Réseaux middlewares et serveurs d'application

Partie 2 - EJB

Jules Chevalier
jules.chevalier@univ-st-etienne.fr

Université Jean Monnet - Télécom Saint Etienne

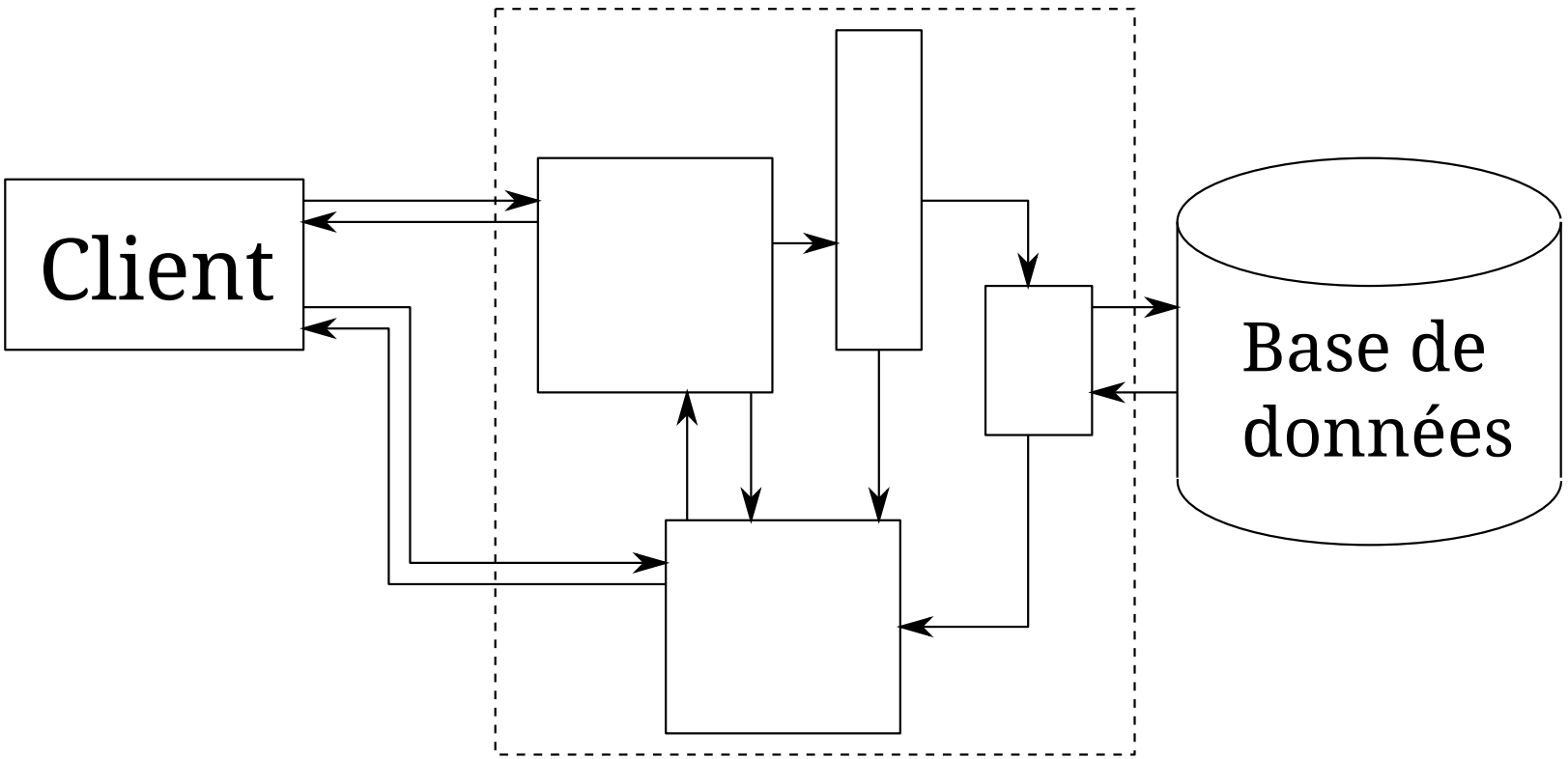
novembre 2014

Entreprise Java Beans

Présentation

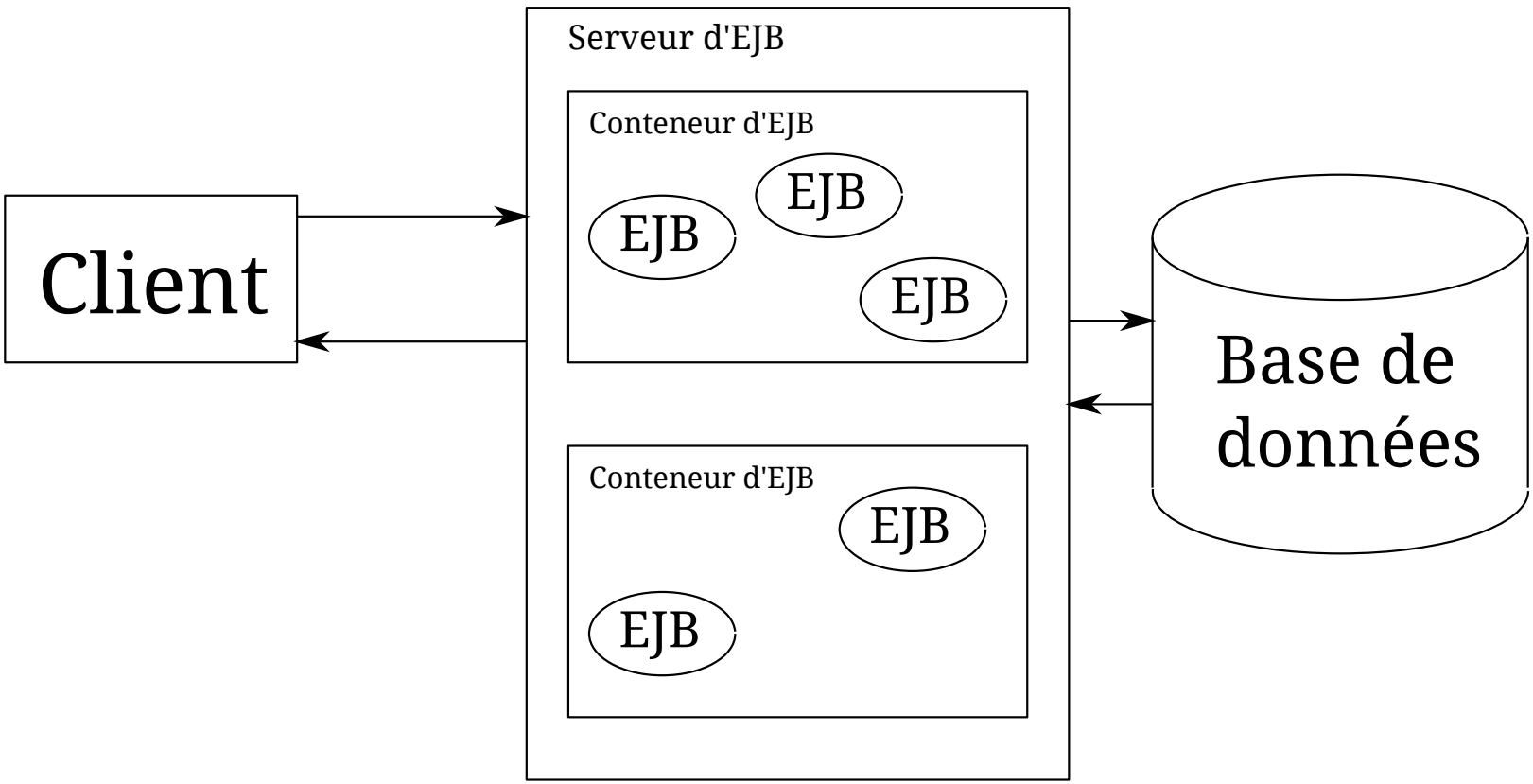
- Éléments de Java EE, ils permettent de développer des applications distribuées
- Ce sont des composants réutilisables, persistants et transactionnels qui proposent des services
- Ces "briques" permettent de construire une application distribuée complexe
- Ils ne peuvent s'exécuter que dans un environnement serveur dédié : un serveur EJB

