Projet DAO/MVC en Java : Application de Gestion des Employés

AMINE ZAGHLOUL

December 8, 2024

Contents

1 Introduction

Ce rapport présente une application développée en Java basée sur l'architecture **MVC** (Modèle-Vue-Contrôleur) avec un accès aux données via la couche **DAO** (Data Access Object). L'application permet la gestion des employés dans une entreprise en incluant différents rôles et postes.

2 Objectifs

- Implémenter une architecture MVC robuste.
- Créer une application avec des rôles et des postes spécifiques.
- Utiliser des classes DAO pour séparer la logique d'accès aux données.

3 Architecture du Projet

L'architecture MVC est composée de :

- **Modèle :** Gère les données (DAO).
- **Vue :** Affiche les informations à l'utilisateur.
- **Contrôleur :** Gère les actions de l'utilisateur.

4 Étapes de Création du Projet

4.1 Création du Modèle (DAO)

Le modèle représente les objets métiers et l'accès aux données.

4.1.1 Classe Employee

```
1
2
3
4  public class Employee {
5      public enum Role {
6          EMPLOYEE,
7          ADMIN
8      }
9
10     public enum Poste {
11         INGENIEUR_ETUDE_ET_DEVELOPPEMENT,
12         TEAM_LEADER,
13         PILOTE
14     }
15     private int id;
16     private String nom;
```

```
17      private String prenom;
18      private String tel;
19      private String email;
20
21
22
23      private Double salaire ;
24      private Poste poste ;
25
26      public Role getRole() {
27          return role;
28      }
29
30      public void setRole(Role role) {
31          this.role = role;
32      }
33
34      private Role role ;
35
36      public Employee(int id, String nom, String prenom, String tel
            , String email,double salaire , Role role , Poste poste) {
37          this.salaire = salaire ;
38          this.id = id;
39          this.nom = nom;
40          this.prenom = prenom;
41          this.tel = tel;
42          this.email = email;
43          this.role = role ;
44          this.poste = poste ;
45      }
46      public Double getSalaire() {
47          return salaire;
48      }
49      public Poste getPoste() {
50          return poste;
51      }
52
53      public void setPoste(Poste poste) {
54          this.poste = poste;
55      }
56      public void setSalaire(Double salaire) {
57          this.salaire = salaire;
58      }
59      public int getId() {
60          return id;
61      }
62
63      public void setId(int id) {
64          this.id = id;
65      }
66
```

```
67      public String getNom() {
68          return nom;
69      }
70
71      public void setNom(String nom) {
72          this.nom = nom;
73      }
74
75      public String getPrenom() {
76          return prenom;
77      }
78
79      public void setPrenom(String prenom) {
80          this.prenom = prenom;
81      }
82
83      public String getTel() {
84          return tel;
85      }
86
87      public void setTel(String tel) {
88          this.tel = tel;
89      }
90
91      public String getEmail() {
92          return email;
93      }
94
95      public void setEmail(String email) {
96          this.email = email;
97
98      }
99
100     @Override
101     public String toString() {
102         return String.format(
103                 "| %-5s | %-15s | %-15s | %-15s | %-25s | %-10s |
                      %-20s | %-10s |",
104                 id, nom, prenom, tel, email, salaire, poste, role
105         );
106     }
107 }
108 package Model;
109
110 import DAO.EmployeeDAOImpl;
111 import java.util.List;
112
113 public class EmployeeModel {
114     private final EmployeeDAOImpl dao;
115
116     public EmployeeModel(EmployeeDAOImpl dao) {
```

```
117         this.dao = dao;
118     }
119 
120     public void addEmployee(Employee emp) {
121         dao.add(emp);
122     }
123 
124     public List<Employee> getAllEmployees() {
125         return dao.findAll();
126     }
127 
128     public void deleteEmployee(int id) {
129         dao.delete(id);
130     }
131 
132     public void updateEmployee(Employee emp, int id) {
133         dao.update(emp, id);
134     }
135 
136     public static int parseEmployeeId(String
        selectedEmployeeString) {
137         try {
138             return Integer.parseInt(selectedEmployeeString.split(
                "\\|")[1].trim());
139         } catch (Exception e) {
140             throw new IllegalArgumentException("Invalid␣employee␣
                format.");
141         }
142     }
143 }
```

