

TD6 : JDBC¹

JDBC est l'API Java permettant l'accès à toutes sortes de source de données et plus particulièrement aux bases de données (relationnelles).

Avertissement. Dans la suite de ce TD, nous approchons l'API JDBC d'accès aux bases de données qui est une API « bas niveau », il existe d'autres outils permettant de gérer des données stockées dans un système de gestion de base de données (database manager system).

1 DÉCOUVERTE DE L'API

Pour découvrir les différentes notions, nous vous invitons à réaliser le tutorial de Sun , disponible à l'adresse <http://java.sun.com/docs/books/tutorial/jdbc/index.html> dont voici le contenu commenté:

JDBC Introduction

Présentation des concepts et de la structure générale de l'API.

```

Connection con = DriverManager.getConnection
    ( "jdbc:mysql:wombat", "myLogin", "myPassword" );

Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM Table1");
while (rs.next()) {
    int x = rs.getInt("a");
    String s = rs.getString("b");
    float f = rs.getFloat("c");
}
    
```

JDBC Architecture

Présentation des architectures à deux (two-tier) ou à trois (three-tier) niveaux. Dans ce TD, nous mettrons en œuvre une architecture à trois niveaux simplifiée².

A Relational Database Overview

Cette section présente les notions élémentaires du langage SQL et des SGBD (DBMS, database manager system). Ceci devrait être un rappel³.

JDBC Basics

Getting Started

Vous aurez besoin d'un JDK, d'un DBMS et d'un driver JDBC permettant la connection « Java/DBMS ». À l'école tout est fait mais ...

Le tutorial de SUN vous propose de travailler avec le DBMS de SUN, soit Sun GlassFish, c'est ce que vous ferez. Sun GlassFish accompagne généralement l'installation de Netbeans. Par contre dans l'exercice qui suivra, nous vous

¹ Ce TD est largement inspiré du travail de *adt*, les exemples étant d'ailleurs de lui.

² Simplifiée car dans notre cas nous n'aurons pas d'aspects client-serveur.

³ Ce n'est pas pour autant que vous devez vous dispenser de le lire ...

proposerons de travailler avec un autre SGBD (de votre choix, MS Access, MySQL, ...).

Quel que soit le SGBD utilisé, il faudra disposer du driver correspondant. Pour ce faire inclure la librairie correspondante ou éventuellement installer le driver.

Setting Up a Database

En quelques clics initiés à partir de « Tools/Services » vous devriez pouvoir lancer votre DB et créer la DB coffeecake. À ce stade, elle ne contient rien.

Establishing a Connection

Pour établir la connexion, vérifiez bien que les librairies nécessaires sont incluses dans votre projet et que l'url renseignée est bien celle que vous trouvez dans l'onglet « services » de Netbeans.

Pour tester que la connexion s'établit bien, lancer votre projet⁴ en ayant au préalable ajouté un `conn.close()`.

Setting Up Tables

Pour créer vos tables vous pouvez utiliser l'outil graphique de Netbeans ou bien le faire via une requête SQL.

Pour exécuter une requête SQL, vous devrez demander un objet de type `Statement` à votre connexion et lui demander d'exécuter votre requête SQL, un peu comme suit pour la table coffees:

```
Statement stmt = conn.createStatement();
String query = "CREATE TABLE coffees (" +
    "cof_name VARCHAR(32), " +
    "sup_id INT, " +
    "price FLOAT, " +
    "sales INT, " +
    "total INT " +
    ") " ;
stmt.execute(query);
```

Il vous reste à la remplir de quelques valeurs (la table suppliers sera créée plus tard).

Retrieving Values from Result Sets

Utilisation de l'objet `ResultSet` contenant les résultats d'une requête.

Updating Tables

Pour mettre une table à jour, il est possible d'envoyer directement une requête SQL via la méthode `execute` de la classe `Statement` ou bien d'utiliser le `ResultSet` obtenu suite à une requête `SELECT`.

Prenez le temps de faire quelques tests des deux manières de faire maintenant que vous disposez de votre DB Coffeecake.

Milestone: The Basics of JDBC

Using Prepared Statements

Outre le fait que les `PreparedStatement` sont plus efficaces, elles permettent le passage de paramètres.

Using Joins

⁴ Je vous conseille d'écrire une méthode statique pour chaque action décrite dans le tutorial. Méthodes que vous « commenterez/décommenterez » à loisir.