Entreprise Java Beans



Sans EJB

Entreprise Java Beans



Avec EJB

Plan

Plan

Historique

Origine

- Années 90 : Apparition des systèmes multi-tiers, une architecture client/serveur couplée avec bases de données et applications. Les middlewares sont imaginés pour faciliter la communication entre les parties, et simplifier la tâche de développement
- Sun propose des middleware sous forme de composants Java standardisés : les EJB

Historique

Versions

- EJB 1.0 (1998)
- EJB 1.1 (1999)
- EJB 2.0 (2001)
- EJB 2.1 (2003)
- EJB 3.0 (2006)
- EJB 3.1 (2009)

Historique

EJB 3

- A partir de EJB 3.0, le processus de création d'applications à base d'EJB est grandement simplifiée
- Les fichiers de configurations disparaissent pour être remplacés par des annotations intégrées au code
- Une configuration "par défaut", suffisante dans la plupart des cas, allège considérablement le code
- La persistance est facilitée grâce à JPA (Java Persistance API)
- L'utilisation de POJO et POJI (Plain Old Java Object/Interface) simplifie le développement
- Il n'est plus utile d'implémenter les méthodes du cycle de vie du Bean

Historique

EJB 3.1

- Les interfaces Local ne sont plus obligatoires, même si elles sont recommandées
- Les interfaces Remote sont elles toujours obligatoires
- Le type EJB Singleton, qui ne peut être instancié qu'une fois dans un conteneur
- Gestion concurrente des EJB
- EJB Lite : Une version allégée pour les applications locales embarquée
- Le packaging des EJB est simplifié
- La standardisation des nom JDNI (annuaire)
- Invocations asynchrones

Plan

Avantages

- Les EJB définissent un standard JavaBean
- Facilitent le développement, l'intégration, la réutilisation et l'interopérabilité des composants
- Ils gèrent les transaction et sont sécurisés
- La mise en place, auparavant complexe, tend à se simplifier, notamment depuis la version 3

Plan

Concept

Le client

- Assure la saisie et l'affichage des données
- Localise le Bean à travers l'annuaire JDNI (Java Naming and Directory Interface)
- Utilise le Bean grâce à ses méthodes accessibles, via un conteneur

Concept

Le Bean

- Implémente les différentes méthodes nécessaires à l'exécution du service
- Implémente une interface locale, et une interface distante
- "S'imbrique" avec les autres EJB pour constituer l'application

Concept

Le conteneur

- Fournit un service de nommage
- Gère le cycle de vie
- Permet au composant d'être persistant
- Sécurisé et transactionnel

- Encapsule un (ou plusieurs) composant(s)
- Transmet les invocations de méthodes du client
- Transmet les invocations internes

- Gère la création et la destruction des composants
- Peut sérialiser l'EJB pour le sauvegarder

Concept

Les interfaces

- Une interface `Local`
  - Utilisée par les autres EJB
  - Utilisée pour l'exécution des services
  - Plus rapide, puisqu'elle ne passe pas par le réseau
- Une interface `Remote`
  - Utilisée par le client
  - Expose les méthodes accessibles au client

Plan

Processus de développement

Développement des EJB

- Définition des interfaces locales et distantes
- Implémentation des EJB

Processus de développement

Déploiement des EJB

- Construction de l'archive JAR
- Déploiement sur une serveur EJB

Processus de développement

Utilisation

- Implémentation du client
  - Servlet
  - Application Java
  - Applet
  - etc

Processus de développement

Convention de nommage

- Aucune règle n'est imposée
- Mais il mieux nommer ses EJB comme suit :
  - Nom du Bean : *CalculEJB*
  - Nom la classe métier : *CalculBean*
  - Interface locale : *CalculLocal*
  - Interface distante : *CalculRemote*

