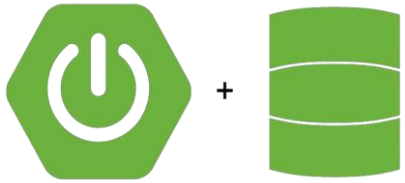
Spring 5, développer des applications d'entreprise

Niveau : 3 ING
Enseignant: Dr. Hafedh NEFZI

2025/2026

Chapitre 4

SPRING DATA JPA – JPQL



Plan

- JPQL
- Requêtes SELECT, UPDATE, DELETE, INSERT avec JPQL et Native Query
- @Query
- @Modifying
- @Param

JPQL : Définition

- JPQL = Java Persistence Query Language
- JPQL peut être considéré comme une version orientée objet de SQL.
- En JPQL, on sélectionne des objets d'une entité (une classe annotée par @Entity) en utilisant les noms des attributs et le nom de la classe de l'entité en question et non ceux de la table de base de données et ses colonnes.

Select

- Récupérer les entreprises avec une adresse donnée :

- **JPQL** :

```
@Query("SELECT e FROM Entreprise e WHERE e.adresse
=:adresse")
List<Entreprise> retrieveEntreprisesByAdresse
(@Param("adresse")String adresse);
```

C'est équivalent à :

```
@Query("SELECT e FROM Entreprise e WHERE
e.adresse = ?1")
List<Entreprise> retrieveEntreprisesByAdresse
(String adresse);
```

Select

L'entité Entreprise est mappée avec la table T_Entreprise :

- **Native Query (SQL et non JPQL) :**

```
@Query(value = "SELECT * FROM T_Entreprise e WHERE
e.entreprise_adresse = :adresse",nativeQuery = true)
List<Entreprise>
retrieveEntreprisesByAdresse(@Param("adresse")String
adresse);
```

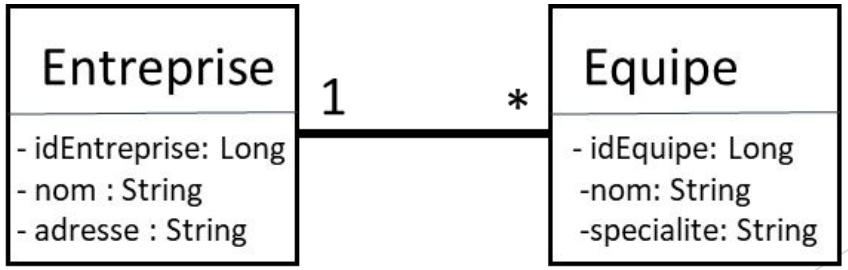
C'est équivalent à :

```
@Query(value = "SELECT * FROM T_Entreprise e WHERE
e.entreprise_adresse = ?1",nativeQuery = true)
List<Entreprise> retrieveEntreprisesByAdresse(String
adresse);
```

Select

- Récupérer les entreprises qui ont une équipe avec une spécialité donnée :
- **JPQL :**

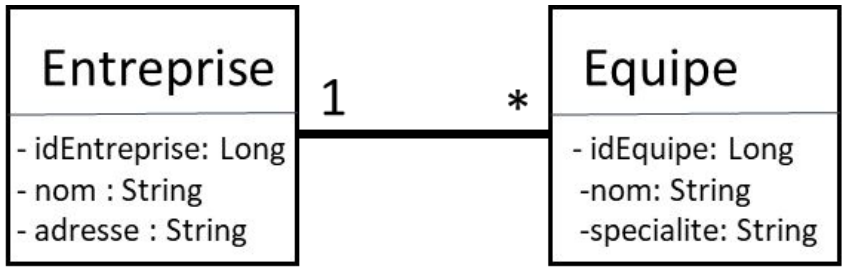
```
@Query("SELECT DISTINCT entreprise
FROM Entreprise entreprise
JOIN entreprise.equipes equipe
WHERE equipe.specialite = :specialite")
List<Entreprise>
retrieveEntreprisesBySpecialiteEquipe(@Param("speci
alite")String specialite);
```



Select

- **Native Query :**

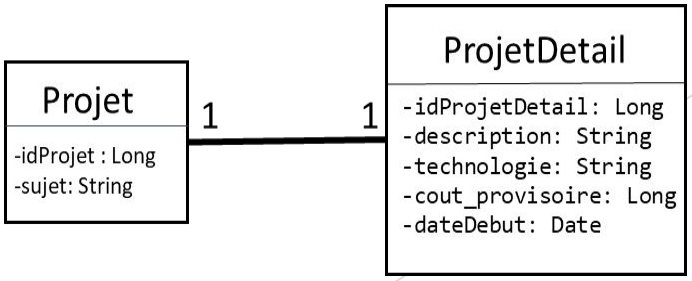
```
@Query(value = "SELECT * FROM T_ENTREPRISE entreprise
JOIN T_EQUIPE equipe ON entreprise.ENTREPRISE_ID =
equipe.ENTREPRISE_ENTREPRISE_ID where
equipe.EQUIPE_SPECIALITE =:specialite", nativeQuery =
true)

List<Entreprise>
retrieveEntreprisesBySpecialiteEquipe(@Param("specialite")
String specialite);
```



