

# Systèmes d'Information Coopératifs et Répartis

Université Lyon 1  
M2 CCI Réseaux

L. Médini, février 2006

## Plan des cours

- Protocole HTTP et programmation serveur
- Architectures réparties
- Objets distribués
  - Javabeans (survol)
  - EJB
- Web services (SOA, WSDL, SOAP, UDDI)
- Projet

## Les JavaBeans (1996)

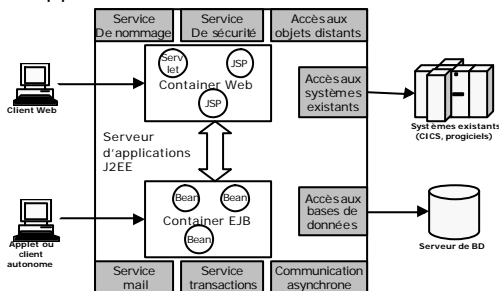
- Définition
  - Composants logiciels réutilisables d'applications **non réparties** (en pratique, des classes Java)
- Structure
  - Les propriétés sont cachées et accessibles par des méthodes publiques  
`public String getNom()` et  
`public void setNom(String valeur)`
  - Les autres méthodes sont privées
- Utilisation depuis une JSP
  - Déclaration : `<jsp:useBean class="package.NomBean" id="testbean" scope="application" />`
  - Accès : `<%= testbean.getProperty0() %>`

## Les JavaBeans

- Communications entre beans : le modèle événementiel
  - Principe des « écouteurs » (*listeners*) Java : un objet s'enregistre comme écouteur d'un certain événement (interface `java.util.EventListener`)
  - Un bean peut émettre un événement (classe `java.util.EventObject`), « capté » par le ou les écouteurs.  
→ Les beans peuvent être assemblés en applications
- Persistance : un bean doit pouvoir être sauvegardé et restitué (en pratique, il doit implémenter `java.io.Serializable`)
- Interface : un bean peut être manipulé *visuellement* dans un outil d'aide à la construction d'applications (dans ce cas, étendre `java.awt.Component`)
- Introspection
  - Les propriétés et événements des beans sont découverts par *introspection* par l'outil de construction d'application (classe `XXXBeanInfo` qui implémente `java.beans.BeanInfo`)
  - La découverte des beans peut être réalisée par une instance de la classe `java.beans.Introspector`

## Les Enterprise JavaBeans (1998)

### □ Rappel : l'architecture J2EE



## Les Enterprise JavaBeans (1998)

### □ Rappel : l'architecture J2EE

- Mêmes principes de base que les infrastructures middleware ...
  - Communication synchrone (RMI/IIOP) et asynchrone (JMS) entre objets distribués
  - Serveurs de nom (JNDI)
  - Intégration d'objets CORBA (JavaIDL)
- ...plus de nombreux services techniques supplémentaires
  - Génération d'objets transactionnels distribués (EJB)
  - Gestion du cycle de vie des objets
  - Gestion des transactions (JTA et JTS)
  - Sécurité
  - Accès aux BD (JDBC)
  - Interfaces graphiques dynamiques (Servlets et JSP)...

## Les Enterprise JavaBeans (1998)

- Définition : composants transactionnels accessibles à distance
- Un EJB est intégré à un serveur d'applications et représente une partie de la logique métier d'une application
- Un EJB est accessible via un conteneur qui permet
  - De l'intégrer à un serveur d'applications (accès local ou distant via une interface réseau)
  - De gérer les aspects distribués
  - De fournir des services supplémentaires
    - Aspects middleware, gestion du cycle de vie, sécurité (accès par les interfaces rendues disponibles par le conteneur), récupération d'un « contexte »...

