Java Persistent API

Auteur : PEREZ Noël

nperez@hinnoya.fr

Date : Octobre 2009

Hinnoya
INNOVONS ENSEMBLE

Plan

- Principes des Frameworks de persistence
- Historique
- JPA 1.0
- JPA 2.0
- Demo
- Questions?

BIO

Noël Perez

- Architecte Java chez Hinnoya

http://www.hinnoya.fr

- Formateur :
  - Java/JEE/Struts/Hibernate (bientôt JPA)

http://www.proatis.fr

- Auteur :
  - Article sur JPA paru dans linuxdevjournal mi 2008

http://www.pere-nono.net

Groupe Hinnoya

Hinnoya Centre Est

Centre de services informatique Lyonnais

Pôle .Net	Pôle Php	Pôle Java
Technos : ASP/C#, DotNetNuke, …	Technos : Symphony, Joomla, PHP5, …	Technos : JEE, Struts, Spring / SEAM, GWT, ...

Délégation de compétences | **Conseil** | **Centre de services**

Hinnoya Training Center

Organisme de formation agrée

- Formation téchnologique
- Gestion de projet
- Bureautique
- Applicatif métier

Hinnoya
INNOVONS ENSEMBLE

Plan

- **Principes des Frameworks de persistence**
- Historique
- JPA 1.0
- JPA 2.0
- Démo
- Questions?

Problématique

- Application inconcevable sans sauvegarde des données
- Beaucoup de codes répétitifs :
  - ouverture de connexions,
  - gestion de requêtes multiples,
  - traitement du résultat, …
- Sauvegarde porte sur des objets métiers
  → 1 objet métier = 1 table