Select

- Récupérer les projets qui ont un coût supérieur à un coût donné et une technologie donnée.
- **JPQL :**

```
@Query("SELECT p
FROM Projet p
JOIN p.projetDetail d
WHERE d.cout_provisoire > :cout
 AND d.technologie = :tech")
List<Projet>
retrieveProjetsByCoutAndTechnologie(@Param
("technologie") String technologie,
@Param("cout_provisoire") Long
cout_provisoire);
```



Select

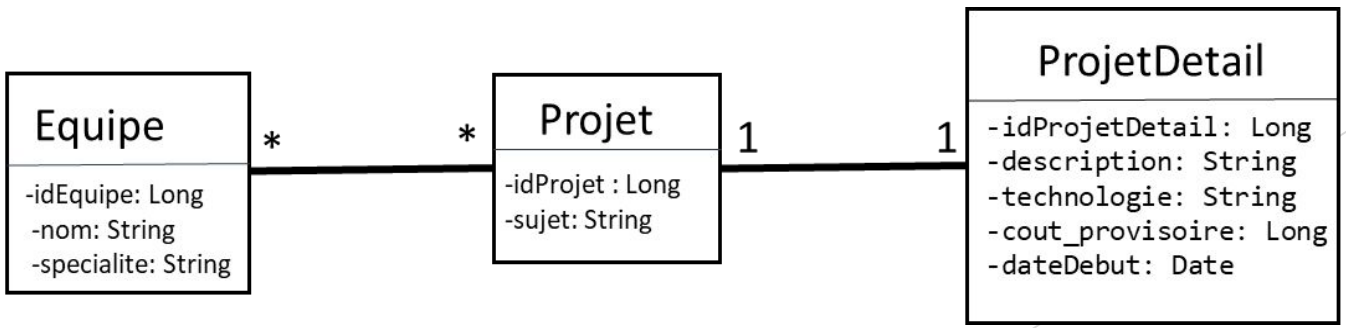
- **Native Query (SQL et non JPQL) :**

```
@Query(value = "SELECT * FROM T_PROJET projet JOIN
T_PROJET_DETAIL detail ON detail.PD_ID =
                projet.PROJET_DETAIL_PD_ID WHERE
detail.PD_TECHNOLOGIE = ?1 and
detail.PD_COUT_PROVISOIRE > ?2", nativeQuery = true)

List<Projet>

retrieveProjetsByCoutAndTechnologie(@Param("technol

ogie")String technologie, @Param("cout_provisoire")

Long cout_provisoire);
```

Select

- Récupérer les équipes qui travaillent sur une technologie donnée dont le projet n'a pas encore commencé.
- **JPQL :**

```
@Query("SELECT equipe FROM Equipe equipe
      + " JOIN equipe.projets projet"
      + " JOIN projet.projetDetail detail"
      + " where detail.dateDebut > current_date"
      + " and detail.technologie =:technologie")
List<Equipe>retrieveEquipesByProjetTechnologie
(@Param("technologie") String technologie);
```



Update

- Si nous souhaitons faire un **UPDATE**, **DELETE** et **INSERT**, nous devons ajouter l'annotaion **@Modifying** pour activer la modification de la base de données.

- Cette méthode permet de mettre à jour l'adresse de l'entreprise.

- **JPQL :**

```
@Modifying

@Query("update Entreprise e set e.adresse = :adresse where
e.idEntreprise = :idEntreprise)

int updateEntrepriseByAdresse
(@Param("adresse")String adresse,
@Param("idEntreprise") Long idEntreprise);
```

Delete

- Cette méthode permet de supprimer les entreprises qui ont une adresse donnée :

- **JPQL :**

```
@Modifying
@Query("DELETE FROM Entreprise e WHERE e.adresse= :adresse")
int deleteEntrepriseByAdresse(@Param("adresse") String adresse);
```

C'est équivalent à :

```
@Modifying
@Query("DELETE FROM Entreprise e WHERE e.adresse= ?1")
int deleteEntrepriseByAdresse(String adresse);
```

Insert

- Cette méthode permet d'insérer des projets dans la table T_Projet:

- JPQL : Nous utilisons Spring Data JPA. Or INSERT ne fait pas partie des spécifications JPA. Donc, nous sommes obligés d'utiliser les Natives Query pour le INSERT.

- **Pas de JPQL pour les requêtes INSERT.**

- **Native Query (SQL et non JPQL) :**

```
@Modifying
@Query(value = "INSERT INTO T_Projet(projet_sujet)
  VALUES (:projetsujet)", nativeQuery = true)
void insertProjet(@Param("projetsujet") String
projetsujet);
```

>> Spring Security