Java EE 6

Université Française d'Egypte
Richard Grin
Version 0.7 – 13/10/12

Plan (1)

- Présentation
- Containers
- Formats de distribution
- Les différents rôles
- JNDI

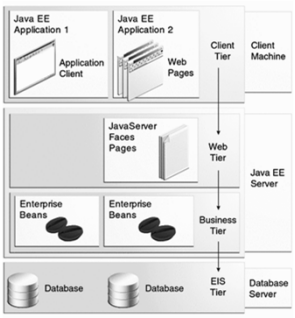
Présentation de Java EE

- Spécifications pour écrire des applications d'entreprise en Java
- Applications multi-tiers avec, le plus souvent une interface Web
- Les processus métier sont implémentés sur le serveur, entre la couche « interface » et la couche « base de données »

Architecture d'une application d'entreprise

- Les applications d'entreprise modernes sont des applications multi-tiers
- Les Entreprise Java Beans (EJB) sont des composants du « *middle tiers* », au cœur des processus métier, placés sur le serveur entre la base de données et les clients
- Le plus souvent l'interface utilisateur est de type Web, implémentée avec Java Server Faces (JSF) et des servlets

Application multi-tiers



Services offerts par Java EE

- Sécurité, accès distants, accès à la base de données, transactions,…
- Grâce à des composants inclus dans des containers

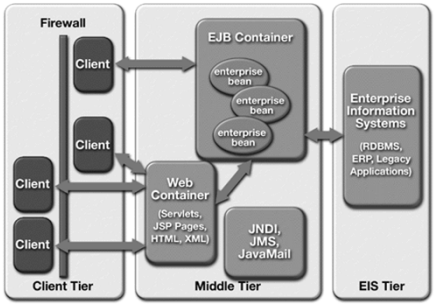
Composants

- Interface utilisateur Web avec JSF et servlets
- Processus métier avec les EJB, composants réutilisables
- Interface base de données avec les entités JPA

Containers

- Les applications Java EE ne peuvent fonctionner que dans un serveur d'application Java EE
- Un serveur d'application utilise des containers pour gérer les divers composants
- Chaque type de composant est géré par un container : client, Web, EJB
- Les containers offrent des services qui facilitent le développement : sécurité, transaction, injection de composants ou référencement par JNDI, appels distants de méthodes,…

Application multi-tiers - containers



Format de distribution

- Les applications sont distribuées dans des fichiers, dits fichiers d'archive (de type fichiers zip), qui réunissent les divers composants
- Chaque unité contient
  - les composants
  - les ressources utilisées par les composants
  - un ou plusieurs fichiers de configuration

Descripteur de déploiement standard

- Pour informer le container des besoins middleware, on utilise un ou plusieurs *descripteurs de déploiement* (XML)
  - Standardisé
  - Descripteurs peuvent être modifiés sans devoir recompiler
- web.xml

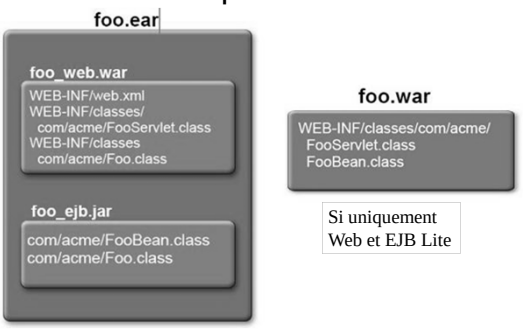
Descripteur de déploiement spécifique

- Descripteurs spécifiques au serveur d'application
- Chaque vendeur ajoute des paramètres spécifiques qui permettent de configurer des fonctionnalités en plus : load-balancing, persistance complexe, clustering, monitoring…
- Dans des fichiers spécifiques (glassfish-resource.xml, glassfish-web.xml ou glassfish-application.xml avec GlassFish)