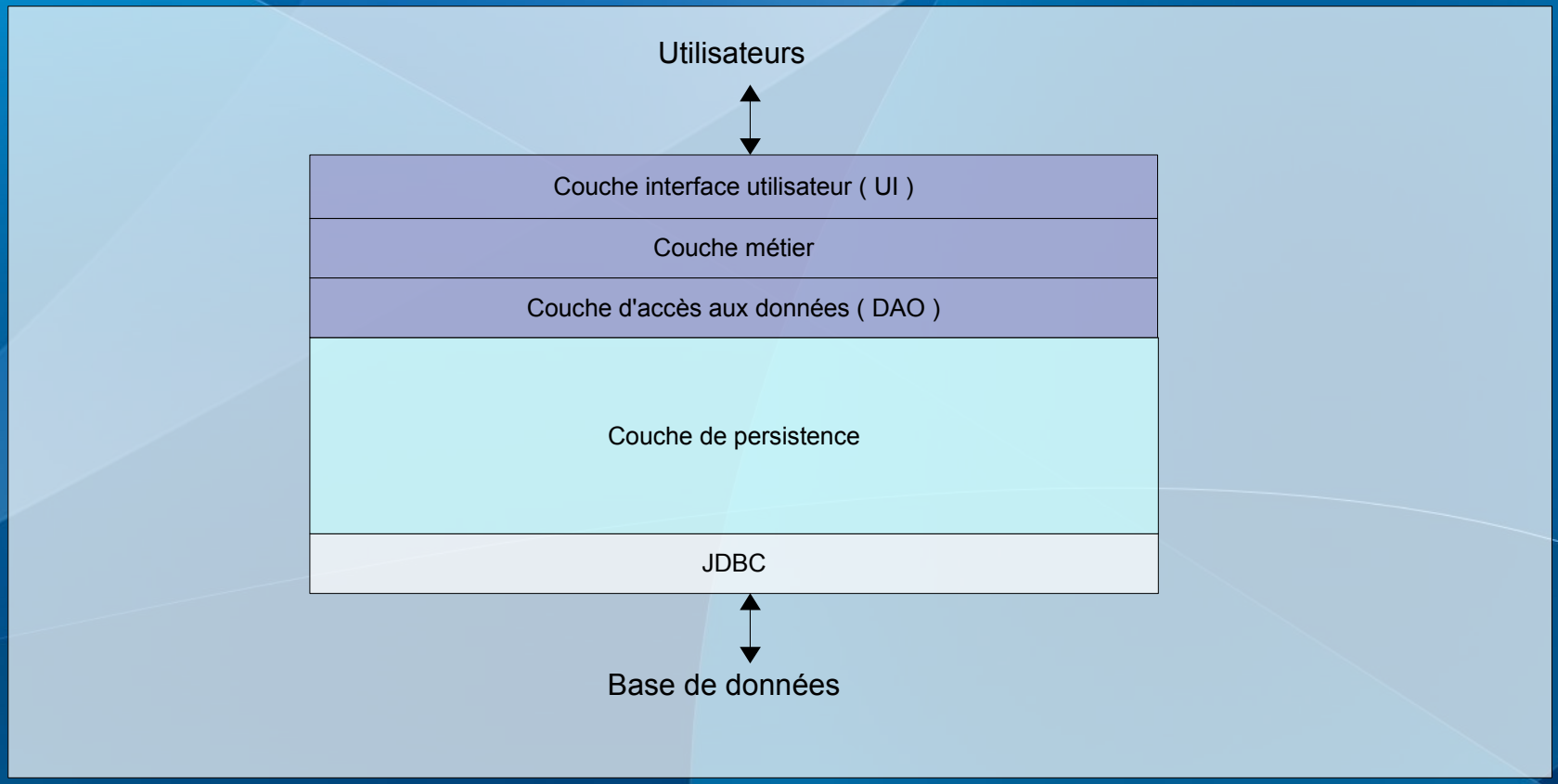
Problème gestion JDBC

- Besoin d'experts pour créer schémas et requêtes de la base,
- difficulté à maintenir ce même schéma et ces requêtes,
  - Opérations coûteuses
  - Peu d'intérêt, très répétitive

- dépendance vis à vis de la base initiale

  → peut nuire à la montée en charge d'une application.

Couches



Solution : frameworks de persistence

- Travail à partir de POJO
- Ajout d'informations permettant de mapper les propriétés d'objets avec les colonnes des tables
- Possibilité de demander à sauver un objet, le supprimer, le mettre à jour
- Possibilité d'interroger la base en manipulant les objets (pur java)

