JAVA Persistence API (JPA)

**Eric Sultan** [e.sultan@ajc-ingenierie.fr](mailto:e.sultan@ajc-ingenierie.fr) 0645104506

# Introduction & Définitions

* API (Application Programming Interface) / ORM (Object-Relational Mapping) destinée à la persistance d’objet (contrairement à transient)
* Stockage d’informations en Bases
* JPA génère les requêtes SQL pour nous
* JPA est une sous-spécification d’EJB3 (Entreprise Java Beans)
* Inclus dans JEE (Java Entreprise Edition)
* Hibernate & ClipsLinq ? sont des implémentations JPA
* JPA est une sorte de définition via interfaces, pas de code.
* Avec ça on code des POJO (Plain Old Java Object)
* Avantage de ces classes pour les tests = on limite les facteurs de modifications = on isole plus rapidement le problème = implémentations de couches
* **EJB1 à fuir, très mal foutu… EJB2 c’est chaud, EJB3 c’est calme.**
* L’idée principale de JPA c’est de **cacher le SQL**. On passe par le **JPQL (Java Persistence Query Language)**. Cela retourne directement les **objets**.
* **Classe maître = EntityManager**
* De cette classe on fait une transaction avec L’**EntityTransaction,** puis ça devient des **EntityBeans**

# Entity Beans

* API EntityManager sert à manipuler et créer des instances d’EntityBeans7
* Application-managed (nous-même) ou Container-managed (appli)
* EntityBean = objet java ordinaire avec des annotations spécifiques à JPA/Hibernate (**@Entity** avant la classe par exemple = je veux l’img de cette classe en BDD)
* Puis méthode de persistance pour enregistrer l’objet en base (**persistance**)
* Règles d’un JavaBeans = **constructeurs, getters & setteurs**
* Par défaut, tous les attributs d’une entité sont persistants, sinon on ajoute **@Transcient**