EJB

Olivier Cailloux

LAMSADE, Université Paris-Dauphine

Version du 12 septembre 2016

Module EJB

Module EJB

- Fichier `.jar`
- Fichiers `.class` EJB et autres
- `META-INF/ejb-jar.xml` : descripteur pour conteneur EJB (attributs de transaction, sécurité, ...)
- `META-INF/glassfish-ejb-jar.xml` : descripteur pour glassfish
- `META-INF/MANIFEST.MF`

Sortes d'EJB

- Session bean ; message-driven bean (et entity bean, facultatif) (diapos valables pour session beans)
- Session bean : `@Stateless` (SLSB), `@Stateful` (SFSB) ou `@Singleton`
- Stateless : conteneur instancie et gère un pool de beans
- Stateful et Singleton : ont un état
- Singleton : conteneur instancie et s'assure de l'unicité
- Stateful : norme EJB ne gère pas son cycle de vie (à gérer par le client sauf si CDI)
- Stateful peut être *passivated* : implémenter Serializable (facultatif)
- Singleton : conteneur s'assure de l'unicité
- Tous types : thread-safe (mais attention aux objects accessibles depuis le bean)

Types d'exposition

- Le conteneur fournit une vue client pour donner accès
- Pattern ?

Types d'exposition

- Le conteneur fournit une vue client pour donner accès
- Pattern ? Proxy !

Types d'exposition

- Le conteneur fournit une vue client pour donner accès
- Pattern ? Proxy !
- Le développeur déclare comment l'EJB sera accessible
  - No-interface view : accès direct et local à l'EJB
  - Local view : accès local via une interface « business »
  - Remote view : accès via une interface « business »
- Ou accès via vue décrite en WSDL via endpoint JAX-WS, pour session bean stateless ou singleton
- Accès local : client et EJB dans même application Java EE

  (peut être relaché hors standard)
- Inconvénient de l'accès Remote ?

Types d'exposition

- Le conteneur fournit une vue client pour donner accès
- Pattern ? Proxy !
- Le développeur déclare comment l'EJB sera accessible
  - No-interface view : accès direct et local à l'EJB
  - Local view : accès local via une interface « business »
  - Remote view : accès via une interface « business »
- Ou accès via vue décrite en WSDL via endpoint JAX-WS, pour session bean stateless ou singleton
- Accès local : client et EJB dans même application Java EE

  (peut être relaché hors standard)
- Inconvénient de l'accès Remote ? Sérialisation !

Types d'exposition

- Le conteneur fournit une vue client pour donner accès
- Pattern ? Proxy !
- Le développeur déclare comment l'EJB sera accessible
  - No-interface view : accès direct et local à l'EJB
  - Local view : accès local via une interface « business »
  - Remote view : accès via une interface « business »
- Ou accès via vue décrite en WSDL via endpoint JAX-WS, pour session bean stateless ou singleton
- Accès local : client et EJB dans même application Java EE

  (peut être relaché hors standard)
- Inconvénient de l'accès Remote ? Sérialisation !
- Recommandations. No-interface view : marquer le bean `@LocalBean`. Business interface : l'implémenter et la marquer `@Local` ou `@Remote`.

Convention de nommage

- Business interface : truc
- Classe (et nom du bean) : trucBean

Invocation

SLSB et Singleton session beans

- Injecter EJB via DI : annoter la référence `@EJB` (v. aussi @Inject via CDI)
- À utiliser pour les EJB dont le conteneur gère le cycle de vie !

Conteneur rend EJBs accessible via JNDI :
`java:global[/<app>]/<module>/<bean>[!<interface>]` ;
`java:app/<module>/<bean>[!<interface>]` ;
`java:module/<bean>[!<interface>]`

SFSB

- Création explicite via JNDI (sauf gestion via CDI)
- Destruction explicite via méthode EJB annotée @Remove

JNDI : aperçu

JNDI

- JNDI : Java Naming and Directory Interface (Java SE)
- API pour accéder à des objets via leur nom
- Et à des « naming contexts »
- Indépendant du stockage du répertoire

Accès

- Créer un `InitialContext`
- Utiliser méthodes de `Context` t.q. `lookup(String name)`
- Accès en lecture uniquement dans cas EJB

Environnement

- L'EJB a un environnement
- Accessible via JNDI dans le naming context `java:comp/env`
- *Ressource* : référence dans env. d'un composant Java EE
- Permet la paramétrisation de l'EJB (exemple : valeur entière)
- Depuis EJB, accéder via JNDI
- Ou utiliser CDI et `@Resource`
- Déclaration : où ?

