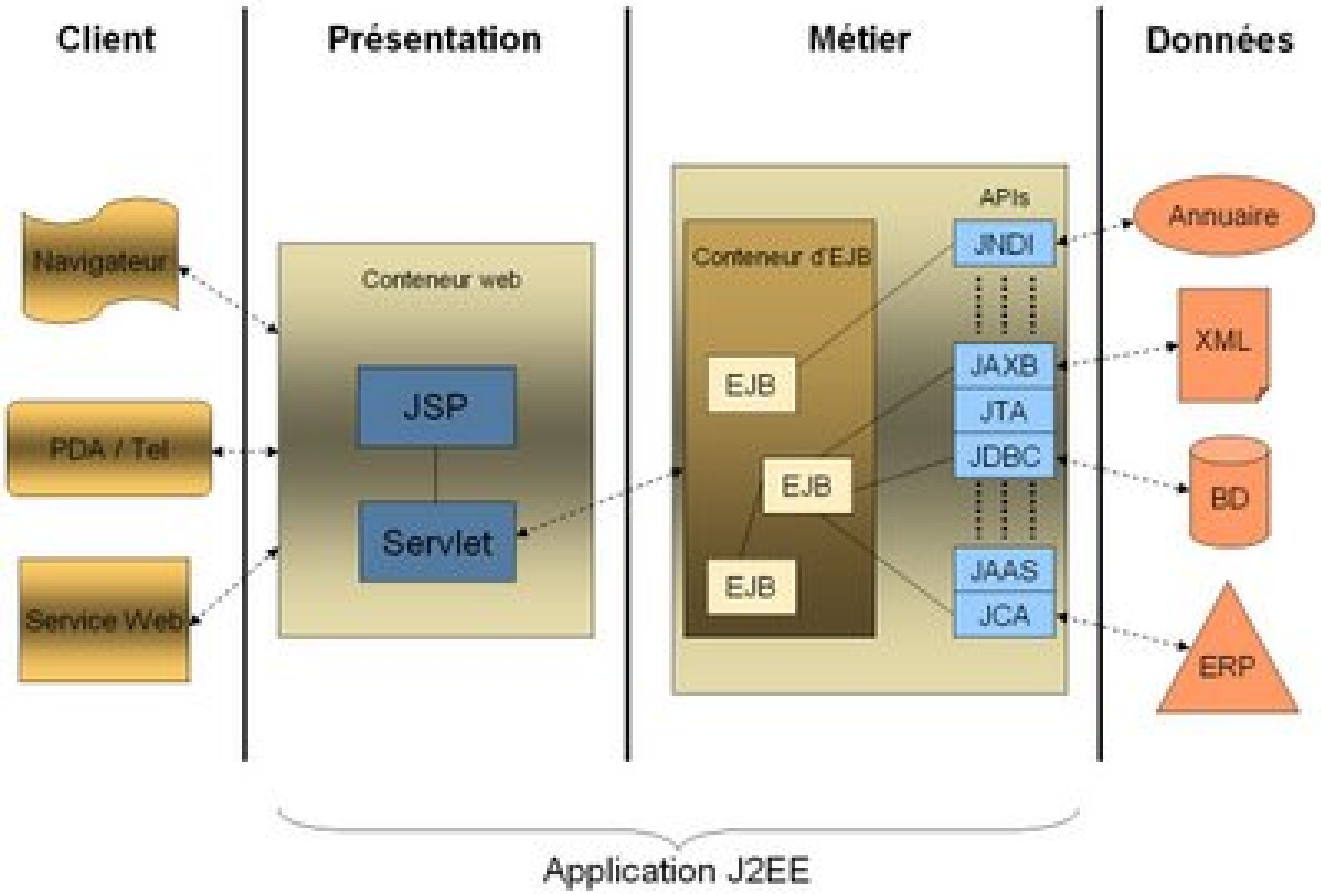
EJB

A.-E. Ben Salem

LRDE and LIP6

17 Octobre 2011

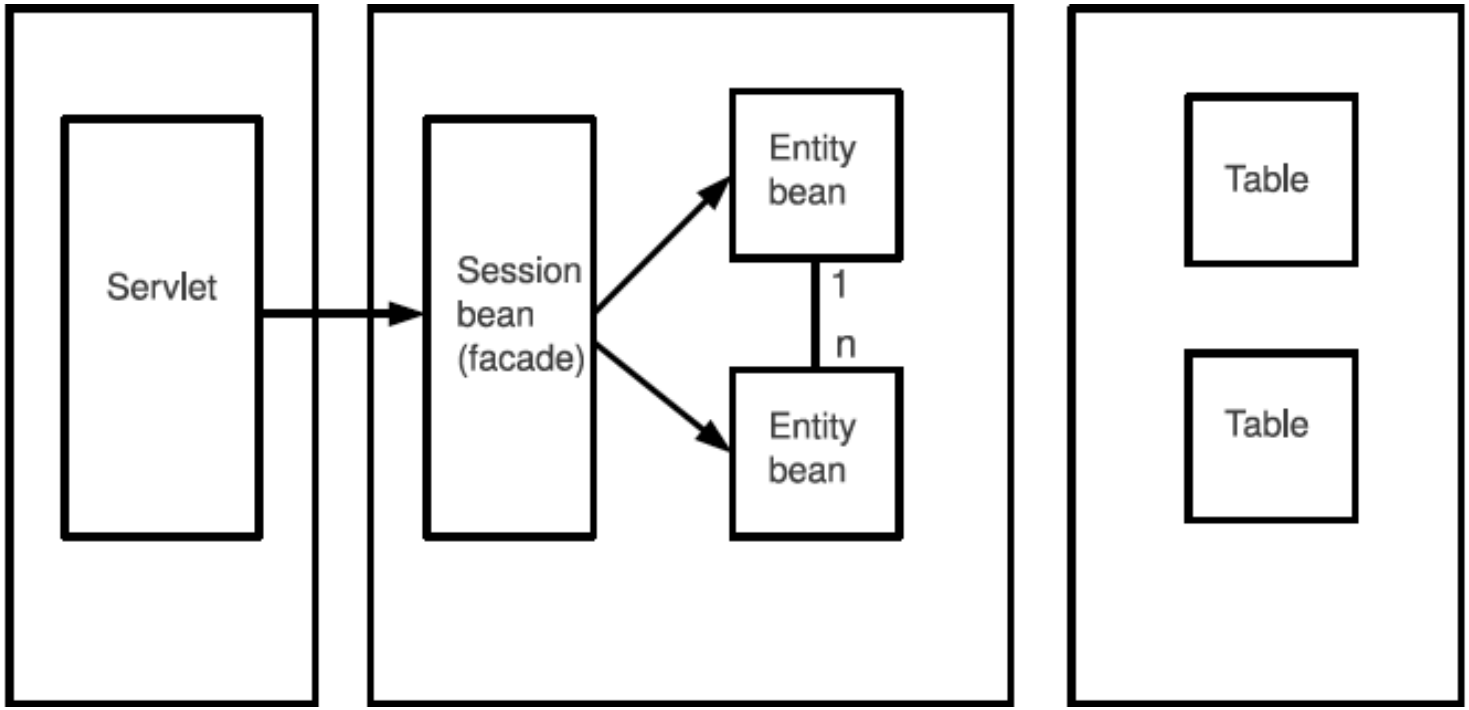
Rappel


Client
Présentation
Métier
Données
Navigateur
PDA / Tel
Service Web
Conteneur web
JSP
Servlet
Conteneur d'EJB
EJB
EJB
EJB
APIs
JNDI
JAXB
JTA
JDBC
JAAS
JCA
Annuaire
XML
BD
ERP
Application J2EE

