JPA – API Criteria

Guillaume Dufrêne – Lionel Seinturier

Université de Lille – Sciences et Technologies

API Criteria

Définition

API liée à JPA pour le requêtage des données

Avantages

- Correction des requêtes avant exécution
- Requêtes typées

Principes

- chaque clause de requête possède un équivalent en méthode Java

Un premier exemple

Recherche de toutes les instances d'une entité Personne

- requête JPQL

```
SELECT p FROM Personne p
```

```
@Entity
public class Personne {
  @Id public String nom;
      public int age;
}
```

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("myapp");
EntityManager em = emf.createEntityManager();
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Personne> cq = cb.createQuery(Personne.class);
Root<Personne> p = cq.from(Personne.class);
cq.select(p);
TypedQuery<Personne> tq = em.createQuery(cq);
List<Personne> personnes = tq.getResultList();
```

Un deuxième exemple

Critère de sélection WHERE pour les entités

- requête JPQL

```
SELECT p FROM Personne p WHERE age >= :valeur
```

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("myapp");
EntityManager em = emf.createEntityManager();
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Personne> cq = cb.createQuery(Personne.class);
ParameterExpression<Integer> valeur = cb.parameter(Integer.class);
Root<Personne> p = cq.from(Personne.class);
cq.select(p).where(cb.ge(p.get("age"),valeur));
TypedQuery<Personne> tq = em.createQuery(cq);
tq.setParameter(valeur,42);
List<Personne> personnes = tq.getResultList();
```

Autres fonctionnalités

Spécialisation des requêtes

- expression en Java des fonctionnalités existant en SQL

- opérateurs logiques : and, or, not
- opérateurs de comparaison : equal, notEqual, gt, ge, lt, le, between, like
- opérateurs de valeur : isNull, isNotNull, in

- opérateur de jointure : join
- tri des résultats : orderBy, asc, desc
- groupement de résultats : groupBy, having

Métamodèle

Définition

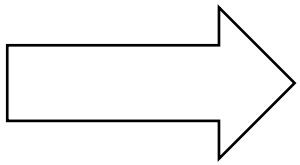
Un métamodèle est un modèle d'un autre modèle

Objectif

- pouvoir manipuler les éléments du modèle

Exemple

```
@Entity
public class Personne {
  @Id public String nom;
      public int age;
}
```



une chaine nommée “nom”
un entier nommé “age”

Métamodèle

Code des métamodèles

- Généré automatiquement (IDE)
- Fourni par le développeur

Métamodèle correspondant à l'entité Personne

```
@Entity
public class Personne {
  @Id public String nom;
      public int age; }
```

```
@StaticMetamodel(Personne.class)
class Personne_ {
    public static volatile SingularAttribute<Personne,String> nom;
    public static volatile SingularAttribute<Personne,Integer> age;
}
```

Exemple

- requête JPQL

```
SELECT p FROM Personne p WHERE age >= :valeur
```

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("myapp");
EntityManager em = emf.createEntityManager();
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Personne> cq = cb.createQuery(Personne.class);
ParameterExpression<Integer> valeur = cb.parameter(Integer.class);
Root<Personne> p = cq.from(Personne.class);
cq.select(p).where(cb.ge(p.get("age"),valeur));
cq.select(p).where(cb.ge(p.get(Personne_.age),valeur));
TypedQuery<Personne> tq = em.createQuery(cq);
tq.setParameter(valeur,42);
List<Personne> personnes = tq.getResultList();
```

En résumé

API Criteria

- Manipuler les requêtes SQL SELECT en Java
- Correspondance des concepts SQL avec des méthodes et des interfaces Java
- Améliore le typage des programmes
- Limite le risque d'erreur