Jonathan MERCIER

- J2EE -
Enterprise JavaBeans

Plan

- Introduction (J2EE)
- Présentation des EJB
- EJB : les concepts
- Processus de développement, de déploiement et d'exécution
- Exemples

Introduction

- Qu'est ce que J2EE ?
- Architecture J2EE
- Serveurs d'applications

Qu'est ce que J2EE ?

- Spécification de java destinée aux applications d'entreprise.
- Extension du framework J2SE
- Architecture multi-tiers
- Facilite la création d'applications réparties
- Bénéficie des avantages et des inconvénients de java

Architecture multi-tiers

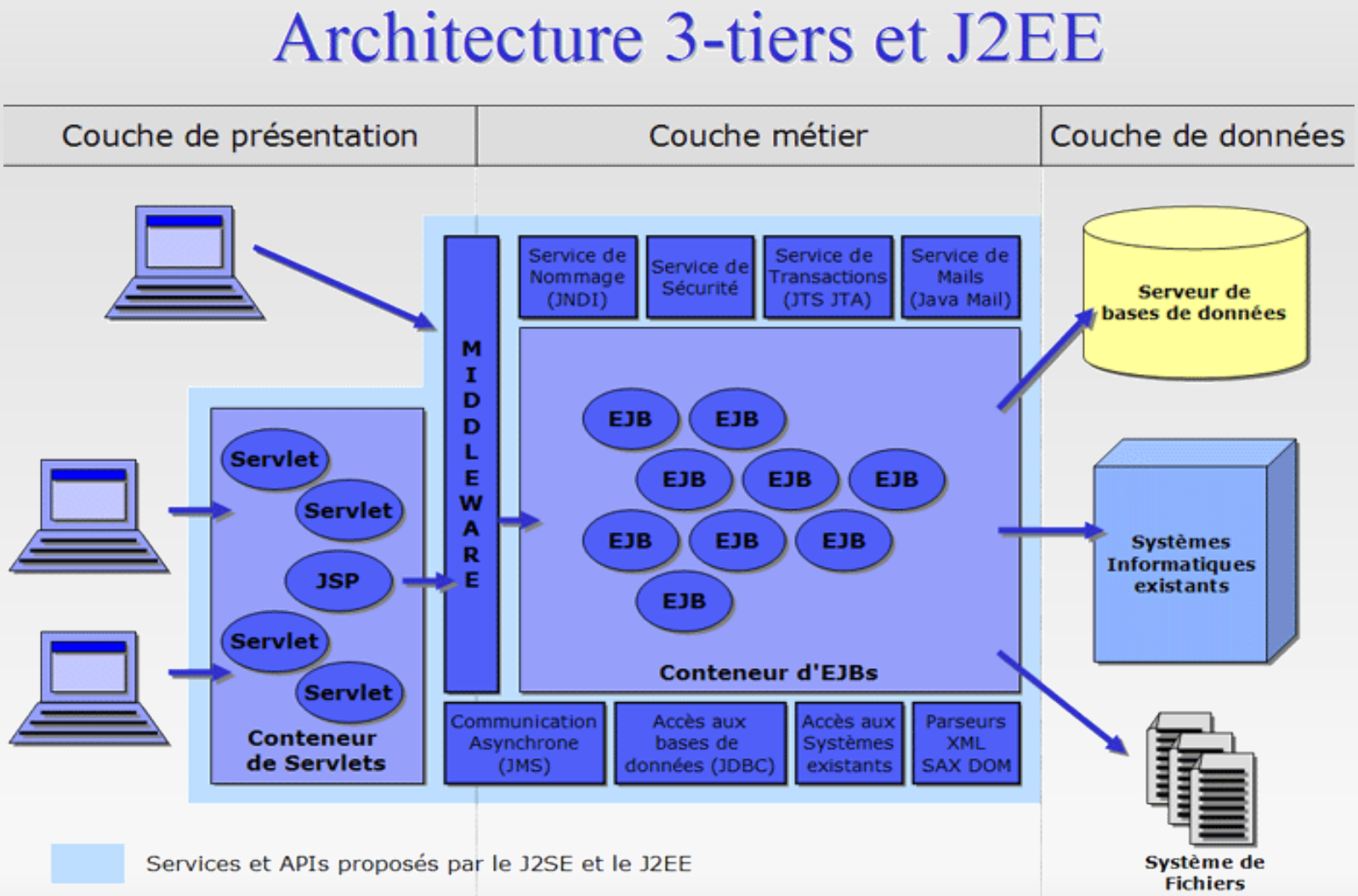
- Clients Légers
  - Browser Web (HTTP/HTML)
  - Applets (RMI)
  - Controle ActiveX (DCOM)
  - Clients Corba (IIOP)
- Serveurs applicatifs
  - Présentation, outils métier
  - CGI,Servlet
- Source de Données
  - Base de données relationnelles
  - Intégrées (ERP)
  - Annuaires (LDAP)
  - Flux (Reuters...)

Architecture J2EE (1)

J2EE comprend :

- Spécification du serveur d'applications
  - *Un environnement d'exécution*
  - *défini rôle et interface pour les applications*
  - *permet à d'autres entreprises de développer leurs serveurs d'applications*
- Des services
  - *API (extensions java indépendantes)*
  - *J2EE SDK (implémentation minimales)*

Architecture J2EE (2)



Serveurs d'applications

- Objectifs
  - Simplifier le développement d'architecture multi-tiers
- Principe
  - Le développeur se concentre sur la logique de son application
  - Le reste est réalisé par la plateforme d'accueil
- Nombreux serveurs d'applications compatible J2EE :
  - SUN/J2EE : la référence
  - JBoss : Gratuit et open source
  - IBM Websphere : le leader incontesté, il n'est pas gratuit
  - WebLogic...

