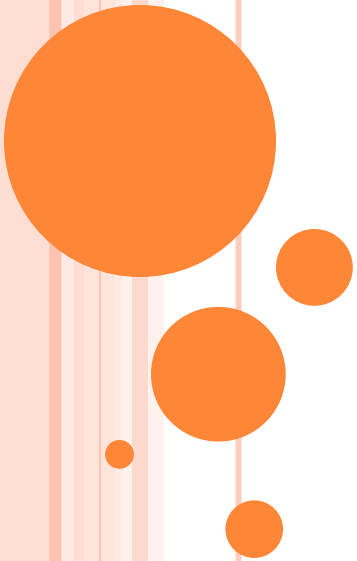

J2EE: Hibernate & JPA

Introduction

- Persistance d'Objet : Désigne le fait que des objets peuvent survivre à l'exécution d'une application. Ces données peuvent être enregistrées sur un support de stockage et récupérées par la suite.

- Le mappage Objet/Relationnel, désigne la persistance automatisée et transparente d'objets au travers d'une application, via des tables relationnelles au moyen de méta données décrivant ce mapping ➔ Il s'agit des ORM

Principales stratégies de persistance

- Java Database Connector via les DAO (JDBC)
- EJB Entity
- Les framework de type Hibernate/JDO/Castor,,,,

Hibernate

- Hibernate voit le jour en 2001

  - Open source
  - Propose un outil de mapping entre les objets (classes) et les données stockées dans une base de données relationnelles
  - Description de la correspondance Classes/Tables via fichier XML
  - Modulaire ⇔ adaptable aux environnements tant légers que lourds

Composition

Hibernate se compose de plusieurs modules développé par des équipes différents:

- Le Core: module principale
  - Transactions
  - SQL
  - Sessions
  - Cache objet

Composition (2)

- Annotations:
  - Apporte le support des Annotations
  - Supprime la description des relations entre Entités Table en XML.
- Entity Manager
  - Permet la prise en charge de JPA

Composition (3)

- Validator
  - Validation des contraintes d'entités au sein de la DB
  - Annotations
  - Vérification de plages de valeurs authorisées, formats de chaine de caractère, valeurs nulles,…
- Autres modules: Shards, Search, Tools,…

Ensuite?

- Succès d'Hibernate en environnement Java
  - Efficacité
  - Rapidité
  - Maintenabilité
- Développement de Nhibernate Tools pour environnement .NET depuis Visual Studio 2005 +

JPA

- JPA = Java Persitence API
- Version actuelle de l'API: 2.1
- Interface de programmation java permettant d'organiser les données relationnelles au sein des applications

Persistance avec JPA

La persistance couvre trois zone:

- L ’API (package javax.persistence)
- Le JPQL (Java Persistence Query Langage)
- L’Objet/Les méta données relationnelles

Fonctionnement

JPA fonctionne essentiellement avec les annotations introduites dans Java 5

- Définition simplifiée des objets métier ➔ interface entre la DB et l'application dans le cadre d'un ORM

Les annotations

Voici les principales annotations qui nous concernent lors du mapping:

Core	Relationships	Inheritance
@Entity	@ManyToOne	@MappedSuperclass
@Table	@OneToOne	@Inheritance
@Id	@OneToMany	@DiscriminatorColumn
@Basic	@JoinColumn	@DiscriminatorValue
@Column		
@Transient		
@Enumerated		
@Temporal		
@Type		

Mappage des associations

Elles se font au moyen des annotations de type « Relationships »

- @OneToMany ⇔ @ManyToOne
  - Nous retrouvons un Object du coté One de la relation
  - Nous retrouvons un Collection (List<T>) du coté Many de la relation

- @OneToOne
  - Maintient une relation avec les Foreign Keys en DB

- @ManyToMany
  - Va utiliser une table de Mapping dans la DB

Hibernate & JPA

- La spécification de JPA est implémentée par Hibernate (entre autre)
- L‘avantage de cette utilisation: un seul fichier de configuration XML (persistence.xml) généré lors de la conversion du projet en projet JPA
- Le fournisseur de persistance doit être indiqué dans ce fichier qui peut également comporter d’autres informations

PERSISTENCE.XML

persistence.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="pu" transaction-type="RESOURCE_LOCAL">
        <provider>org.hibernate.ejb.HibernatePersistence</provider>
        <class>model.People</class>
        <class>model.Role</class>
        <class>model.Status</class>
        <class>model.User</class>
        <properties>
            <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/bfbt1"/>
            <property name="javax.persistence.jdbc.user" value="root"/>
            <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
        </properties>
    </persistence-unit>
</persistence>
```

Entity Manager

- Import de la classe:

  import *javax.persistence.EntityManager;*

  ou

  import javax,persitence.;*

- L' EM est l'interlocuteur principal pour la gestion des entités, il fournit les méthodes pour les gérer
  - Persist(), Merge(), Find(), FindAll(),…

Entity Manager (2)

- La méthode « createEntityManager() » de la classe EntityManagerFactory crée un EntityManager
- Ce dernier est supprimé avec la méthode « close() » de la classe « EntityManager »
- Il n'est plus possible de l'utiliser ensuite, il faudra refabriquer l'EM.

Fabriquer l'EntityManager

- Depuis la Classe Persitence, appeler la fabrique d'EntityManager au moyen de la méthode « createEntityManagerFactory('p-u') »

- Comme vous le voyez cette méthode prend un paramètre, il s'agit du nom de votre unité de persistance (Persistance Unit) déclarée dans votre fichier « persistence.xml »

La classe emFactory()

```
package nom.devotre.package;
import javax.persistence.*;


public class EmFactory {
      public static EntityManager CreateEm()
      {
          EntityManagerFactory emf =
                Persistence.createEntityManagerFactory(«nomdevotrePU»);
          EntityManager em = emf.createEntityManager();
          return em;
      }
}

// Cette classe permet à travers toute votre application de créer un Object
// EntityManager
```

Hibernate VS JPA

- Similitudes
  - Permettre la persistance
  - ORM Déclaratif
  - Adopter un langage de requêtes
- Différences
  - JPA est un API
  - Hibernate est un Framework
  - JPA est une base des standard J2EE 5
- Hibernate peut être utilisé comme fournisseur de persistance pour JPA
- JPA est plus simple à mettre en œuvre que Hibernate (Entity Manager, pas de fichiers de mappage, …)