

2024/2025	AP – Projet Rentrée
BTS SIO	Auteur : PORTOLLEAU Anaïs
2SLAM	Date de rédaction : 15/10/2024

Compte-rendu de AP – Projet Septembre

Objectif Réviser les notions de développement java vues en première année Activité Cette activité fait suite au projet SLAM de fin de 1ère année (projet SIRH, Cf. documents fournis). Le modèle de données de l'application évolue ; il prend en compte les formations suivies par les salariés, comme le montrent les diagrammes cidessous.

Ressources fournies

Diagramme des classes du programme :

A l'étape n°7 du projet de 1ère année, le programme permettait d'afficher la fenêtre de consultation des données relatives à un salarié, ou bien de supprimer ce salarié. On vous fournit le code correspondant à cette étape du projet (« JavaApplication_1Slam_Projet2_SirhInfoware_2024_corrige_etape_7 ») ainsi que le cahier des charges et la bibliothèque du composant de saisie des dates (jcalendar). Le script SQL de création de la BDD est inclus dans le projet NetBeans.

Travail à faire

Ajoutez le code nécessaire pour :

1. Afficher, aussi dans cette même fenêtre : les formations suivies par le salarié ;
Rappel : les informations affichées ne doivent pas être modifiables ;

2. permettre à l'utilisateur d'ajouter une formation existante aux formations suivies par ce salarié.

Dans modele.dao, j'ai commencé par modifier le fichier sirh pour les propriétés d'accès à la base de données ayant wamp et un port 3307 j'ai mit ceci :

```

1 #####
2 # Paramètres de connexion JDBC
3 # Utilisation : décommenter les paramètres spécifiques au SGBD visé
4 # Les paramètres suivants doivent être renseignés : url, utilBD, mdp
5 #####
6
7 # MYSQL / MariaDB local : mysql-connector-java-8.0.18.jar
8 url=jdbc:mysql://localhost:3307/SIRH?serverTimezone=Europe/Paris
9 utilBD=sirh_util
10 mdpBD=secret

```

Test de connexion

```

getConnexion : jdbc:mysql://localhost:3307/SIRH?serverTimezone=Europe/Paris
ConnexionBDD : connexion réussie
BUILD SUCCESSFUL (total time: 0 seconds)

```

J'ai codé ensuite dans modele.metier, j'ai codé les classes qui n'étaient pas présentes comme Formation et Service :

2024/2025	AP – Projet Rentrée
BTS SIO 2SLAM	Auteur : PORTOLLEAU Anaïs
	Date de rédaction : 15/10/2024

```
package modele.metier;

import java.util.ArrayList;
import java.util.Date;

/**
 *
 * @author anaïs
 */
public class Formation {
    public String code;
    public String nom;
    public Date dateDebut;
    private int nbreJours;
    private double coutJourForm;

    // lien avec la classe Salarie
    private ArrayList<Suivre> lesFormationsSuivies = new ArrayList<>();

    public Formation(String code, String nom, Date dateDebut, int nbreJours, double coutJourForm) {
        this.code = code;
        this.nom = nom;
        this.dateDebut = dateDebut;
        this.nbreJours = nbreJours;
        this.coutJourForm = coutJourForm;
    }
    public Formation() {

    }
}
```

J'ai ajouté les différents attributs que cette classe avait. J'ai rajouté son constructeur remplis avec ces différents attributs et j'ai rajouté un constructeur vide.

J'ai ajouté ces différents accesseurs et mutateurs que j'ai généré avec Netbeans :

```
public String getCode() {
    return code;
}

public void setCode(String code) {
    this.code = code;
}

public String getNom() {
    return nom;
}

public void setNom(String nom) {
    this.nom = nom;
}

public Date getDateDebut() {
    return dateDebut;
}

public void setDateDebut(Date dateDebut) {
    this.dateDebut = dateDebut;
}

public int getNbreJours() {
    return nbreJours;
}

public void setNbreJours(int nbreJours) {
    this.nbreJours = nbreJours;
}

public double get CoutJourForm() {
    return coutJourForm;
}

public void set CoutJourForm(double coutJourForm) {
    this.coutJourForm = coutJourForm;
}
```

2024/2025	AP – Projet Rentrée
BTS SIO	Auteur : PORTOLLEAU Anaïs
2SLAM	Date de rédaction : 15/10/2024

Et sa méthode toString pour afficher l'état :

```
@Override
public String toString() {
    return nom;
}
```