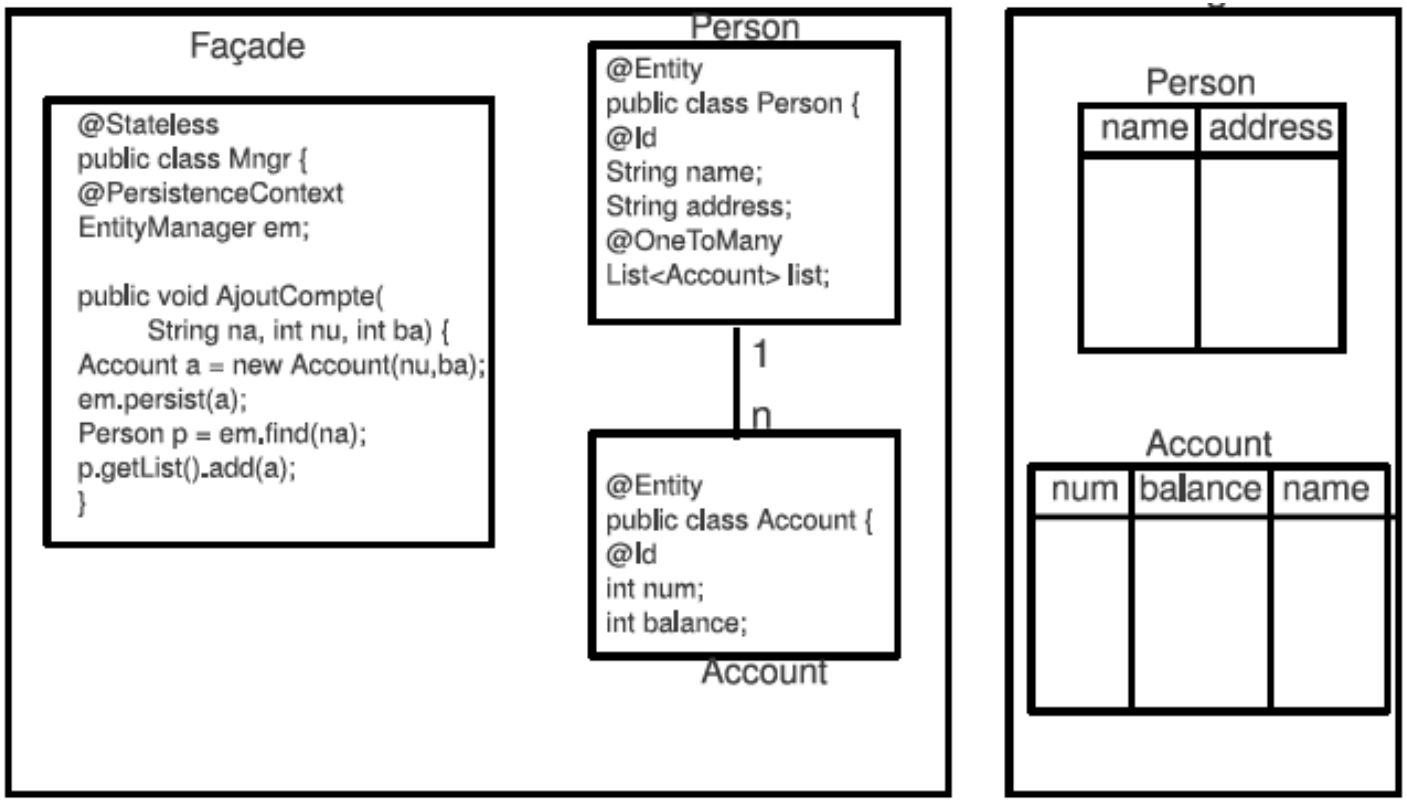
Présentation

- EJB : Enterprise Java Beans
- **Un "composant" qui encapsule de la logique applicative**
  - accés a une base de données
  - gestion de transactions
  - contrôle de concurrence
- Les services systèmes sont fournis par le container:
  - persistence
  - transaction
  - securite
  - cycle de vie
- Utilisation massive des annotations Java sur des POJO
  - separation des concepts metiers et non fonctionnels
  - on garde la puissance objet (héritage, . . .)
- Occupe la partie modéle du design-patern MVC