Plan

Les annotations

Principe

- Avant Java 5, les méta données sont uniquement utilisées par Javadoc
- Java 5 introduit les annotations, des méta données contenues dans le code source
- Les annotations ont une syntaxe dédiée, stockées dans le bytecode à la compilation
- Une certain nombre d'annotations standards sont définies par Java 5, mais il est possible de définir les siennes
- Les annotations précèdent l'entité qu'elle concerne. Elles sont précédées d'un **@**

Les annotations

Types d'annotations

- Les marqueurs (markers) :

```
@Override
```

- Les annotations paramétrées :

```
@SuppressWarnings("unchecked")
@SuppressWarnings(value="unchecked")
@SuppressWarnings(value={"unchecked", "deprecation"})
```

- Les annotations multiparamétrées :

```
@Annotation(param1="A", param2="B")
```

Les annotations

Les paramètres

- Les paramètres peuvent être :
  - Une chaîne de caractère
  - Un type primitif
  - Une énumération
  - Une annotation
  - Le type Class
- Les annotations peuvent avoir des valeurs par défaut

Les annotations

Utilisations

- Génération de documents (Javadoc)
- Génération de code
- Génération de fichiers
- Mappage de bases de données

Les annotations

Traitement des annotations

- Les annotations sont traitées au moment de la compilation
- Un API permet de gérer ces traitements. Elle est regroupée dans :
  - com.sun.mirror.apt
  - com.sun.mirror.declaration
  - com.sun.mirror.type
  - com.sun.mirror.util
- L'API reflexion permet le traitement des annotations au moment de l'exécution

Les annotations

Attention

- Les annotations donnent des informations sur des entités, elles n'ont pas d'effet direct dessus
- Les annotations permettent de générer de nouveaux fichiers, mais pas de modifier le code existant

Les annotations

Les annotations standards

- Un certain nombre d'annotations sont gérées par défaut
  - @Deprecated : Informe que la classe est dépréciée
  - @Override : Marque une méthode qui surcharge une méthode héritée
  - @SuppressWarning : Inhibe certains avertissements (deprecation, unchecked, ...)

Les annotations

Intérêts

- Évite les fichiers de configuration
- Les configurations sont dans le code
- Embarquées dans les classes compilées

Inconvénients

- La configuration est dispersée dans le code
- En cas de changement, il faut parcourir tous les fichiers

Plan

Les types d'EJB

Plusieurs types d'EJB

- EJB Session
  - Stateless
  - Stateful
- EJB Entity
- EJB MessageDriven
- EJB Singleton
- EJB Lite

Plan

Les différents EJB

EJB Session

- Façade des fonctionnalités proposées au client
- Contient la logique métier
- Peut utiliser d'autres composants
- Peut être invoqué :
  - Localement :
    - Dans la même JVM que l'EJB
    - Plus performant (pas d'échanges réseau)
  - A distance :
    - Le client est dans une autre JVM

Les différents EJB

EJB Session - Interfaces

- Deux interfaces permettent de définir les méthodes accessibles localement ou à distance
- L'interface Local pour les appels locaux
- L'interface Remote pour les appels distants

Les différents EJB

EJB Session - Interface Local

```
1  @Local
2  public interface CalculLocal {
3     public long additionner(int valeur1, int valeur2);
4     public long oppose(int valeur1);
5  }
```

Les différents EJB

EJB Session - Interface Remote

```
1  @Remote
2  public interface CalculRemote {
3     public long additionner(int valeur1, int valeur2);
4  }
```