Et la classe Service :

```
/**
 * Classe métier Service
 * @author btssio
 */
public class Service {
    private int code;
    private String designation;
    private String email;
    private String telephone;

    /**
     * Constructeur simple
     * @param code
     * @param designation
     */
    public Service(int code, String designation) {
        this.code = code;
        this.designation = designation;
        this.email = "";
        this.telephone = "";
    }

    /**
     * Constructeur complet
     * @param code
     * @param designation
     * @param email
     * @param telephone
     */
    public Service(int code, String designation, String email, String telephone) {
        this.code = code;
        this.designation = designation;
        this.email = email;
        this.telephone = telephone;
    }

    public int getCode() {
        return code;
    }

    public void setCode(int code) {
        this.code = code;
    }

    public String getDesignation() {
        return designation;
    }
}
```

Dans Categorie.java, j'ai ajouté son lien avec la classe Salarie :

```
private ArrayList<Salarie> categorie = new ArrayList<>();
```

Dans Salarie voici toutes les différentes relations avec les autres classes que j'ai ajouté :

2024/2025	AP – Projet Rentrée
BTS SIO	Auteur : PORTOLLEAU Anaïs
2SLAM	Date de rédaction : 15/10/2024

```

private Categorie categorie; // Relation Many-to-One avec Categorie
private ArrayList<Formation> lesFormationsSuivies; // Relation Many-to-One avec Formation
// Relation avec la classe Suivre
private ArrayList<Formation> formationsSuivies; // Une liste de Suivre
private ArrayList<Suivre> suivis; // Liste des formations suivies
private String code;

```

Et j'ai ajouté une classe Suivre pour faire le lien entre salarié et la classe formations :

```

1 package modele.metier;
2
3 /**
4  *
5  * @author anaïs
6  */
7 public class Suivre {
8     private String codeSal; // Référence au salarié
9     //private String codeFormation; // Référence à la formation
10    private Salarie salarie;
11    private Formation formation;
12
13    // Constructeur
14    public Suivre(String codeSal, Formation formation) {
15        this.codeSal = codeSal;
16        this.formation = formation;
17    }
18
19    // Getter pour la formation
20    public String getSalarie() {
21        return codeSal;
22    }
23
24    public Formation getFormation() {
25        return formation;
26    }
27 }

```

J'ai rajouté un test pour daoFormation :

2024/2025	AP – Projet Rentrée
BTS SIO	Auteur : PORTOLLEAU Anaïs
2SLAM	Date de rédaction : 15/10/2024

Je teste chacune des méthodes contenues dans formations pour pouvoir faire ce que je souhaite coder par la suite :

```
import modele.dao.DaoFormation;
import modele.metier.Formation;

public class TestDaoFormation {
    public static void main(String[] args) {
        try {

            // Test 1: Récupérer une formation par son code
            Formation formationById = DaoFormation.getOneById( code:"F01");
            if (formationById != null) {
                System.out.println(⌘:"Test 1 : DaoFormation.getOneById");
                System.out.println("Formation avec le code F01 trouvée : " + formationById.getNom());
            } else {
                System.out.println(⌘:"Aucune formation trouvée avec le code F01.");
            }

            //Test 2: Récupérer les formations suivies par un salarié
            String codeSalarie = "S07"; // Changez cela avec un code valide pour le test
            ArrayList<Formation> formationsForSalarie = DaoFormation.getFormationsBySalarie(codeSalarie);

            System.out.println("Test 2 : Récupérer les formations pour le salarié " + codeSalarie);
            if (!formationsForSalarie.isEmpty()) {
                System.out.println(formationsForSalarie.size() + " formations trouvées pour le salarié " + codeSalarie + ":");
                for (Formation f : formationsForSalarie) {
                    System.out.println(⌘:f.getNom());
                }
            } else {
                System.out.println("Aucune formation trouvée pour le salarié " + codeSalarie + ".");
            }

            // Test 3: Récupérer les formations pour un salarié avec un code qui n'existe pas
            String codeSalarieInconnu = "S02"; // faut que ce code n'existe pas
            ArrayList<Formation> formationsInconnues = DaoFormation.getFormationsBySalarie( codeSalarie:codeSalarieInconnu);
            System.out.println("Test 3 : Récupérer les formations pour le salarié " + codeSalarieInconnu);
            if (formationsInconnues.isEmpty()) {
                System.out.println("Aucune formation trouvée pour le salarié " + codeSalarieInconnu + ".");
            } else {
                System.out.println("Des formations ont été trouvées pour le salarié " + codeSalarieInconnu + ".");
            }

        } catch (SQLException | IOException e) {
            System.err.println("Erreur lors des tests : " + e.getMessage());
        }
    }
}
```

