**Compte rendu des travaux pratiques du cours de Java avancé N°3**

***Réalisé par : Intissar Lobad***

***Encadré par : Madame Asmaa Elkourchi***

2024-2025

Table des matières

[Gestion des employés : 3](#_Toc186397230)

[Vue (View) 3](#_Toc186397231)

[Contrôleur (Controller) 3](#_Toc186397232)

[Modèle (Model) 4](#_Toc186397233)

[DAO (Data Access Object) 6](#_Toc186397234)

[Conclusion 7](#_Toc186397235)

[Gestion des congés : 8](#_Toc186397236)

[Vue (View) 8](#_Toc186397237)

[Modèle (Model) 8](#_Toc186397238)

[Contrôleur (Controller) 9](#_Toc186397239)

[DAO (Data Access Object) 9](#_Toc186397240)

[Conclusion 10](#_Toc186397241)

# Gestion des employés :

## **Vue (View)**

Dans la classe EmployeeView, l'importation et l'exportation des données sont gérées par les boutons importButton et exportButton :

* **Méthodes associées** :
  + importButton.addActionListener: Ouvre une boîte de dialogue pour choisir un fichier à importer. Une fois un fichier sélectionné, elle déclenche la méthode importerEmployees() dans le contrôleur.
  + exportButton.addActionListener: Ouvre une boîte de dialogue pour choisir un fichier de destination. Une fois un fichier sélectionné, elle déclenche la méthode exporterEmployees() dans le contrôleur.
  + **Méthode showFileChooser** : Elle permet d'afficher une fenêtre de sélection de fichier. Elle est utilisée dans les actions d'importation et d'exportation pour obtenir le chemin du fichier sélectionné.

public String showFileChooser(String title) {

JFileChooser fileChooser = new JFileChooser();

fileChooser.setDialogTitle(title);

int userSelection = fileChooser.showOpenDialog(this);

if (userSelection == JFileChooser.APPROVE\_OPTION) {

File selectedFile = fileChooser.getSelectedFile();

return selectedFile.getAbsolutePath();

}

return null;

}

## **Contrôleur (Controller)**

La classe EmployeeController coordonne les actions entre la vue et le modèle. Elle contient les méthodes suivantes pour gérer l'importation et l'exportation :

* **Méthodes associées** :
  + importerEmployees(): Cette méthode est déclenchée par un clic sur le bouton d'importation dans la vue. Elle utilise la méthode showFileChooser de la vue pour obtenir le chemin du fichier, puis elle appelle la méthode importerEmployees() du modèle.
  + exporterEmployees(): Cette méthode est déclenchée par un clic sur le bouton d'exportation dans la vue. Elle récupère les données affichées dans le tableau et les envoie au modèle pour les exporter.

// Exportation des employés

public void importerEmployees() {

String filePath = view.showFileChooser("Importer des employés");

if (filePath != null) {

boolean importationReussie = model.importerEmployees(filePath);

if (importationReussie) {

view.afficherMessageSucces("Les employés ont été importés avec succès :)");

afficherEmployees();

} else {

view.afficherMessageErreur("Échec de l'importation des employés :(");

}

}

}

// Exportation des employés

public void exporterEmployees() {

int rowCount = view.jt.getRowCount();

boolean exportSuccess = true;

for (int i = 0; i < rowCount; i++) {

int id = Integer.parseInt(view.jt.getValueAt(i, 0).toString());

String nom = view.jt.getValueAt(i, 1).toString();

String prenom = view.jt.getValueAt(i, 2).toString();

String email = view.jt.getValueAt(i, 3).toString();

String telephone = view.jt.getValueAt(i, 4).toString();

double salaire = Double.parseDouble(view.jt.getValueAt(i, 5).toString());

Role role = Role.valueOf(view.getRole().toString());

Poste poste = Poste.valueOf(view.getPoste().toString());

int solde = 0;

boolean ajoutReussi = model.ajouterEmployee(id, nom, prenom, email, telephone, salaire, role, poste, solde);

if (!ajoutReussi) {

exportSuccess = false;

}

}

if (exportSuccess) {

view.afficherMessageSucces("Exportation des données vers la base de données réussie !");

} else {

view.afficherMessageErreur("Une ou plusieurs erreurs se sont produites lors de l'exportation.");

}

}

## **Modèle (Model)**

Dans la classe EmployeeModel, les méthodes responsables de l'importation et de l'exportation des données sont les suivantes :

* **Méthodes associées** :
  + importerEmployees(): Cette méthode lit un fichier CSV ligne par ligne, crée des objets Employee à partir des données lues, puis les ajoute à la base de données via la méthode ajouterEmployee() de la DAO.
  + exporterEmployees(): Cette méthode récupère la liste des employés à partir de la DAO, puis écrit chaque employé dans un fichier CSV.