Les différents EJB

EJB Session Stateless

- L'état de l'EJB n'est pas conservé entre les appels du client
- Plus simples, ils sont plus performants, puisque le conteneur ne doit pas instancier un nouvel EJB à chaque appel
- Le conteneur gère un pool d'instances, créées ou détruites au besoin

Les différents EJB

EJB Session Stateless

```
1  @Stateless
2  public class CalculBean implements CalculRemote, CalculLocal {
3    public long additionner(int valeur1, int valeur2) {
4      return valeur1 + valeur2;
5    }
6    public long oppose(int valeur1){
7      return -valeur1;
8    }
9  }
```

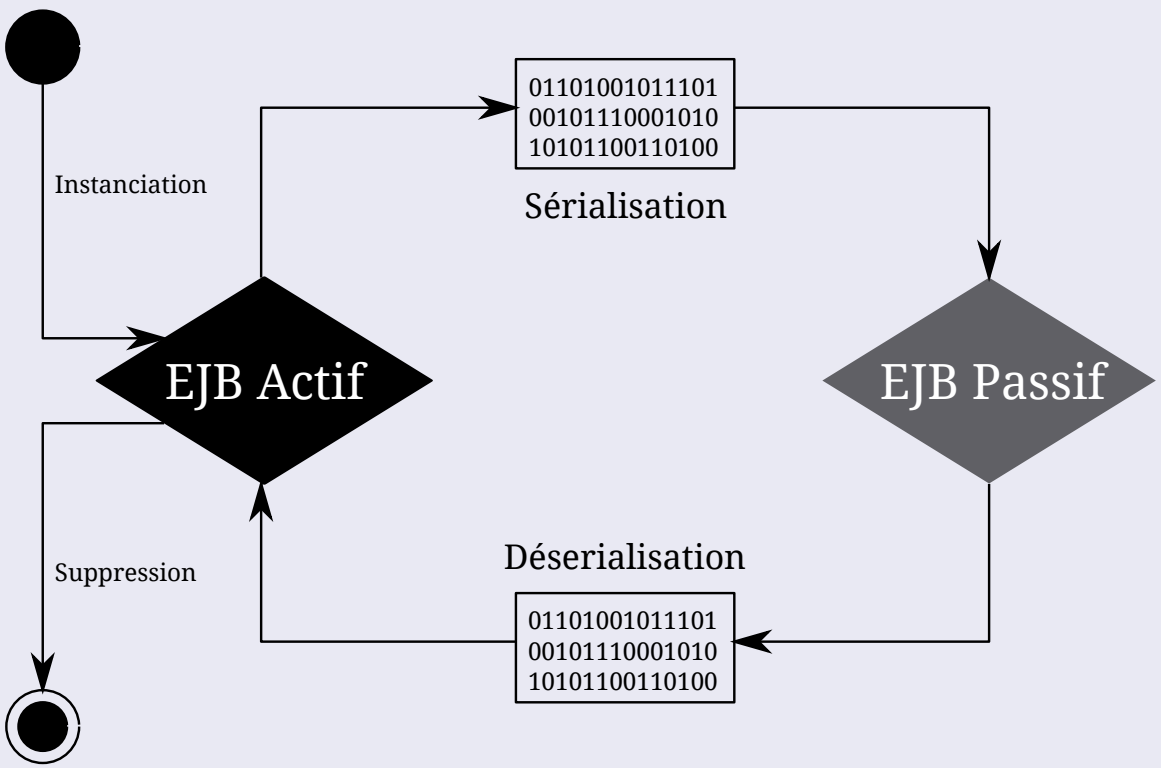
Les différents EJB

EJB Session Stateful

- L'état de l'EJB est conservé d'un appel à l'autre
- Cependant, l'état n'est pas persistant, il est perdu à la destruction de l'objet
- C'est par exemple le panier sur un site de vente en ligne
- Le conteneur peut sérialiser l'EJB et le sauvegarder si la mémoire manque

Les différents EJB

EJB Session Stateful



Cycle de vie

Les différents EJB

EJB Session - Interface Remote

```
1  @Remote
2  public interface BonjourRemote {
3     public void login(String nom);
4     public String bonjour();
5  }
```

