

JÉRÉMY PERROUAULT



DATA-JPA

SPRING DATA-JPA & HIBERNATE

- Déclaration d'interfaces qui étendent l'interface JpaRepository<T, ID>
 - @Repository devient implicite
 - aTransactional devient implicite sur les méthodes classiques

- Utilisation d'une convention pour nommer les méthodes
 - findByNom(String nom)
 - findByReference(String reference)
 - findBy...(arguments attendus)
 - findByNomContaining(String nom)
- Possibilité d'ajouter des Queries spécifiques
 - Utilisation de l'annotation @Query("<Requête JP-QL>") sur la méthode
 - Utilisation de l'annotation a Modifying sur la méthode
 - Si c'est une requête de mise à jour (update ou delete)

- Si la méthode retourne un **Produit**
 - Spring retournera null si rien n'a été trouvé
 - Spring lèvera une exception si plusieurs éléments ont été trouvés
- Si la méthode retourne une liste de **Produit**
 - **Spring** retournera une liste!

Convention	Effet
findByAttribut	Rechercher sur la valeur d'un attribut
findByAttribut1AndAttribut2	Rechercher sur la valeur de deux attributs
findByAttributContaining	Rechercher une valeur contenue dans un attribut
findAllByOrderByAttributDesc	Rechercher tout et ranger par ordre décroissant sur l'attribut
findFirstByAttribut	Rechercher le premier élément sur la valeur d'un attribut
findTop10ByAttribut	Rechercher les 10 premiers éléments sur la valeur d'un attribut
findTop10ByAttributContaining	Rechercher les 10 premiers éléments dont la valeur d'un attribut contient
findTop10ByAttributContainingOrderByIdDesc	Rechercher les 10 derniers éléments dont la valeur d'un attribut contient
countByLibelleContaining	Compter le nombre d'éléments dont la valeur d'un attribut contient

```
public interface IProduitRepository extends JpaRepository<Produit, Integer> {
   public List<Produit> findByNom(String nom);
   public Produit findFirstByNom(String nom);
   public List<Produit> findTop10ByNomContainingOrderByIdDesc(String nom);
   public int countByNomContaining(String nom);
}
```

- Important à savoir
 - Les méthodes find par défaut de **Spring DATA-JPA** retournent
 - Soit une List<T> (c'est le cas de *findAll*)
 - Soit un Optional<T> (c'est le cas de findById)
 - Si la convention n'est pas respectée et que la méthode n'est pas annotée de aQuery
 - **Spring** génère une erreur au démarrage de l'application

- Il est possible de paginer les résultats
 - En utilisant un objet de type Pageable

public List<Produit> findByNomContainingOrderByIdDesc(String nom, Pageable pageable);

```
// Première page de 20 éléments
repoProduit.findByNomContainingOrderByIdDesc("g", PageRequest.of(0, 20));

// Deuxième page de 20 éléments
repoProduit.findByNomContainingOrderByIdDesc("g", PageRequest.of(1, 20));
```

La pagination peut aller plus loin : possibilité de trier avec *PageRequest*!

CONFIGURATION

CONFIGURATION DE SPRING DATA-JPA

CONFIGURATION

- Une dépendance à charger (en plus des dépendances SPRING JPA)
 - spring-data-jpa

CONFIGURATION

- Annotation de la classe de configuration
 - @EnableJpaRepositories("fr.formation.repo")

EXERCICE

• Implémenter SPRING DATA-JPA