Vous pouvez maintenant créer la table suppliers (via la requête suivante) et mettre en œuvre les concepts de jointure que vous connaissez.

```
query = "CREATE TABLE suppliers (" +
        "sup_id INT, " +
        "sup_name VARCHAR(40), " +
        "street VARCHAR(40), " +
        "city VARCHAR(20), " +
        "state CHAR(2), " +
        "zip CHAR(5) " +
        ")";
```

Using Transactions

Stored Procedures⁵

SQL Statements for Creating a Stored Procedure

Creating Complete JDBC Applications

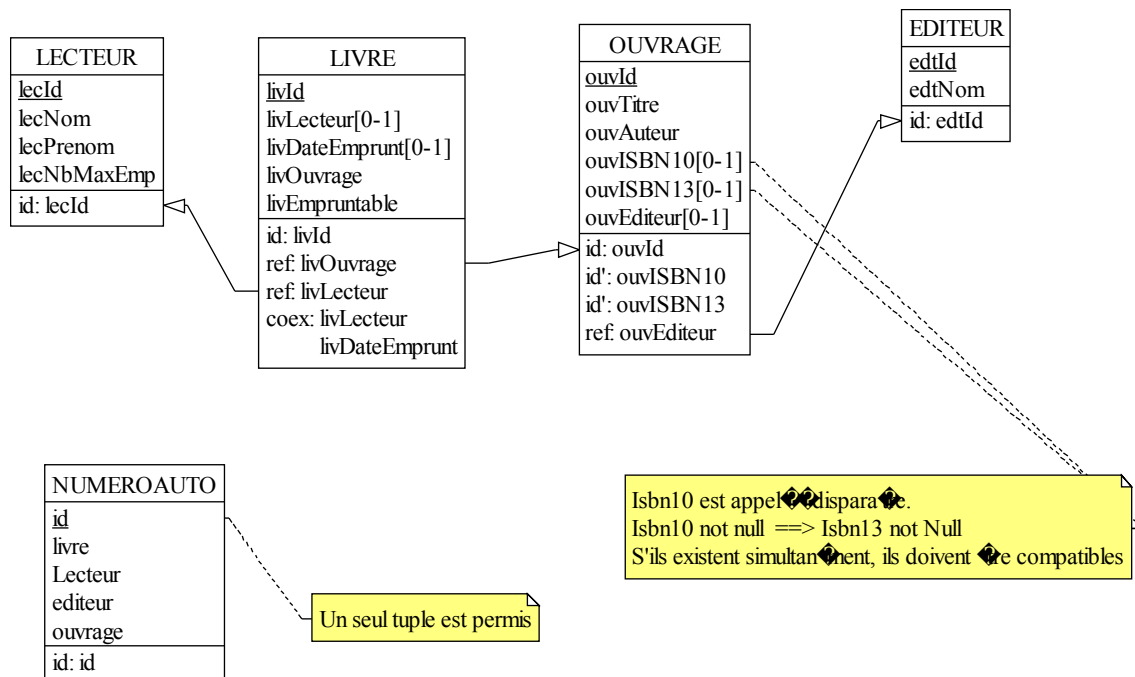
Running the Sample Applications

Creating an Applet from an Application

2 EXEMPLE, MANIPULATION DE LIVRES DANS UNE BIBLIOTHÈQUE

2.1 Présentation de l'environnement

Nous allons travailler avec une DB représentant, de manière simplifiée, les données d'une bibliothèque. La description complète de cette DB se trouve dans un document annexe⁶ (la figure suivante en est un extrait).



⁵ Dans le cadre de ce TD, vous êtes dispensés de la suite du tutorial ... mais l'avant dernier point pourrait cependant être utile.

⁶ Le fichier DescriptionBiblio.pdf

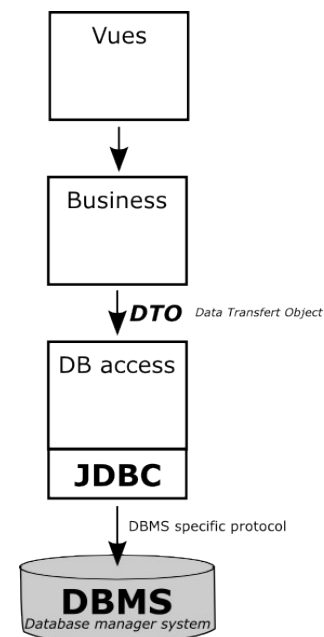
Vous travaillerez avec le SGBD (*DBMS*) de votre choix. Nous vous fournissons le script de création et de remplissage des tables (ddl, *data definition language*) MySQL⁷ (libre à vous de l'adapter pour votre DBMS) ainsi que la BD MS Access.

L'exemple repose sur une architecture à plusieurs niveaux, nous rencontrerons un niveau **db access** (*accès bd*) pour les accès « bas niveaux » à la DB, un niveau **business** (*métier*) pour la manipulation d'objets issus des données (*dto*, *data transfert object*) et un niveau **vues** (*présentation*) comprenant les différentes vues. Dans un premier temps, nous allons nous contenter d'écrire une partie des accès bd et une partie de la logique métier.

