

TD4 : 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 fournisseurs 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 EXERCICE, JEU DE L'ANAGRAMME

L'exercice consiste à modifier⁶ et compléter⁷ du code existant pour jouer au "Jeu de l'anagramme" : l'application de départ, qui respecte l'architecture en 3 couches, doit être transformée pour proposer la résolution d'anagrammes de mots français ou anglais. Pour les descriptions nécessaires à votre travail, reportez-vous aux sections "Travail à réaliser" et "Éléments fournis".

2.1 Présentation de l'environnement

Vous travaillerez avec le SGBD (*DBMS*) de votre choix. Le script de création et de remplissage des tables (ddl - dml) est donné mais libre à vous de l'adapter pour votre DBMS.

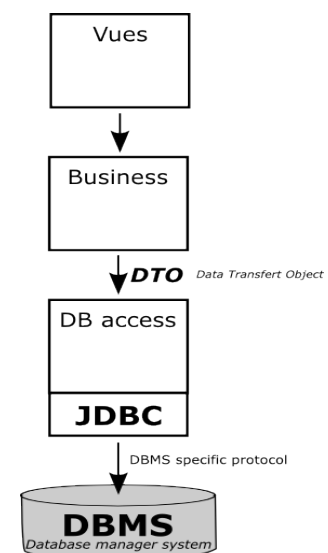
L'application repose sur une architecture à plusieurs niveaux:

- niveau **db access** (*accès bd*) pour les accès « bas niveaux » à la DB
- niveau **business** (*métier*) pour la manipulation d'objets issus des données (*dto*, *data transfert object*)
- niveau **vues** (*présentation*) comprenant les différentes vues.

2.1.1 Les données

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.

Les données transitent entre les différents niveaux de l'application dans des objets *DTO* de classe (dont le nom se termine par DTO).



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 Les éléments fournis pour l'examen ALG2G de juin 2009

⁷ Quelques éléments sont fournis en plus

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.

2.1.2 L'accès aux données

Le rôle de cette couche est 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).

2.1.3 La logique métier

Le nom des méthodes de la logique métier 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 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 présentation⁸. 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.

2.2 Travail à réaliser

Le jeu de l'anagramme est tout simple : il faut retrouver un mot à partir de ses lettres mélangées. Dans l'application demandée, le joueur n'a le droit de faire qu'une proposition de mot mais il peut solliciter de l'aide (une seule fois par partie) qui lui permettra d'obtenir une nouvelle anagramme du même mot où 2 lettres seront placées à leur position. Il peut aussi restaurer l'anagramme du début de la partie ou quitter une partie. Chaque partie entrainera la mise à jour de statistiques pour le mot joué (nombre de fois joué, nombre de fois trouvé,...).

À partir des éléments fournis, il vous est demandé de :

- Modifier l'application de départ : celle-ci doit dorénavant proposer la résolution d'anagrammes de mots anglais ou français issus de la nouvelle base de données. Si une fonctionnalité génère une 'PersistenceException', affichez un message d'excuse reprenant le libellé de l'erreur et terminez l'exécution du programme.
- Compléter l'interface utilisateur du joueur qui est fournie.
- Créer une interface pour l'administrateur de la base de données : il doit pouvoir ajouter un mot (français et/ou anglais) et son anagramme. Attention, il vous est demandé de gérer les règles 'métier' suivantes : un mot ne peut être composé que des 26 caractères minuscules de l'alphabet (a..z), un mot ne peut pas excéder 15 caractères et il ne peut pas y avoir deux mots dans la même langue composés exactement des mêmes lettres.

⁸ Dans ce cas, ces classes du package « business » auront une visibilité « *package* ».

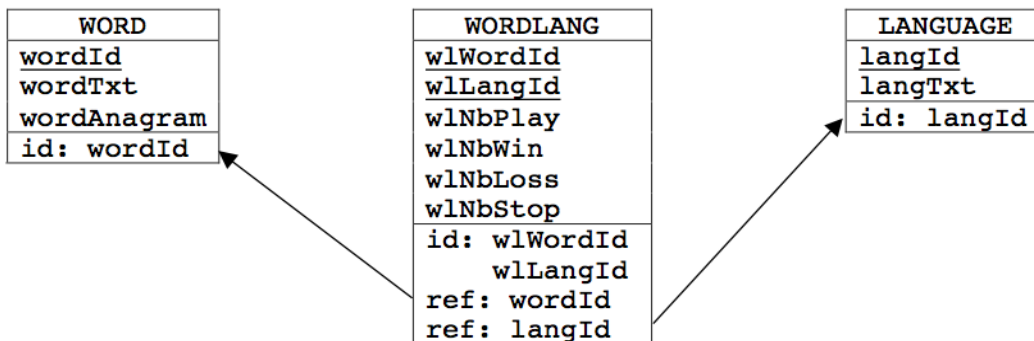
2.3 Éléments fournis

Dans Exa-juin-09-ALG2, vous trouverez les éléments fournis lors de l'examen :

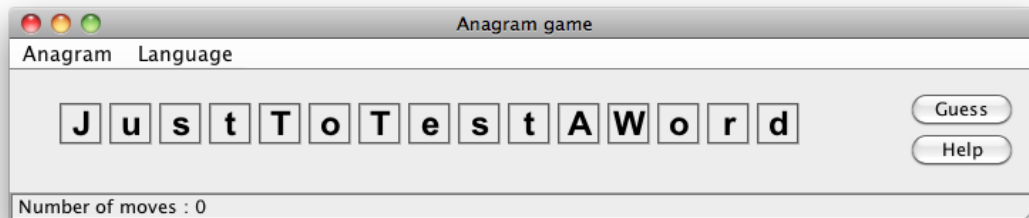
- AnagrammesDdlDml.txt : le script de création, sous javaDb, de la base nécessaire à l'examen
- SaisieAnagramme.jar : le jar du composant 'SaisieAnagramme' permettant d'afficher l'anagramme et d'en modifier l'ordre des lettres par drag and drop (la javadoc est disponible)
- AnagrammeAlg : l'application de départ qu'il fallait faire évoluer

Sont fournis en plus :

- AnagramsSchema.pdf : le schéma de la nouvelle base de données



- AnagramsDdlDml.txt : le script de création, sous javaDb, de la nouvelle base de donnée
- PlayerUI.java : l'interface utilisateur⁹ du joueur, à compléter



- TableLayout-bin-jdk1.5-2009-08-26.jar : le jar du layout TableLayout qui est utilisé dans PlayerUI.java

⁹ La gestion de la langue de l'interface a été simplifiée pour que tout soit dans le même fichier. Idéalement, plus de deux langues devraient être possibles, les textes devraient être mis dans des fichiers ressources et la classe *ResourceBundle* devrait être utilisée.