Application de Gestion des Congés

Fonctionnalités d'Import/Export

Omar Akalay

Introduction

Ce rapport décrit les étapes du développement d'une fonctionnalité d'importation et d'exportation des données dans une application de gestion des congés. L'objectif principal est d'étendre l'application en permettant l'importation des données depuis des fichiers CSV et l'exportation vers ces mêmes formats. Ce processus se fait en plusieurs étapes, en incluant la gestion des données, la logique métier, la mise à jour de l'interface graphique, et l'implémentation d'un contrôleur.

1 Étape 1 : Gestion des Données (DAO)

Objectif :

Dans cette étape, nous avons créé une interface générique pour l'importation et l'exportation des données. L'interface `DataImportExport` définit les méthodes principales utilisées pour ces opérations.

Interface DataImportExport

L'interface suivante définit deux méthodes essentielles pour l'importation et l'exportation des données :

```
public interface DataImportExport<T> {
    void importData(String fileName) throws IOException;
    void exportData(String fileName, List<T> data) throws
       IOException;
}
```

L'implémentation de cette interface utilise les classes `BufferedReader` et `BufferedWriter` pour lire et écrire les données.

2 Étape 2 : Logique Métier

L'étape suivante consiste à effectuer plusieurs vérifications essentielles avant de lancer les opérations d'importation et d'exportation.

Vérifications Essentielles

- **Vérification de l'existence du fichier :** La méthode `checkFileExists()` s'assure que le fichier existe avant toute opération.

```
private boolean checkFileExists(File file) {
    if (!file.exists()) {
        throw new IllegalArgumentException("Le fichier n'
           existe pas : " + file.getPath());
    }
    return true;
}
```

- **Vérification du type de fichier :** La méthode `checkIsFile()` vérifie que le chemin spécifié mène à un fichier valide.

```
private boolean checkIsFile(File file) {
    if (!file.isFile()) {
        throw new IllegalArgumentException("Le chemin
            sp cifi  n'est pas un fichier : " + file.
           getPath());
    }
    return true;
}
```

- **Vérification des droits de lecture :** La méthode `checkIsReadable()` s'assure que le fichier peut être lu.

```
private boolean checkIsReadable(File file) {
    if (!file.canRead()) {
        throw new IllegalArgumentException("Le fichier n'
           est pas lisible : " + file.getPath());
    }
    return true;
}
```

3 Étape 3 : Interface Graphique

La modification de l'interface graphique (vue) inclut l'ajout de boutons pour permettre l'importation et l'exportation des données.

Ajout des Boutons

- **Bouton d'importation :** Un bouton permettant de charger les données à partir d'un fichier.
- **Bouton d'exportation :** Un bouton permettant de sauvegarder les données dans un fichier.

L'organisation des composants utilise `FlowLayout` pour un alignement naturel des boutons.

4 Étape 4 : Contrôleur

La dernière étape implique l'implémentation du contrôleur pour gérer les événements des boutons d'importation et d'exportation.

Gestion des événements des boutons

Lorsqu'un bouton d'importation ou d'exportation est cliqué, le contrôleur appelle les méthodes correspondantes pour importer ou exporter les données.

```
private void handleImport() {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setFileFilter(new FileNameExtensionFilter("
       Fichiers CSV", "csv"));
    if (fileChooser.showOpenDialog(view) == JFileChooser.
       APPROVE_OPTION) {
        String filePath = fileChooser.getSelectedFile().
           getAbsolutePath();
        employeeModel.importData(filePath);
    }
}

private void handleExport() {
    JFileChooser fileChooser = new JFileChooser();
    if (fileChooser.showSaveDialog(view) == JFileChooser.
       APPROVE_OPTION) {
        String filePath = fileChooser.getSelectedFile().
           getAbsolutePath();
        employeeModel.exportData(filePath, employees);
    }
}
```

Cette implémentation permet à l'utilisateur de sélectionner un fichier pour l'importation et l'exportation à l'aide du composant `JFileChooser`.

5 Étape 5 : Main

Objectif :

L'objectif de cette étape est d'initialiser l'application en instanciant les différentes classes nécessaires (DAO, Modèles, Vue, et Contrôleur) et en la rendant visible à l'utilisateur. La classe 'Main' joue un rôle clé en tant qu'entrée principale de l'application.

Code Source de la Classe Main

Le code suivant montre l'implémentation de la méthode 'main' dans la classe 'Main' :

```
public class Main {
    public static void main(String[] args) {
        // Cr ation des instances des DAO
        EmployeeDAOImpl daoEmploye = new EmployeeDAOImpl();
        HolidayDAOImpl daoHoliday = new HolidayDAOImpl();

        // Cr ation de la vue
        PanelView view = new PanelView();

        // Cr ation des mod les avec les DAO correspondants
        EmployeModel modelEmploye = new EmployeModel(
            daoEmploye);
        HolidayModel holidayModel = new HolidayModel(
            daoHoliday);

        // Initialisation du contr leur avec la vue et les
            mod les
        new HolidayController(view, holidayModel);

        // Rendre la vue visible
        view.setVisible(true);
    }
}
```

Description du Code

- **DAO :** Les classes 'EmployeeDAOImpl' et 'HolidayDAOImpl' représentent les implémentations des DAO pour gérer les employés et les congés, respectivement. - **Vue (PanelView) :** Il s'agit de l'interface graphique de l'application. - **Modèles (EmployeModel et HolidayModel) :** Ces modèles sont responsables de la gestion de la logique métier, en liaison avec les DAO.

- **Contrôleur (HolidayController) :** Le contrôleur est initialisé avec la vue et les modèles pour gérer les interactions entre la vue et les données. Il connecte la vue et les données pour mettre à jour l'interface graphique et les données. - **Lancement de l'application :** Enfin, la vue est rendue visible à l'utilisateur avec ‘view.setVisible(true)‘.

Conclusion

Le développement des fonctionnalités d'importation et d'exportation des données dans cette application de gestion des congés a permis d'enrichir son fonctionnement en offrant des méthodes flexibles pour manipuler les fichiers CSV. Grâce à l'ajout des boutons dans l'interface graphique et à la gestion des vérifications de fichiers, cette fonctionnalité devient robuste et facile à utiliser.