

# JPA Mapping

---

## Entités et Composants

adriencaubel.fr

# Table des matières

- 1. Introduction
  - 1. Rappels
- 2. Les entités JPA
  - 1. Entité JPA
  - 2. Entité JPA – La clé
  - 3. Entité JPA – Colonnes
- 3. Les composants JPA (Embeddable)
  - 1. Composant JPA
  - 2. Composant – Exemple

# Introduction

# Rappels

- Nous avons introduit la spécification JPA
  - EntityManager et les opérations persist, remove, find ...
  - Cycle de vie d'une entité
  - Gestion des transactions
- Maintenant, intéressons nous au Mapping Objet/Relationnel
  - Partie 1 : notion d'entités et de composants
  - Partie 2 : les associations

# Les entités JPA

# Entité JPA

```
// définit l'entité mappant la table « sport » dans le schéma par défaut
@Entity
@Table(name = "t_sport")
public class Sport implements Serializable {
    // définit la clé primaire
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Integer id;
    // l'attribut « intitulé » mappe la colonne « intitulé » : optionnel car ont
    // le même nom, le mapping se ferait implicitement et par défaut
    @Column(name = "intitule")
    private String intitulé;
}
```

- `@Entity`, définit une classe comme étant une entité EJB dont les instances pourront être rendues persistantes
- `@Table(name = "t_sport")` (optionnelle)

Par défaut le défaut de la nom de la table sera le nom de la classe

# Entité JPA - La clé

- `@Id` , si l'attribut est un identifiant d'objet
- `@GeneratedValue` , comment la clé primaire sera générée

## Les stratégies

Stratégie	Description
<b>AUTO</b>	Le fournisseur de persistance doit choisir une stratégie appropriée en fonction de la base de données utilisée.
<b>IDENTITY</b>	Attribuer les clés primaires de l'entité en utilisant une colonne d'identité de la base de données.
<b>SEQUENCE</b>	Attribuer les clés primaires de l'entité en utilisant une séquence de la base de données.

# Entité JPA - Colonnes

- `@Column` (optionnelle) permet de préciser plusieurs paramètre
- Plusieurs paramètre
  - `name` indique le nom de la colonne dans la table;
  - `length` indique la taille maximale de la valeur de la propriété;
  - `nullable` (false ou true) indique si la colonne accepte ou non des valeurs à NULL
  - `unique` indique que la valeur de la colonne est unique

```
@Column(name = "email", length = 255, nullable = false, unique = true)  
private String email;
```

# Les composants JPA (Embeddable)

# Composant JPA

Un composant, au contraire, est un objet sans identifiant, qui ne peut être persistant que par rattachement à une entité.

- Réutilisable dans plusieurs entités
- Pas de table séparée
- Mappage intégré dans la table principale
- Permet de structurer des données complexes

# Composant - Exemple

Societe

id	...	rue	ville	code_postal

Client

id	...	rue	ville	code_postal

Via les composants nous n'avons pas à écrire l'ensemble des attributs dans nos deux classes Java mais pouvons coder une classe Adresse

```
@Embeddable
public class Adresse {
    @Column(name = "rue")
    private String rue;

    @Column(name = "ville")
    private String ville;

    @Column(name = "code_postal")
    private String codePostal;
}
```

Aucune table de ne sera créée pour Adresse

```
@Entity
public class Client {
    @Id
    private Long id;
    private String nom;

    @Embedded
    private Adresse adresse;
}

@Entity
public class Societe {
    @Id
    private Long id;
    private String nomEntreprise;

    @Embedded
    private Adresse adresse;
}
```

Chaque attribut dans Adresse sera dans les tables client et société