Object Relational Mapping
Java Persistence API

Alexandre COLLIGNON - Ingénieurs 2000

22 Janvier 2008

Sommaire

- Introduction
- Object Relational Mapping
- Java Persistence API
- Exemple d'utilisation
- Conclusion

Introduction
Les bases de données relationnelles

- Basées sur le langage relationnel
  - Opérations ajout / lecture / modification / suppression
- Eléments de même “constitution” stockés dans des tables
- Stockent de grande quantité de données de façon
  - Uniforme
  - Maîtrisée
  - Interopérable

Introduction
Le monde objet

- Ensemble d'entités
  - Autonomes
  - En relation
  - Qui coopèrent
- Un objet correspond à
  - Ensemble de membres (primitif ou objet)
  - Ensemble de méthodes
  - UNE responsabilité

Object Relational Mapping

- Relier les mondes objet et relationnel
  - Stocker les objets dans des bases de données relationnelles
- Offrir une couche d'abstraction au SGBDR
- Séparer les responsabilités
- Implémentation dans de nombreux langages
  - Framework Ruby on Rails contient un ORM
  - Rose::DB::Object en Perl
  - PDO pour Python

Java Persistence API

- L'O.R.M à la sauce Java
- Une partie de la spécification EJB3 (JSR220)
  - Intégrée à Java Enterprise Edition 5
  - Fruit du rassemblement des travaux des ORMs Java existants
- Différentes implémentations disponibles
  - Toplink Essential (Reference Implementation)
  - Hibernate (JBoss)
  - Open JPA (Apache)

Java Persistence API

- Environnement d'exécution
  - Application Stand-alone (Architecture 2-Tiers)
  - Conteneur d'EJB (Architecture 3-Tiers)
- Code indépendant de l'environnement
  - Souplesse évidente
  - Maintenabilité accrue

Java Persistence API

- Cinq composants principaux
  - L'unité de persistance
  - Le modèle des objets du domaine
  - Les méta-données
  - API
  - Requêtes

Java Persistence API
Unité de persistance

- Correspond à l'environnement
  - Configuration de l'accès à la base de données
  - Paramètrage de la couche de persistance
- Spécifiée à l'aide du fichier persistence.xml

Java Persistence API
Unité de persistance

- Exemple hors conteneur d'EJBs

```
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence">
     <persistence-unit name="xposejpa-javadb">
          <provider>
               oracle.toplink.essentials.PersistenceProvider
          </provider>
          <class>fr.umlv.etudiant.acollign.jpa.bom.Person</class>
          ...
          <properties>
               <property name="toplink.jdbc.driver" value="org.apache.derby.jdbc.ClientDriver" />
               <property name="toplink.jdbc.url" value="jdbc:derby://localhost/xpose" />
               <property name="toplink.jdbc.user" value="xpose" />
               <property name="toplink.jdbc.password" value="china" />
               <property name="toplink.ddl-generation" value="create-tables" />
          </properties>
     </persistence-unit>
</persistence>
```

Java Persistence API
Unité de persistance

- Exemple dans un conteneur d'EJB

```
<persistence version="1.0">
      <persistence-unit name="xpose-jpa">
            <jta-data-source>jdbc/xpose</jta-data-source>
      </persistence-unit>
</persistence>
```

Java Persistence API
Objet du domaine - Entity

- Objet métier sous forme de POJO (Plain Old Java Object)
  - Création par un simple new
  - Simple
  - Maîtrisé
- Aucune contrainte d'implémentation / héritage
- Entity = POJO + Clé primaire + Méta données

Java Persistence API
Objet du domaine - Entity

- Un simple JavaBean
  - Constructeur par défaut
  - Accesseur pour chaque champ
    - getFieldName / setFieldName (non booléen)
    - isFieldName / setFieldName (booléen)
  - Gestion d'évènement sur les modifications des propriétiés
    - non requis par JPA

Java Persistence API
Objet du domaine - Entity

- Un exemple de JavaBean

```
public class Person {
	private String lastname;
	private String firstname;
	private Address address;

	public Person() {}

	public Person(String lastname, String firstname, Address address) {
		this.firstname = firstname;
		this.lastname = lastname;
		this.address = address;
	}

	public String getLastname() {
		return lastname;
	}
	public void setLastname(String lastname) {
		this.lastname = lastname;
	}
	...
}
```

Java Persistence API
Objet du domaine – Clé primaire

- Obligatoire
- Référence unique vers un Entity en base de données
- Deux formes possibles
  - Simple (un type primitif)
  - Composé (N type(s) primitif(s) ou objet(s))

Java Persistence API
Objet du domaine – Clé primaire

- Génération automatique des clés primaires simples
  - AUTO
    - Au choix du provider de peristance
  - TABLE
    - Une valeur dans une table utilisée pour tous les entities
  - SEQUENCE
    - Fonction sequence du SGBDR (pas toujours portable)
  - IDENTITY
    - Fonction identity du SGBDR (pas toujours portable)

