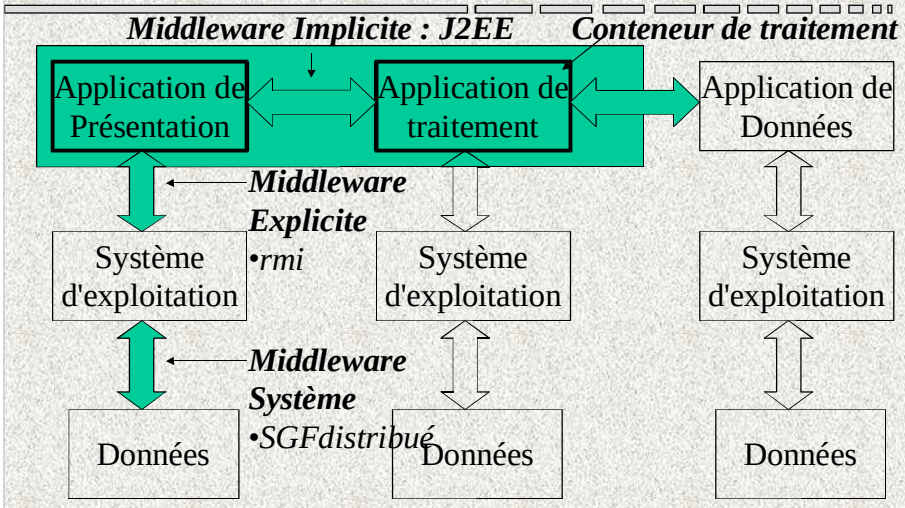
Les Ejb

Les Enterprise Java Bean

Répartition d'une application



Les composants

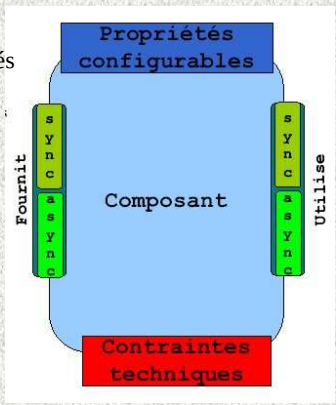
- Quoi ?
  - Définition usuelle
    - module logiciel autonome pouvant être installé sur plusieurs plates-formes
    - qui exporte différents attributs, propriétés ou méthodes
    - qui peut être configuré
    - qui peut être transporté / déployé
  - But
    - Brique de base pour concevoir des applications par composition

Les approches à composants

- Exemple :
  - Java Beans : Composant interface utilisateur
  - Enterprise JavaBeans :
  - Objets
  - CORBA
  - Avalon
  - Fractal
  - …

Modèle à composant : Composant

- Coopération
  - Interface fournie
    - Composantes, interfaces, opérations, propriétés
  - Interface requise
    - Composition et références
  - Mode de communication
    - sync, async, flots
- Propriété configurable du composant
- Contraintes techniques (Qos)
  - middleware : placement, sécurité, transaction
  - interne : cycle de vie, persistance
  - implantation : os, bibliothèque, version