Présentation des EJB

- Qu'est ce qu'un EJB ?
- Caractéristiques principales
- Avantages des EJB
- Quand utiliser des EJB ?
- Les types d'EJB

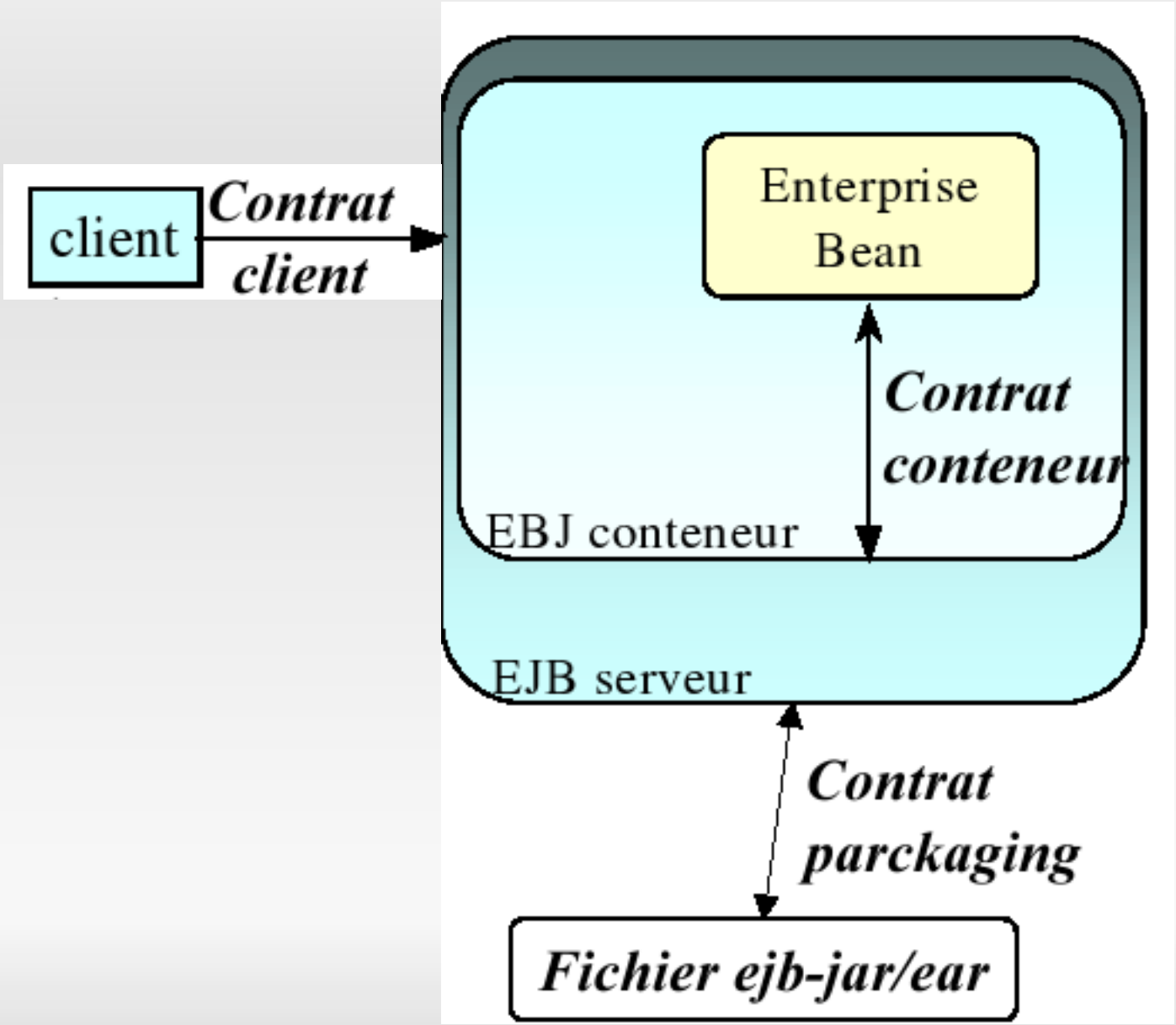
Qu'est ce qu'un EJB ?

- Ecrit en JAVA
- Composant coté serveur
- Objets distants utilisant RMI/IIOP
- Encapsulation de la logique métier
- Logique métier = code qui accomplit le but de l'application

Caractéristiques Principales

- S'intéresser aux activités liées au développement, au déploiement, et à l'exécution d'une application
- Définir différents rôles associés aux différentes parties intervenant dans la production d'une application
- Définir des contrats associé à un bean

Contrats (1)


client
Contrat client
Enterprise Bean
Contrat conteneur
EBJ conteneur
EJB serveur
Contrat parckaging
Fichier ejb-jar/ear

Contrats (2)

- Fournir un modèle de développement uniforme pour les applications qui utilisent les composants

  - Contrat coté client
    - *fournir une vue uniforme du bean au client. En particulier cette vue est indépendante de la plateforme de déploiement*
  - Contrat coté conteneur
    - *permettre la portabilité des beans sur différents serveurs EJB.*
  - Contrat coté « packaging »
    - *fournir un format de fichier standard pour « packager » les beans. Ce format doit être supporté par tous les outils liés aux EJB.*

Avantages des EJB

- Simplification du développement de grandes applications
- Le développeur se concentre :
  - uniquement à résoudre les problèmes métiers
  - sur la présentation client
- Le client n'a pas besoin d'un gros équipement (client léger)
- Portable, réutilisable sur n'importe quel serveurs d'applications J2EE.

Quand utiliser les EJB ?

- L'application doit être évolutives
  - ➔ les EJB peuvent fonctionner sur différentes machines
  - ➔ localisation transparentes pour le client.
- Intégrité des données d'une transaction
- L'application aura une variété de clients.