Types de fichiers d'archive

- Jar (Java ARchive) : fichiers d'archive habituels qui contiennent les EJB, les classes Java ordinaires et les ressources associées
- War (Web ARchive) : modules liés au Web, qui contiennent les servlets, fichiers HTML, JSF, EJB, et les ressources associées
- Ear (Entreprise ARchive) : réunissent des modules jar ou war
- Une application Java EE peut être distribuée sous une de ces 3 formes (jar, war ou ear)

Avant et après Java EE 6



Si uniquement Web et EJB Lite

Modules

- Java EE définit plusieurs types de modules qui ont leur propre format de distribution :
  - Module d'application cliente dans un fichier jar qui peut être exécuté dans un environnement Java SE (pas EE) ou un container d'application cliente (ACC) ; si ACC, le jar peut contenir un fichier descripteur de déploiement META-INF/application-client.xml
  - Module EJB qui contient des EJB dans un fichier jar ; peut contenir un fichier descripteur de déploiement META-INF/ejb-jar.xml

Modules

- Module d'application Web qui contient des servlets, pages JSF, services Web, pages HTML et XHTML, feuilles CSS,… dans un fichier jar avec une extension .war ; peut contenir un fichier descripteur de déploiement WEB-INF/web.xml ; peut aussi contenir des EJB « lite » (pas de MDB) avec un fichier descripteur de déploiement optionnel WEB-INF/ejb-jar.xml ; classes Java dans WEB-INF/classes et librairies externes utilisées dans des fichiers jar placés dans WEB-INF/lib
- Module entreprise qui peut contenir des modules EJB ou Web et des librairies externes, dans un fichier jar avec une extension .ear

Technologies et profiles

Technologies

- Java EE utilise de nombreuses technologies
- Pour les applications Web : servlet, JSF,…
- Pour les applications d'entreprise : CDI, EJB, JPA, JTA, validation bean, JMS, JavaMail,…
- Pour les services Web : JAX-RS, JAX-WS, JAXB,…
- Pour la sécurité : SSL, cryptographie,…

Alléger Java EE

- Des technologies utilisées dans les versions précédentes de Java EE ont été remplacées par d'autres technologies, par exemple les EJB entités
- La spécification Java EE permet le « pruning » (taille, élagage) de certaines technologies qui deviennent optionnelles dans les serveurs d'applications

Profile

- Toujours pour alléger, la spécification Java EE permet de définir des profiles qui n'utilisent pas toutes les technologies
- Le premier profile défini est le profile Web ; il inclut les technologies qui suffisent le plus souvent pour écrire une application Web : servlets, JSF, CDI, EJB Lite, JPA, JTA, validation bean

EJB Lite

- Ne contient qu'une partie de la spécification EJB :
  - beans stateless, stateful et singleton (donc pas EJB message)
  - interfaces locales seulement (pas d'interface distante)
  - gestion des transactions par le container ou le bean
  - définition des restrictions d'accès déclarative et par programmation
  - intercepteurs
  - API embarquée

Les différents rôles

Fournisseur d'EJB

- Il écrit le code des EJB, leurs interfaces et leurs fichiers de déploiement

Assembleur d'applications

- Il assemble différents éléments, dont les EJB, pour écrire une application
- Autres types de composants : clients Java GUI, applets, servlets, JSP, Java beans
- Il travaille sur les fichiers de déploiement des EJB pour ajouter des informations d'assemblage

Responsable du déploiement

- Il connaît les systèmes dans lesquels l'application va se déployer et adaptent l'application à cet environnement d'exécution
- Il intègre l'application à d'autres applications existantes
- Pour cela il est amené à modifier les descripteurs de déploiement

Administrateur système

- Il contrôle l'application quand elle s'exécute
- Il intègre l'application dans l'environnement d'exécution
- Il configure et administre cet environnement pour permettre l'exécution efficace de l'application

Contenu du cours

- Ce cours sera orienté vers le développement Web ; il couvrira les parties suivantes :
  - Servlet
  - JSF
  - JPA
  - EJB
- Il utilisera le serveur d'applications GlassFish, version 3