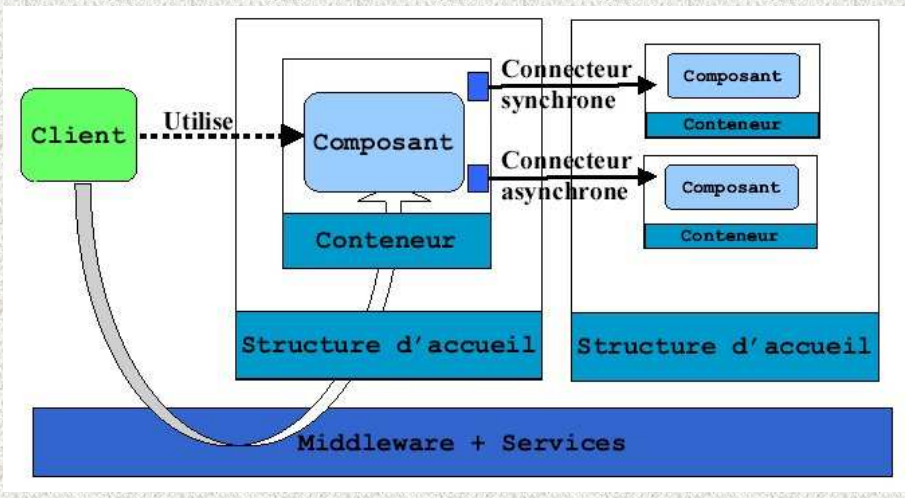


@ src : didier Donsez

Modèle à composants : Conteneur et Structure d'accueil

- Conteneur
  - Encapsulation d'un composant (et ses composantes)
  - prise en charge (masque) les services systèmes
  - nommage, sécurité, transaction, persistance ...
  - prise en charge partielle des connecteurs
  - invocations et événements
  - techniquement par interposition (ou délégation)
- Structures d'accueil
  - espace d'exécution des conteneurs et des composants
  - médiateur entre les conteneurs et les services systèmes
  - des + comme le téléchargement de code (navigateur)

Modèle à composants : conteneurs & structures d'accueil



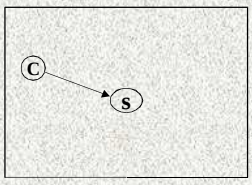
Composants : de l'installation à l'introspection

- Installer les composants
  - technologie de packaging
  - production des conteneurs
- Créer les composants
  - par des fabriques (maisons / « home »)
  - configuration des valeurs initiales
- Retrouver les composants
  - services de désignation (Nommage ou Vendeur) ou maisons
- Utiliser
  - invocation synchrone et événements
- Introspection
  - découvrir leurs APIs (fonctionnelle)
  - découvrir les connecteurs (structurelle)

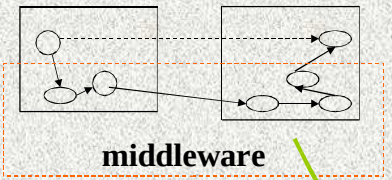
Construction par assemblage de composants

- Construction par assemblage plutôt que ingénierie de développement
  - réduire les besoins en compétence technique
  - focaliser l'expertise sur les problèmes du domaine
- Langage de description d'Architecture (ADL)
  - capturer les composants
  - fonctionnalités et besoins
  - capturer les connecteurs
  - composition et modes de communication
  - impédance entre composants => adaptateurs
  - C'est le point faible des solutions industrielles !

Le middleware




```
Class Client {
  public void test (){
    res=s.calcul();
  }
}
```




```
Class Client {
  public void test (){
    res=s.calcul();
  }
}
```

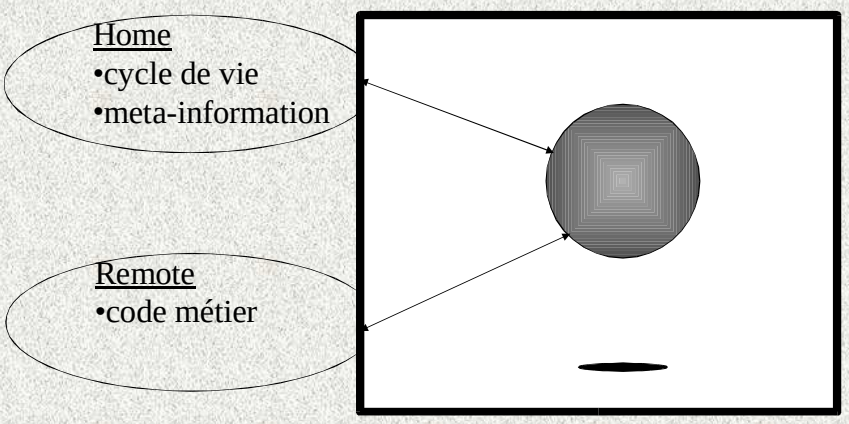
Enterprise JavaBeans

- Enterprise Java Beans :
  - Composants logiciels serveur
- Objectif
  - Standardiser le développement et le déploiement de composants serveurs écrits en Java
  - Le développement ne se fait que sur l'interface métier de l'objet (code fonctionnel)
  - Le conteneur d'EJB prend en charge tout ce qui n'est pas du code fonctionnel