2024/2025	AP – Projet Rentrée
BTS SIO 2SLAM	Auteur : PORTOLLEAU Anaïs
	Date de rédaction : 15/10/2024

J'ai créé ensuite une classe de test pour la classe formation :

```
package test.metier;

import java.util.ArrayList;
import java.util.Date;
import modele.metier.Formation;

/**
 * Classe de test pour la classe Formation
 * @author anaïs
 */
public class TestFormation {
    public static void main(String[] args) {

        // Créer des instances de Formation
        Formation formation1 = new Formation( code:"F001", nom:"Formation Java", new Date(), nbreJours: 5, coutJourForm:200.0);
        Formation formation2 = new Formation( code:"F002", nom:"Formation SQL", new Date(), nbreJours: 3, coutJourForm:150.0);
        Formation formation3 = new Formation( code:"F003", nom:"Formation Python", new Date(), nbreJours: 7, coutJourForm:250.0);

        // Afficher les noms des formations
        System.out.println("Nom de la formation 1: " + formation1.getNom());
        System.out.println("Nom de la formation 2: " + formation2.getNom());
        System.out.println("Nom de la formation 3: " + formation3.getNom());

        // Tester la méthode toString()
        System.out.println("Détails de la formation 1: " + formation1);
        System.out.println("Détails de la formation 2: " + formation2);
        System.out.println("Détails de la formation 3: " + formation3);

        // Ajouter les formations à une liste pour simuler l'utilisation dans Salarie
        ArrayList<Formation> formations = new ArrayList<>();
        formations.add(formation1);
        formations.add(formation2);
        formations.add(formation3);

        // Vérifier le nombre de formations dans la liste
        System.out.println("Nombre de formations suivies: " + formations.size());

        // Parcourir et afficher les formations
        for (Formation formation : formations) {
            System.out.println("Formation suivie: " + formation.getNom());
        }
    }
}
```

On peut voir en testant ensuite que cela marche bien cela récupère bien :

```
run:
Nom de la formation 1: Formation Java
Nom de la formation 2: Formation SQL
Nom de la formation 3: Formation Python
Détails de la formation 1: Formation Java
Détails de la formation 2: Formation SQL
Détails de la formation 3: Formation Python
Nombre de formations suivies: 3
Formation suivie: Formation Java
Formation suivie: Formation SQL
Formation suivie: Formation Python
BUILD SUCCESSFUL (total time: 0 seconds)
```

Ensuite dans le JDialog j'ai ajouté une JList pour pouvoir afficher les formations des salariés :

Et un bouton ajouter pour la question 2 pour pouvoir ajouter une formation au salarié souhaité.

Voici le jdialog :

2024/2025	AP – Projet Rentrée
BTS SIO 2SLAM	Auteur : PORTOLLEAU Anaïs
	Date de rédaction : 15/10/2024

Fiche salarié

code :

nom :

prénom :

Date de naissance :

Date d'embauche :

Fonction :

Taux horaire :

Situation familiale :

Nombre d'enfants :

Catégorie :

Service :

Créer

Modifier

Supprimer

Annuler

Valider

Retour

Liste Formations :

Ajouter

Item 1

Item 2

Item 3

Item 4

Item 5

Pour pouvoir aller chercher les formations du salarié j'ai utilisé la méthode qui était déjà présente de base remplirFormulaire()

Ce code est conçu pour gérer l'affichage et la mise à jour des formations d'un salarié dans une interface utilisateur Java Swing. Il utilise une JList pour afficher les formations disponibles et un modèle de liste (DefaultListModel) pour gérer les éléments de cette liste. Voici une explication détaillée des différentes parties du code :

Dans un premier temps après avoir remplis toutes les informations du salarié je fait un modèle de liste (DefaultListModel) est initialisé pour contenir des chaînes de caractères représentant les noms des formations. Ce modèle est ensuite associé à la JList (jListFormations), permettant à cette dernière d'afficher les éléments qu'elle contient.

