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

Votre Nom

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;
```

```java
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
```

```java
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) {
```

```java
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

```java
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) {
```

```java
                    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",
```

```java
                                JOptionPane.INFORMATION_MESSAGE);
                    view.getAfficher().doClick();
                }
            } catch (IllegalArgumentException ex) {
                JOptionPane.showMessageDialog(view, ex.getMessage
                    (), "Error", JOptionPane.ERROR_MESSAGE);
            }
        });

        view.getModifier().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);
                Employee emp = new Employee(
                        id,
                        view.getNom().getText(),
                        view.getPrenom().getText(),
                        view.getTel().getText(),
                        view.getEmail().getText(),
                        Double.parseDouble(view.getSal().getText
                            ()),
                        Employee.Role.valueOf((String) view.
                            getRoleComboBox().getSelectedItem()),
                        Employee.Poste.valueOf((String) view.
                            getPostesComboBox().getSelectedItem())
                );
                model.updateEmployee(emp, id);
                JOptionPane.showMessageDialog(view, "Employee 
                    updated successfully!", "Success", JOptionPane
                    .INFORMATION_MESSAGE);
                view.getAfficher().doClick();
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(view, "Invalid 
                    salary value!", "Error", JOptionPane.
                    ERROR_MESSAGE);
            } catch (IllegalArgumentException ex) {
                JOptionPane.showMessageDialog(view, ex.getMessage
                    (), "Error", JOptionPane.ERROR_MESSAGE);
            }
        });
    }
}
```

4.3 Création de la Vue

La vue affiche les résultats.

```
1   package Vue;
2   import DAO.*;
3   import Model.Employee;
4
5   import javax.print.DocFlavor;
6   import javax.swing.*;
7   import javax.swing.border.Border;
8   import java.awt.*;
9   import java.util.ArrayList;
10  import java.util.List;
11
12  public class Vue extends JFrame {
13
14      private JPanel p1;
15      private JPanel p2;
16      private JPanel p3;
17      public JPanel getP3(){
18          return p3 ;
19      }
20
21      private JPanel p4;
22      private JComboBox<String> postesComboBox ;
23      private JComboBox<String> roleComboBox ;
24      private JButton ajouter ;
25      private JButton modifier;
26      private JButton supprimer ;
27      private JButton afficher ;
28
29
30      private JTextField tel;
31      private JTextField sal  ;
32      private JTextField nom ;
33      private JTextField prenom ;
34      private JTextField email  ;
35      private  JList<String> employeeList ;
36      public  JList<String> getEmployeeList(){
37          return employeeList ;
38      };
39      public  void setEmployeeList(JList<String> employeeList){
40          this.employeeList =employeeList   ;
41      } ;
42      public Vue() {
43          setTitle("App");
44          setSize(1920, 1080);
45          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
46          EmployeeDAOImpl eImp = new EmployeeDAOImpl();
47          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
48          p1 = new JPanel();
```

```java
        p2 = new JPanel();
        p3 = new JPanel();
        p4 = new JPanel();
        p1.setLayout(new BorderLayout());
        p2.setLayout(new GridLayout(7,2,10,10));

        p3.setLayout(new GridLayout());
        p4.setLayout(new GridLayout());
        add(p1);
        p1.add(p2 , BorderLayout.NORTH);
        p1.add(p3 , BorderLayout.CENTER);
        p1.add(p4 , BorderLayout.SOUTH);


        p2.add(new JLabel("Nom:"));
        nom =  new JTextField();p2.add(nom);
        p2.add(new JLabel("Prenom:"));
        prenom =  new JTextField();p2.add(prenom);
        p2.add(new JLabel("Email:"));
        email =  new JTextField();p2.add(email);
        p2.add(new JLabel("Telephone:"));
        tel = new JTextField();p2.add(tel);
        p2.add(new JLabel("Salaire:"));
        sal =  new JTextField();p2.add(sal);

        p2.add(new JLabel("Role:"));
        List<Employee.Role> roles = eImp.findAllRoles() ;
        String[] roleStrings = roles.stream()
                .map(Enum::name)
                .toArray(String[]::new);
        roleComboBox = new JComboBox<String>(roleStrings);
        p2.add(roleComboBox);

        p2.add(new JLabel("Poste:"));
        List<Employee.Poste> postes = eImp.findAllPostes() ;
        String[] postesStrings = postes.stream()
                .map(Enum::name)
                .toArray(String[]::new);
        postesComboBox = new JComboBox<String>(postesStrings);
        p2.add(postesComboBox);



        //P3 Container
        List<Employee> all_e = eImp.findAll();
        List<String> allString = new ArrayList<String>();
        allString.add(String.format(
                "|␣%-5s␣|␣%-15s␣|␣%-15s␣|␣%-15s␣|␣%-25s␣|␣%-10s␣|
                    ␣%-20s␣|␣%-10s␣|",
                "ID", "Nom", "Prenom", "Tel", "Email", "Salaire",
                    "Poste", "Role"
```

```java
 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

```java
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 applica-
tion simple de gestion des employés. Chaque couche est bien séparée, permettant une
évolutivité et une maintenance optimales.