ISI - Guide projet

Gaël Guibon, Omar Boucelma

20 mars 2017

https://github.com/gguibon/teaching-isi-projet

Sommaire

Chapitre 1

Projet ISI

1.0.1 Objectif

Chaque groupe/élève devra choisir un scenario (Section 2), ou alors proposer le sien.
Les scenarii servent de base et peuvent être modifiés ou même totalement inventés.
Ce document regroupe les principales informations nécessaires au projet.

Chapitre 2

Scenarii

2.0.1 Sport

L'investisseur veut faire le rapprochement entre les installations sportives mises à disposition et la valeur sportive de chaque ville. Pour cela vous devrez croiser les classements sportifs du sport de votre choix : football, tennis, handball, etc. avec les installations des villes concernées.
Classements : http://www.lequipe.fr/Football/ligue-2-classement.html (ou autres ligues françaises et autre sport)
Données d'équipements : http://www.data.gouv.fr/fr/datasets/recensement-des-equipements-sportifs-espace

2.0.2 Jeux vidéo

L'utilisateur veut automatiquement être guidé pour son parcours professionnel dans le jeu video. Il veut se voir proposé une école pour chaque stade de son parcours (L,M,D), en fonction de son budget, et ensuite une liste d'offres d'emplois et leurs compétences associées. Il souhaite également savoir quel poste est le plus demandé.
Données des écoles : http://www.letudiant.fr/etudes/ecoles-specialisees/le-banc-d-essai-des-ecoles-de-je
html
Offres d'emploi à parser : http://emploi.afjv.com/index.php

2.0.3 Tourisme de cinéphile

L'utilisateur souhaite visiter les lieux de tournages de films de Paris en toute tranquilité. Il n'a pas beaucoup de temps et souhaite donc choisir le lieu avec le plus de concentration de cinéma et lieux de tournage. Sans le sou, il souhaite également pouvoir y circuler en velib et utiliser les hotspots pour streamer son vlog.
Tournages en exterieur à Paris : https://www.data.gouv.fr/fr/datasets/lieux-de-tournage-de-films-long-metra
Wifi gratuits : https://www.data.gouv.fr/fr/datasets/liste-des-sites-des-hotspots-paris-wifi-prs/
Velib : https://www.data.gouv.fr/fr/datasets/velib-paris-et-communes-limitrophes-idf/
salles et fréquentation : https://www.data.gouv.fr/fr/datasets/geographie-du-cinema-equipement-et-frequenta

2.0.4 Relations Internationales en situation de crise

On souhaite pouvoir repérer les ambassades par continent ainsi leurs ambassadeurs rapidement selon l'année en cours, et le lieu où l'on se trouve.
Geoloc des ambassades : https://www.data.gouv.fr/fr/datasets/geolocalisation-des-ambassades-de-france/
Ambassadeurs : https://www.data.gouv.fr/fr/datasets/les-ambassadeurs-de-france-depuis-1945/

Chapitre 3

Wrapper csv-sql

Le but est de prendre un CSV et automatiquement créer une base de données à partir de ce CSV.

3.0.1 Wrapper rapide

1. Lire un fichier ligne par ligne

```
/**
 * Lire le contenu d'un file et le retourne en String
 *
 * @param path
 * @param encoding
 * @return
 * @throws IOException
 */
public static String readFile(String path)
                throws IOException {
        byte[] encoded = Files.readAllBytes(Paths.get(path));
        return new String(encoded, StandardCharsets.UTF_8);
}
```

2. Découper chaque ligne par la virgule

```
myString.split(",");
```

3. Prendre les informations et les mettre dans une requête SQL

```
StringBuilder sb = new StringBuilder();
sb.append("CREATE TABLE blabla .....");
sb.append("mes informations issues du CSV");
sb.append("INSERT blabla ....");
sb.append("mes valeurs issues du CSV");
// retourner la chaÃ®ne de caractÃ¨re finale.
return sb.toString();
// Ensuite l'envoyer en requete Ã  la base de données (voir section suivante)
```

3.0.2 Wrapper modulaire

Pour aller plus loin

Bien entendu le mieux serait d'avoir un wrapper adaptatif modulaire. Pour cela :
- Créer une classe Wrapper
- Rendre son instanciation adaptative aux formats et nombre de colonnes

Chapitre 4

Base de données

Les bases de données se font de préférence avec HSQLDB (mais libre à vous d'utiliser celui de votre choix ou les accès disponibles en salle de TP).

— Télécharger HSQL (c'est juste un jar à mettre en dépendance à votre projet) : http://hsqldb.org/
— En utilisant le wrapper csv2sql, créer vos différentes bases de données à l'aide du code présent ici : https://github.com/gguibon/teaching-isi-projet, contenant le code plus bas.

