EPSI I4 Web Services / **David Gayerie** - 2013-2014

ENTERPRISE JAVA BEANS (EJB)

ENTERPRISE JAVA BEANS (EJB)

Les EJB sont des composants fournis par les développeurs d'application. Ils encapsulent la logique de traitement de l'application (logique métier).

Les EJB donnent accès à des services et des fonctionnalités permettant le développement d'application scalable, sécurisée et transactionnelle.

Il existe plusieurs types d'EJB :

- Les EJB de session : Stateful EJB, Stateless EJB et Singleton EJB
- Les EJB orientés message : Message Driven Bean
- Les entity bean pour la persistance : Cf. JPA

Les EJB offrent les services suivants :

- modèle d'exécution thread-safe
- gestion du cycle de vie des instances pour une meilleure scalabilité
- accés aux services Java EE par injection
- gestion des transactions
- contrôle des droits d'accès pour l'invocation des méthodes
- gestion des traitements asynchrones
- accès distant pour un client à travers une interface remote

LES EJB SESSION

Il existe trois types d'EJB session :

- les EJB avec état conversationnel (stateful)
- les EJB sans état conversationnel (stateless)
- les EJB singleton (singleton)

EJB stateful : le panier utilisateur

```
import javax.ejb.Stateful;

@Stateful
public class UserBasket {
  private List<Item> items = new ArrayList<>();

  public void addItem(Item i) {
    items.add(i);
  }

  // ...
}
```

L'EJB stateful maintient l'état conversationnel : cela signifie qu'une instance particulière de cet EJB est créée par le conteneur pour chaque client. L'état de l'instance est donc conservé entre deux appels de méthodes.

EJB stateless : la gestion d'une ressource

```
import javax.ejb.Stateless;

@Stateless
public class ArticleRepository {

  public void addArticle(Article a) {
    // ...
  }

  // ...
}
```

L'EJB stateless ne maintient pas d'état conversationnel. Le conteneur crée un pool d'instances et en attribut une à chaque demande du client.

Le conteneur garantit cependant qu'une même instance de cet EJB n'est utilisé que par un client à la fois.

EJB singleton : la gestion d'une ressource unique

```java
import javax.ejb.*;

@Singleton
@Lock(LockType.WRITE)
public class SharedResource {

  @Lock(LockType.READ)
  public SharedResource getSharedResource() {
    // ...
  }
}
```

Le conteneur veille à ce qu'il n'existe qu'**UNE** instance d'un EJB singleton pour une application.

L'annotation **@Lock (http://docs.oracle.com/javaee/6/api/javax/ejb/Lock.html)** permet de contrôler si l'instance ou une méthode autorise des accès concurrents (lock de type READ) ou des accès avec acquisition d'un verrou (lock de type WRITE).

Par défaut, un EJB singleton dispose d'un verrou en écriture pour toutes ses méthodes (lock de type write).

Pour avoir accès à une instance d'un EJB, une application **ne la crée pas**, elle demande au conteneur EJB de la lui fournir.

La méthode la plus simple, consiste à utiliser l'annotation **@EJB (http://docs.oracle.com/javaee/6/api/javax/ejb/EJB.html)** sur un attribut d'un composant Java EE.

Un client EJB

```java
import javax.ejb.EJB;
import javax.ws.rs.Path;

// Une classe avec l'annotation @Path désigne une ressource racine JAX-RS.
// Cette classe est bien un composant Java EE géré par le conteneur Web.
@Path("article")
public class ArticleResource {

  @EJB
  private ArticleRepository;

  // ...
}
```

LA GESTION DES TRANSACTIONS

Les EJB permettent d'ajouter le support transactionnel sur chacune de leur méthode.

L'utilisation la plus courante consiste à gérer les transactions vers les bases de données.

Lors de l'appel d'une méthode d'un EJB, le conteneur démarre une transaction de base de données et à la fin de la méthode, le conteneur effectue un commit ou un rollback.

Une transaction a les propriétés **ACID (http://fr.wikipedia.org/wiki/Propri%C3%A9t%C3%A9s_ACID)** pour garantir sa fiabilité :

- **A**tomicité
- **C**ohérence
- **I**solation
- **D**urabilité

Deux annotations permettent de déclarer le support transactionnel pour les EJB :

@TransactionManagement (http://docs.oracle.com/javaee/6/api/javax/ejb/TransactionManagement.html)

Définit si la transaction est gérée par le conteneur (valeur CONTAINER par défaut) ou si la transaction est gérée par le bean lui-même (valeur BEAN). Une transaction gérée par le bean signifie que le développeur souhaite gérer la transaction par programmation.

@TransactionAttribute (http://docs.oracle.com/javaee/6/api/javax/ejb/TransactionAttribute.html)

Permet de gérer sous quelle condition une transaction gérée par le conteneur peut être démarrée lors de l'appel à la méthode de l'EJB.

Transaction gérée par le conteneur

```
import javax.ejb.*;

@Stateless
// Il s'agit de la valeur par défaut
@TransactionManagement(TransactionManagementType.CONTAINER)
// Il s'agit de la valeur par défaut
@TransactionAttribute(TransactionAttributeType.REQUIRED)
public class ArticleRepository {

  public void addArticle(Article a) {
    // ...
  }

  // ...
}
```

Transaction gérée par le bean

```
import javax.ejb.*;
import javax.annotation.Resource;
import javax.transaction.UserTransaction;

@Stateless
// signale que la transaction est gérée dans le code de l'EJB
@TransactionManagement(TransactionManagementType.BEAN)
public class ArticleRepository {
  @Resource
  private UserTransaction tx;

  public void addArticle(Article a) {
    // démarrer la transaction
    tx.begin();
    // ...
    // commiter la transaction
    tx.commit();
  }

  // ...
}
```

Le bean peut gérer la transaction grâce à l'objet **UserTransaction (http://docs.oracle.com/javaee/6/api/javax/transaction/UserTransaction.html)** injecté par le conteneur grâce à l'annotation **@Resource (http://docs.oracle.com/javaee/6/api/javax/annotation/Resource.html)** .

Dans le cas d'une gestion des transactions par le conteneur, une transaction sera rollbackée si :

- la méthode de l'EJB se termine par une exception runtime
- la méthode de l'EJB se termine par une exception portant l'annotation **@ApplicationException (http://docs.oracle.com/javaee/6/api/javax/ejb/ApplicationException.html)** avec l'attribut **rollback** avec la valeur true

Dans tous les autres cas, la transaction est **commitée**.

L'exception ci-dessous provoque un rollback de la transaction gérée par le conteneur lorsqu'elle est jetée lors de l'exécution d'une méthode d'EJB.

Une exception applicative

```
import javax.ejb.ApplicationException;

@ApplicationException(rollback = true)
public class ArticleNotAvailableException extends Exception {

  // ...

}
```

POUR ALLER PLUS LOIN

Cours de José Paumard sur les EJB
http://blog.paumard.org/cours/jpa/chap07-ejb.html

Java EE 7 Essentials
Arun Gupta - O'Reilly 2013