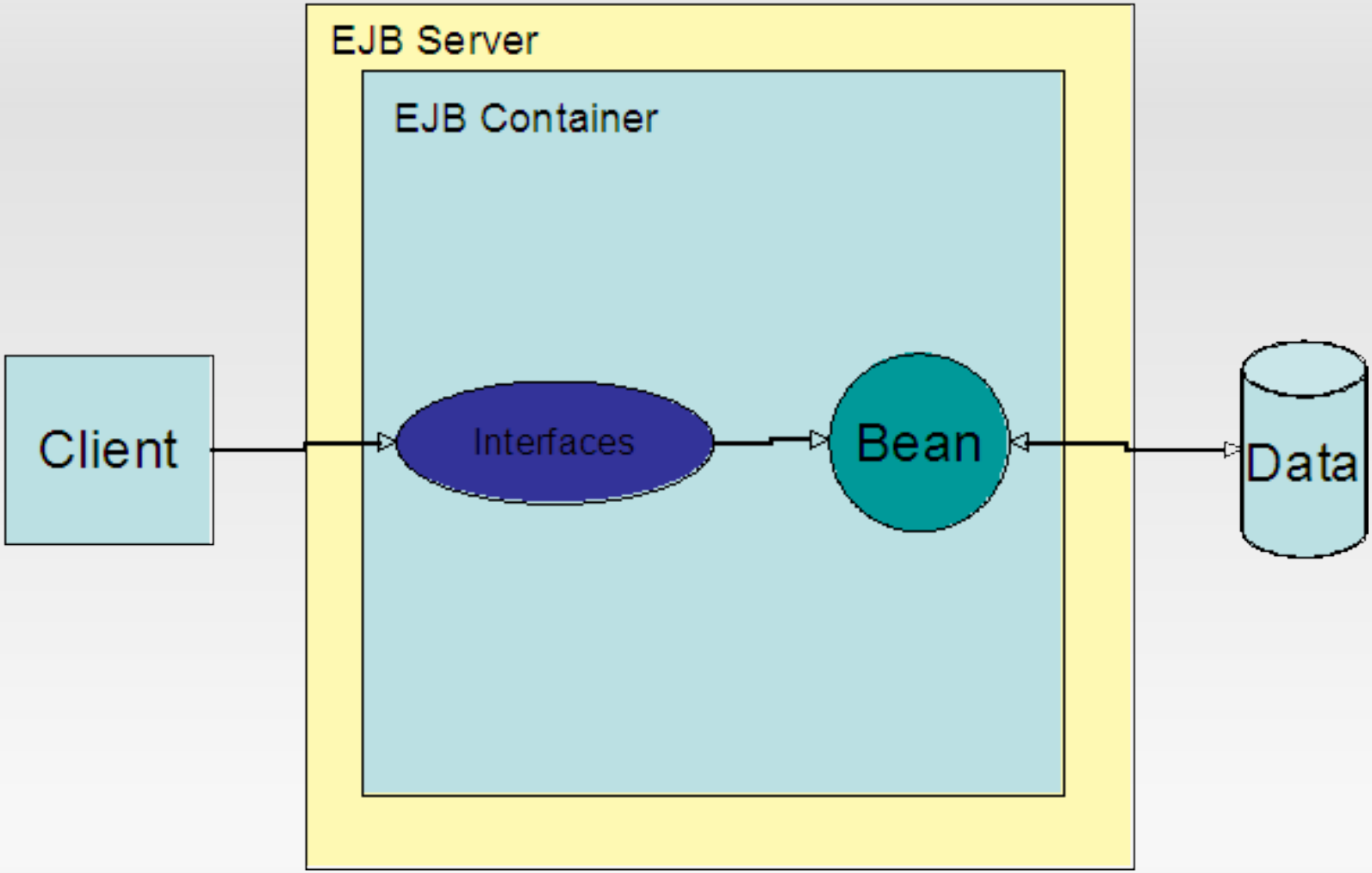
Les types d'EJB

- Session :
  - Représente les données du client et les comportements associés.
  - Ex: caddie virtuel
- Entity :
  - Représente une donnée persistante (existence sur un support physique)
  - Ex: articles stockés dans le caddie
- Message Bean :
  - Pour les applications basés sur le traitement asynchrone des messages
  - Basé sur Service JMS (Java Message Service)

EJB : les concepts

- Architecture EJB
- Server EJB
- Container EJB
- Les différents types de beans
  - *Session Bean*
  - *Entity Bean*
  - *Message Bean*

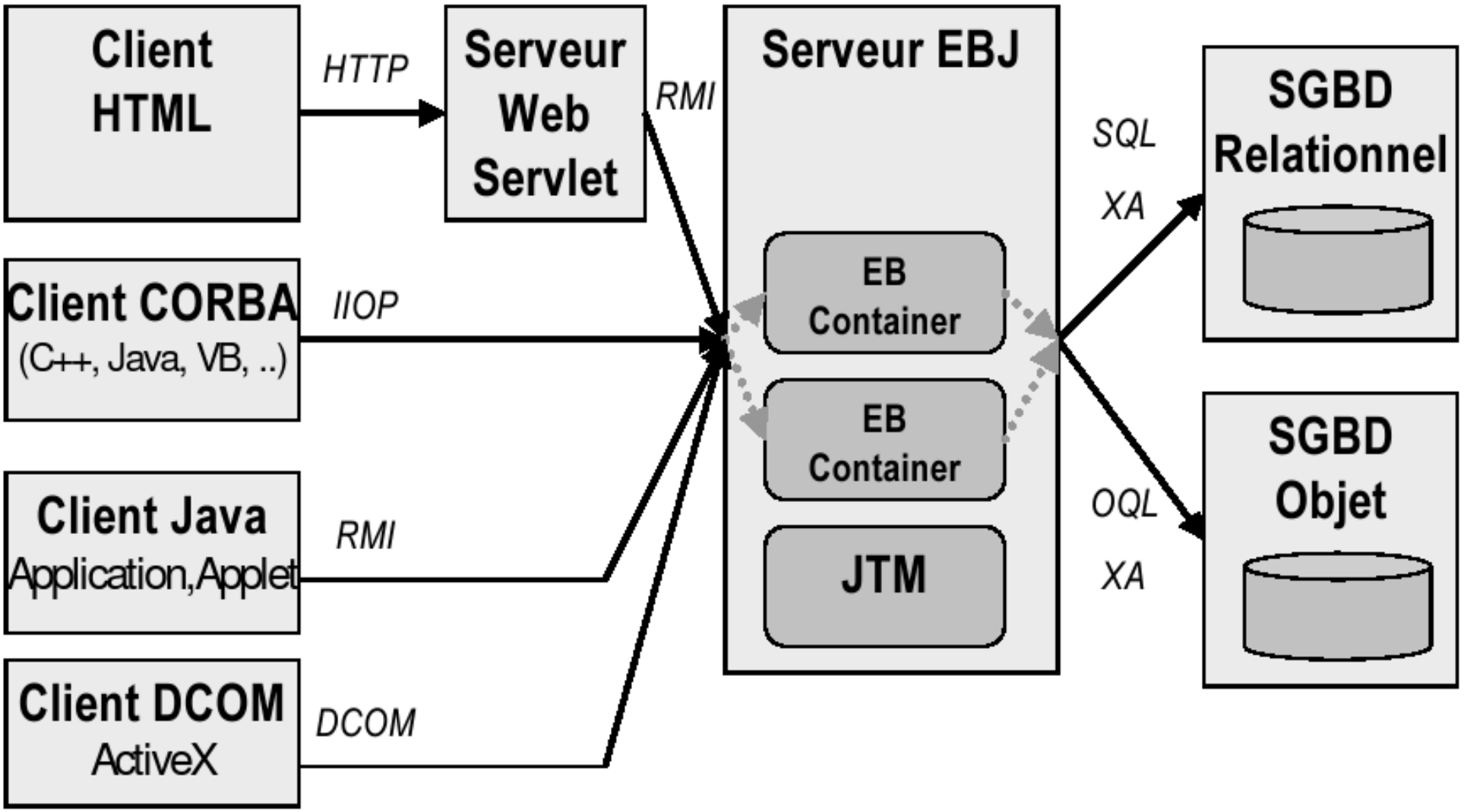
Architecture EJB



Server EJB (1)

