TP JDBC épicerie

1 Objectif

Le but de ce TP est de vous familiariser avec l'accès à une base de données relationnelle à partir d'un programme JAVA en utilisant la librairie JDBC. La base utilisée sera une base de donnée MYSQL.

2 Préparation du tp

2.1 Description de la base de données épicerie

Soit une base de données **`épicerie`** qui comporte trois tables ayant la signature suivante :

```
fournisseur(F:chaîne, Nom:chaîne, Remise:réel, Ville:chaîne)
mafourniture(F:chaîne, P:chaîne, Qte:réel)
produit(P:chaîne, Nom_P:chaîne, Couleur:chaîne, Origine:chaîne)
```

Les clés primaires de chaque table sont soulignées.

Voici un diagramme décrivant cette base:

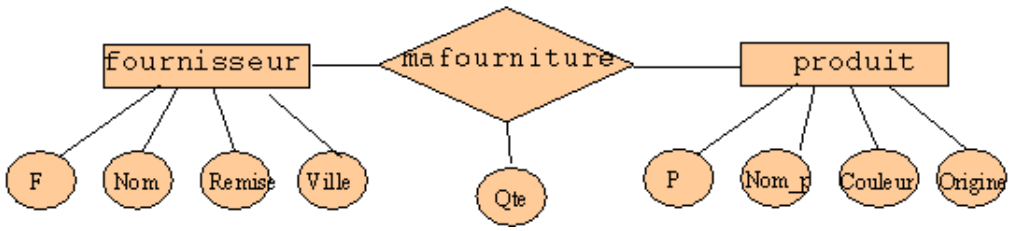

FIGURE 1. Schéma de la base de données Epicerie

2.2 Base de donnée MYSQL sur le serveur `anteros` de l'IFSIC

Accéder au serveur **`anteros`** de l'IFSIC à l'aide d'un navigateur à l'URL : **`http://anteros.ifsic.univ-rennes1.fr`** : Chaque utilisateur de l'IFSIC peut créer une base sur le serveur **`anteros.`** Le nom de la base et le nom de l'utilisatuer sont de manière conventionnelle :

Nom de la base : **`base_<identification de l'utilisateur>`**

Nom de l'utilisateur : **`user_<identification de l'utilisateur>`**

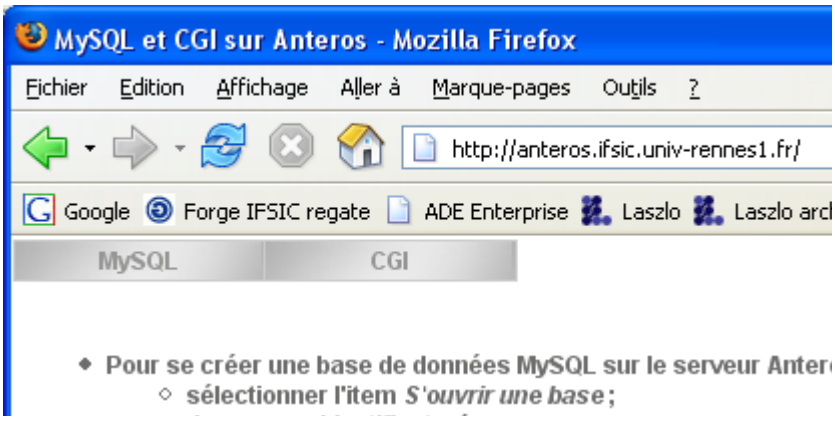

FIGURE 2. Page d'accueil du serveur `anteros`

Laissez vous guider par les instructions de la page d'accueil du serveur **`anteros`** pour créer une base de données MySQL si vous n'en avez pas déjà une. Donnez un mot de passe que vous utiliserez plus loin pour accéder à votre base. A sa création la base n'a pas de table, pour ce TP, c'est vous qui instancierer les tables via JDBC.

Remarque La base que vous venez de créer vous appartient en propre, elle peut être utilisée dans plusieurs projets ou TP tout au long de votre cursus. Le menu MySQL de la page d'accueil d'**`anteros`** vous permet de revenir ultérieurement gérer cette base, c'est à dire visualiser/effacer des tables. Par cette page vous pourrez changer le mot de passe si vous l'avez oublié. Nous vous encourageons à faire périodiquement cette tâche d'administration.