Principe technique

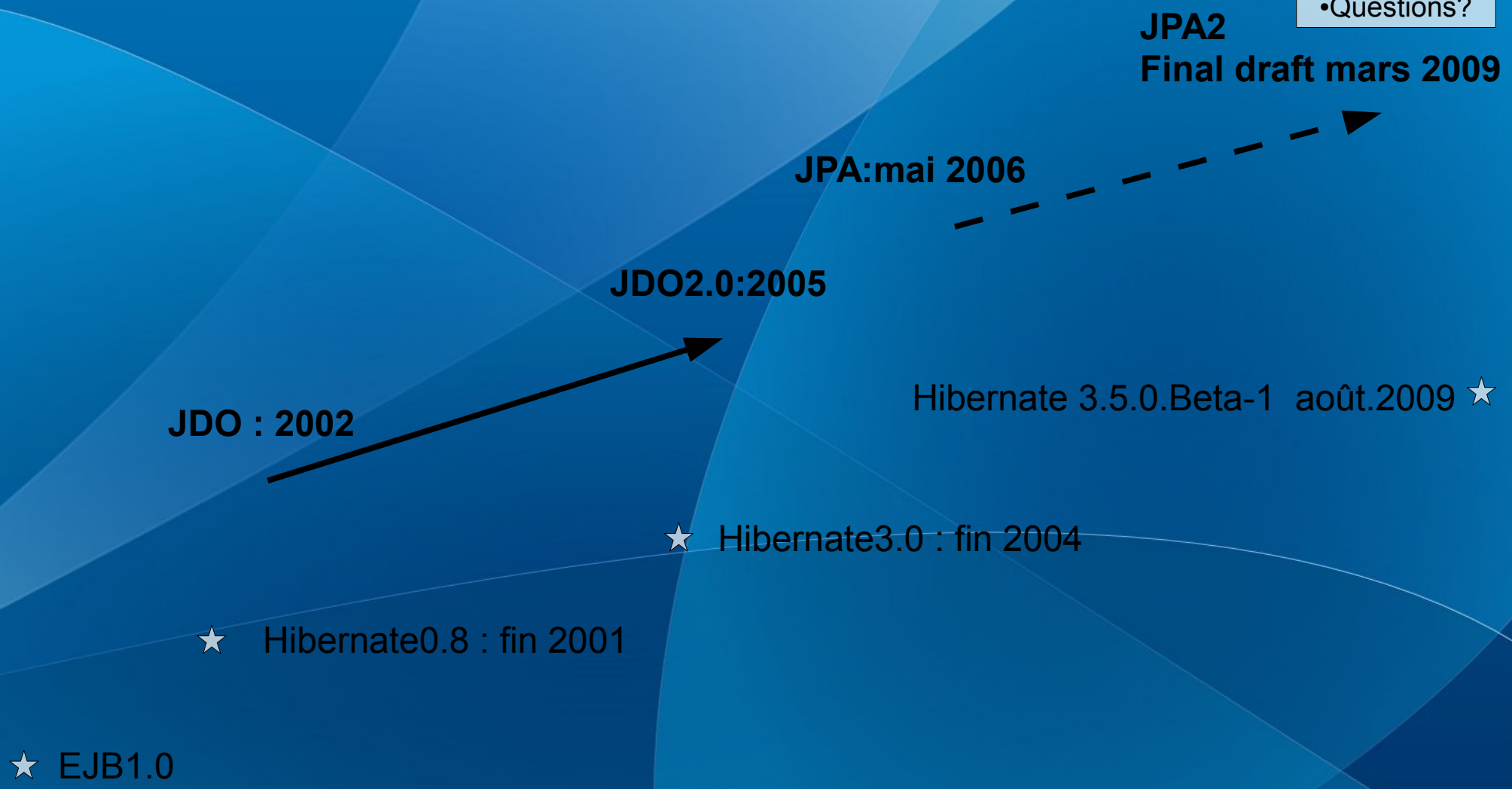
- 1 classe à persister = 1 description du mode persistance
  - Indique la table
  - Indique les correspondances de champs
    - Type
    - Obligation
    - Clé
    - Lien avec d'autres objets
- 1 fichier de configuration générale décrivant l'accès à la base de données

Hinnoya
INNOVONS ENSEMBLE

Plan

- Principes des Frameworks de persistence
- **Historique**
- JPA 1.0
- JPA 2.0
- Démo
- Questions?