4.1.2 Classe DAO

```
1  import java.util.ArrayList;
2  import java.util.List;
3  
4  public class EmployeeDAO {
5      private List<Employee> employees = new ArrayList<>();
6  
7      public void addEmployee(Employee e) {
8          employees.add(e);
9      }
10 
11     public List<Employee> getAllEmployees() {
12         return employees;
13     }
14 }
```

4.2 Création du Contrôleur

Le contrôleur gère la logique métier.

```
1  package Controller;
2
3  import Model.Employee;
4  import Model.EmployeeModel;
5  import Vue.Vue;
6
7  import javax.swing.*;
8  import java.util.ArrayList;
9  import java.util.List;
10
11 public class EmployeeController {
12     private final EmployeeModel model;
13     private final Vue view;
14
15     public EmployeeController(EmployeeModel model, Vue view) {
16         this.model = model;
17         this.view = view;
18
19         initializeListeners();
20     }
21
22     private void initializeListeners() {
23         view.getAjouter().addActionListener(e -> {
24             try {
25                 Employee emp = new Employee(
26                         0,
27                         view.getNom().getText(),
28                         view.getPrenom().getText(),
29                         view.getTel().getText(),
30                         view.getEmail().getText(),
31                         Double.parseDouble(view.getSal().getText
                               ()),
32                         Employee.Role.valueOf((String) view.
                               getRoleComboBox().getSelectedItem()),
33                         Employee.Poste.valueOf((String) view.
                               getPostesComboBox().getSelectedItem())
34                 );
35                 model.addEmployee(emp);
36                 JOptionPane.showMessageDialog(view, "Employee␣
                       added␣successfully!", "Success", JOptionPane.
                       INFORMATION_MESSAGE);
37                 view.getAfficher().doClick();
38             } catch (NumberFormatException ex) {
39                 JOptionPane.showMessageDialog(view, "Invalid␣
                       salary␣value!", "Error", JOptionPane.
                       ERROR_MESSAGE);
40             } catch (IllegalArgumentException ex) {
```

```
                JOptionPane.showMessageDialog(view, ex.getMessage
                    (), "Error", JOptionPane.ERROR_MESSAGE);
            }
        });

        view.getAfficher().addActionListener(e -> {
            List<Employee> allEmployees = model.getAllEmployees()
                ;
            List<String> employeeStrings = new ArrayList<>();
            employeeStrings.add(String.format(
                    "| %-5s | %-15s | %-15s | %-15s | %-25s | %
                        -10s | %-20s | %-10s |",
                    "ID", "Nom", "Prenom", "Tel", "Email", "
                        Salaire", "Poste", "Role"
            ));
            for (Employee emp : allEmployees) {
                employeeStrings.add(emp.toString());
            }
            String[] employeeArray = employeeStrings.toArray(new
                String[0]);
            JList<String> updatedList = new JList<>(employeeArray
                );
            view.setEmployeeList(updatedList);
            JPanel p3 = view.getP3();
            p3.removeAll();
            p3.add(new JScrollPane(updatedList));
            p3.revalidate();
