2023/2024	Projet Java – Etape 05
BTS SIO 1SIOB - SLAM	Auteur : AUGEREAU Eliott et PORTOLLEAU Anaïs Date de rédaction : 16/04/2024

Étape n°5 - fonctionnalité de consultation des salariés d'un service

Cette étape et les suivantes sont destinées à programmer le comportement des interfaces graphiques utilisateurs
conçues à l'étape n°3.

Un ticket de demande d'évolution du code a été émis :
Si on se réfère au scénario du cas d'utilisation « C2-SN - Consulter les salariés par service » (Cf. étape 3), la sélection d'un service dans la liste déroulante doit filtrer la liste des salariés en fonction de l'appartenance de ceux-ci au service.

Or, actuellement, cette fonctionnalité est inopérante.

Travail à faire :
Déterminer la cause du problème et y remédier.

A remettre à l'issue de l'étape
Dans une archive zip respectant la nomenclature :
le répertoire de votre projet NetBeans ; le code source est normalisé et commenté ;
un compte-rendu de l'étape comportant les éléments suivants :
o vos explications ;
o un rapport de tests fonctionnels :
copies d'écran montrant le comportement de l'application lors du test des scénarios prévus ;
interprétation des résultats obtenus ;
o un bilan de l'étape (fait, non fait, difficultés rencontrées).

On a importé toutes les librairies ou classes qui ont été nécessaire dans notre code :

```
package vue;

import java.io.IOException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.DefaultComboBoxModel;
import javax.swing.table.DefaultTableModel;
import modele.dao.ConnexionBDD;
import modele.metier.Salarie;
import modele.dao.DaoSalarie;
import modele.metier.Service;
/**
```

Puis ce code permet de remplir selon les colonnes de la zone d'affichage dans chaque colonne pour les différentes informations du client en utilisant les méthodes getColumnModle() et setPreferredWidth. Puis j'appelle la méthode remplirJComboBoxAvecDonneDeLaBase().
Puis utilisé

```
private void jTableSalariesMouseClicked(java.awt.event.MouseEvent evt) {
    // Un salarié sélectionné ?
    if (jTableSalaries.getSelectedRow() > -1) {
        String codeSalarie = (String) jTableSalaries.getValueAt(jTableSalaries.getSelectedRow(), 0);
        try {
            Salarie unSalarie = DaoSalarie.getOneById(codeSalarie);
            System.out.println("Salarié sélectionné :\n" + unSalarie.toString());
        } catch (Exception ex) {
            System.out.println("JFrameListeSalaries - Erreur lors de la lecture du salarié sélectionné : " + ex.getMessage());
        }
    }

}

private void jButtonQuitterActionPerformed(java.awt.event.ActionEvent evt) {
    this.dispose();
    System.exit(0);
}
```

Et avec le constructeur : La méthode JFrameListeSalaries() est le constructeur de la classe. Elle initialise les composants de l'interface graphique, définit les modèles pour la table des salariés et la liste déroulante des services, et remplit la liste déroulante avec les services disponibles.
Méthodes d'Action des Composants : Ces méthodes sont déclenchées lorsque certains composants sont activés, par exemple lorsqu'un bouton est cliqué ou qu'une sélection est effectuée dans la liste déroulante. Elles appellent d'autres méthodes pour effectuer les actions correspondantes, comme la consultation, la modification ou la suppression d'un salarié.