- Gère :
  - l'ensemble des services systèmes
  - les containers dans lequels les beans s'exécutent
- Un serveur se distingue par rapport aux options que le serveur EJB propose.

Server EJB (2)


Client HTML
HTTP
Serveur Web Servlet
RMI
Serveur EBJ
SQL
XA
SGBD Relationnel
Client CORBA
(C++, Java, VB, ..)
IIOP
EB Container
EB Container
JTM
Client Java
Application,Applet
RMI
OQL
XA
SGBD Objet
Client DCOM
ActiveX
DCOM

Container EJB

- Permet de minimiser la charge du développeur en fournissant :

  - La connectivité clients/EJB
  - Une gestion :
    - La persistance
    - Transactions
    - La sécurité
    - La concurence
    - Du cycle de vie

Les différents types de Bean

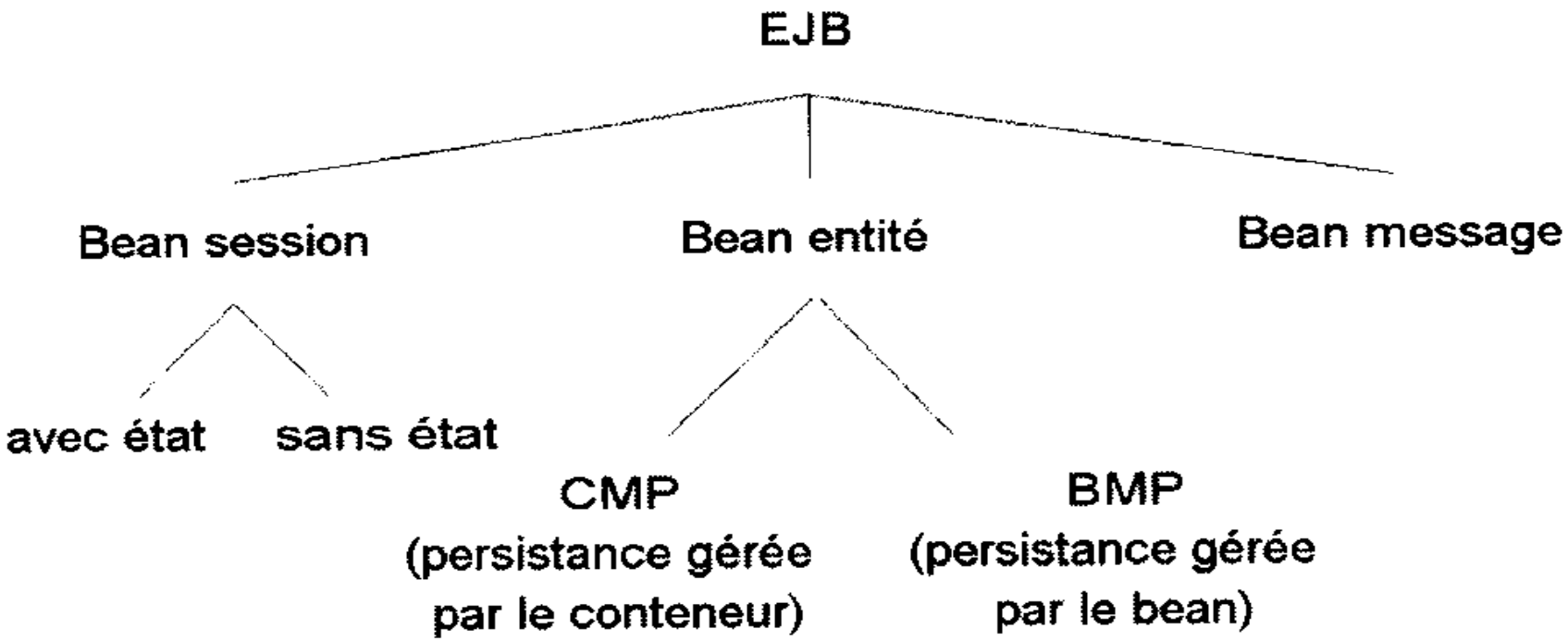


Figure 1.6 — Les différents types de beans.

Session Bean (1)

