Atelier Framework Spring-03
JPA –Partie05

Objectifs

- **JPA et Spring Data**
  - Utiliser les méthodes requêtes
  - Utiliser les opérateurs de Query

- **Spring Data JPA** est capable aussi de générer la requête SQL automatiquement en partant du nom de votre méthode. On parle de **méthode requête**
- **Spring Data JPA** propose un ensemble de conventions qui lui permettront de déduire la bonne requête à partir du nom de la méthode. **Toute la logique est fournie par le nom de la méthode** (vous pouvez vous documenter à travers le lien suivant :

https://docs.spring.io/spring-data/data-jpa/docs/1.1.x/reference/html/#jpa.query-methods.query-creation

1. Reprendre le projet « **jpa-spring-data-spring-boot** »
2. Déclarer dans l'interface «**ProduitRepository**» la méthode suivante :

```
List<Produit> findByPrixGreaterThan(double prixMin);
```

- Toute la logique est fournie par le nom de la méthode :
  - **findBy** : indique que l'opération à exécuter est un SELECT
  - **Prix** : fournit le nom de la propriété sur laquelle le SELECT s'applique
  - **GreaterThan** : mot clé qui définit une condition "**plus grand que**"

- La valeur à appliquer à la condition est définie par le paramètre « **prixMin** ».
- Cette méthode génère une requête équivalente à l'instruction SQL suivant :

select * from produit where prix > ?

3. Passer au test et remarquer dans la console la requête HQL générée :

Hibernate: select produit0_.id as id1_0_, produit0_.designation as designat2_0_, produit0_.prix as prix3_0_, produit0_.quantite as quantite4_0_ from produit produit0_ where produit0_.prix>?

4. Réécrire avec la logique de **Spring Data** les deux méthodes suivantes (en utilisant les méthodes requêtes) at lancer l'exacution.

✓ **findByDesignation :**

```
// Retourner la liste des Produits par recherche par designation
List<Produit> findByDesignationLike(String mc);
```

✓ **findByDesignationAndPrix :**

```
// Retourner la liste des Produits par recherche par designation
// et dont le prix est supérieur à un prix minimal
List<Produit> findByDesignationLikeAndPrixGreaterThan(String mc, double prixMin);
```

NB :

Les opérateurs disponibles de Query sont les suivants :

- **By** : Opérateur principale actant le where SQL
- **Dinstinct** : Spécifie un Elément unique
- **Or, And** : Critère d'agrégation ou d'exclusion
- **OrderBy,Asc, Desc** :Acte le tri par propriété et politique Asc/Desc
- **LessThan, GreaterThan** :Règles numériques
- **Between, After, Before** Règle pour des dates
- **Like, NotLike** : Like SQL
- **IsNull, isNotNull, NotNul** : Opérateurs sur valeurs null
- **Not** :Différent de
- **In, NotIn** : Est contenu ou pas dans la liste de valeurs
- **True, False** : Opérateurs sur valeurs booléennes
- **IgnoreCase, AllIgnore , StartingWith, EndingWith , Containing** : Opérateurs sur chaines de caractères

Exercice

5. Définir les méthodes qui correspondent aux traitements suivants :
   - ✓ Retourner la liste de tous les produits en ordre croissant selon le prix
   - ✓ Retourner la liste des produits dont la date d'achat est supérieure à « **2020-03-15** »
   - Ajouter un attribut « **dateAchat** » dans la définition de l'entité « **Produit** »
   - Modifier la classe «**JpaSpringBootApplication** » pour insérer des produits avec date d'achat.
6. Réaliser les tests nécessaires