Environnement

- L'EJB a un environnement
- Accessible via JNDI dans le naming context `java:comp/env`
- *Ressource* : référence dans env. d'un composant Java EE
- Permet la paramétrisation de l'EJB (exemple : valeur entière)
- Depuis EJB, accéder via JNDI
- Ou utiliser CDI et `@Resource`
- Déclaration : où ? Dans descripteur standard EJB.

Environnement

- L'EJB a un environnement
- Accessible via JNDI dans le naming context `java:comp/env`
- *Ressource* : référence dans env. d'un composant Java EE
- Permet la paramétrisation de l'EJB (exemple : valeur entière)
- Depuis EJB, accéder via JNDI
- Ou utiliser CDI et `@Resource`
- Déclaration : où ? Dans descripteur standard EJB.
- Resource non injectée si pas de valeur dans le descripteur : utilise la valeur donnée dans le code

Déclarer et accéder à une ressource

Exemple de déclaration d'une ressource (de type « simple »)

`java:comp/env/ejb.MySesEjbBean/zeNumber` via `ejb-jar.xml`

```
<enterprise-beans>
  <session>
    <ejb-name>MySesEjbBean</ejb-name>
    <env-entry>
      <env-entry-name>ejb.MySesEjbBean/zeNumber</env-entry-name>
      <env-entry-type>java.lang.Integer</env-entry-type>
      <env-entry-value>150</env-entry-value>
    </env-entry>
  </session>
</enterprise-beans>
```

Accès à la ressource

```
@Stateless
public class MySesEjbBean implements MySesEjb {
  @Resource
  int zeNumber = 0;
```

Responsabilités

- L'EJB doit avoir des responsabilités business
- À usage interne (typiquement)
- EJB (et data model) fournissent le modèle et les services
- Les servlets au sens large fournissent une vue
- *Contrat* clair requis... comme toujours !
- Contrat : devoirs de l'utilisateur ; droits attendus en échange
- À indiquer dans la javadoc
- Utiliser des copies défensives ! Pourquoi ?

Responsabilités

- L'EJB doit avoir des responsabilités business
- À usage interne (typiquement)
- EJB (et data model) fournissent le modèle et les services
- Les servlets au sens large fournissent une vue
- *Contrat* clair requis... comme toujours !
- Contrat : devoirs de l'utilisateur ; droits attendus en échange
- À indiquer dans la javadoc
- Utiliser des copies défensives ! Pourquoi ? Toujours bien ; important si EJB devient Remote

Références

- The Java EE Tutorial: Enterprise Beans
- JSR 345 (EJB 3.2) (direct)
- JSR 342 (Java EE 7)
- JSR 346 (Context and Dependency Injection 1.1 et 1.2) (direct).
- JNDI doc

Exercices : EJB I

- Programmer un SLSB qui implémente une méthode `add(int, int): int`. Il s'occupera de la partie algorithmique de votre premier servlet.
- Votre EJB logge (au niveau `info`) les valeurs reçues et renvoyées
- Si le résultat est nul, votre EJB logge ce fait (en plus) au niveau `warn`
- Programmer un servlet qui accède à l'EJB pour additionner deux paramètres fournis par le client web
- Déployer et tester l'application résultante ; vérifier les logs
- Ajouter une ressource à votre EJB, valant 1 par défaut, qui est ajoutée au résultat du calcul avant renvoi à l'appelant. Tester le servlet.

Exercices : EJB II

- Exporter cette ressource, prendre son chapeau de responsable du déploiement, et modifier la valeur par défaut de la ressource, sans toucher au code source donc. Vérifier l'effet.
- Programmer un SFSB qui incrémente un compteur à l'aide d'une valeur passée en paramètre (reçue via votre servlet)
- Utiliser votre SFSB à la place de votre SLSB dans votre servlet. Durant toute la session HTTP utilisateur, l'utilisateur voit donc le compteur augmenter avec les valeurs qu'il fournit en paramètre.
- Programmer la destruction de votre SFSB à la destruction de la session. Que se passe-t-il si vous ne le faites pas ?
- Programmer l'expiration de la session HTTP après un délai court, via votre servlet (indice : nous avons déjà rencontré l'objet qui permet de le faire)

Exercices : EJB III

- Vérifier (via les logs) que votre SFSB est bien créé et détruit à l'ouverture et à l'expiration des sessions HTTP. Utiliser pour ce faire différents onglets ou fenêtres de votre navigateur et la fonction de suppression des cookies.

Licence