TD4 : Carnet d'adresses

On cherche à créer une application minimaliste de gestion des contacts, tout en permettant son extension future.

La gestion des contacts peut être extrêmement complexe, selon que l'on s'interconnecte ou non avec d'autres services, selon que l'on cherche à implémenter les standards en vigueur ou non, etc. Nous appliquerons d'abord le concept *KISS* (*Keep It Simple Stupid*) en considérant qu'un contact est simplement défini par un *nom*, un *prénom* et une *adresse électronique*. Vous pourrez étendre à votre convenance l'application une fois le prototype opérationnel.

1 Interface

Proposer une application *simple* avec une interface utilisateur (graphique ou non) permettant de créer et supprimer des contacts, d'afficher et modifier les contacts enregistrés. Attention, il faudra respecter les principes de développements vus dans les cours précédents pour rendre possible l'intégration future d'autres interfaces dans l'application.

2 Persistance des données

2.1 Sauvegarde et chargement dans un fichier à plat

Comme toute application de gestion des contacts, on souhaite évidemment pouvoir conserver ces contacts sans avoir à les saisir à chaque redémarrage de l'application.

On considère intialement un format de stockage très simple : un fichier CSV (*Comma Separated Value*) qui contient tous les contacts et que l'on peut charger au démarrage de l'application. Le format du fichier CSV est le suivant :

```
nom1,prenom1,courriel1
nom2,prenom2,courriel2
...
```

Étendre votre application pour pouvoir charger et sauvegarder un tel fichier. Pour vous aider, un exemple simple de code est mis à disposition sur l'ENT. Un fichier de test `addressbook.csv` est aussi disponible. Voir section 4.

2.2 Changement de support de stockage : base de données

Avec l'augmentation du nombre de contacts, on souhaite stocker nos données dans une base de données plutôt que dans un simple fichier à plat. Adapter l'application afin de permettre le stockage des contacts dans une base de données. Par souci de simplicité, on pourra utiliser la base de données intégrée à Netbeans.

Pour vous aider, un tutoriel ainsi qu'un exemple de code sont mis à disposition sur l'ENT. Un script SQL pour créer et peupler la base de données est aussi disponible (`addressbook.sql`). Voir section 4.

3 Pour aller plus loin : diversification des supports

Si vous avez réussi à écrire un logiciel de gestion des contacts, vous pouvez maintenant ajouter la gestion multiple des sources :

Un format de contact extrêmement répandu est le format vCard, qui est complexe à mettre en œuvre si l'on souhaite l'implémenter complètement. Une pratique courante des utilisateurs manipulant des contacts vCard est d'importer ou exporter les informations d'un seul contact stocké dans une vCard. Par souci de simplicité, on n'implémentera pas ce format et on considérera uniquement un format CSV. Intégrer la possibilité de charger et sauvegarder un seul contact à la fois depuis un fichier. On prendra en compte le fait que l'utilisateur peut manipuler des contacts provenant de diverses sources (la base de données, des fichiers vCard qu'il reçoit, le fichier CSV global, un fichier XML global, un fichiers vCard global, etc.) pour peupler son carnet d'adresses.

4 Aide

Pour ce TD, on pourra se servir des différents fichiers d'aide mis à disposition. Notamment pour charger/sauvegarder un fichier, ainsi que pour utiliser une base de données.

4.1 Charger et lire un fichier CSV

Pour lire un fichier CSV, on pourra s'inspirer de l'exemple `CSVReader.java` (avec `addressbook.csv` en entrée) et l'adapter.

4.2 Écrire un fichier

Pour écrire un fichie, il suffit d'écrire dans un fichier classique en concaténant les champs avec le séparateur voulu. On pourra s'inspirer de l'exemple `CSVWriter.java` et l'adaptes (il n'est pas du tout générique et ne pourra donc être réutiliser tel quel dans votre projet).

4.3 Base de données

4.3.1 Création de la base de données

Dans Netbeans, aller dans l'onglet « services », sélectionner « Databases » puis faire un clic droit sur « Java DB ». S'assurer dans « Properties » que les champs sont bien renseignés (les compléter si ce n'est pas le cas), puis refaire un clic droit sur « Java DB » et cliquer sur « Create Database ». Donner un nom à votre base de données (par exemple « addressbook »), puis saisir un nom d'utilisateur et un mot de passe. Ils serviront à se connecter à la base de données. Valider. La base de données apparait. Faire un clic droit dessus pour se connecter. Une fois la connexion établie, refaire un clic droit sur le nom de la base de données et choisir d'exécuter une commande. Une fenêtre devrait apparaitre, vous pouvez y écrire des requêtes SQL. Tester en copiant le script SQL fourni.

En cas de souci, on pourra se référer à `https://netbeans.org/kb/docs/ide/java-db.html`.

4.3.2 Utilisation de la base de données avec Java

Un fichier d'exemple `DBConnectionExample.java` est à disposition sur l'ENT. Il vous montr comment vous connecter, et comment effectuer une requête de chaque type (SELECT, INSERT, UPDATE, DELETE) sur la base de données. Le fichier README donne les détails de la base et du programme.