// Importation des employés depuis un fichier CSV

public boolean importerEmployees(String filePath) throws NumberFormatException {

try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {

String line;

List<Employee> importedEmployees = new ArrayList<>();

while ((line = reader.readLine()) != null) {

String[] fields = line.split(",");

if (fields.length != 9) {

System.out.println("Invalid data format in file.");

return false;

}

int id = Integer.parseInt(fields[0].trim());

String nom = fields[1].trim();

String prenom = fields[2].trim();

String email = fields[3].trim();

String telephone = fields[4].trim();

double salaire = Double.parseDouble(fields[5].trim());

Role role = Role.valueOf(fields[6].trim());

Poste poste = Poste.valueOf(fields[7].trim());

int solde = Integer.parseInt(fields[8].trim());

Employee employee = new Employee(id, nom, prenom, email, telephone, salaire, role, poste, solde);

importedEmployees.add(employee);

}

for (Employee employee : importedEmployees) {

dao.ajouterEmployee(employee);

}

return true;

} catch (IOException | IllegalArgumentException e) {

System.out.println("Erreur lors de l'importation des employés: " + e.getMessage());

return false; }

}

// Exportation des employés vers un fichier CSV

public boolean exporterEmployees(String filePath) {

List<Employee> employees = dao.afficherEmployees();

try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {

for (Employee employee : employees) {

String line = String.format("%d,%s,%s,%s,%s,%.2f,%s,%s,%d",

employee.getId(),

employee.getNom(),

employee.getPrenom(),

employee.getEmail(),

employee.getTelephone(),

employee.getSalaire(),

employee.getRole().name(),

employee.getPoste().name(),

employee.getSolde());

writer.write(line);

writer.newLine();

}

return true;

} catch (IOException e) {

System.out.println("Erreur lors de l'exportation des employés: " + e.getMessage());

return false;

}

}

## **DAO (Data Access Object)**

Dans la classe EmployeeDAOImpl, l'importation et l'exportation des données sont réalisées via l'interface DataImportExport<Employee>. Les méthodes importData() et exportData() permettent d'importer et d'exporter les employés en utilisant des requêtes SQL pour interagir avec la base de données.

* **Méthodes associées** :
  + importData(): Cette méthode importe les données depuis un fichier CSV en les insérant dans la base de données.
  + exportData(): Cette méthode exporte les données des employés depuis la base de données vers un fichier CSV.

// Importation des données depuis un fichier CSV

public void importData(String filePath) throws IOException {

String query = "insert into employe (nom, prenom, email, telephone, salaire, roleId, posteId) values (?, ?, ?, ?, ?, (select id from role where nom=?), (select id from poste where nom=?))";

try (BufferedReader reader = new BufferedReader(new FileReader(filePath));

PreparedStatement pstmt = DBConnection.getConnection().prepareStatement(query)) {

String line;

while ((line = reader.readLine()) != null) {

String[] data = line.split(",");

if (data.length == 6) {

pstmt.setString(1, data[0]);

pstmt.setString(2, data[1]);

pstmt.setString(3, data[2]);

pstmt.setString(4, data[3]);

pstmt.setString(5, data[4]);

pstmt.setString(6, data[5]);

pstmt.setString(7, data[6]);

pstmt.setString(8, data[7]);

pstmt.executeUpdate();

}

}

}

}

// Exportation des données vers un fichier CSV

public void exportData(String filePath) throws IOException {

String query = "SELECT \* FROM employe";

try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath));

Statement stmt = DBConnection.getConnection().createStatement();

ResultSet rs = stmt.executeQuery(query)) {

while (rs.next()) {

String line = String.format("%d,%s,%s,%s,%s,%.2f,%s,%s,%d",

rs.getInt("id"),

rs.getString("nom"),

rs.getString("prenom"),

rs.getString("email"),

rs.getString("telephone"),

rs.getDouble("salaire"),

rs.getString("role"),

rs.getString("poste"),

rs.getInt("solde"));

writer.write(line);

writer.newLine();

}

}

}

## Conclusion

Les fonctionnalités d'importation et d'exportation des employés sont mises en place en suivant une architecture MVC avec la classe EmployeeView pour l'interaction utilisateur, EmployeeController pour gérer les actions, et EmployeeModel pour l'importation et l'exportation via la DAO. Cela assure une gestion claire et séparée des responsabilités tout en maintenant la modularité et la maintenabilité du code.

# **Gestion des congés :**

## **Vue (View)**

La classe HolidayView est responsable de l'affichage de l'interface graphique. Elle présente une table contenant les informations sur les congés et permet d'interagir avec l'utilisateur. Les boutons pour importer et exporter les données sont intégrés, et leurs actions sont liées aux méthodes du contrôleur.