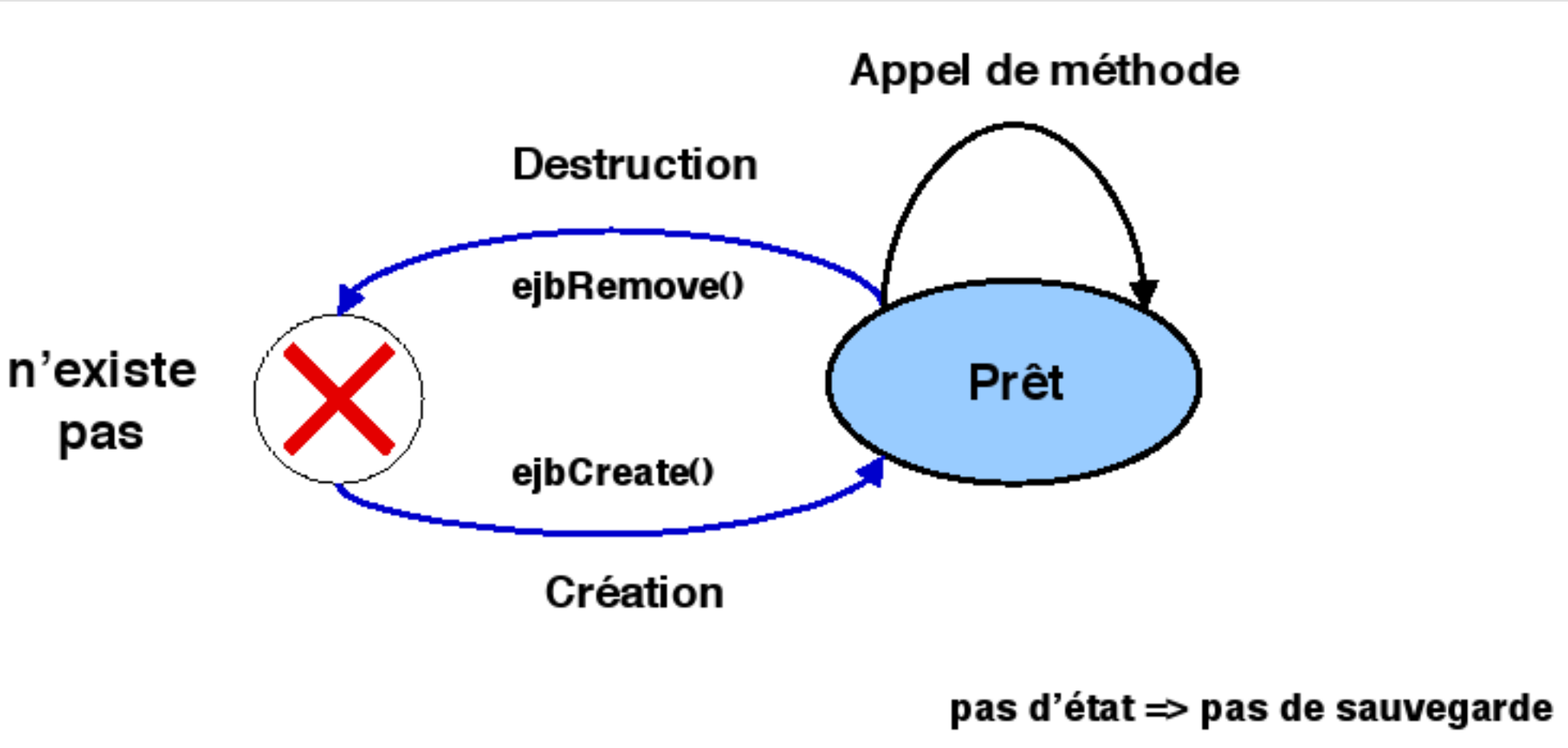
- sont non persistants
- Associés à un seul client
- détruit après un arrêt (ou une panne) du serveur EJB.
- Il existe deux types de session beans
  - stateless (sans état)
  - statefull (avec état)

Session Bean (2)

- Stateless
  - Représente un client le temps d'un traitement
  - Lors de l'appel d'une méthode, les variables d'instances n'existent que jusqu'à la fin de l'appel
- Statefull
  - Maintient les données client en mémoire

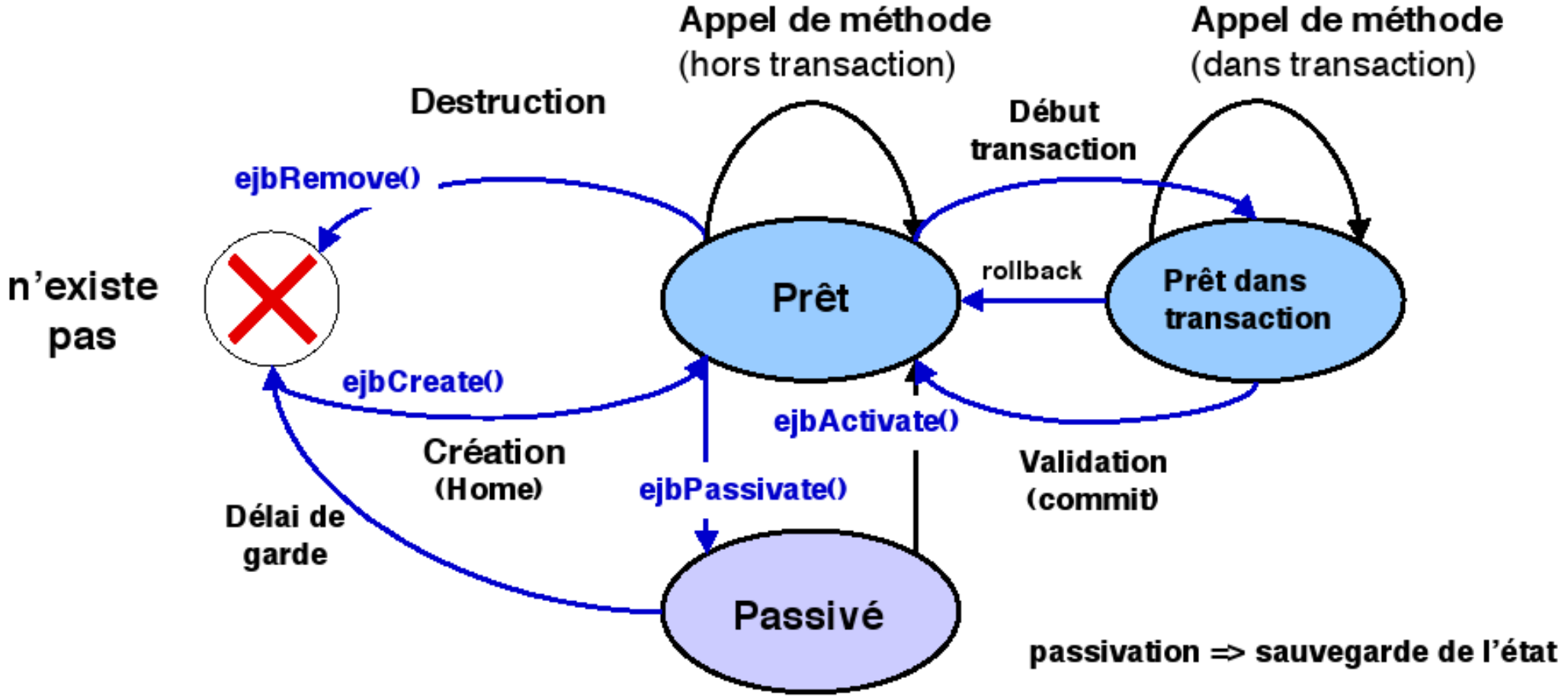
Cycle de vie d'un Bean Session (1)

