Systèmes d'Information Avancés (et répartis)

Université Lyon 1
MIAGE

L. Médini, mars 2005

Plan des cours

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

Dire que les trois premiers prennent un cours, le quatrième 2 et le dernier 3.

Programmation serveur : outils pour les cours suivants

Types d'architectures (modularité, standardisation)

Middleware (CORBA, RMI, objets répartis)

Rappel : les EJB

- Un EJB
  - Est un composant transactionnel accessible à distance
  - Est intégré à un serveur d'applications
  - 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 »…

JTA : Java Transaction API ; JTS : Java Transaction Service

JMS : Java Message Service

JNDI : Java Naming and Directory Interface

Rappel : les EJB

- Trois types d'EJB
  - Session : représente un processus métier ; ne peut avoir qu'un seul client à un moment donné
    - Avec états : c'est le même client qui réalise toutes les invocations (exemple : panier électronique)
    - Sans état : peut être utilisé successivement par plusieurs clients (exemple : calcul d'une distance)
  - Entité : représente un objet métier persistant (exemple : une commande, un article, un compte bancaire…) ; deux moyens de gérer la persistance
    - CMP : Container Managed Persistance
    - BMP : Bean Managed Persistance
  - Message : composant asynchrone qui peut échanger des messages par l'intermédiaire de Java Message Service

JTA : Java Transaction API ; JTS : Java Transaction Service

JMS : Java Message Service

JNDI : Java Naming and Directory Interface

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 des beans entités (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)

JTA : Java Transaction API ; JTS : Java Transaction Service

JMS : Java Message Service

JNDI : Java Naming and Directory Interface

Les EJB entités

- Persistance des beans entités (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 desactivé, et n'est associé à aucune donnée
      - ⇒Peut être activé et passivé
    - 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

JTA : Java Transaction API ; JTS : Java Transaction Service

JMS : Java Message Service

JNDI : Java Naming and Directory Interface

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;
  }
}
```

Un Handle est une représentation sérialisable de l'objet EJB distant. Peut être stocké sur un support persistant par le client pour accéder à l'instance du bean et restaurer son état. N'est utile que pour les beans session avec étés et les beans entités.

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>
```

Ici, « MonEJB » représente une référence sur l'interface métier de l'EJB.

Méthodes create : create(), createSansId()… L'ordre et les types des paramètres sont libres.

Méthodes find : findByPrimaryKey(), findByCategory() ; ne doivent pas être implémentées dans la classe d'implémentation (le sont par le conteneur).

Les EJB entités

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

Ici, « MonEJB » représente une référence sur l'interface métier de l'EJB.

Méthodes create : create(), createSansId()… L'ordre et les types des paramètres sont libres.

Méthodes find : findByPrimaryKey(), findByCategory()

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)

JTA : Java Transaction API ; JTS : Java Transaction Service

JMS : Java Message Service

JNDI : Java Naming and Directory Interface

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);
```

JTA : Java Transaction API ; JTS : Java Transaction Service

JMS : Java Message Service

JNDI : Java Naming and Directory Interface

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

      ...

JTA : Java Transaction API ; JTS : Java Transaction Service

JMS : Java Message Service

JNDI : Java Naming and Directory Interface

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; }
```

JTA : Java Transaction API ; JTS : Java Transaction Service

JMS : Java Message Service

JNDI : Java Naming and Directory Interface

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) { }

JTA : Java Transaction API ; JTS : Java Transaction Service

JMS : Java Message Service

JNDI : Java Naming and Directory Interface

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

JTA : Java Transaction API ; JTS : Java Transaction Service

JMS : Java Message Service

JNDI : Java Naming and Directory Interface

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>
```

JTA : Java Transaction API ; JTS : Java Transaction Service

JMS : Java Message Service

JNDI : Java Naming and Directory Interface

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( ) { }

  […]
}
```

JTA : Java Transaction API ; JTS : Java Transaction Service

JMS : Java Message Service

JNDI : Java Naming and Directory Interface

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>)

JTA : Java Transaction API ; JTS : Java Transaction Service

JMS : Java Message Service

JNDI : Java Naming and Directory Interface

```
<!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>
   <persistence-type>Container</persistence-type>
   <prim-key-class>java.lang.String</prim-key-class>
   <reentrant>False</reentrant>
   <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-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>
   <query>...</query>
  </entity>
 </enterprise-beans>
 <assembly-descriptor>
  ...
```

JTA : Java Transaction API ; JTS : Java Transaction Service

JMS : Java Message Service

JNDI : Java Naming and Directory Interface

Les EJB entités

- Références
  - Code source
    http://java.sun.com/developer/codesamples/ejb.html
  - Tutorial
    http://java.sun.com/j2ee/1.4/docs/tutorial/doc/