Architecture



Exemple


Façade
@Stateless
public class Mngr {
@PersistenceContext
EntityManager em;
public void AjoutCompte(
String na, int nu, int ba) {
Account a = new Account(nu,ba);
em.persist(a);
Person p = em.find(na);
p.getList().add(a);
}
Person
@Entity
public class Person {
@Id
String name;
String address;
@OneToMany
List<Account> list;
1
n
@Entity
public class Account {
@Id
int num;
int balance;
Account
Person
name
address
Account
num
balance
name

Annotations Java 5

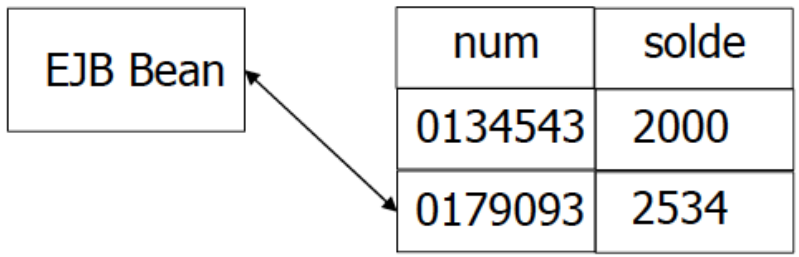
- Mécanisme standard dans le langage Java depuis version 5 (1.5)
- Idée similaire aux commentaires Javadoc
  - informations attachées à des éléments de programme (classe, méthode, attributs, ...)
  - pour ajouter de l'information sur cet élément
- @Identificateur
- Avec des paramètres : @Identificateur(attribut=value,...)
  - types autorisés:
    - primitifs, String, Class, annotation
    - tableaux de primitifs, String, Class, annotation
- Eventuellement plusieurs annotations par élément

Exemple:

```
@Entity
@Table(name="STOCK")
public class StockEntityBean { . . . }
```

Entity Bean (EJB Entity)

- Représentation d'une donnée manipulée par l'application
- Donnée typiquement stockée dans un SGBD (accessible en JDBC)
- Correspondance Objet $\leftrightarrow$ tuple relationnel (mapping O/R)