- Stateless



Cycle de vie d'un Bean Session (2)

- Statefull



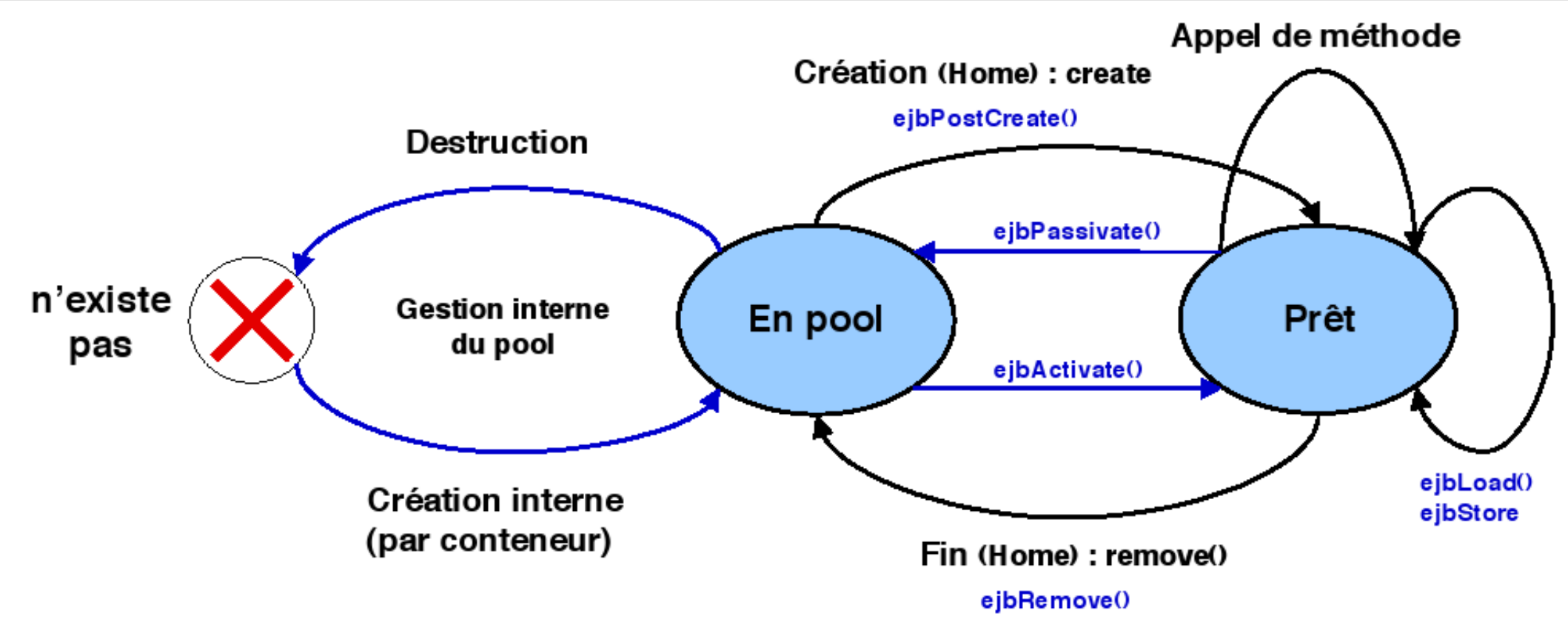
Entity Bean (1)

- Représente les données d'une base de données
- Sont persistants
- La gestion de la persistance par le bean (bean managed persistence) ou délégué à son conteneur (container managed persistence)

Entity Bean (2)

- Acceptent les accès multiples effectués par plusieurs clients
  - → **Gestion de la concurrence**
- Peuvent participer à des transactions
- Survivent aux pannes d'un serveur EJB.

Cycle de vie d'un Bean Entity



Persistance

- Persistance gérée par le Bean (BMP)
  - le fournisseur du Bean écrit les opérations d'accès aux données permettant de gérer dans les callback appropriés (ejbCreate, ejbStore, ejbLoad, ejbFind)
- Persistance gérée par le Container (CMP)
  - Le container EJB s'occupe de tous les accès à la base de données
  - Le code bean ne contient aucun accès à la base de données
  - Pas de spécificité à une base de données