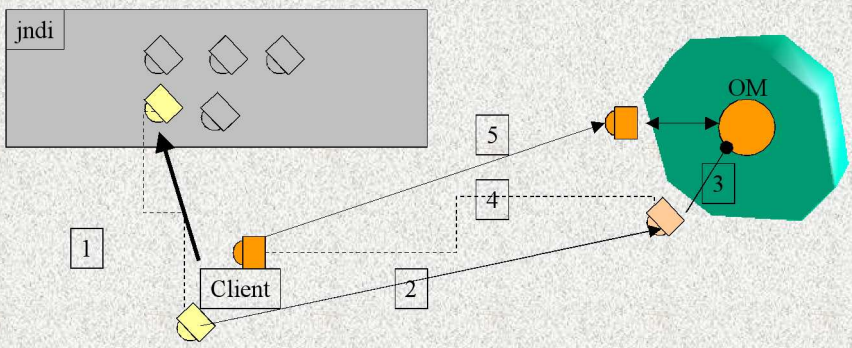
2 principes

- Les interfaces :
  - Représentent la vision "client" d'un objet
    - L'interface Home, représente la vision "cycle de vie" de l'objet
    - L'interface Remote, représente la vision "fonctionnelle » de l'objet
  - Elles sont concrétisées par une classe d'implantation
- Le conteneur
  - Exécute la classe d'implantation
  - Gère les aspects non-fonctionnels

La vision d'un EJB



Le fonctionnement global



Les services offerts par le conteneur

- **Contrôle de la charge** : Il permet de contrôler le nombre d'instances actives, par l'intermédiaire de l'usine de fabrication
- **Transparence du réseau** : Les interfaces sont concrétisées par des stubs rmi lorsque client et serveur sont distants
- **Surveillance des instances inactives** : Une instance non utilisée pendant un certain temps est automatiquement sauvée et purgée de la mémoire
- **Gestion des time-out** : Une requête d'un objet qui met plus de 30 secondes est automatiquement annulée
- **Gestion des données** : Les données d'un objet sont automatiquement synchronisées avec une base de données
- **Gestion des événements** : Le conteneur notifie automatiquement un objet si un message réseau arrive
- **Cryptage, Sécurité** ...

Principe de conception des EJB

- Le modèle des EJB est fondé sur quatre concepts pour la conception de systèmes distribués
  - Approche de serveurs sans-états
  - Approche orientée session
  - Approche objet persistant
  - Approche objet orientés messages
- Les spécifications EJB parlent de
  - Session Beans
    - Stateless Session Bean
    - Statefull Session Bean
  - Entity Beans
    - Container-Managed Persistence
    - Bean-Managed Persistence
  - Message-Driven Bean

Les beans session

Le Bean "stateless Session"

- Leurs comportements
  - Fournissent un service pour un seul utilisateur
  - Ne maintiennent pas d'état par rapport au client
  - Ne survivent pas à un crash du serveur d'EJB
  - Sont plutôt destinés à vivre sur une courte période
  - Deux instances d'un même bean « sans état » sont identiques

Les Bean "statefull Session"

- Leurs comportements
  - Interagissent d'une manière conversationnelle
  - Maintiennent un état sur le client connecté
  - Ne survivent pas à un crash du serveur d'EJB
  - Une instance est hébergée par un seul thread
  - Une instance peut être partagée par plusieurs clients

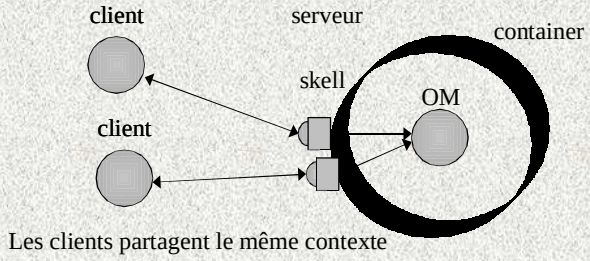
Stateful vs. Stateless

- Un bean stateful maintient un état conversationnel qui est récupéré entre deux invocations
- Un bean stateless n'a pas d'état entre les appels sur ses méthodes