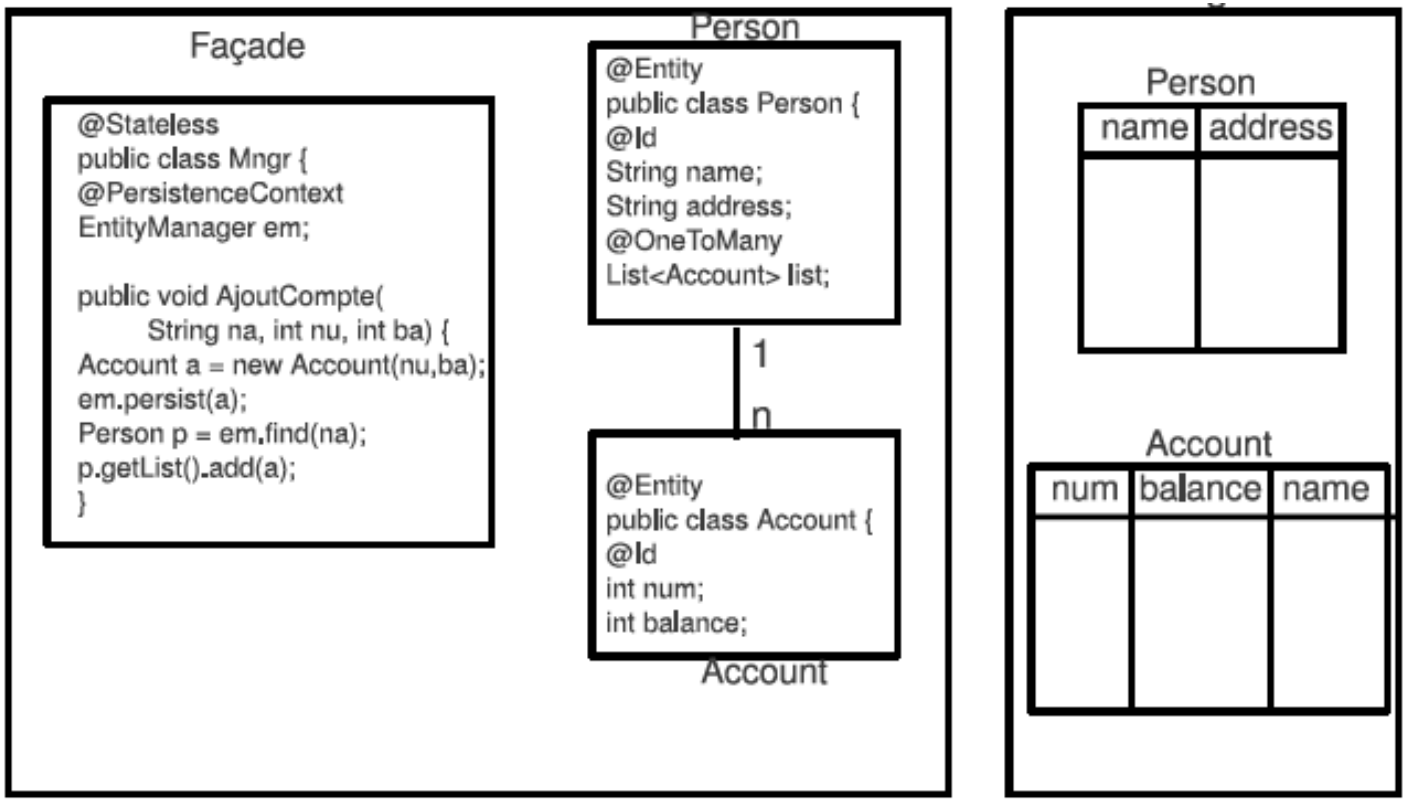


- Avantage : manipulation d'objets Java plutôt que de requêtes SQL
- Mise en oeuvre à l'aide:
  - d'annotations Java 5 et de la généricité Java 5
  - de l'API JPA (Java Persistence API): EntityManager, ...

Entity Bean (développement)

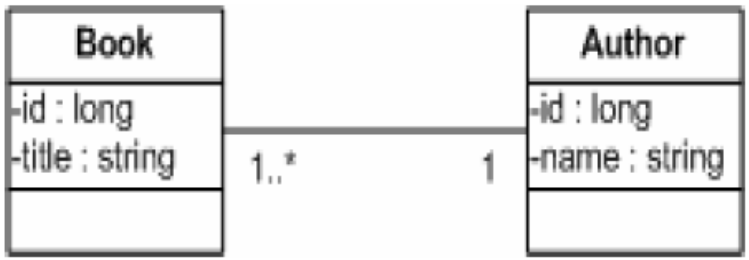
- Annotation @Entity : déclare une Classe correspondant à un entity bean
- Chaque Classe d'Entity Bean est mise en correspondance avec une Table de la BD:
  - par défaut Table avec même nom que la Classe
  - sauf si annotation @Table(name="...")
- mapping des colonnes des tables:
  - par défaut colonne avec même nom que l'attribut de la Classe
  - sauf si annotation @Column(name="...")
  - annotation @Id : définit une clé primaire
  - colonne auto-incrémenté: Annotation @GeneratedValue Exemple: @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "book_sequence")