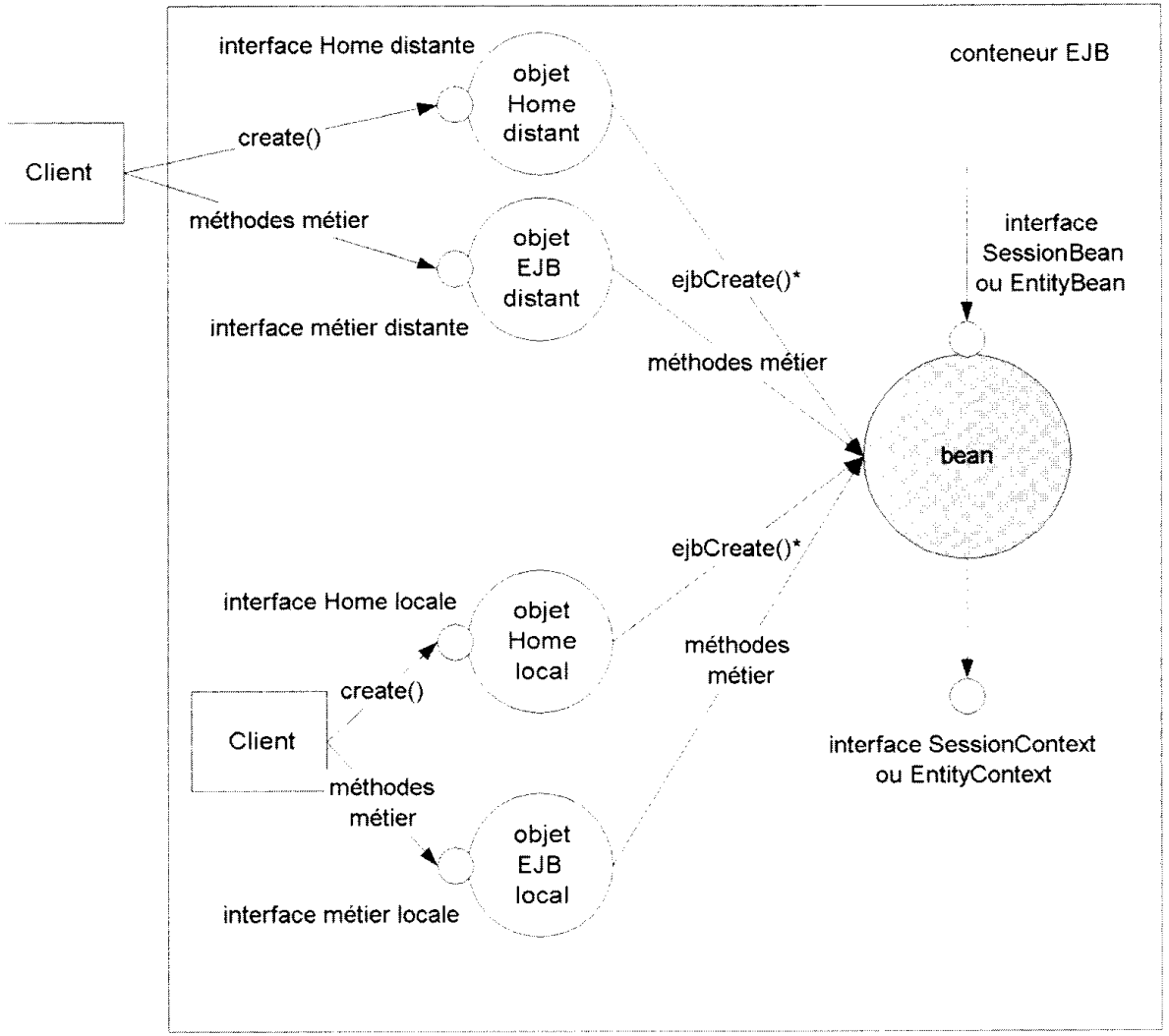
Message bean

- Permettre à des applications J2EE de traiter des messages asynchrones.
- Ils s'exécutent à la réception d'un message simple du client
- Sont appelés de manière asynchrone
- Ils sont relativement de courte durée
- Ils ne représentent pas de données BD mais peuvent y accéder et mettre à jour.
- Ils n'ont pas d'états.

Processus de développement, de déploiement et d'exécution

- Principe
- Interfaces et classes
- Le descripteur de déploiement

Principe (1)



* Pour les beans session sans état, l'invocation de la méthode create() sur une interface Home ne déclenche pas nécessairement la méthode ejbCreate() sur le bean

Figure 3.1 — La structure d'un bean.

Principe (2)

- Développement
  - Ecrire la Home interface
  - Ecrire la Remote (ou Local) interface
  - Ecrire l'implémentation du bean
  - Compiler ces classes et interfaces
- Déploiement
  - Construire le descripteur de déploiement
- Exécution
  - Activer J2EE
  - Exécuter le client
    - *qui peut être une servlet*
    - *ou une application java*

Interfaces et classes (1)

- Pour Entity et Session Bean
- EJB est constitué au moins 3 fichiers
  - Interface de fabrique (Home interface)
    - *Définit le cycle de vie du bean*
    - *Etend de javax.ejb.EJBObject et java.rmi.Remote*
  - Interface Métier (Remote ou Local Interface)
    - Définit les services proposés par le beans
  - Bean
    - Implémente les deux interfaces précédentes

Interfaces et classes (2)

- Autres fichiers
  - Descripteur de déploiement
    - *Qualifie la nature du bean*
    - *Décrit les aspects sécurités*
    - *Décrit l'utilisation du bean dans les transactions*

Le descripteur de déploiement (1)

- Fonction
  - Spécification déclarative de diverses propriétés du bean
    - *Identification,attributs de transaction, champ persistant, environnement, gestion ou non par conteneur, rôle pour la sécurité...*
    - *Utilisé par le conteneur pour mettre en oeuvre ses fonctions.*
- Forme
  - Écrit en XML (depuis EJB 1.1)
  - Pour chaque propriété il y a une balise particulière

Le descripteur de déploiement (2)

- Exemple :

```
...
<persistence-type>Container</persistence-type>

<container-transaction>
        <method>
                <ejb-name>MyBean</ejb-name>
                <method-name>*</method-name>
        </method>
        <trans-attribute>Required</ trans-attribute>
</container-transaction>
```

persistance gérée par le conteneur

transactions gérées par le conteneur

exécution transactionnelle obligatoire pour toutes les méthodes

- Un par bean !
- Généré automatiquement (ouf !)

Les EJB 3.0

- En cours de normalisation
- Utilise JDK 1.5
- Développement plus simple
  - Plus besoin d'héritage de super interface/classe
  - Plus de méthodes callback inutiles
  - Plus d'interface Home
  - plus d'obligation de descripteur de déploiement
  - Apparition des annotations
- Compatibilité ascendante

Exemple

- La calculatrice
  - EJB 2.0
  - EJB 3.0

Calculatrice - EJB 2.0

- Interface métier distante
- Interface Home distante
- Classe d'implémentation
- Le client

Interface métier distante

```
import java.ejb.EJBObject;
import java.rmi.Remote Exception;

public interface Calc extends EJBObject {
public double add ( double val1, double val2)
throws Remote Exception;
public double mult( double val1, double val2)
throws Remote exception;
}
```

Interface Home distante

```
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHOME;

public interface CalcHome extends EJBHOME {
   public Calc create() throws RemoteException,
CreateExeption;
}
```

La classe d'implémentation

```
import java.rmi.RemoteException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;

public class CalcBean implements SessionBean {
   public double add(double val1, double val2){
      return val1 + val2;
   }
   public double mult(double val1, double val2){
      return val1 * val2;
   }
   public CalcBean(){}
   public void ejbCreate(){}
   public void ejbRemove(){}
   public void setSessionContext(SessionContext sc){}

   /*piège*/ <-- bean sans état !!!!!
   public void ejbActivate(){}
   public void ejbPassivate(){}
}
```

Le client (1)