JNDI

Généralités

- Les composants ont besoin d'accéder à d'autres composants ou à des ressources (source de données JDBC, service de messagerie, ressource javaMail,…)
- Ces composants ou ressources sont enregistrés dans un annuaire pour être disponibles dans le code Java
- JNDI est l'API de Java EE pour enregistrer et récupérer dans un annuaire des objets, des services ou des ressources

Nom JNDI

- Les ressources JNDI sont identifiées par des noms
- Exemple de nom :
  `java:global/appli1/ejb1/MonEJB`

Code pour récupérer un EJB avec JNDI

```
import javax.naming.InitialContext;
...
try {
   InitialContext ic = new InitialContext();
   MonEJB monEjb = (MonEJB)
     ic.lookup("java:global/ejb1/MonEJB");
   ...
} catch (NamingException e) {
   e.printStackTrace();
}
```

Si c'est possible, plus simple d'injecter l'EJB !
Voir cours sur CDI.

Code pour injecter un EJB

```
@EJB
MonEJB monEjb;
```


Injection et JNDI

- Depuis l'injection de dépendance des dernières versions de Java EE, l'utilisation explicite de JNDI est bien moins fréquente
- Cependant, lorsque l'injection de dépendance n'est pas possible, le développeur peut être amené à utiliser JNDI explicitement ; par exemple, pour l'écriture d'un validateur JSF 2.0
- En interne le serveur d'application utilise JNDI pour l'injection

JNDI pour les clients

- Pour un client non Web il faut indiquer comment trouver le service JNDI qui va, par exemple, fournir une référence vers un bean session enregistrer sur le serveur
- On utilise alors souvent un fichier de propriétés qui contient les informations nécessaires à la localisation du service JNDI

Standardisation dans Java EE 6

- EJB 3.1 a standardisé les noms associés à des composants ou ressources, ce qui facilite le transfert d'une application sur un autre serveur d'application
- Les exemples suivants concernent des noms d'EJB. On suppose que l'application se nomme « app1 » ; un module est un fichier « .war » ou « .jar » dans lequel l'objet est inséré (enlever l'extension pour désigner le module)

3 niveaux de noms JNDI (1/2)

- Niveau global (<application> seulement si dans un fichier « .ear »)
  java:global[/<application>]/<module>/<bean>[!<interface>]
  <bean> est, par défaut, le nom de la classe du bean (**sans le paquetage**)
  <interface> est le nom **complet** de l'interface ou de la classe pour les beans sans interface
- Exemples : java:global/app1/ejb1/Bean1
  java:global/app1/ejb1/Bean1!p1.Bean1

3 niveaux de noms JNDI (2/2)

- Niveau application : java:app/<module>/<bean>[!<interface>]
- Exemple : java:app/ejb1/bean1
- Niveau module : java:module/<bean>[!<interface>]
- Exemple : java:module/bean1

Donner un autre nom

- Il est possible de donner un autre nom (en plus du nom que l'on vient de voir) à un bean avec l'attribut « name » de l'annotation @EJB (voir cours sur les EJB)
- Exemple : @EJB(name="java:global/module1/autrenom")

Pour connaître le nom JNDI

- Pour les débutants il est parfois difficile de trouver le nom JNDI
- Si on travaille avec GlassFish, une bonne façon de trouver le nom global est de lire les logs du serveur lorsque l'application est déployée en cherchant les lignes qui commencent par « INFO: Portable JNDI names for »
- Se renseigner pour savoir comment avoir la même information avec les autres serveurs

Niveau de nom JNDI pour les ressources

- Il y a aussi le niveau composant si la portée de la ressource est l'EJB dans lequel la ressource est désignée : java:comp/env/nom-ressource

Définir une ressource JNDI

- La façon de définir les ressources désignées par un nom JNDI dépend le plus souvent du serveur d'application
- Avec GlassFish on peut, par exemple, définir une source de données avec la console d'administration

Bibliographie

- Tutoriel Java d'Oracle (en anglais, gratuit) : http://download.oracle.com/javaee/6/tutorial/doc/
- Beginning Java EE 6 Platform with GlassFish 3, Antonio Goncalves, éditions Apress