2.3 Ressource pour ce tp

2.3.1 Documents xml

Nous vous fournissons trois document XML. Ces document sont structurés selon le modèle de la base **`épicerie`** décrite ci-dessus. Vous pouvez parcourir les documents XML pour voir ce qu'ils contiennent. Chaque document contient des valeurs qui correspondent à une instance de table.

TABLEAU 1. Documents xml

Relation	Document
Fournisseur	fournisseur.xml
Produit	produit.xml
MaFourniture	fourniture.xml

Les éléments **`<fournisseur>`** et **`<produit>`** possèdent des clés dans leurs attributs respectifs **`<f>`** et **`<p>`**. Elles verifient une contrainte d'unicité. L'élément **`<fourniture>`** contient deux clés étrangères **`<f>`** et **`<p>`**. Ces deux clés vérifient la contrainte d'intégrité de référencement.

Voici l'arbre XML correspondant au document **`fournisseur.xml`** :

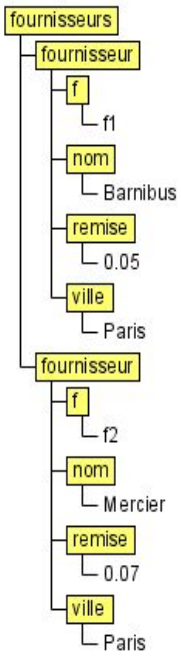

FIGURE 3. Document fournisseur.xml

2.3.2 Classes Java

En ressource de ce TP nous vous fournissons une librairie **`fr.ifsic.epicerie`** qui contient, entre autres, trois classes JAVA **`Fournisseur`**, **`Produit`**, et **`Fourniture`**. Ces classes ont le même modèle que les tables de la base de données décrite ci-dessus.

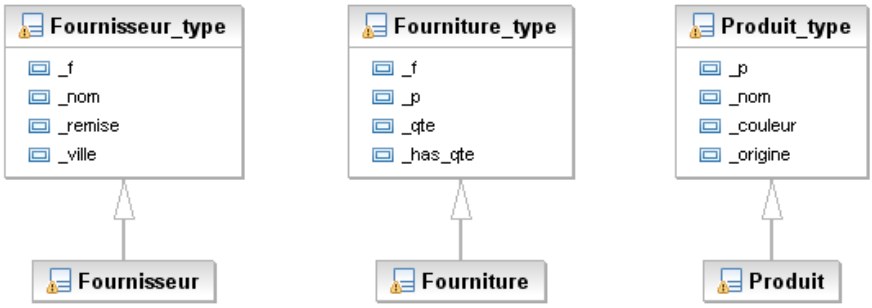

FIGURE 4. Modèle uml des classes fournies en ressource

Ainsi la classe **`Fournisseur`** possède les accesseurs **`getF()`**, **`getNom()`**, **`getRemise()`** et **`getVille()`**. Pour votre information ces classes ont été générées automatiquement par la librairie CASTOR à partir d'un schema de données **`xmlschema`**.

Nous vous fournissons aussi une classe **`fr.ifsic.epicerie.Epicerie`** qui possède trois attributs déclarés comme suit :

```
private Fournisseur[] lesFournisseurs;
private Produit[] lesProduits;
private Fourniture[] maFourniture;
```

FIGURE 5. La classe Epicerie

Ces tableaux ont des accesseurs **`public getLesFournisseur()`** etc. Dans son constructeur, la classe **`Epicerie`** intancie les trois tableaux à partir des valeurs lues dans les documents XML fournis en ressource. Il s'agit donc d'une conversion *instance de document XML* vers *instances de classes JAVA* effectuée par les méthodes **`unmarshall`** fournie par *CASTOR*. Vous n'aurez pas à programmer cette convertion. Elle sera effectuée automatiquement lors de la création d'une instance de la classe **`épicerie`**.

2.3.3 Librairies tiers

Vous trouverez aussi dans les ressources un répertoire **`lib`** qui contient un certain nombre de librairies tiers qu'il conviendra d'ajouter au **`classpath`** de votre projet. Voici la hérarchie fournie en ressource

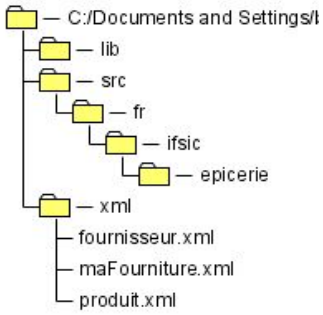

FIGURE 6. Ressources pour ce tp

Question 1. Initialiser un projet java

Chargez ces ressources dans un nouveau projet Java. Vous devez retrouver dans votre projet l'arborescence ci-dessus.