## Les Enterprise JavaBeans

- Trois types de beans
  - Beans session
    - Représentent les processus métiers
    - Méthodes accessibles par le client
    - Ne peuvent avoir qu'un seul client à un moment donné
    - Deux types de Beans session
      - Avec états : c'est le même client qui réalise toutes les invocations (exemple : panier électronique)
      - Sans état : le Bean peut être utilisé successivement par plusieurs clients (exemple : calcul d'une distance)

## Les Enterprise JavaBeans

- Trois types de beans
  - Beans entités
    - Représentent des objets métiers persistants (exemple : une commande, un article, un compte bancaire...)
    - Permettent d'accéder aux données persistantes
    - Identifiés par des clés primaires (*Int*, *String*, *Object*)
    - Deux moyens de gérer la persistance
      - CMP : Container Managed Persistence
      - BMP : Bean Managed Persistence

## Les Enterprise JavaBeans

- Trois types de beans
  - Beans messages
    - Composant asynchrone qui peuvent échanger des messages par l'intermédiaire de Java Message Service
    - Les beans message ne sont en général pas accédés par les clients, mais par d'autres beans (pas d'interface home)

## Les Enterprise JavaBeans

- Principes de base
  - Un EJB n'est pas accédé directement par le client, il dispose d'interfaces au niveau du conteneur
    - Interfaces « Home »
      - Gestion du cycle de vie (création de l'instance)
    - Interfaces métier
      - Présente les méthodes mises à disposition par la classe d'implémentation
  - ⇒ Depuis EJB 2.0, il existe des interfaces distantes et des interfaces locales

## Les Enterprise JavaBeans

- Principes de base
  - Exemples de nommage pour un EJB session sans état Calc (calculatrice)
    - Interfaces home

```
public interface CalcHome extends EJBHome {
    public Calc create() throws RemoteException,
        CreateException; }
public interface CalcHomeLocal extends EJBLocalHome {
    public CalcLocal create() throws CreateException; }
```
    - Interfaces métier

```
public interface Calc extends EJBObject {
    public double add(double val1, double val2) throws
        RemoteException; }
public interface CalcLocal extends EJBLocalObject {
    public double add(double val1, double val2); }
```

## Les Enterprise JavaBeans

### Principes de base

- Classe d'implémentation (nom : *CalcBean*)
  - Code de la logique métier
  - Implémente une interface spécifique au type d'EJB
    - *javax.ejb.SessionBean*
    - *javax.ejb.EntityBean*
    - *javax.ejb.MessageDrivenBean*
 } Dérivent de l'interface *javax.ejb.EnterpriseBean*
- Contenu
  - Méthodes métiers : *add()*
  - Constructeur : *CalcBean()*
  - Méthode de création : *ejbCreate()*
  - Méthodes de gestion du cycle de vie : *ejbActivate()*, *ejbPassivate()*, *ejbRemove()*

## Les Enterprise JavaBeans

### Principes de base

- Classe d'implémentation (nom : *CalcBean*)
  - Exemple
 

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

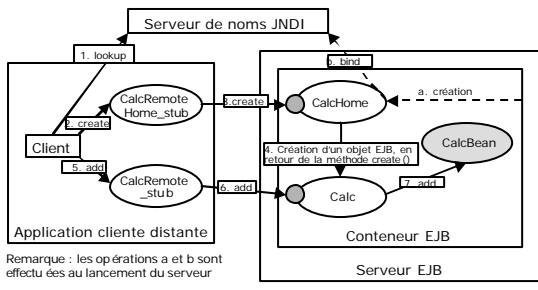
public class CalcBean implements SessionBean {

    public double add(double val1, double val2) {
        return val1+val2; }

    public CalcBean() { }
    public void ejbCreate() { }
    public void setSessionContext(SessionContext sc) { }
    public void ejbActivate() { }
    public void ejbPassivate() { }
    public void ejbRemove() { }
}
```

## Les Enterprise JavaBeans

### Fonctionnement du serveur et du client



## Les Enterprise JavaBeans

### Programmation du client

- Tâches à effectuer
  - Contact du serveur de noms
  - Récupération d'une référence sur l'interface home distante du bean
  - Création d'un objet EJB pour accéder au bean
  - Invocation des méthodes métier

```
import javax.naming.InitialContext;
import javax.naming.Context;
import javax.rmi.PortableRemoteObject;

public class CalcClient {
    public static void main(String args[]) {
        try {
            // Obtention du contexte initial par défaut
            Context initialContext = new InitialContext();