Historique

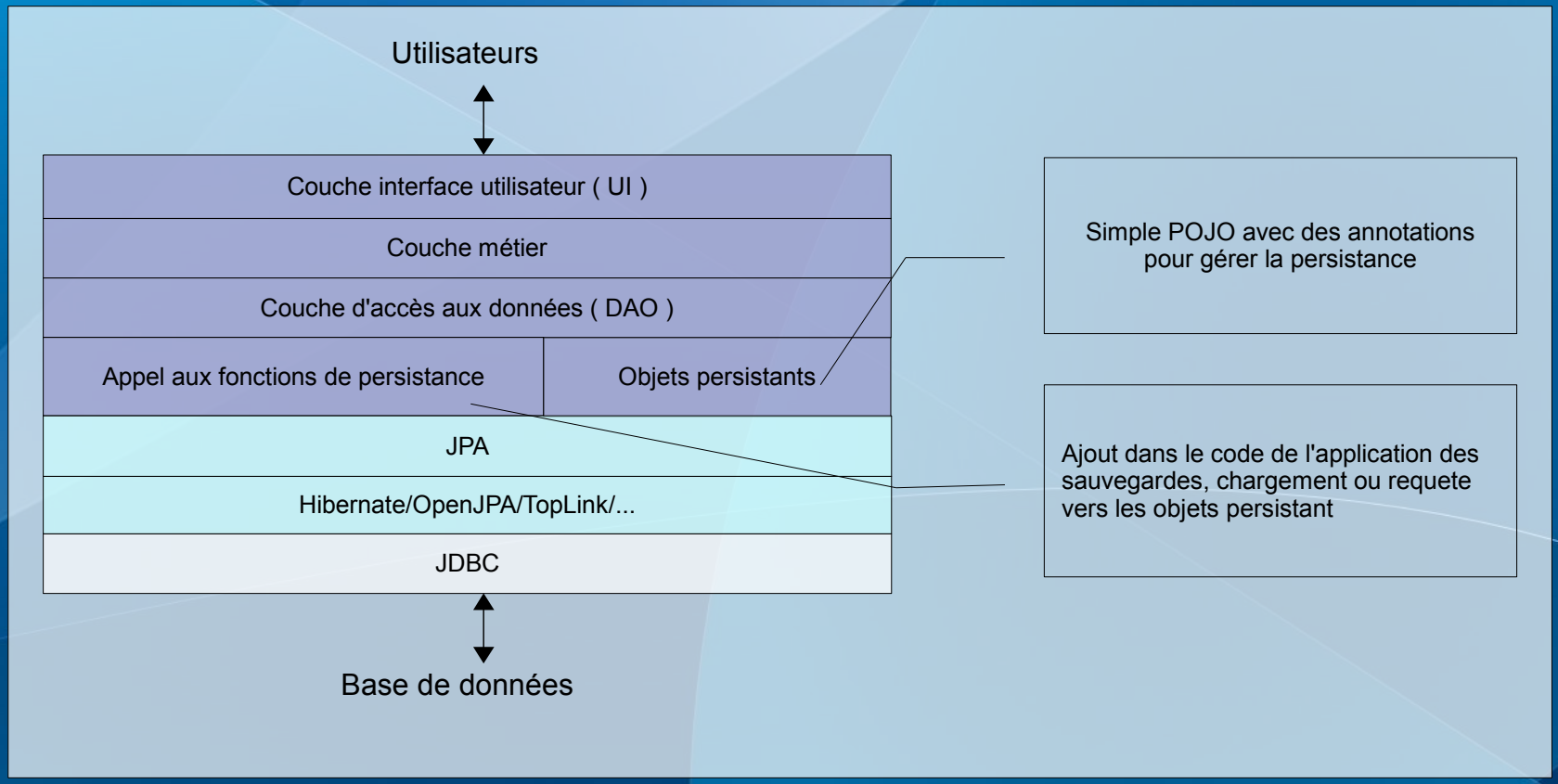


Participant JSR 317

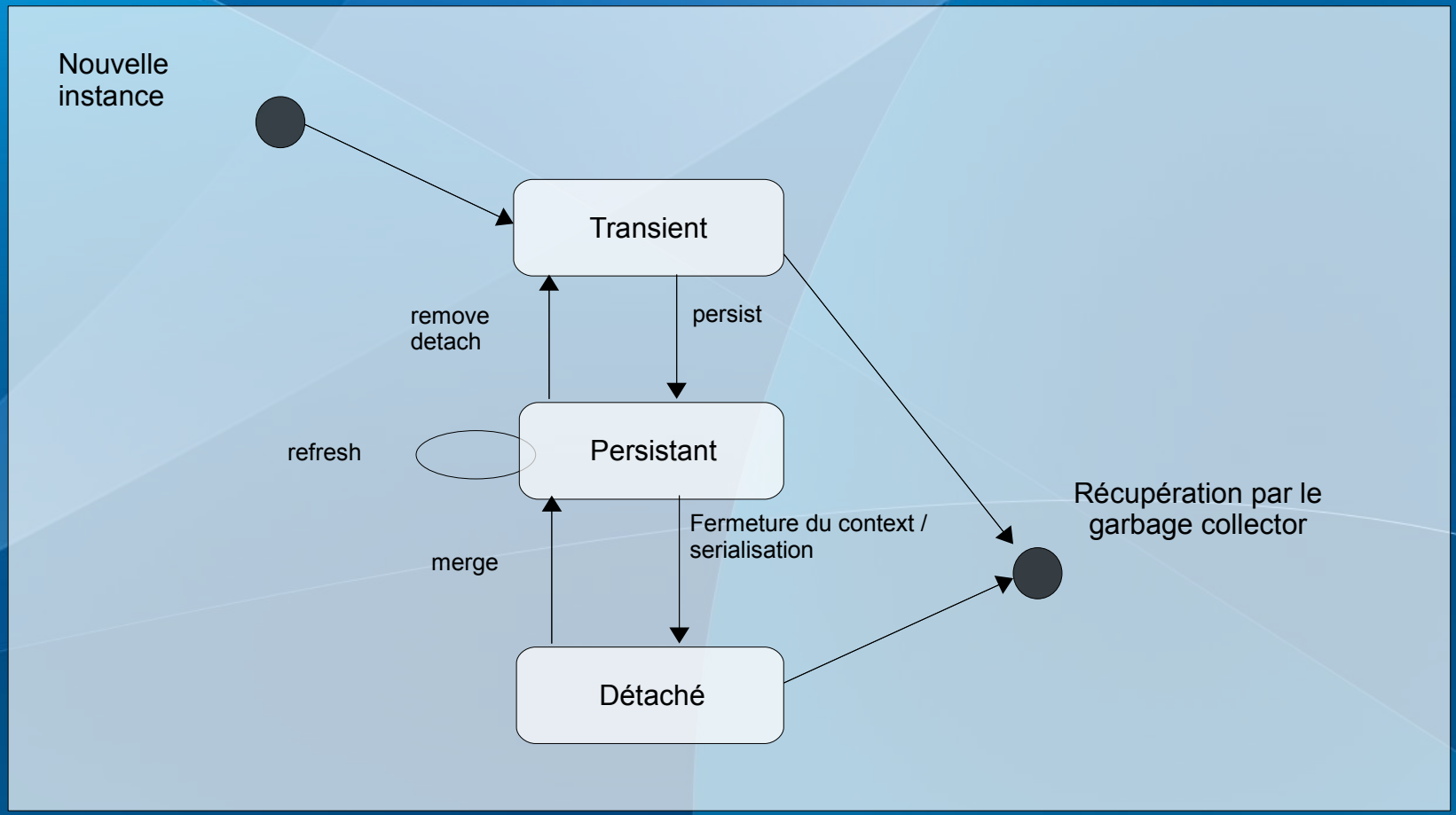
Plan

- Principes des Frameworks de persistence
- Historique
- **JPA 1.0**
- JPA 2.0
- Demo
- Questions?