Les différents EJB

EJB Session - Interface Local

```
1  @Local
2  public interface BonjourLocal {
3     public void login(String nom);
4     public String bonjour();
5     public String nom();
6  }
```

Les différents EJB

EJB Session - Bean Bonjour

```
1   @Stateful
2   public class BonjourBean implements BonjourLocal, BonjourRemote{
3     String nom = null;
4     public void login(String nom){
5       this.nom = nom;
6     }
7     public String bonjour(){
8       if(this.nom==null)
9         return "Bonjour";
10      else
11        return "Bonjour "+this.nom;
12    }
13    public String nom(){
14      return this.nom;
15    }
16  }
```

Les différents EJB

EJB Session - Interfaces

- Avec EJB 3.1, les interfaces locales sont optionnelles
- On peut annoter directement l'EJB Stateless ou Stateful

```
@Stateless
public class MonBean {
public String saluer(){ return "Bonjour";}
}
```

- Pour l'utiliser dans un autre EJB, on utilise *l'injection de dépendance*

```
@EJB
private MonBean monBean;
```

- Cela remplace l'utilisation de new

Les différents EJB

Les exceptions

- Les EJB permettent de lever des exceptions qui seront envoyées au client par le conteneur
- L'exception peut déclencher un rollback

Plan

Les différents EJB

EJB Entity

- Assure la gestion de la persistance des données
- Depuis la version 3.0, c'est un simple POJO qui utilisent l'API Java Persistence (JPA)
- *Mappe* une table de la base de données dans laquelle ses champs sont inscrits
- Est annoté par @Entity

Plan

Les différents EJB

EJB MessageDriven

- Apparu avec la version 3.0
- Permet de réaliser des traitements asynchrones
- Utilise des messages dans une file JMS pour déclencher les exécutions
- Pas d'interface locale ou distante, mais implémente javax.jms.MessageListener
- Implémente la méthode OnMessage(Message)

Plan

Les différents EJB

EJB Singleton

- Une seule instance de l'EJB est utilisable et partagée dans le conteneur
- Par défaut, toutes les méthodes d'un EJB Singleton sont thread-safe et transactionnelles
- Peut être utilisé simultanément par plusieurs threads

Les différents EJB

EJB Singleton

- Possibilité de demander au conteneur de le lancer au démarrage de l'application
- D'autres EJB peuvent être lancés avant, en définissant des dépendances entre les EJB
- Souvent utilisé pour partager ou mettre en cache des données dans l'application
- Offre tous les services d'un EJB : sécurité, transaction, injection de dépendances, gestion du cycle de vie et intercepteurs

Plan

Les différents EJB

EJB Lite

- Version légère d'un conteneur d'EJB
- Utile pour des applications desktop, embarquées ou dans un conteneur web

Les différents EJB

Supporte

- les EJB Session (Stateless, Stateful, et Singleton)
- les EJB avec interface locale ou sans interface
- L'injection
- Les intercepteurs
- La sécurité et les transactions

Les différents EJB

Ne supporte pas

- Les EJB 2.x
- L'invocation via RMI/IIOP
- Les EJB Session avec interface distante
- Les EJB MessageDriven
- Le support des endpoints pour les services web
- Le service Timer

Plan

Les intercepteurs

Présentation

- Exécuté selon deux types d'évènements :
  - L'invocation de méthodes métiers
  - Les évènements liés au cycle de vie de l'EJB
- Permet de définir les traitements exécutés lors de ces évènements
- Ils sont utilisables avec les EJB Session et Message Driven
- Les annotations dédiées sont dans le package javax.interceptor

Bibliographie

- http://www.jmdoudoux.fr/
- http://www.journaldunet.com/solutions/0104/010420_ejb.shtml
- http://cedric.cnam.fr/~farinone/SAR/EJB.pdf
- http://wiki.eclipse.org/