Puis dans la JComboBoxk Méthode remplirJComboBoxAvecDonneesDeLaBase() : Cette méthode se connecte à la base de données pour récupérer la liste des services disponibles, puis remplit la liste déroulante avec ces services.
Méthode remplirJComBoxServices(List<String> desServices) : Cette méthode met à jour le modèle de la liste déroulante avec une liste de services fournie en paramètre.

```java
// SERVICES
private void remplirJComboBoxAvecDonneesDeLaBase() throws IOException {
    try {
        modele.dao.ConnexionBDD connexionBDD = new modele.dao.ConnexionBDD();
        java.sql.Connection laConnection = ConnexionBDD.getConnexion();
        PreparedStatement St = laConnection.prepareStatement("select designation from service");
        ResultSet Rs = St.executeQuery();
        List<String> services = new ArrayList<>();
        while (Rs.next()) {
            services.add(Rs.getString("designation"));
        }
        remplirJComBoxServices(services);
    } catch (IOException | SQLException e) {
        System.out.println("Impossible de se connecter à la base : " + e.getMessage());
    }
}

// Mettre à jour la JComboBox avec une liste de services
public void remplirJComBoxServices(List<String> desServices) {
    modeleJComboLesServices.removeAllElements();
    modeleJComboLesServices.addElement("*** Tous les Services ***");
    for (String service : desServices) {
        modeleJComboLesServices.addElement(service);
    }
}
```

```
private void jComboBoxLesServicesActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    // Récupérer le service sélectionné
    String serviceSelectionne = (String) jComboBoxLesServices.getSelectedItem();
    // Afficher les salariés pour ce service dans la table
    afficherSalariesPourService(serviceSelectionne);
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)
    //</editor-fold>

    //</editor-fold>
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                new JFrameListeSalaries().setVisible(true);
            } catch (IOException ex) {
                Logger.getLogger(JFrameListeSalaries.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    });
}
```

Puis ici on récupère d'abord le service sélectionné par l'utilisateur puis on utilise la méthode afficherSalariePourService en fonction du service sélectionné.

Puis Méthode main(String args[]) : La méthode main() est la méthode principale qui démarre l'application. Elle crée une instance de la classe JFrameListeSalaries et la rend visible.

La méthode afficherSalariesPourService(String service) récupère les salariés associés à un service donné et les affiche dans la table des salariés. Voici une explication détaillée de cette méthode :
En Paramètre : Cette méthode prend en paramètre le nom du service sélectionné dans la liste déroulante.
Récupération des Salariés : En fonction du service sélectionné, la méthode appelle la méthode DaoSalarie.getSalariesByService(service) pour récupérer les salariés associés à ce service depuis la base de données.
Mise à Jour de la Table des Salariés : Une fois les salariés récupérés, la méthode efface le contenu actuel de la table des salariés (modeleJTableLesSalaries.setRowCount(0)) pour la rafraîchir. Ensuite, elle parcourt la liste des salariés récupérés et ajoute chaque salarié à la table des salariés sous forme de lignes. Pour chaque salarié, elle crée un tableau d'objets contenant

les informations du salarié (code, nom, prénom, etc.) et utilise modeleJTableLesSalaries.addRow(row) pour ajouter cette ligne à la table.
Une fois que toutes les lignes de salariés sont ajoutées à la table, celle-ci est mise à jour pour afficher les nouveaux salariés associés au service sélectionné.
Cette procédure est essentielle pour mettre à jour dynamiquement la liste des salariés affichés en fonction du service sélectionné par l'utilisateur dans la liste déroulante.

```
private void afficherSalariesPourService(String service) {
    try {
        // Récupérer les salariés pour le service sélectionné
        List<Salarie> salaries = DaoSalarie.getSalariesByService(service);
        //Si tous services sélectionné
        if (service.equals("*** Tous les Services ***")) {
            salaries = DaoSalarie.getAll();
        } else {
            salaries = DaoSalarie.getSalariesByService(service);
        }


        // Effacer le contenu actuel de la table
        modeleJTableLesSalaries.setRowCount(0);

        // Ajouter chaque salarié à la table
        for (Salarie salarie : salaries) {
            Object[] row = {
                salarie.getCode(),
                salarie.getNom(),
                salarie.getPrenom(),
                salarie.getDateNaiss(),
                salarie.getDateEmbauche(),
                salarie.getFonction(),
                salarie.getService().getDesignation(),
                salarie.getCategorie().getLibelle()
            };
            modeleJTableLesSalaries.addRow(row);
        }
    } catch (SQLException | IOException ex) {
        ex.printStackTrace();
    }
}
```

Bilan de l'étape :
Sur cette étape, à la fin on s'est rendu compte que le projet complet ne focntionnait pas et donc on a passé beaucoup de temps à régler les problèmes de main, et divers d'autrs problèmes d'exécution