            p3.repaint();
        });

        view.getSupprimer().addActionListener(e -> {
            String selectedEmployeeString = view.getEmployeeList
                ().getSelectedValue();
            if (selectedEmployeeString == null) {
                JOptionPane.showMessageDialog(view, "No employee
                    selected!", "Error", JOptionPane.ERROR_MESSAGE
                    );
                return;
            }

            try {
                int id = EmployeeModel.parseEmployeeId(
                    selectedEmployeeString);
                int confirm = JOptionPane.showConfirmDialog(view,
                    "Are you sure you want to delete this
                    employee?", "Confirm", JOptionPane.
                    YES_NO_OPTION);
                if (confirm == JOptionPane.YES_OPTION) {
                    model.deleteEmployee(id);
                    JOptionPane.showMessageDialog(view, "Employee
                        deleted successfully!", "Success",
```

```
                        JOptionPane.INFORMATION_MESSAGE);
78                      view.getAfficher().doClick();
79                  }
80              } catch (IllegalArgumentException ex) {
81                  JOptionPane.showMessageDialog(view, ex.getMessage
                        (), "Error", JOptionPane.ERROR_MESSAGE);
82              }
83          });
84
85          view.getModifier().addActionListener(e -> {
86              String selectedEmployeeString = view.getEmployeeList
                    ().getSelectedValue();
87              if (selectedEmployeeString == null) {
88                  JOptionPane.showMessageDialog(view, "No employee 
                        selected!", "Error", JOptionPane.ERROR_MESSAGE
                        );
89                  return;
90              }
91
92              try {
93                  int id = EmployeeModel.parseEmployeeId(
                        selectedEmployeeString);
94                  Employee emp = new Employee(
95                          id,
96                          view.getNom().getText(),
97                          view.getPrenom().getText(),
98                          view.getTel().getText(),
99                          view.getEmail().getText(),
100                         Double.parseDouble(view.getSal().getText
                            ()),
101                         Employee.Role.valueOf((String) view.
                            getRoleComboBox().getSelectedItem()),
102                         Employee.Poste.valueOf((String) view.
                            getPostesComboBox().getSelectedItem())
103                 );
104                 model.updateEmployee(emp, id);
105                 JOptionPane.showMessageDialog(view, "Employee 
                        updated successfully!", "Success", JOptionPane
                        .INFORMATION_MESSAGE);
106                 view.getAfficher().doClick();
107             } catch (NumberFormatException ex) {
108                 JOptionPane.showMessageDialog(view, "Invalid 
                        salary value!", "Error", JOptionPane.
                        ERROR_MESSAGE);
109             } catch (IllegalArgumentException ex) {
110                 JOptionPane.showMessageDialog(view, ex.getMessage
                        (), "Error", JOptionPane.ERROR_MESSAGE);
111             }
112         });
113     }
114 }
```