4.0.1 Quelques sources d'aide

— https://www.tutorialspoint.com/jdbc/index.htm
— http://baptiste-wicht.developpez.com/tutoriels/java/db/hsql/#LI-A

4.0.2 Examples HSQLDB

```
package main;

import java.sql.*;

public class HsqlMain {

        // JDBC driver name and database URL
        static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
        static final String DB_URL = "jdbc:hsqldb:file:testdb;shutdown:true";

        // Database credentials
        static final String USER = "username";
        static final String PASS = "password";
        private static Statement stmt;
        private static Connection conn;

        public static void main(String[] args) throws Exception {

                try{
                        // Création de la base de données (ne marchera que si elle n'
                            existe pas)
                        create_database();
                }catch(Exception e){
                        // Connexion Ã  la base de données si elle existe déjÃ
                        connect_database();
                }


                // Création d'une table dans la base de données (ne marchera pas si elle
                    existe déjÃ )
                try{
                        create_table();
                }catch(Exception e){
                        System.out.println("Table already exists.");
                }
```

```
            try{
                    // Insertion de données dans la base
                    insert_values();
            }catch(Exception e){
                    System.out.println("Data already exists");
            }

            // Affichage de certaines données de la base
            display_values();

            // Modification de certaines valeurs
            update_values();

            // Affichage de certaines données pour voir les différences aprÃ¨s
            // l'update
            display_values();

            // Suppression de certaines valeurs et affichage des changements
            delete_values();
            display_values();

            // Fermeture de la base de données et sauvegarde de ces données
            close_and_save_database();

    }

    private static void create_database() throws Exception {
            System.out.println("Creating databse...");
            Class.forName("org.hsqldb.jdbcDriver").newInstance();
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            System.out.println("Database created.");
    }

    private static void connect_database() throws Exception {
            System.out.println("Establishing connexion to database...");
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            System.out.println("Connexion established.");
    }

    private static void create_table() throws Exception {
            System.out.println("Creating table in given database...");
            stmt = conn.createStatement();

            String sql = "CREATE TABLE REGISTRATION " + "(id INTEGER not NULL, "
                            + " first VARCHAR(255), " + " last VARCHAR(255), "
                            + " age INTEGER, " + " PRIMARY KEY ( id ))";

            stmt.executeUpdate(sql);
            System.out.println("Table created in given database.");
    }

    private static void insert_values() throws Exception {
            System.out.println("Inserting records into the table...");
            stmt = conn.createStatement();

            String sql = "INSERT INTO Registration "
                            + "VALUES (100, 'Zara', 'Ali', 18)";
            stmt.executeUpdate(sql);
            sql = "INSERT INTO Registration "
                            + "VALUES (101, 'Mahnaz', 'Fatma', 25)";
            stmt.executeUpdate(sql);
            sql = "INSERT INTO Registration " + "VALUES (102, 'Zaid', 'Khan', 30)";
            stmt.executeUpdate(sql);
            sql = "INSERT INTO Registration "
                            + "VALUES(103, 'Sumit', 'Mittal', 28)";
            stmt.executeUpdate(sql);
            System.out.println("Inserted records into the table...");
    }
```

```
        private static void display_values() throws Exception {
                System.out.println("Displaying values...");
                stmt = conn.createStatement();

                String sql = "SELECT id, first, last, age FROM Registration";
                ResultSet rs = stmt.executeQuery(sql);

                while (rs.next()) {
                        // Retrieve by column name
                        int id = rs.getInt("id");
                        int age = rs.getInt("age");
                        String first = rs.getString("first");
                        String last = rs.getString("last");

                        // Display values
                        System.out.print("ID: " + id);
                        System.out.print(", Age: " + age);
                        System.out.print(", First: " + first);
                        System.out.println(", Last: " + last);
                }
                rs.close();
        }

        private static void update_values() throws Exception {
                System.out.println("Updating values...");
                stmt = conn.createStatement();
                String sql = "UPDATE Registration "
                                + "SET age = 30 WHERE id in (100, 101)";
                stmt.executeUpdate(sql);
                System.out.println("Values updated.");
        }

        private static void delete_values() throws Exception {
                System.out.println("Deleting some data...");
                stmt = conn.createStatement();
                String sql = "DELETE FROM Registration " + "WHERE id = 101";
                stmt.executeUpdate(sql);
                System.out.println("Data deleted.");
        }

        private static void close_and_save_database() throws Exception {
                // Close connexion to the database and save changes
                System.out.println("Closing database ...");
                Statement statement = conn.createStatement();
                statement.executeQuery("SHUTDOWN");
                statement.close();
                conn.close();
                System.out.println("Database closed and changes saved!");
        }

}
```

Chapitre 5

Création de vues intéressantes

La création de vues est ce qui donnera une plus-value à votre projet.

En fonction des besoins de l'utilisateur, une vue différente devra être créée. Cette vue permet d'aggréger des informations spécifiques de plusieurs bases de données en une, et ainsi d'avoir un accès dédié et personnalisé aux données.
Voici le format d'une création de vue :

```
CREATE VIEW <viewname>[(<viewcolumn>,..) AS SELECT ... FROM ... [WHERE Expression]
[ORDER BY orderExpression [, ...]]
[LIMIT <limit> [OFFSET <offset>]];
```

http://www.hsqldb.org/doc/guide/ch09.html#create_view-section