3 Initialisation de la base épicerie en Java

Répondez aux questions suivantes dans la méthode **`main()`** d'une classe **`TestEpicerie`**. Noter que le pilote d'accès aux bases MSQL est la classe **`org.gjt.mm.mysql.Driver`**. Il vous est fourni en ressource pour ce TP dans la librairie **`mysql-connector-java-5.0.4-bin.jar`**

Question 2. Charger le pilote

Ecrire le code de chargement du pilote avec réception des erreurs de chargement de pilote.

Question 3. Connexion à la base

*Créer une connection que vous conserverez dans une variable statique. Fermez cette connection en fin de programme. Réceptionnez les erreurs de connection. Voici la forme de l'*URL *pour une connection à une base mysql de l'*IFSIC

`jdbc:mysql://anteros.ifsic.univ-rennes1.fr:3306/base_identSesam`

3.1 Création des tables

Le tableau **`lesFournisseurs`** de la classe **`Epicerie`** contient les valeurs suivantes

```
f1 - Barnibus - Paris - 0.05
f2 - Mercier - Paris - 0.07
f3 - Colbert - Reims - 0.03
f4 - Bossuet - Dijon - 0.06
f5 - Tanguy - Riec - 0.1
f6 - Dupont - Paris - 0.0
f8 - bridoux - Vannes - 0.045
```

Question 4. Créer les tables fournisseur et produit

On vous demande de créer dans la base de données MYSQL *la table* **`fournisseur`** *à l'aide de l'instruction* SQL *suivante :*

```
create table fournisseur (
   f         int unsigned not null auto_increment primary key,
   nom       varchar(25) not null,
   remise    float(24),
   ville     varchar(20)
)
```

Notez bien la transformation de la clé « **`f1`***» qui est une chaîne dans le tableau* JAVA*, en un entier de valeur* **`1`** *dans la table* SQL*. Vérifiez à l'aide du gestionnaire* MYSQL *que la table a bien été créée.*

Procéder de même pour la table **`produit`**.

Si par hazard les tables que vous voulez créer existent déjà, celles-ci devront d'abord être effacée par une instruction **`drop table`**

Question 5. Créer la table mafourniture

Créez la table **maFourniture** *à l'aide de l'instruction* SQL *suivante :*

```
create table mafourniture (
   f        int unsigned not null,
      constraint Foreign Key(f) references fournisseur(f),
   p        int unsigned not null,
      constraint foreign key(p) references produit(p),
      constraint primary key(f,p),
   qte      int unsigned
)
```

3.2 Innitialiser les tables

Pour initialiser les tables vous devrez d'abord créer une instance d'objet de la classe **Epicerie**. Vous aurez ainsi une copie en mémoire des instances de document XML que vous pourrez parcourir.

Question 6. initialiser les tables

En parcourant les tableaux **mesFournisseurs**, **mesProduits**, **maFourniture** *de l'intance d'objet* **Epicerie** *initialisez les trois tables de la base* MYSQL.

*Vous utiliserez des instructions «***preparedStatement***» de la forme*

```
insert into fournisseur values(?,?,?,?)
```

4 Accéder/modifier les tables

Question 7. Liste des fournisseurs de produits verts

Ecrire une méthode qui affiche la liste des fournisseurs de produits verts.

Question 8. Effacer un tuple

Ecrire une méthode effectuant l'effacement d'un n-uplet dans la table Fournisseur.

Que se passe-t-il si vous réexécutez le même effacement? Essayez et observez les messages d'erreurs. Traitez ce genre d'erreur.

Question 9. Modifiez un fourniseur

Ecrire une méthode qui modifie le nom d'un fournisseur donné.

5 Utiliser les méta données

Question 10. Utiliser les méta données

Ecrire une méthode qui prend en paramètre un resultSet et qui affiche ce dernier par introspection. Prévoir le cas où le resultSet est vide.

Tester votre méthode en écrivant une méthode sql qui prend en paramètre n'importe quelle requête d'interrogation et qui en affiche les résultats. Tester.