2024/2025	AP – Projet Rentrée
BTS SIO	Auteur : PORTOLLEAU Anaïs
2SLAM	Date de rédaction : 15/10/2024

```

public void remplirFormulaire() throws SQLException, IOException {
    if (leSalarie == null) {
        System.out.println("Erreur : le salarié est null");
        return;
    }
    jTextFieldCode.setText(e: leSalarie.getCode());
    jTextFieldNom.setText(e: leSalarie.getNom());
    jTextFieldPrenom.setText(e: leSalarie.getPrenom());
    jDateChooserNaissance.setDate(date: leSalarie.getDateNaiss());
    jDateChooserEmbauche.setDate(date: leSalarie.getDateEmbauche());
    jTextFieldFonction.setText(e: leSalarie.getFonction());
    jTextFieldTauxHoraire.setText(e: String.valueOf(d: leSalarie.getTauxHoraire()));
    jTextFieldSituationFamilliale.setText(e: leSalarie.getSituationFamilliale());
    jTextFieldNombreEnfants.setText(e: String.valueOf(i: leSalarie.getNbrEnfants()));
    int idxService = comboBoxModelServices.indexOf(anObject: leSalarie.getService());
    int idxCategorie = comboBoxModelCategories.indexOf(anObject: leSalarie.getCategorie());
    jComboBoxService.setSelectedIndex(anIndex: idxService);
    jComboBoxCategorie.setSelectedIndex(anIndex: idxCategorie);

    DefaultListModel<String> listModel = new DefaultListModel<>();
    jListFormations.setModel(model: listModel); // Définir le modèle de la JList
    try {
        // Récupérer les formations pour le salarié sélectionné
        ArrayList<Formation> formations = DaoFormation.getFormationsBySalarie(codeSalarie: leSalarie.getCode());

        // Vider le modèle avant de remplir pour éviter les doublons ou garder des éléments non souhaités
        listModel.clear();

        // Ajouter chaque formation au modèle de la JList
        if (formations.isEmpty()) {
            // Si aucune formation n'est trouvée, ajouter le message dans la liste
            listModel.addElement("Aucune formation trouvée pour le salarié " + leSalarie.getCode());
        } else {
            // Si des formations existent, supprimer le message "Aucune formation trouvée" s'il a été ajouté auparavant
            listModel.removeElement("Aucune formation trouvée pour le salarié " + leSalarie.getCode());

            // Ajouter les formations trouvées dans la liste
            for (Formation formation : formations) {
                listModel.addElement(element: formation.getNom()); // Ajoute le nom de la formation
            }
        }
    }
}

```

Ensuite j'appelle la méthode `getFormationsBySalarie` du `DAOFormation` est appelée pour récupérer la liste des formations associées au salarié dont le code est spécifié. Cette méthode renvoie une `ArrayList` d'objets `Formation`. Ensuite je vide la liste pour éviter les doublons avant de pouvoir ajouté de nouveaux.

Si la liste est vide cela affiche `Aucune formation` et si ce n'est pas le cas cela affiche le code salarié et si pendant le code si une formation est ajouté cela enlève le message aucune formations.

J'ai utilisé également cette méthode pour remplir les formations :


```
public void remplirComboBoxFormations() throws SQLException, IOException {
    DefaultComboBoxModel<String> comboBoxModel = new DefaultComboBoxModel<>();
    jComboBoxListFormations.setModel(aModel: comboBoxModel); // Définir le modèle de la JComboBox

    try {
        // Récupérer toutes les formations
        ArrayList<Formation> formations = DaoFormation.getAllFormations();

        // Vérifiez si aucune formation n'a été trouvée
        if (formations.isEmpty()) {
            comboBoxModel.addElement(anObject: "Aucune formation disponible.");
        } else {
            // Ajouter chaque formation au modèle de la JComboBox
            for (Formation formation : formations) {
                comboBoxModel.addElement(anObject: formation.getNom()); // Ajoute le nom de la formation
            }
        }
    } catch (SQLException e) {
        System.err.println("Erreur lors de la récupération des formations : " + e.getMessage());
        JOptionPane.showMessageDialog(parentComponent: this, "Erreur lors de la récupération des formations : " + e.getMessage(), title: "Erreur", messageType: JOptionPane.ERROR_MESSAGE);
    }
}
```