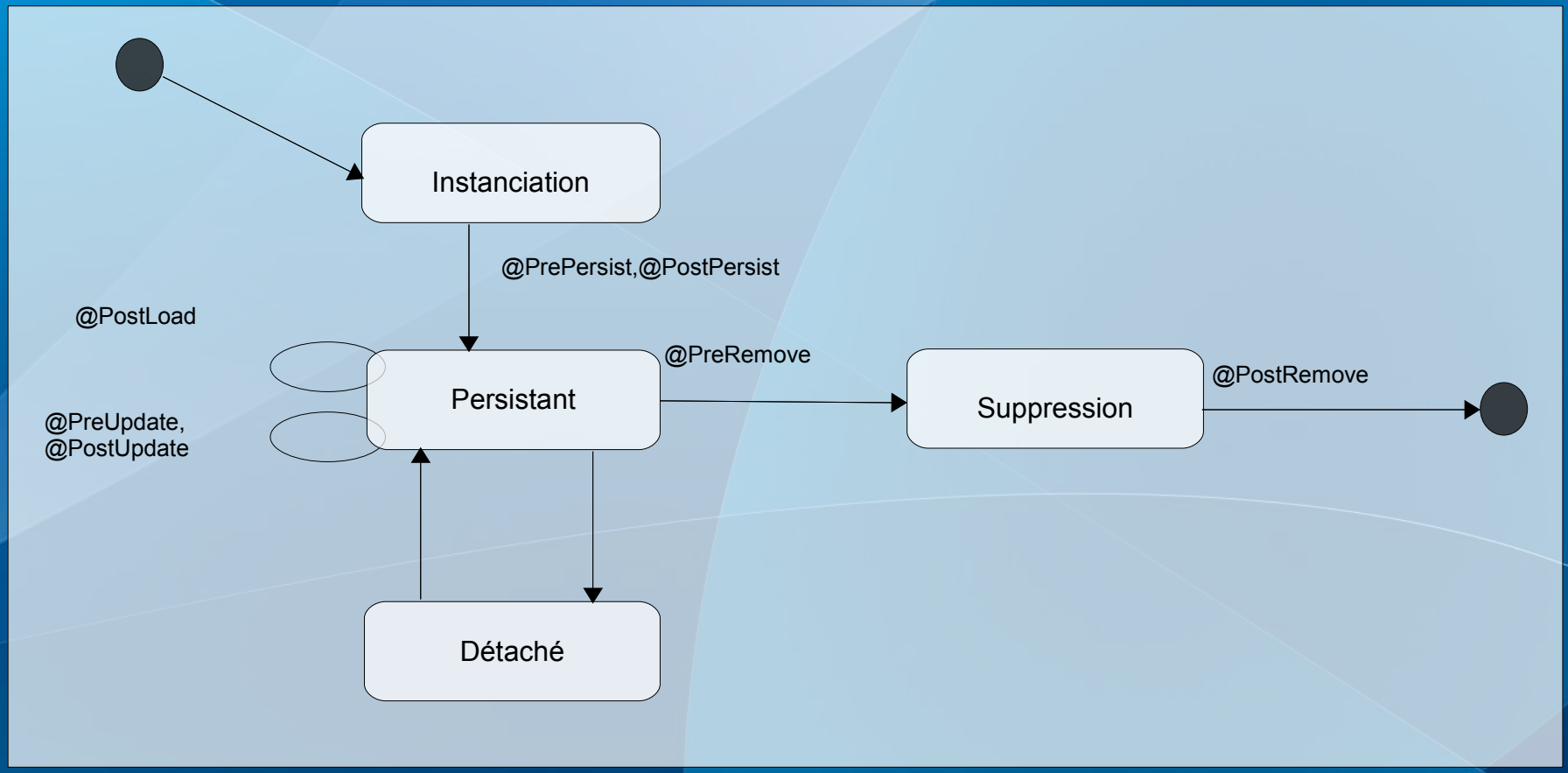
Couches



Cycle de vie


Nouvelle instance
Transient
remove
detach
persist
refresh
Persistant
Fermeture du context / serialisation
merge
Détaché
Récupération par le garbage collector

Injections (callback method)



Injections 2/2

- Tient compte de l'héritage
  - Les injections classe mère en dernier
  - Possible de désactiver les appels de la classe mère
- Tient compte de l'ordre dans le fichier (même injection présente plusieurs fois)

Requetage

- SQL

- JPQL

```
select distinct joueur
from Club club,
   Personne joueur
where joueur member of club.organisateurs
```

Fonctionnalités standards

- Lazy strategie sur les Fetch
- Traitement en cascade
- Gestion des transactions
- Lock d'objet

Autres fonctionnalités

- Création de requêtes nommées (Named Queries)

Plan

- Principes des Frameworks de persistence
- Historique
- JPA 1.0
- **JPA 2.0**
- Démo
- Questions?

Hinnoya
INNOVONS ENSEMBLE

JPQL

Aucune nouveauté depuis la version 1.0

Criteria API 1/3

- Requetage en objet
- Construction de requêtes dynamiques très aisée
- Voir modification après coup

- 2 modes de création :
  - Metamodel
  - Réference aux attributs

Criteria 2/3

En utilisant le metamodel :

```
   CriteriaQuery q = qb.create();
Root<Customer> cust = q.from(Customer.class);
Join<Order, Item> item =
cust.join(Customer_.orders).join(Order_.lineitems);
q.select(cust.get(Customer_.name))
.where(
qb.equal(item.get(Item_.product).get(Product_.productType),
"printer"));
```

Order *est un objet et la correspondance de sa classe dans le metamodel est* ***Order_***

Criteria 3/3

- En utilisant les réferences aux attributs :

```
QueryBuilder qb = ...
CriteriaQuery q = qb.create();
Root<Customer> cust = q.from(Customer.class);
Join<Order, Item> item =
cust.join("orders").join("lineitems");
q.select(cust.get("name"))
.where(
qb.equal(item.get("product").get("productType"),
"printer"));
```

Cached API

```
@Entity
@Table(schema="jpa", name = "participe")
@Cacheable(true)
public class Participe {
```