            // Recherche de l'interface home de distante de l'EJB
            Object objref = initialContext.lookup("Calc");
            CalcHome home = (CalcHome)PortableRemoteObject.narrow (objref,
            CalcHome.class);

            // Création et utilisation du bean
            Calc myCalc = home.create();
            System.out.println("3 + 2 = " + myCalc.add(3., 2.));

        } catch (Exception e) {
            e.printStackTrace(); }
    }
}
```

## Les EJB session

### Cycle de vie

- Pooling d'instances
  - Le conteneur possède un « pool » (réserve) d'instances qui lui évitent de créer et de détruire des objets pour chaque client
  - Le conteneur crée une instance de la classe d'implémentation d'un bean et la place dans le pool
    - À l'appel de la méthode *create()* par un client
    - S'il n'en a aucun de disponible
    - Dans la limite du nombre fixé par l'administrateur
  - Fonctionnement pour les différents types de beans session
    - Les beans sans état, une fois créés, restent opérationnels. Après utilisation, ils sont remis dans le pool ou détruits par le conteneur
    - Les beans avec états doivent être désactivés (*ejbPassivate()*) pour être remis dans le pool, et réactivés (*ejbActivate()*) avant de re-servir

## Les EJB entités

- Principes généraux
  - Permettent l'accès aux données métier stockées
    - Dans une BD
    - Dans un système propriétaire
    - Dans un système de stockage hétérogène
  - Chaque EJB entité représente un concept métier
    - Exemple : un compte en banque, un client, un achat...
- ⇒ Il y a autant d'instances d'EJB entités que de données archivées
- ⇒ Les EJB ont une « identité », matérialisée par une classe de clé primaire

## Les EJB entités

- Principes généraux
  - Les données sont accessibles par des méthodes spécifiques
  - Plusieurs utilisateurs peuvent utiliser le même bean en même temps
    - Accès successifs sans transaction
    - Accès transactionnels simultanés
    - ⇒ La gestion des accès concurrents est effectuée par le container

## Les EJB entités

- Persistance (gérée par le conteneur)
  - Rappels
    - un bean entité encapsule des données métier pour sécuriser leur utilisation
    - Chaque ensemble de données est représentée par un bean identifié (clé primaire)
    - Les données continuent d'exister après l'utilisation du bean
- ⇒ Il faut pouvoir sauvegarder l'état d'un bean (sérialisation)
- ⇒ La sauvegarde et la restauration des données sont des étapes critiques (gestion des transactions)

## Les EJB entités

- Persistance (gérée par le conteneur)
  - États d'un bean entité
    - Inexistant
      - Levée d'une exception à partir de n'importe quelle méthode
      - ⇒ Doit être créé par une méthode `newInstance()`, suivie de `setEntitycontext()`
    - Dans le pool
      - Le bean est désactivé, et n'est associé à aucune donnée
      - ⇒ Peut être activé
    - Prêt
      - Le bean est associé à un objet entité déterminé (il connaît sa clé primaire)
      - Il peut exécuter des méthodes métier
      - Le conteneur appelle les méthodes `ejbLoad()` et `ejbStore()` pour gérer la synchronisation du bean avec le support de persistance

## Les EJB entités

```
public interface Exemple extends EJBObject {
    public InfosExemple getInfosExemple() throws RemoteException;
    public void setInfosExemple(InfosExemple ie) throws
    RemoteException;
    ...
}

public class InfosExemple implements java.io.Serializable {
    public final Integer champInt;
    public final String champString;

    public InfosExemple(Integer i, String s) {
        champInt = i;
        champString = s;
    }
}
```

## Les EJB entités

```
public Collection findByCategorie(String categorie);