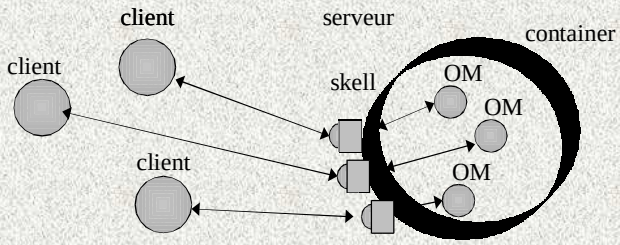
Exemples de Beans Session

- Session Stateless
  - Un EJB qui calcule le cos(x)
  - Un décompacteur
  - Un EJB qui vérifie si une donnée est valide
  - Un EJB qui calcule le coût d'une communication téléphonique
- Session Statefull
  - Un EJB qui gère une réservation
  - Un EJB qui commande les pièces détachées d'un véhicule
  - Un EJB qui gère le flux d'information d'un centre d'appel

Stateful vs. Stateless EJB



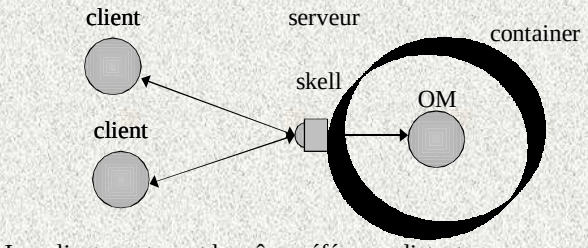
Les clients partagent le même contexte



Un client possède son propre contexte sur un EJB stateful

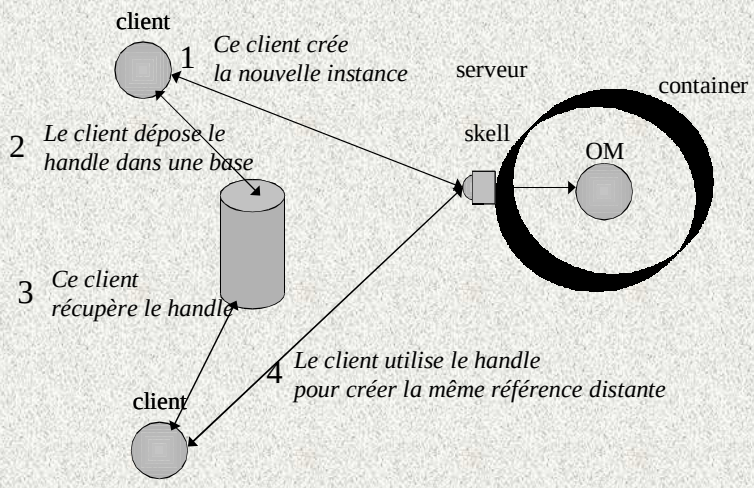
EJB Stateful identiques ?

- Deux EJB stateful sont identiques s'ils partagent le même contexte client !

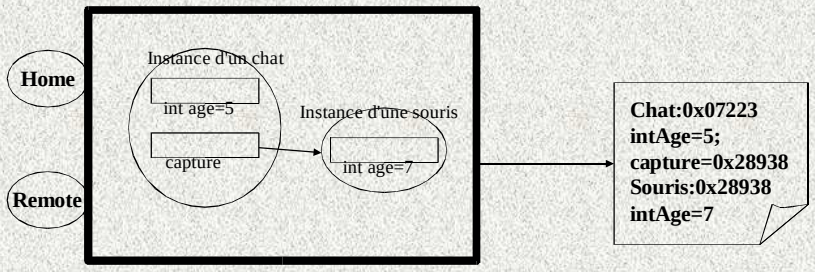


Les clients partagent la même référence distante

Comment obtenir la même référence distante ?

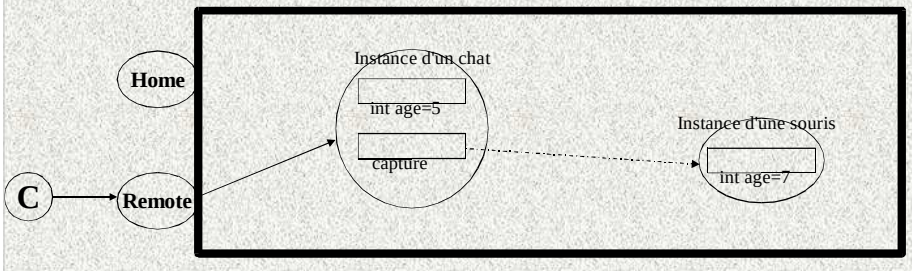


Service de surveillance de l'activité



- Si un ejb n'est pas utilisé pendant un certain temps, son instance est "sérialisée" sur un support secondaire
- Passivation