Les fonctionnalités « **accès base de données** » seront offertes à la logique métier dans le package `xxx.biblio.bd`.

Les fonctionnalités « **métier** » seront offertes au gestionnaire de présentation (nous ne l'aborderons que plus tard) au travers de méthodes reprises dans des classes du package `xxx.biblio.businessLogic`.

Les données elles-mêmes transiteront entre les différents niveaux de l'application dans des objets (*dto*, *data transfert object*) de classe dont le nom devra se terminer par DTO. Chaque niveau de l'application doit manipuler des données. Afin de limiter les accès aux données et les requêtes, les données sont regroupées dans un objet, cet objet sert donc au transfert des données entre les différents intervenants de l'application.



Notez-bien: Les **dto** représentent l'état des données au moment de leur instanciation. Il doit être clair que les données peuvent évoluer pendant la manipulation d'un dto.

2.1.1 L'accès aux données

(`xxx.biblio.bd`)

Le package `xxx.biblio.bd` permet de masquer à la logique métier les spécificités de la gestion de persistance de données (bd relationnelle, bd objet, fichiers, ...).

Les classes d'accès aux données regroupent généralement des fonctionnalités d'accès aux données d'une même classes ou facette (cf analyse).

Les méthodes de ce package ne pourront lancer que des exceptions du type `Bibliothèque-DbException`.

Remarque: Il existe des API récentes⁸ permettant d'accéder plus facilement aux données. Dans ce TD, nous nous contenteront d'utiliser JDBC.

2.1.2 La logique métier

(`xxx.biblio.businessLogic`)

Le nom de ces méthodes se termine par *facade*. Elles mettent en œuvre le *design pattern Facade* qui, en bref, offre une interface simplifiée d'un sous-système complexe.

Ces classes offriront la plupart des fonctionnalités au travers de méthodes de classe. Par contre, dans les cas où une session nécessite de mémoriser des données qui lui sont propres (l'exemple le plus classique est celui de l'élaboration d'un « panier » d'achat sur un site de vente. Le panier pourra être rempli au fur et à mesure de la promenade dans le site) il sera nécessaire de prévoir

⁷ Le fichier `biblioMySQL.ddl`

⁸ Je pense par exemple à Hibernate.

des classes pouvant être instanciées pour mémoriser des informations relatives à la session de l'utilisateur.

Différentes classes « facades » peuvent utiliser du code commun. Pour cela, on utilisera des classes utilitaires qui ne seront pas exposées au gestionnaire de présentation⁹. Les différentes classes façades regroupent généralement les fonctionnalités liées à un type d'utilisateur (rôle). Ceci n'apparaîtra pas dans notre application puisque nous n'abordons pas la problématique de l'identification et l'authentification des utilisateurs.

Les méthodes de ce package ne lanceront que des erreurs de type `BibliothèqueException`.

Chacune de ces méthodes, si elle accède –directement ou indirectement– au package `xxx.biblio.bd` et accède aux données persistantes, devra s'exécuter dans une **transaction**.

2.2 Terminer l'exemple

2.2.1 Terminer l'implémentation commencée

La méthode `main` dans le *package* `bibliobis` jette des exceptions envoyant le message: « Fonctionnalité non encore implémentée ». Commencez par repérer ces méthodes et complétez le projet afin que cette méthode `main` fonctionne parfaitement.

Vous devrez au préalable installer correctement la DB *bibliothèque* et adapter les paramètres de connection à votre DB, ceci se fait dans la classe `adt.biblio.db.DBManager`.

2.2.2 Un peu de couche présentation

Vous trouverez dans le package `adt.biblio.gui.emprunts` une *frame* `SaisieEmprunt` offrant une interface de saisie d'emprunts. Complétez-la pour que l'utilisateur puisse demander (via la pression d'un bouton) de voir la liste des emprunts en cours du lecteur sélectionné.

De plus, dans le même package, réalisez une *frame* `SaisieRetourLivre` permettant de saisir aisément le retour des livres. Pour ce faire, utilisez le(s) composant(s) déjà présent(s) dans `SaisieEmprunt`. Veillez à ce que l'utilisateur puisse vérifier qu'il s'agit du bon livre et du bon emprunteur avant de valider le retour.

3 PROJET JDBC

Inspirez-vous de cet exemple pour réaliser au mieux votre projet.

Votre projet consiste en la réalisation d'un exercice utilisant une base de données. Cet exercice sera à présenter lors de l'examen¹⁰.

Vous pouvez choisir votre projet. Le plus facile pour vous étant probablement d'utiliser la base de données que vous manipulez au cours d'analyse.

⁹ Dans ce cas, ces classes du package « *businessLogic* » auront une visibilité « *package* ».

¹⁰ Ce sera donc le deuxième travail à présenter lors de l'examen.