Fabien Coelho
MINES ParisTech
fabien.coelho@mines-paristech.fr

Composé avec LaTeX

ORM – Object-Relational Mapping
programme objet & stockage relationnel

Concepts : objets vs relations

- données + fonctions
- programmation
- langage procédural
- typage fort, statique/dynamique

- classe, encapsulation
  listes, ensembles, tableaux
- référence/pointeurs
- héritage
- levée d'exceptions, à l'exécution
- modifications locales

- données + contraintes
- déclaration
- algèbre relationnelle
- typage fort
  . . . mais pas les mêmes types
- relation, tuples à plat (struct ?)
  scalaire et relations. . .
- associations
- héritage possible *(modèle OR)*
- contraintes d'intégrité déclarées
- transactions globales

ORM - opérations

(S)CRUD *(Search) Create, Read, Update, Delete*

code ORM très similaire, très répétitif, très spécifique

Génération automatique de code : portabilité

3 stratégies

1. Relationnel vers objet
2. Objet vers relationnel
3. Description vers relationnel et objet

Java JPA, Hibernate, iBATIS, MyBatis

Scala Squeryl, SORM, (Anorm)

Python Django, SQLAlchemy

Ruby Active Records, RBATIS

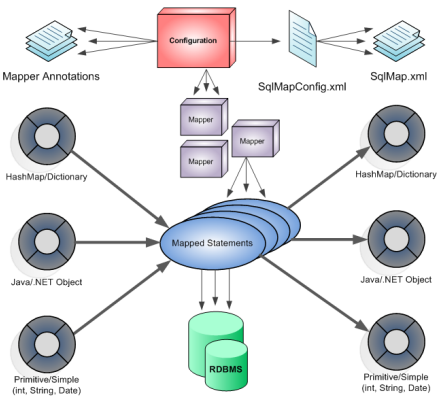
.NET NHibernate, iBATIS

Approche

- séparation code application / lien base de donnée
  couche d'abstraction qui cache JDBC et SQL
- philosophie base POJO : *Plain Old Java Object*
  vs objects contraints (héritage, méthode)
- déclaration via XML ou annotations. . .
  génération code Java & SQL, ou introspection. . .
  efficacité SQL ? en général, non :
  pourquoi une requête avec 100 résultats si je peux faire 100 requêtes ?
- services : connexion, cache (policies), pooling, prefetch. . .
  persistence *automatique* ou *à la demande*

MyBatis – SQL Mapping Framework for Java

- basé sur Apache iBATIS, fork en 2010 `http://mybatis.org`
  il existe aussi une version .Net
- fichiers XML pour l'environnement et les mappings
  contrôle complet des requêtes (appels de procédures. . .)
  support SQL dynamique : variantes de requêtes
  éventuellement spécifié avec des annotations
- utilisation par un *runtime*
  lecture fichiers XML
  configuration connexion et définition des tables/objets
- appel explicite des requêtes



doc MyBatis

MyBatis – fichier XML

```
<sqlMap namespace="StudentMapper">
  <!-- correspondance des attributs -->
  <resultMap id="studentResult" class="Student">
    <result property="id" column="pk"/>
    <result property="name" column="nom"/>
  </resultMap>
  <select id="allStudents" resultMap="studentResult">
    SELECT * FROM Student
  </select>
  <select id="studentById" parameterClass="int" resultClass="Student">
    SELECT ... FROM Student WHERE pk = #{id}
  </select>
  <insert id="insertStudent" parameterClass="Student">
    INSERT INTO Student(pk,nom) VALUES(#{id}, #{name});
  </insert>
  <update id="updateStudent" parameterClass="Student">
    UPDATE Student SET nom=#name# WHERE pk=#{id}
  </update>
  <delete id="deleteStudent" parameterClass="int">
    DELETE FROM Student WHERE pk=#{id}
  </delete>
</sqlMap>
```

MyBatis – Java annoté

— plutôt pour des requêtes assez simples

```
public interface StudentMapper
{
  @Select("SELECT * FROM Student WHERE pk=#{id}")
  Student selectStudent(int id);

  @Insert("INSERT INTO Student(pk,nom) VALUES(#{id},#{name})")
  @Options(useCache=true,flushCache=true,keyProperty="pk")
  void insertStudent(Student s);

  // etc.
}
```


MyBatis – code Java

```
SqlSession session = ...("student.xml");
StudentMapper sm = session.getMapper(StudentMapper.class);
List<Student> ls = sm.allStudents();
Student s = sm.studentById(12);
s.setNom("Calvin");
sm.updateStudent(s);
session.close();
```


MyBatis Generator

— dérivation des classes et requêtes
à partir du schéma de la base de données
— **DAO** – *Data Access Object* par table
— éventuellement, construction d'objets *au dessus*
requêtes plus complexes impliquant des jointures...

Ne jamais modifier du code généré automatiquement !

JPA – Java Persistence API

— Standard Java 5.0, package `javax.persistence`
— utilise des annotations Java...
— **implémentations** OpenJDA (Apache), Hibernate (Red Hat)

Red Hat – Hibernate

— similaire à MyBatis, mais SQL plutôt généré...
possibilité de ré-écrire les requêtes trop inefficaces
— est un *JPA provider*
— intégré à JBoss – implémentation J2EE Red Hat

Comparaison rapide

MyBatis simple, souple, SQL donc spécifique, contrôle

Hibernate standard, complet, rigide, portable mais HQL, communauté

Philosophies

Domaine application OO, DB simple stockage — OO > DB
— une seule application
— données non partagées entre applications
— intégrité assurée par l'application
— MySQL, Hibernate

Base de données a les données, utilisée en OO — DB > OO
— applications multiples
— données partagées
— intégrité assurée par la DB
— PostgreSQL, MyBatis

List of Slides