<entity>
<display-name>ExempleEJB</display-name>
<ejb-name>ExempleEJB</ejb-name>
...
<abstract-schema-name>ExempleSN</abstract-schema-name>
...
<query>
<query-method>
<method-name>findByCategorie</method-name>
<method-params>
<method-param>java.lang.String</method-param>
</method-params>
</query-method>
<ejb-ql>
select Object(a) from ExempleSN a where a.categorie = ?1
</ejb-ql>
</query>
```

## Les EJB entités

---

- Développement d'un bean CMP
  - L'interface home locale
    - Mêmes méthodes que l'interface distante *a priori*
    - Pas d'exception RemoteException

## Les EJB entités

---

- Développement d'un bean CMP/BMP
  - La classe d'implémentation
    - Bean BMP : codage de la sérialisation dans cette classe
    - Bean CMP : classe déclarée abstraite pour que le conteneur puisse la sous-classer et implémenter les méthodes de gestion de la persistance
    - Contenu (BMP)
      - Déclaration des méthodes de gestion des champs
      - Méthodes métiers (déclarées dans les interfaces métier)
      - Constructeur sans paramètre (appelé par le conteneur)
      - Méthodes de création (déclarées dans les interfaces home)
      - Méthodes sans entités (déclarées dans les interfaces home)
      - Méthodes internes d'accès aux données (méthodes Select)
      - Méthodes de rappel (appelées par le conteneur)

## Les EJB entités

---

- Développement d'un bean CMP
  - La classe d'implémentation
    - Déclaration des méthodes de gestion des champs

```
public abstract String getChampTexte ();
public abstract void setchampTexte (String texte);
public abstract Integer getChampInt ();
public abstract void setchampInt (Integer int);
```

## Les EJB entités

---

- Développement d'un bean CMP
  - La classe d'implémentation
    - Déclaration des méthodes de gestion des champs

```
public abstract String getChampTexte ();
public abstract void setchampTexte (String texte);
public abstract Integer getChampInt ();
public abstract void setchampInt (Integer int);
```
    - Implémentation des méthodes métier
      - Cf. beans session
    - Constructeur
      - ...

## Les EJB entités

---

- Développement d'un bean CMP
    - La classe d'implémentation
      - Méthodes de création
        - `ejbCreateXxx()` : initialise la création du bean
          - Met à jour les champs à partir des paramètres (utilise les *setters*)
          - Implémentation des méthodes déclarées dans les interfaces home
          - Type de la valeur de retour = type de la clé primaire
          - Doit retourner `null` (c'est le conteneur qui retrouve la clé primaire)
- ```
public Integer ejbCreateAvecUnParametre(Type1 param1,
throws CreateException {
    if (param1 == null) throw new CreateException;
    else setParam1(param1);
    return null; }
```

## Les EJB entités

---

- Développement d'un bean CMP
    - La classe d'implémentation
      - Méthodes de création
        - `ejbPostCreateXxx()` : appelée une fois le support de persistance (BD) mis à jour
          - Traitements nécessaires pour finaliser la création du bean
          - Ne doivent pas obligatoirement comporter une clause `throw CreateException`
          - Type de retour void
          - Mêmes noms et paramètres que la méthode `create()` correspondante
- ```
public void ejbPostCreateAvecUnParametre(Type1 param1) { }
```

## Les EJB entités

### ▣ Développement d'un bean CMP

- La classe d'implémentation
  - ▣ Méthodes sans entité : `ejbHomeXXX`
    - Doivent être déclarées dans les interfaces `Home`
    - Permettent de faire des traitements de lots  
Ex : trouver combien de clients se sont connectés à une certaine date.
    - Peuvent utiliser des méthodes `Finder` ou des requêtes à la base
    - Peuvent nécessiter de nombreux accès aux données  
⇒ À ne pas utiliser

## Les EJB entités

### ▣ Développement d'un bean CMP

#### ■ La classe d'implémentation

```
<query>
<query-method>
  <method-name>
    ejbSelectObjetsCommeCa
  </method-name>
  <method-params>
    < method-param>java.lang.String</ method-param>
  </method-params>
</query-method>
<ejb-ql>
select a.idDonnees from DonneesSN a where a.champCa = ?1
</ejb-ql>
</query>
```

## Les EJB entités

### ▣ Développement d'un bean CMP

#### ■ La classe d'implémentation