Gestion des délais de garde (time-out)



- Lorsqu'un client invoque une méthode, un délais de garde est déclenché. Si la méthode ne retourne pas dans ce délais le conteneur annule la méthode et renvoie une exception au client

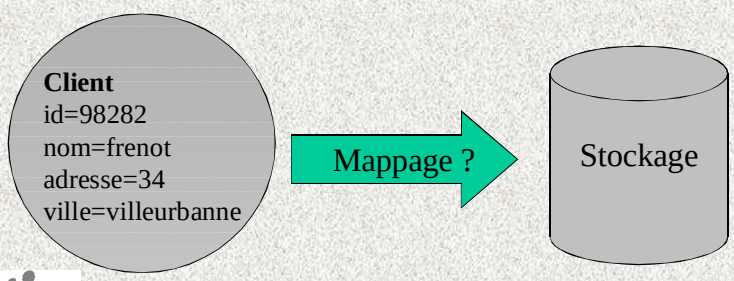
Les Beans Entity

Les EJB Entité (Entity EJB)

- Leurs comportement
  - Ils représentent les données persistantes
  - Ils survivent à un crash
  - Plusieurs clients peuvent utiliser des EJB qui "pointent" sur les mêmes données
  - L'instance EJB contient une copie des données du système de stockage

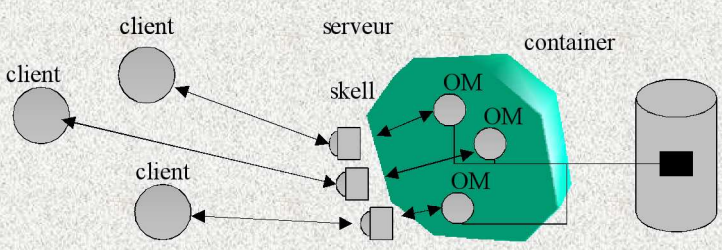
Gestion de la persistance

- Les attributs d'un objet doivent être déposés sur un support persistant
- Exemples de supports de persistance :
  - Sérialisation
  - Accès à une base sur le JDBC (mappage)



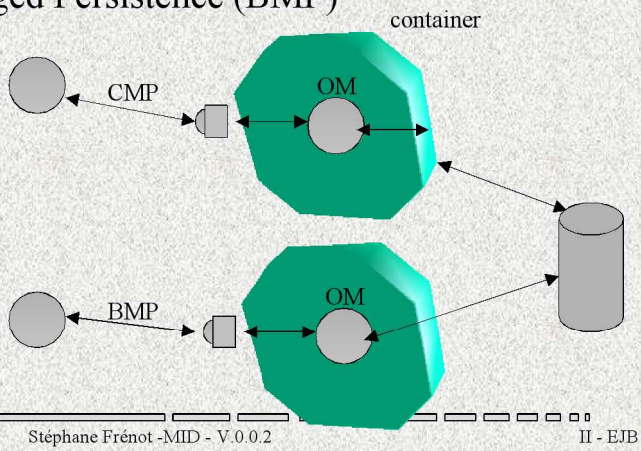
Partage d'EJB entity

- Quand plusieurs clients partagent un EJB entity ils
  - reçoivent leurs propres instance d'EJB
  - partagent les données sous-jacentes
  - n'ont pas à gérer la synchronisation sur les données



Type de persistance

- Il y a deux mode de gestion de la persistance
  - Container-Managed Persistence (CMP)
  - Bean-Managed Persistence (BMP)



EJB Entité

- Exemples :
  - Un EJB qui représente les statistiques d'un site
  - Un EJB qui représente l'évolution d'une cotation
  - Un EJB qui représente une séquence du génôme
  - ...

