Application pour la gestion de personnel

Séance #7 - 30/11/2017

Université Paris 1 Panthéon Sorbonne

Carmen Brando

L'objectif de cet exercice est de développer une application pour la gestion de personnel à partir du modèle MVC et en accédant à des informations stockées dans une base de données. L'interface graphique est illustrée en figure 1s :

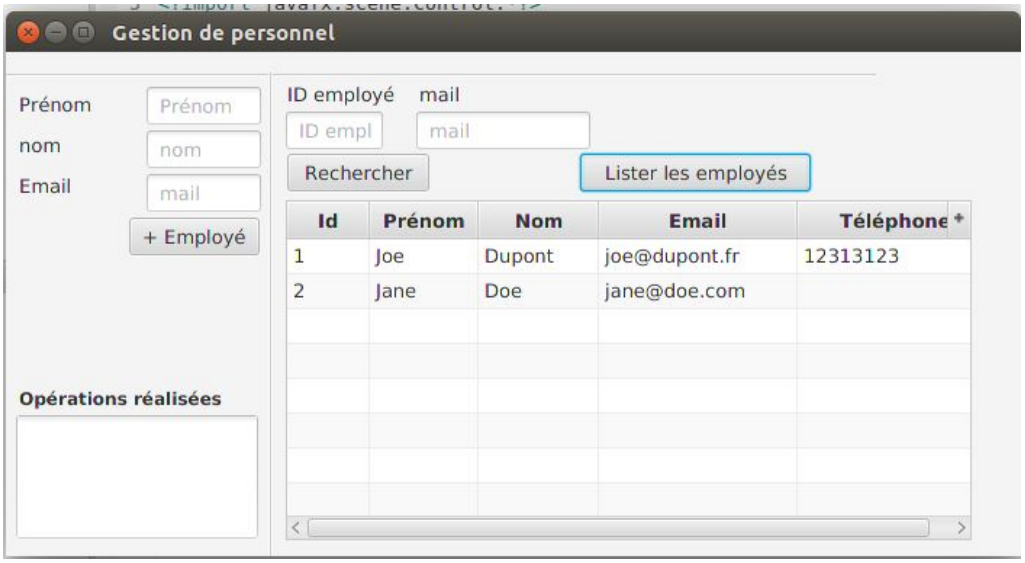

Figure 1 : interface graphique de l'application gestion de personnel

LA VUE :

Créez l'interface en FXML (ou si vous préférez en Java) présentée en figure 1 à partir des contrôles graphiques (TextField, Label, Button et TextArea) et une liste de données comme TableView. Ajoutez des références vers trois actions (attribut 'onAction' en FXML) pour lister toutes les employés, ajouter un nouvel employé et rechercher par identifiant et adresse électronique.

MODELE :

Il est composé de deux classes que vous pouvez recuperer : Employee.java et EmployeeDAO.java. Cette dernière illustre l'utilisation du pattern de conception DAO, acronyme pour *Data Access Object*. Celui-ci permet de faire le lien entre la couche métier et la couche de base de données. Vous allez retrouver la classe Singleton DBUtil.java, elle gère la connexion avec la base de données.

Pour simplicité, la base sera implémentée sous le système SQLITE. Vous pouvez télécharger et installer ce logiciel de manière simple et rapide via ce lien : https://sqlite.org/quickstart.html. Créez une table “employee” avec la commande SQL suivante :

```
create table employees (
EMPLOYEE_ID INTEGER PRIMARY KEY AUTOINCREMENT,
FIRST_NAME,
LAST_NAME,
EMAIL,
PHONE_NUMBER,
JOB_ID,
SALARY,
COMMISSION_PCT,
MANAGER_ID,
DEPARTMENT_ID);
```

Ensuite, ajoutez une nouvelle entrée via une commande SQL de type INSERT.

LE CONTROLEUR :

Complétez le code dans la classe EmployeeController.java. En particulier, complétez la méthode pour ajouter une nouvelle entrée dans la base de données, vous devez faire appel à la méthode correspondante définie dans le modèle. Ici, vous allez retrouver un exemple d’utilisation de valeurs observables (*ObservableValue*) en JavaFX : elles représentent une entité qui enveloppe une valeur et permet d'observer la valeur pour les changements.

Enfin, créez une **classe main** qui construit l’interface comme vous avez fait lors des cours précédents.

+) ajouter un bouton ‘Actualiser’ pour mettre à jour le champ mail d’un employé (celui ci est retrouvable par son identifiant) à partir de la valeur saisie par l’utilisateur sur le champ de texte correspondant.