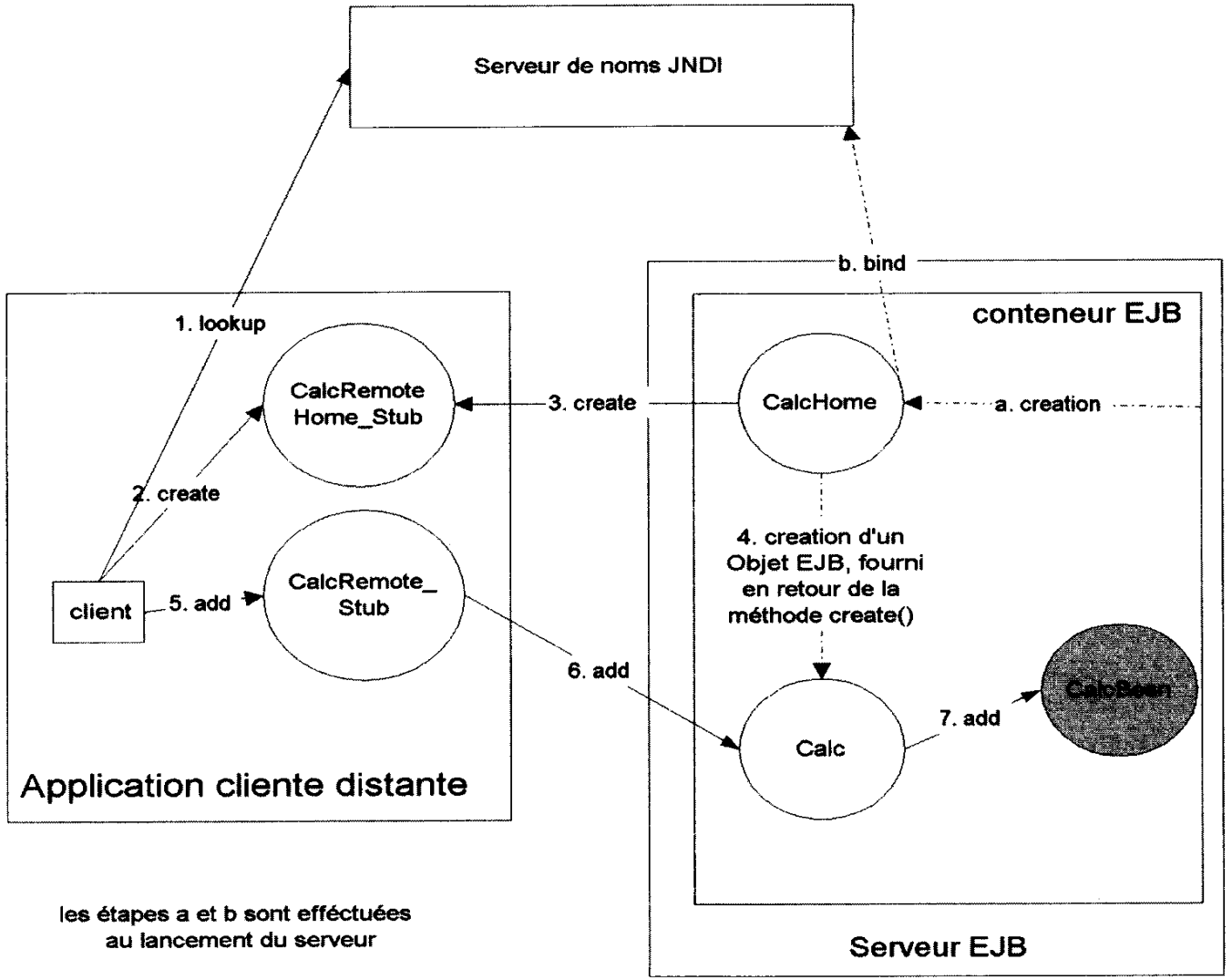


Figure 2.5 — Interactions entre un client et un EJB.

Le client (2)

```
import javax.rmi.*;
import javax.naming.*;
public class ClientCalc {
   CalcHome CalcHome;
   Calc mycalc;
   try{
      Context ctx = new InitialContext();
      //étapes 1,2,3
      Object ref = ctx.lookup("Calc");
      calcHome = (CalcHome)PortableRemote-
Object.narrow(ref, CalcHome.class);
      myCalc = calcHome.create();
      //étapes 4,5,6
      myCalc.add(2.564,8.876);
      myCalc.mult(98.99,6.65);
   }catch(Exception e){
      e.printStackTrace();
   }
}
```

Calculatrice – EJB 3.0

- Interface métier distante
- Classe d'implémentation
- Le test

Interface distante

```
package demo.ejb3.calculatrice;
import javax.ejb.Remote;

@Remote
public interface CalculatriceRemote {
     double add(double val1, int val2);
     double mult(double val1, double val2);

}
```

Classe d'implémentation

```
package demo.ejb3.calculatrice;
import javax.ejb.Stateless;

@Stateless
public class CalculatriceBean implements CalculatriceRemote
{

    public CalculatriceBean() {}
      public double add(double val1, double val2) {
         return x + y;
      }
      public double mult(double val1, double val2){
         return x*y;
      }
}
```

Le Test (1)

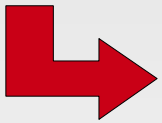
```
package demo.ejb3.calculatrice;

public class CalculatriceBeanTest extends TestCase {

    public CalculatriceBeanTest(String testName) {
        super(testName);
    }
    public void testAdd() {
        double x = 3;
        double y = 6;
        CalculatriceBean instance = new CalculatriceBean();
        int result = instance.add(x, y);
```

Le Test (2)

```
InitialContext ctx;
        try {
            ctx = new InitialContext();
            Object ref =
ctx.lookup("demo.ejb3.calculatrice.CalculatriceRemote");
            CalculatriceRemote calc =
(CalculatriceRemote)PortableRemoteObject.narrow(ref,Calculat
riceRemote.class);
            result = calc.add(x,y);
        } catch (NamingException ex) {
            fail(ex.getMessage());
        }
    }
}
```

Conclusion

- Avantages
  - Le développeur s'occupe uniquement de la logique métier.
  - Indépendant du fournisseur du serveur



réutilisation du code

- Inconvénient
  - Subtil
  - Difficile à comprendre
  - Déploiement non évident

Bibliographie

- Basé sur le présentation de :
  - Sacha KRAKOWIAK
  - Didier DONSEZ
- Livres
  - EJB 2.0 – Mise en oeuvre ; Christophe Calandreau, Alain Fauré, Nader Soukouti.
  - Enterprise JavaBeans ; Richard Monson-Haefel
- Web
  - http://java.sun.com/products/ejb/
  - http://www.labo-sun.com/