Pour que les formations ne soient pas modifiables j'utilise la méthode existante et je mets la jlist à faux :

```
public void modeVisualiser() {

    jTextFieldCode.setEditable(b: false);
    jTextFieldNom.setEditable(b: false);
    jTextFieldPrenom.setEditable(b: false);
    jDateChooserNaissance.setEnabled(enabled: false);
    jDateChooserEmbauche.setEnabled(enabled: false);
    jTextFieldFonction.setEditable(b: false);
    jTextFieldTauxHoraire.setEditable(b: false);
    jTextFieldSituationFamilliale.setEditable(b: false);
    jTextFieldNombreEnfants.setEditable(b: false);
    jListFormations.setEnabled(enabled: false);
    jComboBoxCategorie.setEnabled(b: false);
    jComboBoxService.setEnabled(b: false);
    jButtonValider.setVisible(aFlag: false);
    jButtonCreer.setVisible(aFlag: false);
    jButtonModifier.setVisible(aFlag: false);
    jButtonSupprimer.setVisible(aFlag: false);
    jButtonAnnuler.setVisible(aFlag: false);
    jButtonRetour.setVisible(aFlag: true);
}
```

Quand on teste et qu'on sélectionne un salarié:

InfoWare - SIRH

Liste des salariés par service

Service : *** Tous services ***

Code	Nom	Prenom	Date naiss.	Date emb.	Fonction	Service
77	ff	ff	2024-04-01	2024-04-04	fff	Adminstration
C77	ff	ff	2024-04-01	2024-04-04	fff	Adminstration
C89	ccc	ccc	2024-04-05	2024-04-07	cxccc	Informatique
cc	cc	cc	2024-04-03	2024-04-01	cc	Comptable
S02	BERNARD	Céline	1972-08-14	2015-09-01	Directrice commerciale	Commercial
S04	DOMARD	Marie	1960-06-14	2015-09-01	Directrice générale	Adminstration
S05	LALOIS	Régis	1982-07-25	2015-09-01	Chef comptable	Adminstration
S07	AIAVAR	Karima	1996-11-25	2016-08-01	Standardiste	Adminstration

Supprimer

2024/2025	AP – Projet Rentrée
BTS SIO	Auteur : PORTOLLEAU Anaïs
2SLAM	Date de rédaction : 15/10/2024

Et qu'on fait consulter on a toujours les informations du salarié (déjà présent dans le TP de base) :

Fiche salarié

code : S02

nom : BERNARD

Date de naissance : 14 août 1972

Fonction : Directrice commerciale

Situation familiale : Marié.e

Catégorie : Cadre supérieur

prénom : Céline

Date d'embauche : 1 sept

Taux horaire : 32.5

Nombre d'enfants : 2

Service : Comm

Liste Formations : Java débutant

Ajouter

Java débutant

La conduite d'un p

Relations clientèl

Fixer des objectifs

A droite on voit bien les formations déjà suivies par le salarié et dans le cas où le salarié n' a pas de formation cela donne ceci :

Fiche salarié

code : CC

nom : CC

Date de naissance : 3 avr. 2024

Fonction : CC

Situation familiale : ddd

Catégorie : Employé niveau 1

prénom : CC

Date d'embauche : 1 avr. 2024

Taux horaire : 4.0

Nombre d'enfants : 4

Service : Comptable

Liste Formations : Java débutant

Ajouter

Aucune formation trouvée pour le salarié cc

Et si on lui ajoute une formation cela enlève bien Aucune formations :

2024/2025	AP – Projet Rentrée
BTS SIO 2SLAM	Auteur : PORTOLLEAU Anaïs
	Date de rédaction : 15/10/2024

Fiche salarié

code : cc

nom : cc

Date de naissance : 3 avr. 2024

Fonction : cc

Situation familiale : ddd

Catégorie : Employé niveau 1

prénom : cc

Date d'embauche : 1 avr. 2024

Taux horaire : 4.0

Nombre d'enfants : 4

Service : Comptable

Liste Formations : Java initié

Ajouter

Java initié

On voit une liste déroulante qui est proposé et qui permet de choisir entre toutes les formations, une d’entre elles à ajouter au salarié :

Administration d'un réseau local sous Linux

Ajouter

Java débutant
La conduite d'un p
Relations clientèle
Fixer des objectifs
Administration d'un

On peut bien ajouter de nouvelles formations comme on peut le voir ci-dessus :

Et j’ai ajouté une condition pour éviter les doublons et si on choisit une formation déjà existante cela fait une erreur et empêche l’ajout :

tion familiale : Marié.e

Catégorie : Cadre supér

Alerte

La formation est déjà dans la liste.

OK

itions : Administration d'un réseau local sous Linux

Ajouter