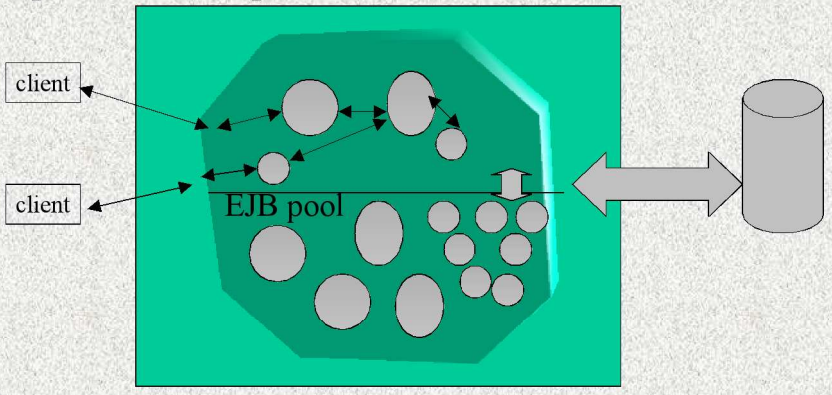
Exercice

- Un EJB qui implante C-MesCources ?
- Un EJB qui gère un cours en vidéo ?
- Un EJB qui gère le passage du TOEIC ?
- Un EJB qui calcule de la vitesse de rotation des planètes ?
- Un EJB qui représente le compte bancaire d'une personne ?
- Un EJB qui gère un appel de support technique ?
- Un EJB qui calcule la valeur de remboursement d'un prêt ?
- Un EJB de gestion de chaîne Hi-Fi ?
- Un EJB qui liste les personnes connectées sur un réseau ?
- Un EJB qui contrôle la validité d'un cookie d'une page Web ?
- Un EJB qui représente un document Word ?
- Un EJB qui représente une enchère sur un site ?

Les optimisations

Pooling d'objets

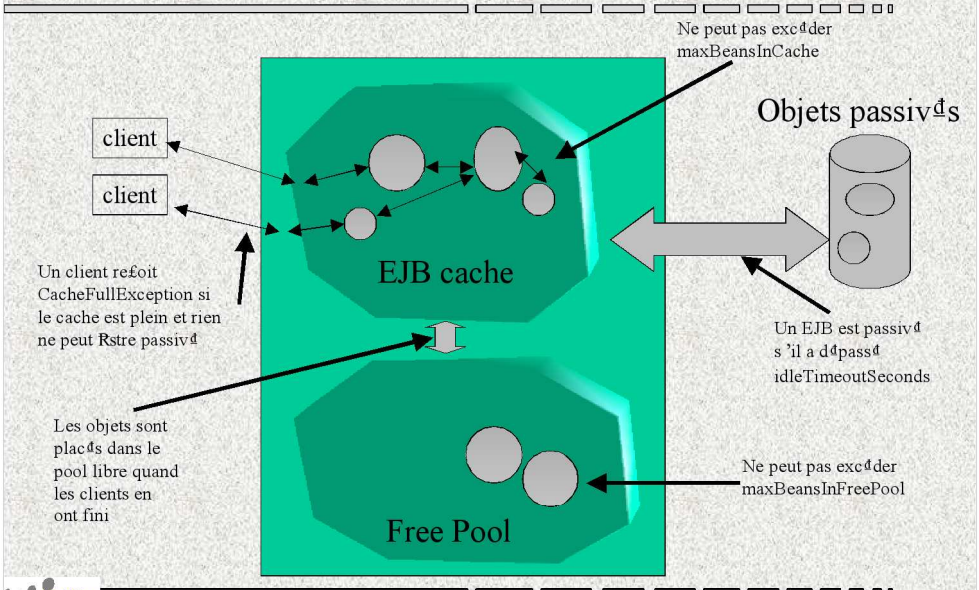
- Un serveur d'application peut créer un pool d'objets « nus » qui peuvent être utilisés quand des requêtes sont faites



