository<Produit, Long> {}
```

4. Vous n'avez pas besoin pour l'instant d'aucun code pour cette interface ni d'une classe d'implémentation : Le fait de définir une interface qui hérite de

«**JpaRepository**», nous disposons de toutes les méthodes de base nécessaires (les méthodes classiques):

- **save (Produit p)**
- **delete (Produit p)**
- **findAll()**
- **findOne(Produit p)**
- **count()**

5. Passons maintenant au test, nous utilisons la même application «**JpaSpringBootApplication**» en changeant la manière de persistance (Approche2) dans la classe «**JpaSpringBootApplication** » (utiliser le type «**ProduitRepository**» au lieu du type «**IProduitDao**») :

- Remplacer l'instruction suivante :

```
//Récupérer une implémentation de l'interface "IProduitDao" par injection de dépendance
IProduitDao daoProduit =  contexte.getBean(IProduitDao.class);
```

- Par celle-ci :

```
//Récupérer une implémentation de l'interface "ProduitRepository" par injection de dépendance
ProduitRepository produitRepos =  contexte.getBean(ProduitRepository.class);
```

- Réaliser les modifications nécessaires dans la classe : «**JpaSpringBootApplication** » :
  - **Remplacer « daoProduit » par « produitRepos »**
  - **Ajouter les instructions « import »**

NB : commenter les instructions d'appel des méthodes de recherches « **findByDesignation** » et « **findByDesignationAndPrix** » qui ne sont pas encore définies.

6. Lancer l'exécution pour visualiser le même résulat que la première approche.
7. Pour définir des méthodes personnalisées (comme la recherche par mot clé), nous utilisons maintenant l'annotation «**@Query**» pour déclarer des requêtes (**HQL : Hiberante Query Language ou bien JPQL :Java Persistence Query Language**). Modifier le code de l'interface « **ProduitRepository** » comme suit :

```
package spring.jpa.repository;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import spring.jpa.model.Produit;

public interface ProduitRepository extends JpaRepository<Produit, Long>
    {
     @Query ("select p from Produit p where p.designation like %:x% ")
     public List<Produit> findByDesignation(@Param ("x") String mc);
    }
```

8. Vous remarquez que nous n'avons pas besoin d'implémenter l'interface pour donner du code à la méthode « **findByDesignation** ». C'est Spring qui va se charger de ça. (Par injection de dépendance).
    - Annotée par « **@Query** », la méthode « **findByDesignation** » est appelée d'exécuter la requête HQL « `select p from Produit p where p.designation like %:x%` ».
    - Cette requête présente un paramètre « **x** » dans la valeur est récupérée de celle de l'argument « **mc** » de la méthode via l'annotation « **@Param** ».

9. Lancer par la suite le test de la méthode « **findByDesignation** » dans la classe « **JpaSpringBootApplication** »

Exercice

10. De même, définir et tester une méthode «**findByDesignationAndPrix**» qui recherche les produits ayant une désignation « **mc** » et un prix supérieur à « **prix** » sachant que :
    ✓ « **mc** » est un argument de type « **String** »
    ✓ « **prix** » est un argument de type « **double** »

11. Passer maintenant à définir une méthode de modification de données. Ajouter dans l'interface «**ProduitRepository**» la méthode «**mettreAJourDesignation**» comme suit :

```
@Query("update Produit p set p.designation =:designation where p.id = :id")
@Modifying
@Transactional
  public int mettreAJourDesignation(
@Param("designation")String designation,
@Param("id") Long idProduit);
```

✓ @Query : définir la requête HQL
✓ @Modifying : indiquer qu'il s'agit d'une requête d'écriture dans la BD
✓ @Transactional : passer la requête dans une transaction (tout ou rien). Cette annotation est définie dans le package « jakarta.transaction »

NB : Ajouter les instructions de « import » nécessaires

12. Passer au test :

- Exemple de test dans la classe « **JpaSpringBootApplication** » :

```
        //Mette à jour la désignation du produit ayant l'id=2
      System.out.println("-------------------------------------");
      System.out.println("***Mise à jour du protuit (id=2) :" );
      System.out.println("*************************************");
      produitRepos.mettreAJourDesignation("Pain", 2L);
    //Afficher le produit modifié s'il est présent
        Produit pm= produitRepos.findById(2L).get();
        if (pm!=null )
        {
          System.out.println("Désignation:"+pm.getDesignation());
        }
        else      {
            System.out.println("Produit non existant...");
        }
```

✓ finbById : méthode prédéfinie dans l'interface «**ProduitRepository**»

✓ 2L : pour spécifier un entier de type long