Formation Java EE

EJB

Rôle d'un EJB

Propose d'automatiser ce qui a trait au système :

- Transactions
- Sécurité
- Evolutivité
- Concurrence
- Communications
- Gestion des ressources
- Persistance
- Gestion des erreurs
- ...

3 types d'EJB

Session

- composant métier réutilisable
- Stateless
  - 1 instance par appel
- Stateful
  - 1 instance par session

Entity

- JPA

Message-Driven

- JMS

Configuration ejb-jar.xml

Facultatif

Dans META-INF

Priorité aux annotations

EJB Session Stateless

Stateless

- les attributs de l'EJB sont réinitialisés entre chaque appel même s'il s'agit du même client
- Sont spécialement pensés pour être robustes et fiables lorsqu'il y a beaucoup d'appels en concurrence

```
@Stateless
public class PersonService {
```

EJB Session Stateful

@Stateful

- L'état de l'EJB est sauvegardé durant toute la session
- Une instance de l'EJB est créée, cette instance reste disponible pour les futurs appels de ce client uniquement. Cette instance sera détruite a la fin de la session (timeout ou appel à une méthode portant l'annotation @Remove)

```
@Stateful
public class PersonService {
```

Injecter un Entity Manager

```
@Stateless
public class PizzaService {

    @PersistenceContext(unitName="pizza-db") private EntityManager em;
```

Tâche planifiée

Testez !

```
@Stateless
public class PizzaService {

    @Schedule(second="10", minute="*", hour="*")
    public void insererPizza() {
        ....
    }
```

Asynchronisme (1)

Testez !

```
@Stateless
public class PizzaService {

    @Asynchronous
    public void insererPizza() {
        ....
    }
```

Asynchronisme (2)

Testez !

```
@Stateless
public class PizzaService {

    @Asynchronous
    public Future<List<Pizza>> findAllPizzas() {
        ...
        return new AsyncResult<List<Pizza>>(pizze);
    }
```

Injecter un EJB

```
@WebServlet("/pizzas")
public class PizzaWeb {

   @EJB private PizzaEJB pizzaEJB;
```

Transactions

ACID

Atomicité

- Plusieurs opérations => toutes les opérations sont effectuées ou aucune.

Cohérence

- Après une transaction, la ressource doit se trouver dans un état cohérent

Isolation

- Pas de dépendance avec d'autres transactions qui peuvent avoir lieu en même temps

Durabilité

- Le résultat d'une transaction est durable (persisté).

Transactions

Les transactions EJB peuvent être gérées par le container ou programmatiquement dans l'EJB Session ou Message-Driven

CMT

- Transaction par défaut
- Le container démarre et valide la transaction

BMT

- L'implémentation de l'EJB doit démarrer et valider une transaction fournie par le Container
- Les transactions dans servlets sont gérées comme des BMT

Exemple de CMT

```
@Stateless
@TransactionManagement(value=TransactionManagementType.CONTAINER)
public class Bean implements LocalBean{

    @TransactionAttribute(TransactionAttributeType.REQUIRED))
    public void myMethod() {
    }
}
```

Type d'attributs de transaction

Attribut	Si une transaction existe	Si pas de transaction
REQUIRED	Utiliser la transaction existante	Démarrer une nouvelle transaction
REQUIRES_NEW	Suspendre la transaction en cours. Démarrer une nouvelle transaction.	Démarrer une nouvelle transaction.
MANDATORY	Utiliser la transaction existante	Lancer une exception
NEVER	Lancer une exception	Traiter sans transaction
NOT_SUPPORTED	Suspendre la transaction en cours.	Traiter sans transaction
SUPPORTS	Utiliser la transaction existante	Traiter sans transaction

Rollback Applicatif

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


}
```

Rollback Applicatif

```
@Resource
private SessionContext sessionContext;

private void save() {

        // Du code
        // Du code
        // Du code

        sessionContext.setRollbackOnly();

}
```

Exemple de BMT

```
@Stateless
@TransactionManagement(TransactionManagementType.BEAN)
public class Beanimplement implements LocalBean {
    @Resource private SessionContext sessionContext;

    public void myMethod(ReminderForm reminder) {
        UserTransaction utx = sessionContext.getUserTransaction();
        try {
            utx.begin();
            //implementation
            utx.commit();
        } catch(Exception e) {
            utx.setRollbackOnly();
        }
    }
}
```

Exemple de persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<persistence xmlns="http://java.sun.com/xml/ns/persistence"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://java.sun.com/xml/ns/persistence persistence_1_0.xsd"
	version="1.0">
	<persistence-unit name="banko" transaction-type="JTA">
		<jta-data-source>jdbc/banko</jta-data-source>
		<properties>
			<property name="javax.persistence.schema-generation.database.action" value="drop-and-create"></
property>
			<property name="hibernate.ejb.naming_strategy"
value="org.hibernate.cfg.ImprovedNamingStrategy"></property>

<property name="hibernate.show_sql" value="true"></property>

		</properties>
	</persistence-unit>
</persistence>
```