```
public abstract class MonEJBBean implements
javax.ejb.EntityBean {
    EntityContext ec = null;
    public void setEntitycontext(javax.ejb.EntityContext param) {
        ec = param;
    }
    public void unsetEntitycontext() {
        ec = null;
    }
    public void ejbActivate() { }
    public void ejbLoad() { }
    public void ejbStore() { }
    public void ejbRemove() { }
    public void ejbPassivate() { }
}
[...]
```

## Les EJB entités

### ▣ Développement d'un bean CMP

#### ■ La clé primaire

- ▣ Permet d'identifier le bean de manière unique
- ▣ Classe sérialisable
  - Dérivant de `java.lang.Object` (`Integer`, `String`..)
  - Spécifique au bean `MonBeanId`, mais la clé doit être composée d'un seul champ
  - Classe annexe sérialisable : méthodes à implémenter
    - Constructeur par défaut
    - `public boolean equals(Object other)`
    - `public int hashCode()`
- ▣ Le descripteur de déploiement doit indiquer
  - Le type de la classe (balise `< prim-key-class>`)
  - Le nom du champ (balise `< primkey-field>`)

## Les Enterprise JavaBeans

### ▣ Packaging et déploiement

- Packaging : assembler tous les éléments dans un fichier `ejb-jar`

- ▣ Interfaces métier et home (locales et/ou distantes)

*Calc.class, CalcLocal.class, CalcHome.class, CalsHomeLocal.class*

- ▣ Classe d'implémentation

*CalcBean.class*

- ▣ Classe de clé primaire (beans entités)

*MyEntityBeanKey.class*

- ▣ Descripteur de déploiement

*ejb-jar.xml*

## Les Enterprise JavaBeans

### ▣ Packaging et déploiement

- Déploiement : réalisé par le serveur d'applications

- ▣ Extraction du contenu du JAR

- ▣ Génération du code déployé (code nécessaire à la communication du bean et de ses clients)

- Objets EJB local et distant

- Implémentent l'interface métier

- Peuvent prendre en charge les fonctions de gestion de l'EJB (persistance, sécurité, transactions...)

- Objets EJB Home local et distant

- Implémentent l'interface home (méthode `create()`)

- Stubs, skeletons, etc. en fonction du type de serveur

- ▣ Enregistrement d'une référence à l'objet EJB Home dans un serveur de noms accessible via JNDI

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!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>
<description> Descripteur de d'eploiment du bean Calc</description>
<display-name> Calc</display-name >
<enterprise-beans>
<session>
<description> Calculatrice simple</description>
<display-name> Calc</display-name >
<ejb-name> Calc</ejb-name>
<home> calcul.CalcHome</home>
<remote> calcul.Calc</remote>
<ejb-class> calcul.CalcBean</ejb-class >
<session-type> Stateless</session-type >
<transaction-type> Container</transaction-type >
</session>
</enterprise-beans>
<assembly-descriptor >
<container-transaction >
<method >
<ejb-name> Calc</ejb-name >
<method-name > * </method-name>
</method >
<trans-attribute > Required</trans-attribute >
</container-transaction >
</assembly-descriptor >
</ejb-jar >

```

```

<!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>
<entity>
<ejb-name>ExempleBean</ejb-name>
<home>com.demo.ExempleHome</home>
<remote>com.demo.ExempleObject</remote>
<ejb-class>com.demo.ExempleBean</ejb-class >
<persistance-type >Container</persistance-type >
<prim-key-class>java.lang.String</prim-key-class >
<entrant>False</entrant>
<abstract-schema-name>ExempleSN</abstract-schema-name>
<cmp-field>
<field-name>ExempleId</field-name >
</cmp-field >
<cmp-field >
<field-name>ExempleData</field-name >
</cmp-field >
<primkey-field>ExempleId</primkey-field >
<query>
<query-method>
<method-name>findByCategorie</method-name >
<method-param>
<method-param> java.lang.String</method-param >
</method-param >
</query-method >
<ejb-ql>select Object(a) from ExempleSN a where a.categorie = ?1</ejb-ql >
</query >
<query>...</query >
</entity >
</enterprise-beans >
<assembly-descriptor >
...

```

## Les Enterprise JavaBeans

### □ Conclusion

#### ■ Les EJB sont des composants

- Sûrs
- Pratiques
- puissants
- Extensibles
- ... mais pas simples

⇒ Nécessitent une modélisation approfondie de l'application