Exemple


Façade
@Stateless
public class Mngr {
@PersistenceContext
EntityManager em;
public void AjoutCompte(
String na, int nu, int ba) {
Account a = new Account(nu,ba);
em.persist(a);
Person p = em.find(na);
p.getList().add(a);
}
Person
@Entity
public class Person {
@Id
String name;
String address;
@OneToMany
List<Account> list;
1
n
@Entity
public class Account {
@Id
int num;
int balance;
Account
Person
name
address
Account
num
balance
name

Entity Bean: mapping OneToMany

Mapping “OneToMany”: Relation (1 – n) entre 2 Tables (utilisant le “FOREIGN KEY”)

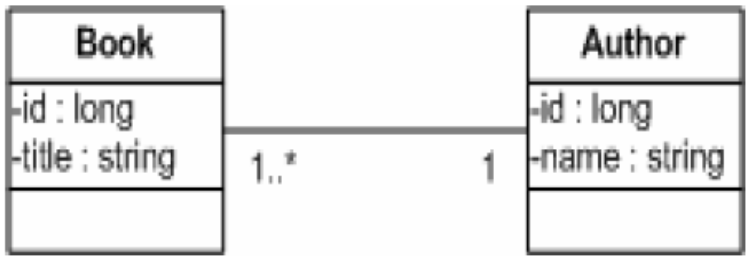



```
@Entity
public class Author {
    private long id;
    private String name;
    private Collection<Book> books;
    public Author() { books = new ArrayList<Book>(); }
    public Author(String name) { this.name = name; }
    @OneToMany
    public Collection<Book> getBooks() { return books; }
    public void setBooks(Collection<Book> books) { this.books=books; }
    ...
}
```

Entity Bean: mapping ManyToOne

Mapping “ManyToOne”: Relation (n – 1) entre 2 Tables (Relation inverse du “OneToMany”):

```
@Entity
public class Book {
    private long id;
    private Author author;
    private String title;
    public Book() {}
    public Book(Author author, String title) {
        this.author = author;
        this.title = title; }
    @ManyToOne
    @JoinColumn(name="Author_id")
    public Author getAuthor() { return author; }
    public void setAuthor(Author author) { this.author = author; }
    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }
    ...
```



Nom de la colonne de jointure

Session Bean (EJB Session)

- Session Bean : représente un traitement (services fournis à un client)
- Stateless session bean
  - sans état
  - ne conserve pas dínformation entre 2 appels successifs
  - 2 instances quelconques dún tel bean sont équivalentes
- Stateful session bean
  - avec un état (en mémoire)
  - similaire session servlet/JSP
  - même instance pendant toute la durée d'une session avec un client
  - 1 instance par client

Utilisation d'un EntityManager (Gestionnaire d'entités) dans un "Session Bean"

Exemple de Session Bean (Facade) : création de trois enregistrements dans la table des livres

```
@Stateless
public class MyBean implements MyBeanItf {
    @PersistenceContext
    private EntityManager em;
    public void init() {
        Book b1 = new Book("Honore de Balzac","Le Pere Goriot");
        Book b2 = new Book("Honore de Balzac","Les Chouans");
        Book b3 = new Book("Victor Hugo","Les Miserables");
        em.persist(b1);
        em.persist(b2);
        em.persist(b3);
} }
```

Appel d'un Session Bean à partir d'une Servlet

Client local (colocalisée sur le même serveur que le Bean) = une Servlet possédant:

- un attribut de type l'interface "Local" du Bean:

```
private  MyBeanItf  myBeanFacade;
```

- annoté par @EJB (injection de dépendance)

```
public class ServletClient extends HttpServlet {

    @EJB
    private MyBeanItf myBeanFacade;

    protected void doGet(...) {            ...
        List<Book> books = myBeanFacade.findAllBooks();
        ...
    }
}
```

Archives Ĺ déployer dans les 2 Conteneurs