#### Actions importantes :

* exporterButton: permet d'exporter les congés vers un fichier.
* importerButton: permet d'importer les congés à partir d'un fichier.

public JButton importerButton = new JButton("Importer");

public JButton exporterButton = new JButton("Exporter");

view.exporterButton.addActionListener(e -> exporterHolidays());

view.importerButton.addActionListener(e -> importerHolidays());

## **Modèle (Model)**

Le modèle HolidayModel gère les données des congés, en s'assurant que les informations sont valides avant de les ajouter ou de les modifier. Il contient aussi les méthodes pour exporter et importer les données via le DAO.

#### Actions importantes :

* La méthode exporterHolidays: Exporte les congés dans un fichier CSV.
* La méthode importerHolidays: Importe les congés depuis un fichier CSV.

public void exporterHolidays() {

JFileChooser fileChooser = new JFileChooser();

int option = fileChooser.showSaveDialog(view);

if (option == JFileChooser.APPROVE\_OPTION) {

String filePath = fileChooser.getSelectedFile().getAbsolutePath();

model.getDao().exporterHolidays(filePath);

view.afficherMessageSucces("Les congés ont été exportés avec succès !");

}

}

public void importerHolidays() {

JFileChooser fileChooser = new JFileChooser();

int option = fileChooser.showOpenDialog(view);

if (option == JFileChooser.APPROVE\_OPTION) {

String filePath = fileChooser.getSelectedFile().getAbsolutePath();

model.getDao().importerHolidays(filePath);

afficherHolidays();

view.afficherMessageSucces("Les congés ont été importés avec succès !");

}

}

## **Contrôleur (Controller)**

Le contrôleur HolidayController relie la vue et le modèle. Il gère les actions utilisateur comme l'ajout, la modification, la suppression, l'importation et l'exportation des congés.

#### Actions importantes :

* **Exporter les congés**: appelle la méthode exporterHolidays du modèle pour créer un fichier CSV.
* **Importer les congés**: appelle la méthode importerHolidays du modèle pour charger un fichier CSV et ajouter les congés dans la base de données.

public void exporterHolidays() {

JFileChooser fileChooser = new JFileChooser();

int option = fileChooser.showSaveDialog(view);

if (option == JFileChooser.APPROVE\_OPTION) {

String filePath = fileChooser.getSelectedFile().getAbsolutePath();

model.getDao().exporterHolidays(filePath);

view.afficherMessageSucces("Les congés ont été exportés avec succès !");

}

}

public void importerHolidays() {

JFileChooser fileChooser = new JFileChooser();

int option = fileChooser.showOpenDialog(view);

if (option == JFileChooser.APPROVE\_OPTION) {

String filePath = fileChooser.getSelectedFile().getAbsolutePath();

model.getDao().importerHolidays(filePath);

afficherHolidays();

view.afficherMessageSucces("Les congés ont été importés avec succès !");

}

}

## **DAO (Data Access Object)**

La classe HolidayDAOImpl implémente les méthodes spécifiques d'accès aux données pour les congés. Elle est responsable de l'exportation des congés dans un fichier CSV et de l'importation des congés depuis un fichier CSV.

#### Exportation des congés (fichier CSV) :

public void exporterHolidays(String filePath) {

List<Holiday> holidays = afficher();

try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {

writer.write("Id,Nom Employe,Date Debut,Date Fin,Type,Solde\n");

for (Holiday holiday : holidays) {

writer.write(String.format("%d,%s,%s,%s,%s,%d\n",

holiday.getId(),

holiday.getEmployeeNom(),

new java.text.SimpleDateFormat("dd/MM/yyyy").format(holiday.getStartDate()),

new java.text.SimpleDateFormat("dd/MM/yyyy").format(holiday.getEndDate()),

holiday.getHolidayType(),

holiday.getSolde()

));

}

System.out.println("Exportation réussie !");

} catch (IOException e) {

e.printStackTrace();

System.out.println("Erreur lors de l'exportation !");

}

}

#### Importation des congés (fichier CSV) :

public void importerHolidays(String filePath) {

try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {

String line = reader.readLine(); // Skip header

while ((line = reader.readLine()) != null) {

String[] parts = line.split(",");

if (parts.length == 6) {

String employeeNom = parts[1];

Date startDate = new java.text.SimpleDateFormat("dd/MM/yyyy").parse(parts[2]);

Date endDate = new java.text.SimpleDateFormat("dd/MM/yyyy").parse(parts[3]);

Holiday.HolidayType holidayType = Holiday.HolidayType.valueOf(parts[4]);

ajouter(new Holiday(employeeNom, startDate, endDate, holidayType));

}

}

System.out.println("Importation réussie !");

} catch (IOException | java.text.ParseException e) {

e.printStackTrace();

System.out.println("Erreur lors de l'importation !");

}

}

## Conclusion

Les fonctionnalités d'importation et d'exportation pour la gestion des congés ont été intégrées à l'architecture MVC avec une gestion claire des données à travers le modèle et l'accès au système de fichiers dans le DAO. Cela permet à l'utilisateur d'importer et d'exporter les congés au format CSV de manière simple et fluide.