# Java Persistence API

Guillaume Dufrêne – Lionel Seinturier

Université de Lille – Sciences et Technologies

# Java Persistence API

## Définition

**J**ava **P**ersistence **A**PI
interface de programmation pour la gestion de données relationnelles

- Manipulation de données et de leurs relations
- Spécification Java (version 2.1, JSR 338, mai 2013)
- Différentes mise en oeuvre :
  Hibernate, Toplink, EclipseLink, Apache OpenJPA, etc.
- 3 parties :
  - les annotations
  - l'API pour manipuler les données
  - JPQL

## Définition

***O**bject **R**elation **M**apping*

Technique de programmation


ORM

- opérations SQL courantes
- vision homogène
- haut niveau
- transfert automatique des données

# Vocabulaire JPA

## Définitions

Persistance / persister :

- action de sauvegarder des données dans une base de données (SGBD)

Entité :

- un ensemble de données d'un même type (eg. Personnes)

## Deux approches

1. Création du modèle de données depuis JPA

2. Schéma et Données existantes dans un SGBD

## Principes de mise en correspondance

| | |
|---|---|
| • une classe Java | une table SQL |
| • une propriété d'une classe | une colonne dans une table SQL |
| • une référence entre classes | une relation entre tables SQL |
| • un objet Java | un enregistrement dans une table SQL |

Remarques

- nb objets ≠ nb enregistrements
- les données peuvent être chargées de façon dite paresseuse (*lazy*)

# Annotation @Entity

## Définition

- à déclarer sur les classes dont les données doivent être prises en charge par JPA

```
@Entity
public class Personne {

  private String nom;
  private int age;

  public Personne( String nom, int age ) {
    this.nom = nom;
    this.age = age;
  }

  public String getNom() { return nom; }
  public void setNom( String nom ) { this.nom = nom; }
  public int getAge() { return age; }
  public void setAge( int age ) { this.age = age; }

}
```

# Annotations @Id et @GeneratedValue

## Définitions

`@Id` : clé primaire

`@GeneratedValue` : générer automatiquement la clé

```
@Entity
public class Personne {
  private long id;
  private String nom;
  private int age;
  public Personne( String nom, int age ) {
    this.nom = nom;
    this.age = age;
  }
  @Id
  @GeneratedValue
  public long getId() { return id; }
  public void setId( long id ) { this.id = id; }
  // méthodes getNom, setNom, getAge, setAge
}
```

## Relations entre entités

Liens entre des types de données

Cardinalités

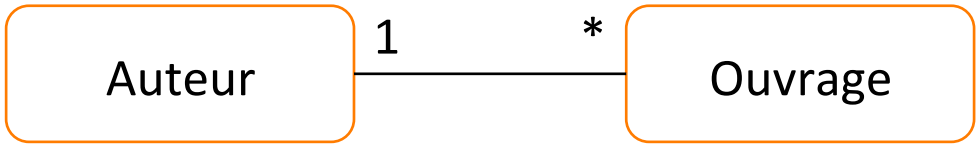
- 1-n
- n-1
- 1-1
- n-n

Annotation JPA

- @OneToMany
- @ManyToOne
- @OneToOne
- @ManyToMany

# Relations entre entités

## Exemple




```
@Entity
public class Auteur {
  String nom;
  @OneToMany
  List<Ouvrage> oeuvres;
  // constructeur
  // getter/setter nom
  // getter/setter oeuvres
}
```

```
@Entity
public class Ouvrage {
  String titre;
  @ManyToOne
  Auteur auteur;
  // constructeur
  // getter/setter titre
  // getter/setter auteur
}
```

# Modèles de données existant

## Principe

- Correspondance attributs et colonnes
- Annotations `@Table` et `@Column`

```
@Entity
@Table(name="ECRIVAINS")
public class Auteur {
  @Column("NOM_ID")
  String nom;
  @OneToMany(mappedBy="auteur")
  List<Ouvrage> oeuvres;
  // constructeur
  // getter/setter nom
  // getter/setter oeuvres
}
```

```
@Entity
@Table(name="LIVRES")
public class Ouvrage {
  @COLUMN("TITRE_ID")
  String titre;
  @ManyToOne
  @JoinColumn("NOM_ID")
  Auteur auteur;
  // constructeur
  // getter/setter titre
  // getter/setter auteur
}
```

# Entity Manager

## Définition

- Assure la liaison entre SGBD et Java
- fournit les méthodes courantes d'accès aux données

Exemple

- création de deux nouvelles entités Personne

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("myapp");
EntityManager em = emf.createEntityManager();

Personne anne = new Personne("Anne", 37);
Personne bob = new Personne("Bob", 42);

em.persist(anne);
em.persist(bob);
```

# Recherche d'entités

## Recherche par requête

- méthode find

Exemple

- recherche de la personne dont l'id vaut 123

```
Personne p123 = em.find( Personne.class, 123 );
```

# Recherche d'entités

## Recherche par requête

- méthode createQuery

1. Requête JPQL paramétrée
2. Valeur des paramètres
3. Récupération de la liste de résultats

Exemple

- recherche des personnes dont l'âge est supérieur à une valeur donnée

```
TypedQuery<Personne> q =
  em.createQuery("SELECT p FROM Personne p WHERE age >= :valeur");
q.setParameter("valeur", 42);
List<Personne> l = q.getResultList();
```

# CRUD JPA

## Définition

- acronyme de **C** reate, **R** ead, **U** pdate, **D** elete

## Exemple

```
Personne bob = new Personne("Bob", 42);
em.persist(bob);

Personne p123 = em.find( Personne.class, 123 );

bob.setAge( bob.getAge() + 1 );
em.merge(bob);

em.remove(bob);
```

# En résumé

## Java Persistence API

- Interface de programmation pour la gestion de données relationnelles
- Manipulation des données d'un SGBD comme des objets
- Sauvegarde des informations transparentes
- Abstraction de SQL
- Différentes implémentations