4.3 Création de la Vue

La vue affiche les résultats.

```
package Vue;
import DAO.*;
import Model.Employee;

import javax.print.DocFlavor;
import javax.swing.*;
import javax.swing.border.Border;
import java.awt.*;
import java.util.ArrayList;
import java.util.List;

public class Vue extends JFrame {

    private JPanel p1;
    private JPanel p2;
    private JPanel p3;
    public JPanel getP3(){
        return p3 ;
    }

    private JPanel p4;
    private JComboBox<String> postesComboBox ;
    private JComboBox<String> roleComboBox ;
    private JButton ajouter ;
    private JButton modifier;
    private JButton supprimer ;
    private JButton afficher ;


    private JTextField tel;
    private JTextField sal  ;
    private JTextField nom ;
    private JTextField prenom ;
    private JTextField email  ;
    private  JList<String> employeeList ;
    public  JList<String> getEmployeeList(){
        return employeeList ;
    };
    public  void setEmployeeList(JList<String> employeeList){
        this.employeeList =employeeList   ;
    } ;
    public Vue() {
        setTitle("App");
        setSize(1920, 1080);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        EmployeeDAOImpl eImp = new EmployeeDAOImpl();
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        p1 = new JPanel();
```

```
49          p2 = new JPanel();
50          p3 = new JPanel();
51          p4 = new JPanel();
52          p1.setLayout(new BorderLayout());
53          p2.setLayout(new GridLayout(7,2,10,10));
54
55          p3.setLayout(new GridLayout());
56          p4.setLayout(new GridLayout());
57          add(p1);
58          p1.add(p2 , BorderLayout.NORTH);
59          p1.add(p3 , BorderLayout.CENTER);
60          p1.add(p4 , BorderLayout.SOUTH);
61
62
63          p2.add(new JLabel("Nom:"));
64          nom =  new JTextField();p2.add(nom);
65          p2.add(new JLabel("Prenom:"));
66          prenom =  new JTextField();p2.add(prenom);
67          p2.add(new JLabel("Email:"));
68          email =  new JTextField();p2.add(email);
69          p2.add(new JLabel("Telephone:"));
70          tel = new JTextField();p2.add(tel);
71          p2.add(new JLabel("Salaire:"));
72          sal =  new JTextField();p2.add(sal);
73
74          p2.add(new JLabel("Role:"));
75          List<Employee.Role> roles = eImp.findAllRoles() ;
76          String[] roleStrings = roles.stream()
77                  .map(Enum::name)
78                  .toArray(String[]::new);
79          roleComboBox = new JComboBox<String>(roleStrings);
80          p2.add(roleComboBox);
81
82          p2.add(new JLabel("Poste:"));
83          List<Employee.Poste> postes = eImp.findAllPostes() ;
84          String[] postesStrings = postes.stream()
85                  .map(Enum::name)
86                  .toArray(String[]::new);
87          postesComboBox = new JComboBox<String>(postesStrings);
88          p2.add(postesComboBox);
89
90
91
92          //P3 Container
93          List<Employee> all_e = eImp.findAll();
94          List<String> allString = new ArrayList<String>();
95          allString.add(String.format(
96                  "|␣%-5s␣|␣%-15s␣|␣%-15s␣|␣%-15s␣|␣%-25s␣|␣%-10s␣|
                        ␣%-20s␣|␣%-10s␣|",
97                  "ID", "Nom", "Prenom", "Tel", "Email", "Salaire",
                        "Poste", "Role"
```

```
98          ));
99          for (Employee e : all_e){
100             allString.add(e.toString());
101         }
102         String[] allStringArray = allString.toArray(new String
                [0]);
103
104         employeeList = new JList<String>(allStringArray);
105         p3.add(employeeList);
106
107
108         //p4 Container
109
110         p4.setLayout(new FlowLayout());
111         this.ajouter = new JButton("Ajouter");
112         this.modifier = new JButton("Modifier");
113         this.supprimer = new JButton("Supprimer");
114         this.afficher = new JButton("Afficher");
115         p4.add(this.ajouter);p4.add(this.modifier);p4.add(this.
                supprimer);p4.add(this.afficher);
116         setVisible(true);
117     }
118
119     public JComboBox<String> getPostesComboBox() {
120         return postesComboBox;
121     }
122
123     public JComboBox<String> getRoleComboBox() {
124         return roleComboBox;
125     }
126
127     public JButton getAjouter() {
128         return ajouter;
129     }
130
131     public JButton getModifier() {
132         return modifier;
133     }
134
135     public JButton getSupprimer() {
136         return supprimer;
137     }
138
139     public JButton getAfficher() {
140         return afficher;
141     }
142
143     public JTextField getTel() {
144         return tel;
145     }
146
```

```
147     public JTextField getSal() {
148         return sal;
149     }
150 
151     public JTextField getNom() {
152         return nom;
153     }
154 
155     public JTextField getPrenom() {
156         return prenom;
157     }
158 
159     public JTextField getEmail() {
160         return email;
161     }
162 
163     public JPanel getP1() {
164         return p1;
165     }
166 
167 }
```

4.4 Création de la Classe Main

```
1  import DAO.EmployeeDAOImpl;
2  import Model.EmployeeModel;
3  import Vue.Vue;
4  import Controller.EmployeeController;
5  
6  public class Main {
7      public static void main(String[] args) {
8          EmployeeDAOImpl dao = new EmployeeDAOImpl();
9          EmployeeModel model = new EmployeeModel(dao);
10         Vue view = new Vue();
11 
12         // Pass the model and view to the controller
13         new EmployeeController(model, view);
14     }
15 }
```

5 Conclusion

Ce projet met en œuvre l'architecture MVC et l'utilisation des DAO pour une application simple de gestion des employés. Chaque couche est bien séparée, permettant une évolutivité et une maintenance optimales.