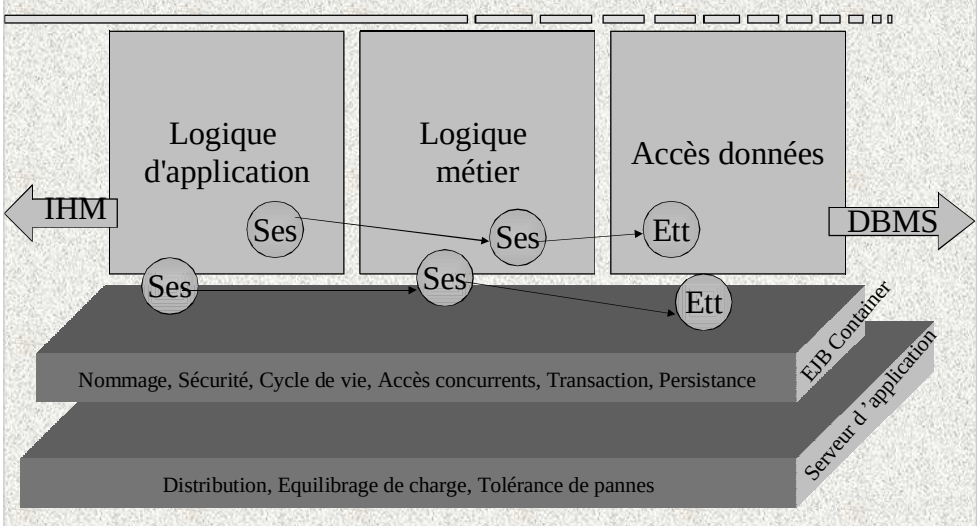
La passivation

- La passivation est le fait de placer un EJB sur un support secondaire
- La passivation :
  - A lieu quand un EJB dépasse son idle-timeout
  - Permet à un serveur d'application de réclamer des ressources
  - Placer une instance sérialisée de l'EJB dans son support de stockage

Pooling / Passivation sur Weblogic



Application et logique métier



Les rôles dans le monde EJB

- Fournisseur de serveur d'application
  - Fournit le serveur intégré avec des services de base
- Fournisseur de container
  - Fournit les containeurs d'EJB
- Fournisseur d'EJB
  - Expert dans un domaine vertical; crée les composants EJB
- Développeur d'application
  - Assemble les applications à partir de composants EJB préfabriqués
- Spécialiste du déploiement
  - Déploie les applications
  - Maîtrise les principes d'architecture

Cycle de développement

- Implantation de la base de données (opt.)
- Implantation des EJB (SL/SF/Entity)
- Implantation des JSP

Cycle de développement d'un EJB session

1) Définir les interfaces métier
   Interface Remote
2) Définir les méthodes de gestion du cycle de vie
   Interface Home
3) Définir les caractéristiques intrinsèques du bean dans un fichier de description
4) Définir les caractéristiques de déploiement
5) Déployer le bean
6) Réaliser le(s) clients qui utilisent le service

Interface Métier (Remote)

```
package exemple.fibonacci;

public interface Fibonacci extends javax.ejb.EJBObject {
   public int getFibonacciNumber(int n) throws java.rmi.RemoteException;
}
```

Interface Home

```
package exemple.fibonacci;
import javax.ejb;
import java.rmi;

public interface FibonacciHome extends EJBHome {
 public Fibonacci create() throws CreateException, RemoteException;
}
```


Développement du Bean

```
package exemple.fibonacci;
import java.rmi.*;
import javax.ejb.*;
public class FibonacciBean implements SessionBean {
  public void ejbPassivate(){}
  public void ejbActivate(){}
  public void ejbRemove(){}
  public void setSessionContext(SessionContext ctx){}
  public void ejbCreate() {
    System.out.println("Cet EJB Fibonacci est créé";
  }
  public int getFibonacciNumber(int n){...}
}
```

Descripteur du Bean

```
<?xml version="1.0"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise
    JavaBeans 2.0//EN" "http://java.sun.com/dtd/ejb-jar_2_0.dtd">
<ejb-jar>
  <enterprise-beans>
    <session>
        <ejb-name>fibonacci</ejb-name>
        <home>exemple.fibonacci.FibonacciHome</home>
        <remote>exemple.fibonacci.Fibonacci</remote>
        <ejb-class> exemple.fibonacci.FibonacciBean</ejb-class>
        <session-type>Stateless</session-type>
        <transaction-type>Container</transaction-type>
      </session>
      ...
  </enterprise-beans>
</ejb-jar>
```


