TP 3 : JPA

Architecture JEE

Master 2 Génie Logiciel

Préparation de la base de données

- PostgreSQL 9.4 :

  http://www.postgresql.org/download/

  -> création de la base de données MIDB

- Driver JDBC PostgreSQL (type 4) :

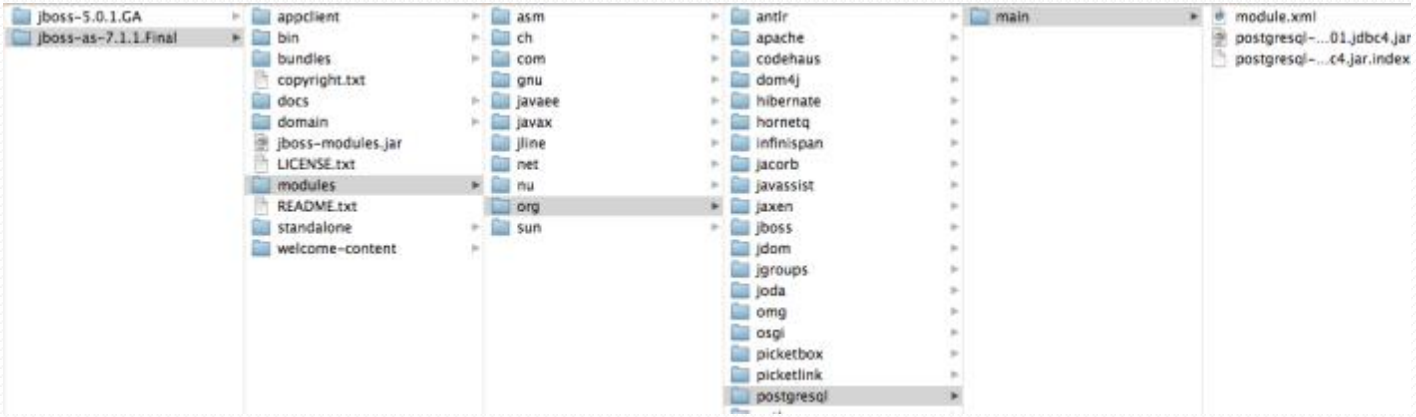
  http://jdbc.postgresql.org/download.html

  -> postgresql-9.2-1000.jdbc4.jar

Installation driver JDBC dans JBoss

Dans Jboss 7, toutes les librairies additionnelles sont à installer dans le répertoire « module » :

- dans un emplacement packagé ./org/postgresql/main,
- avec un descripteur de module « module.xml ».



Descripteur « module.xml »

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

<module xmlns="urn:jboss:module:1.1" name="org.postgresql">
    <resources>
        <resource-root path="postgresql-9.1-901.jdbc4.jar"/>
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

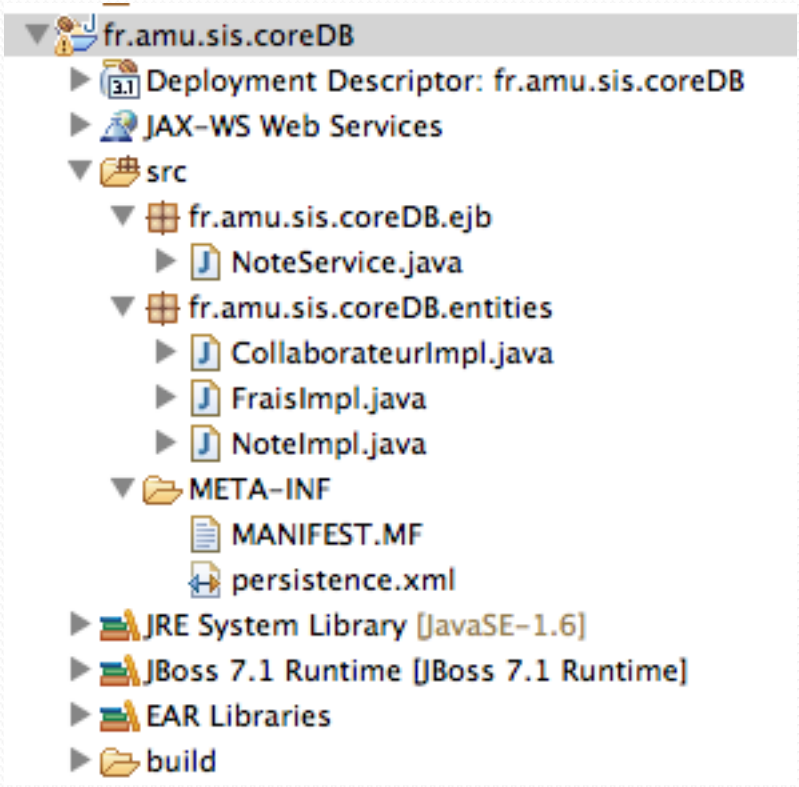
Déclaration de la source de données

Source de données définie dans le fichier de configuration de Jboss « standalone.xml » (ou fichier de configuration personnalisé) :

