Projection

adriencaubel.fr

Table des matières

Introduction

Rappels

- Nous avons étudié les concepts suivants
  - La notion `@Entity`
  - Les associations `@OneToMany` ... et `mappedBy` (bidirectionnelle)
  - L'héritage `@Inheritance`
  - L'optimisation des lectures (`Fetch.LAZY` / `Fetch.EAGER`)
  - L'API Criteria

Nous allons maintenant nous intéresser à **la notion de Projection**

Objectifs des projection ?

Définition
Les projections sont des techniques permettant de ne récupérer que les champs strictement nécessaires depuis une base de données, plutôt que l'ensemble des informations d'une entité.

Elles permettent ainsi de

- réduire la quantité de données transférées
- améliorer donc la performance du système

Exemple

Ainsi au lieu de tout récupérer (`SELECT *`), nous sélectionnons les colonnes qui nous intéressent

```
em.createQuery("SELECT b.title, b.publisher, b.author.name FROM Book b");
```

3 types de projection

3 Types de Projections

JPA supportent principalement trois types de projections :

- **Projections d'entités** : Récupération complète des entités.
- **Projections de valeurs scalaires** : Sélection de colonne spécifique
- **Projections DTO**: Utilisation d'objets personnalisés pour transporter des données spécifiques.

Projection d'entité

C'est la plus commune, on récupère toute l'entité soit via un `find()` soit avec JPQL

```
TypedQuery<Book> query = entityManager.createQuery("SELECT b FROM Book b", Book.class);
List<Book> books = query.getResultList();
```

Equivalent SQL

```
SELECT *
FROM book;
```

Projections de Valeurs Scalaires

Les projections de valeurs scalaires permettent de sélectionner des colonnes spécifiques ou des résultats de fonctions agrégées, sans charger des entités complètes.

```
// Récupère un tableau d'Object
TypedQuery<Object[]> query = entityManager.createQuery(
  "SELECT b.title, b.publisher FROM Book b", Object[].class);

List<Object[]> results = query.getResultList();
for (Object[] result : results) {
    String title = (String) result[0];
    String publisher = (String) result[1];
}
```

Équivalent SQL

```
SELECT title, publisher
FROM book;
```

Très pratique, mais pas très lisible, il serait intéressant d'utiliser un type précis au lieux de Object ⇒ DTO

=> Les projections DTO impliquent l'utilisation de classes personnalisées pour encapsuler les données nécessaires, offrant une structure claire.

Projections DTO (Data Transfer Object)

```java
public class BookDTO {
    private String title;
    private String authorName;
    private String publisher;
}
```

```java
// Pas Object[] mais typé avec le DTO
TypedQuery<BookDTO> query = entityManager.createQuery(
  "SELECT new com.example.BookDTO(b.title, b.author.name, b.publisher) FROM Book b",
  BookDTO.class);

List<BookDTO> bookDTOs = query.getResultList();
```

Équivalent SQL

```sql
SELECT
    b.title,
    a.name AS author_name,
    b.publisher
FROM book b
JOIN author a ON b.author_id = a.id;
```

ATTENTION : Note ultra importante

Contrairement au projection d'entité, **les projections scalaire et DTO ne font plus partie du contexte de persistance et ne suivent plus le cycle de vie**.

- Ainsi si vous mettez à jour le DTO, la nouvelle donnée ne sera pas persister en base de donnée
- Par conséquent les projections scalaire et DTO ne peuvent être utilisé que pour de la lecture

```
TypedQuery<BookDTO> query = entityManager.createQuery(
    "SELECT new com.example.BookDTO(b.title, b.publisher) FROM Book b",
    BookDTO.class
);

List<BookDTO> books = query.getResultList();

BookDTO dto = books.get(0);
dto.setTitle("Nouveau titre"); // ❌ Résultat : aucune mise à jour en base
```

Car BookDTO :

- n'est pas géré par le contexte de persistance
- ne possède pas de cycle de vie JPA
- **⇒ est juste une copie des données**

"Choisir sa projection"

Le choix du type de projection dépend du cas d'utilisation :

- **Opérations d'écriture** : Les projections d'entités sont obligatoire en raison de la gestion automatique du cycle de vie des entités par le contexte de persistance.
- **Opérations en lecture seule** : Les projections de valeurs scalaires ou DTO sont préférables pour réduire le surcoût et améliorer les performances.