Descripteur du déploiement

```
<?xml version="1.0"?>
<!DOCTYPE weblogic-ejb-jar PUBLIC "-//BEA Systems, Inc.//DTD WebLogic 6.0.0
    EJB//EN" "http://www.bea.com/servers/wls600/dtd/weblogic-ejb-jar.dtd" >
<weblogic-ejb-jar>
  <weblogic-enterprise-bean>
    <ejb-name>fibonacci</ejb-name>
    <stateless-session-descriptor>
      <pool><max-beans-in-free-pool>100</max-beans-in-free-pool></pool>
    </stateless-session-descriptor>
    <transaction-descriptor>
      <trans-timeout-seconds>300</trans-timeout-seconds>
    </transaction-descriptor>
    <jndi-name>fibonnaci</jndi-name>
  </weblogic-enterprise-bean>
</weblogic-ejb-jar>
```

Déploiement du bean

- Eventuellement (pré-compilation)
- Jar + cp

- Au déploiement
  - Inscription de l'Usine (interface Home) sur le containeur
  - Dépôt du stub de manipulation de l'Usine sur le service de nommage

Développement d'un client

```
public class Client{
  public static void main(String [] argv){
    try{
      FibonacciHome home=(FibonacciHome)context.lookup("fibonaci");
      Fibonacci jacques=home.create();
      jacques.getFibonnaciNumber(10);
    ...
  }
}
```

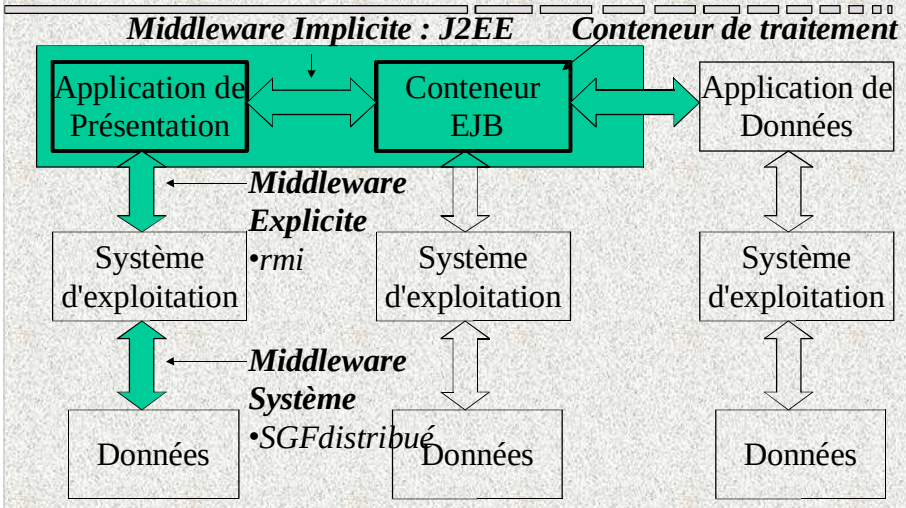
Développement d'un client JSP

```
<html><head><title> <%= pagetitle %> </title></head>
<h2><font color=#DB1260><%= pagetitle %></font></h2>

<%@ page import="example.fibonacci.*"%>

<%!String pagetitle = "Suite de fibonnaci";%>
<%try {
 ctx = getInitialContext();
 FibonnaciHome homeFib = (FibonnacciHome)ctx.lookup("fibonnaci");
 uneSuite=homeFib.create();
 out.println("fib(7)="+uneSuite.getFibonacciNumber(7));
}catch(Exception e){e.printStackTrace();
%>

</body></html>
```


Points forts

- Notion de container
  - Robustesse, standardisation, évolution
- Interface de développement standardisées
  - Pas / Peu de phase de prise en main
- Automatisation de nombreuses tâches
  - Gestion de la persistance, transactions…
- Intégration à l'API java
- Marché explosant

Répartition d'une application



Evolution récentes

- Xdoclet
  - Intégration de la description de déploiement dans les commentaires
  - Notion d'annotations intégrées au jdk 1.5

Annotations

Nouveautés dans le langage Java : les annotations

- **Avant 1.5**

```
public interfaceIHello extends Remote{
  public StringsayHello(Stringname) throws RemoteException;
}
public class Helloimplements IHello{
  public StringsayHello(Stringname) throws RemoteException{
  return "Hello "+name;
}}
```

- **Avec les annotations du 1.5**

```
public class Hello { //with1.5 annotations
public @remote String sayHello(Stringname) {
  return "Hello "+name;
}}
```

- **Impact sur les outils de génération comme rmic**