Dans persistence.xml : <caching>ALL</caching>

- ALL
- NONE
- ENABLE_SELECTIVE
- DISABLE_SELECTIVE

Validation (javax.validation)

- Optionnel
  - TraversableResolver accessible par l'EntityManager
  - @valid
- javax.persistence.ValidationMode
  - **Auto** (defaut) : validation utilisée si présente sans exeception
  - **Callback** : validation obligatoire
  - **None**
- Completée par la JSR-303 (Bean validation)

Autres nouveautés

- Apparition de la méthode ”detach”
  - Passe objet en transient
- Changement de signification de la méthode ”remove”
  - Supprime les entrées associées en base
- cascade=DETACH
- ValidatorFactory
- Possibilité d'accéder au metamodel (disponible avec hibernate mais pas en JPA1)

Implementations disponibles

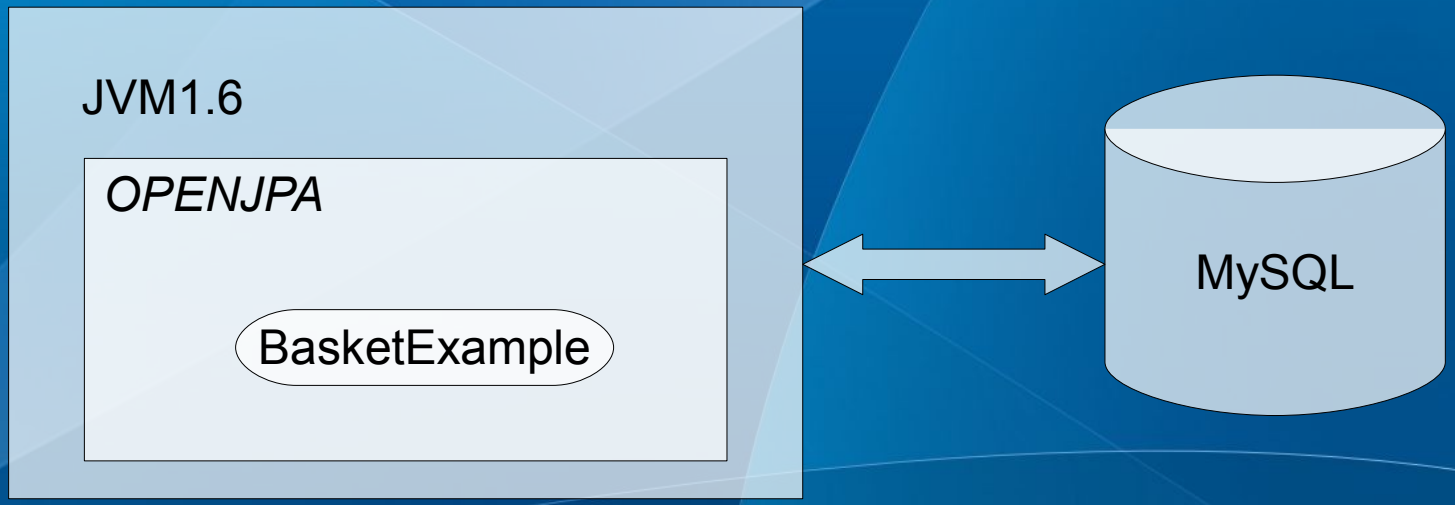
- OpenJPA2.0 - 12/2009 (Geronimo / Weblogic)
  - Version déjà disponible et fonctionnelle (partiel)
- Hibernate - (JbossAS-5.2.0.Beta1 - non grand public)
  - preview disponible, non fonctionnelle JPA2
- Glassfish V3 (preview était disponible)
- EclipseLink2.0 11/2009
- ...