- Adresse jndi de la source de données,
- URL de connexion JDBC,
- Driver JDBC à utiliser (module).

```
<subsystem xmlns="urn:jboss:domain:datasources:1.0">
    <datasources>
        <datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="ExampleDS" enabled="true" use-java-context="true">
            <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</connection-url>
            <driver>h2</driver>
            <security>
                <user-name>sa</user-name>
                <password>sa</password>
            </security>
        </datasource>
        <datasource jndi-name="java:jboss/datasources/MIDS" pool-name="MIDS" enabled="true" use-java-context="true">
            <connection-url>jdbc:postgresql://localhost:5432/MIDB</connection-url>
            <driver>postgresql</driver>
            <security>
                <user-name>postgres</user-name>
                <password>postgres</password>
            </security>
        </datasource>
        <drivers>
            <driver name="h2" module="com.h2database.h2">
                <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
            </driver>
            <driver name="postgresql" module="org.postgresql"/>
        </drivers>
    </datasources>
</subsystem>
```

Structure du module coreDB



Définition du contexte de persistance

Fichier persistence.xml :

- Déclare l'unité de persistance,
- Référence la source de données (adresse JNDI),
- Recense les entités mappées,
- Configure les paramètres du fournisseur d'ORM (ici Hibernate).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
    version="2.0" xmlns="http://java.sun.com/xml/ns/persistence">
    <persistence-unit name="MIDB" transaction-type="JTA">
        <jta-data-source>java:jboss/datasources/MIDS</jta-data-source>

        <!-- Les entités communes -->
        <class>fr.amu.sis.coreDB.entities.CollaborateurImpl</class>
        <class>fr.amu.sis.coreDB.entities.FraisImpl</class>
        <class>fr.amu.sis.coreDB.entities.NoteImpl</class>

        <properties>
            <property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver" />
            <property name="javax.persistence.jdbc.url" value="jdbc:postgres://localhost/ejb3" />
            <property name="javax.persistence.jdbc.user" value="postgres" />
            <property name="javax.persistence.jdbc.password" value="postgres" />
            <property name="hibernate.hbm2ddl.auto" value=""/>
            <property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQLDialect"/>
            <property name="hibernate.show_sql" value="false"/>
        </properties>

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

Annotations des entités

Annotations de base :

@Entity,

@Table,

@Id,

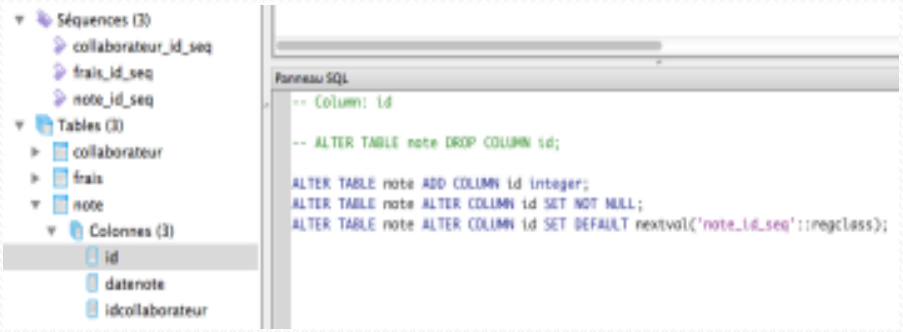
@Column,

@GeneratedValue,

@SequenceGenerator.

Id auto-généré à partir d'une séquence

Dans la BDD



Dans l'entité

```
@Entity
@Table(name="note")
@SequenceGenerator(name="note_id", sequenceName="note_id_seq",allocationSize=1)
@NamedQueries({
    @NamedQuery(name="findNotesByCollaborateur", query="SELECT n FROM NoteImpl
})
public class NoteImpl implements Note{
    /**
     * Numéro de série de la classe
     */
    private static final long serialVersionUID = 9865757549545881361L;

    /**
     * Identifiant de la note de frais
     */
    @Id
    @GeneratedValue(strategy=GenerationType.SEQUENCE, generator="note_id")
    private Long id;
```

On référence au niveau de l'entité la séquence utilisée par le SGBD pour attribuer un numéro de série à la clé primaire grâce aux annotations suivantes :

- @GeneratedValue,
- @SequenceGenerator.

Paramètres d'associations

Paramétrez les associations entre entités (@OneToMany, @OneToOne etc.) avec les propriétés suivantes :

- targetEntity : spécifie la classe de l'entité si l'association est exprimée par « interface »,
- fetch (EAGER ou LAZY) : comportement de chargement de(s) entité(s) associée(s),
- cascade : comportement de propagation des actions de l'entity manager.

Développement EJB NoteService

Reprendre l'EJB NoteService du module coreM et mettre à jour toutes les méthodes en utilisant l'Entity Manager :

- Injection de l'Entity Manager par annotation,
- Utilisation des méthodes CRUD de l'Entity Manager,
- Utilisation des requêtes nommées en JPQL,

```
/**
 * Contexte de persistance
 */
@PersistenceContext(unitName="MIDB")
private EntityManager em;
```

```
note = em.find(NoteImpl.class, id);
```

```
public Collaborateur setCollaborateur(String nom, String prenom){
    try{
        Query query = em.createNamedQuery("findCollaborateurByName");
        query.setParameter("nom", nom);
        query.setParameter("prenom", prenom);
        this.collaborateur = (Collaborateur) query.getSingleResult();
    }
    catch(NoResultException e){
```

Mise à jour du client de test

Si programmation par « interfaces » (ejb, entités), aucune modification du programme client de test de la note de frais n'est nécessaire. Seules les dépendances sont à mettre à jour :

- ✓ gestion-frais-api : pour les interfaces référencées dans le code,
- ✓ gestion-frais-coreDB : pour les entités manipulées à l'exécution,
- ✓ jboss-as-7.1.1.Final/modules/org/hibernate/main/hibernate-core-4.0.1.Final.jar : pour les classes techniques d'Hibernate manipulées à l'exécution.