Java Persistence API
Objet du domaine – Clé primaire

- Clé primaire composée
  - Plusieurs champs de l'objet
  - Un objet externe contenant plusieurs champs

Java Persistence API
Méta données

- Définition dudit mapping
  - Correspondance entre entities et tables
  - Correspondance entre membres et colonnes
  - Renseignement des relations entre entities
  - Spécification de la génération des clés primaires

Java Persistence API
Méta données

- Deux méthodes
  - Annotation Java
  - Fichier de description XML orm.xml
- Ces deux méthodes peuvent être mélangées
  - Technique appelée “merged meta-data”
  - Parfois utile
  - Souvent dangereux

Java Persistence API
Méta données - Exemple

- Utilisation des annotations

```
@Entity
public class Person {

      @Id
      @GeneratedValue
      private long id;

      @Column(name="LASTNAME")
      private String lastname;

      private String firstname;

      private Address address;

      public Person() {}

      public String getLastname() { return lastname;}

      public void setLastname(String lastname){
            this.lastname = lastname;
      }
      ...
}
```

Java Persistence API
Méta données - Exemple

- Utilisation du fichier orm.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<entity-mappings version="1.0">
      <entity class="fr.umlv.etudiant.acollign.jpa.bom.Person">
            <attributes>
                  <id name="id">
                        <generated-value />
                  </id>
                  <basic name="lastname">
                        <column name="LASTNAME" />
                  </basic>
            </attributes>
      </entity>
</entity-mappings>
```

Java Persistence API
Le manager d'entité (EntityManager)

- Entity managé par l'objet EntityManager
  - Lecture / ajout / modification / suppression
  - Factory de requêtes
  - Récupération de la transaction d'entité
  - Attribué à une unité de persistance
- EntityManagerFactory
  - Créé sur la factory method Persistence.createEntityManagerFactory
  - Méthode createEntityManager

Java Persistence API
Requête (Query)

- JPA dispose du langage JPQL (aka EJBQL)
  - Inspiré fortement du langage SQL
  - Plus proche de l'objet que SQL
  - Une abstraction supplémentaire
  - Facile à manipuler (quand on connaît SQL ou pas)
- SQL reste cependant utilisable
  - Parfois nécessaire
  - Mais déconseillé

Java Persistence API
Requête (Query)

- L'objet Query
  - Représente une requête
  - Instantiation à l'aide de factory method sur un EntityManager
    - em.createQuery
    - em.createNativeQuery
    - em.createNamedQuery

Java Persistence API
Requête (Query)

- Exemple de requêtes JPQL
  - Select s From Student s
  - Select s From Student s Where s.firstname = “Alexandre”
- Les requêtes paramètrées
  - Select s From Student s Where s.firstname = :fs
  - Peuvent être compilées -> performances
  - Enrichies des paramètres (place holder) après création
    - query.setParameter(“fs”, “Alexandre”);

Java Persistence API
Requête (Query)

- Named Query
  - Utilisation d'annotation @NamedQuery sur un Entity
  - Associe un nom à une requête JPQL
  - Plus de cohérence
  - Compilée (et donc plus rapide)
  - @NamedQuery(“Student.findAll”, “Select s from Student s”)
- Utilisation via factory method spécifique
    - em.createNamedQuery(“Student.findAll”);
  - /!\ Espace de nom partagé -> Attention au conflit
    - Convention ClassName.QueryDescription

Java Persistence API

- Exemple d'utilisation avec
  - Des étudiants (et donc des personnes) et leurs filières
  - Toplink Essential comme implémentation JPA
  - Deux SGBDR
    - Java DB (aka Apache Derby DB)
    - PostgreSQL
  - SQuirreLSQL
  - Eclipse en IDE

Conclusion

- Les avantages
  - Séparation couche métier – accès à la base de données
  - Simplification de l'utilisation des données
  - Abstraction du SGBDR sous-jacent
- Les faiblesses (subjectives)
  - Jeunesse de la norme
  - Implémentation aux compléments hétérogènes
    - Utilisation des bonus empêche de changer d'implémentation

Conclusion

- Le futur de JPA
  - Ecriture de la version 2.0 en cours
  - Spécification sera extraite de celle des EJBs
- Les alternatives aux ORMs (et au SGBDR)
  - Utilisation de SGBD Objet (Caché, Versant)
- Ce qui n'a pas pu être vu en détail
  - Gestion des relations
  - Gestion des transactions

Références

- Approche objet
  http://www-igm.univ-mlv.fr/~dr/DESS/Objet/
- ORM
  http://www.objectmatter.com/vbsf/docs/maptool/ormapping.
- JPA – Informations JEE
  http://www.parleys.com/display/PARLEYS/Home
  - Conférence “Writing JPA Applications”, SpringOne
  - Conférence “JEE 5 Blueprints JPA”, JavaOne 2006
  - Conférence “Advanced Topic in JPA”, JavaZone 2007

Questions ?