Hinnoya
INNOVONS ENSEMBLE

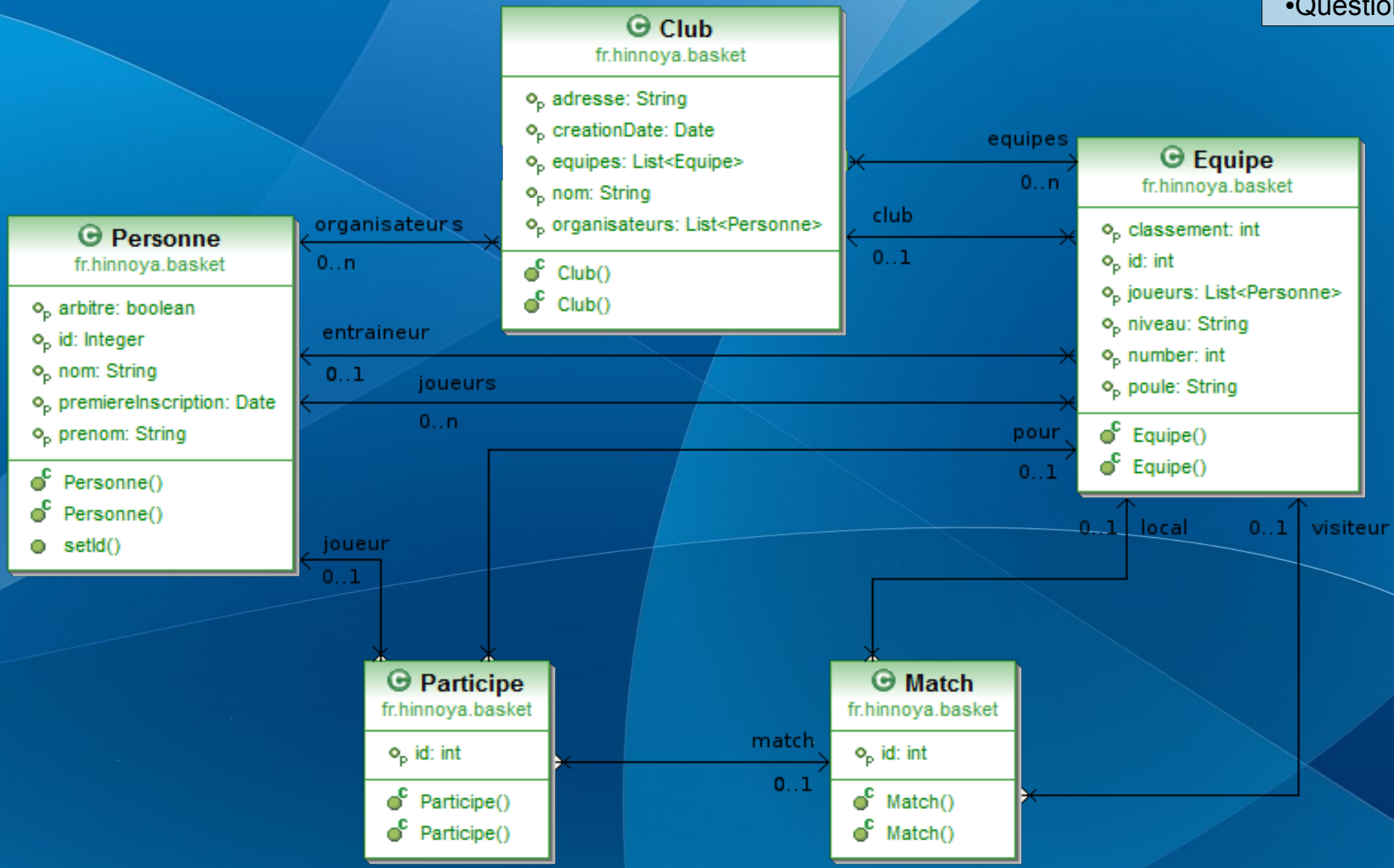
Plan

- Principes des Frameworks de persistence
- Historique
- JPA 1.0
- JPA 2.0
- **Démo**
- Questions?

Démo : architecture



Démo : modèle



Pour aller plus loin

References :

- *JSR 220: Enterprise JavaBeans™, Version 3.0*
- *JSR 317: Java™ Persistence API, Version 2.0*

Outils :

- *OpenJPA: http://openjpa.apache.org/*

Articles sur JPA :

- *Initiation : Java Persistence API : Persistance universelle - linuxdevjournal - Perez N.*
- *Avancé : http://java.sun.com/javaee/5/docs/tutorial/doc/bnbpz.html*

Hinnoya
INNOVONS ENSEMBLE

A vous de jouer :

Question?