

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

Étape n°4 - classes d'accès aux données (DAO)

Le code source est structuré selon ce qu'on nomme un design pattern. Il est architecturé en couches, chaque couche ayant son propre rôle :

Interfaces graphiques : interfaces utilisateurs avec contrôles ;

Classes métiers : structurent les données traitées par le programme (Cf. Diagramme UML des classes métier ci-dessous);

Classes Dao : c'est une couche intermédiaire entre les classes métier et la base de données ; c'est là que sont exécutées les requêtes SQL

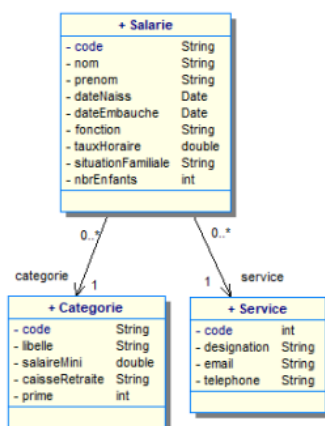
Pour chaque classe métier, il existe une classe DAO

Les méthodes qui exécutent un SELECT transforment les données récupérées depuis la base de données en objets métier ;

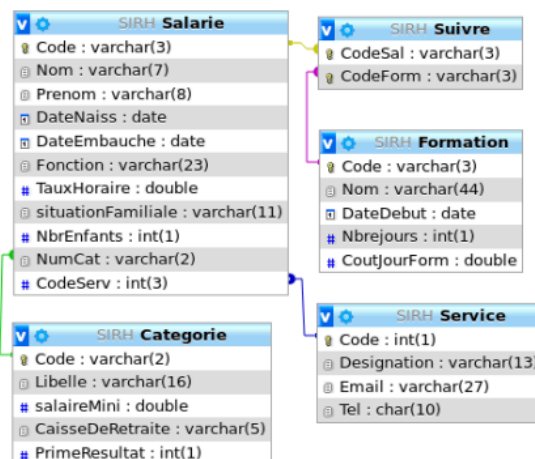
Les méthodes qui exécutent un INSERT ou un UPDATE utilisent les objets métier pour créer ou modifier les enregistrements de la base de données ;

Cette étape consiste à de tester les classes d'accès à la base de données (couche DAO).

Diagramme des classes métier actuel (rappel)



Modèle relationnel de la base de données SIRH (toutes les tables ne sont pas utilisées dans l'application)



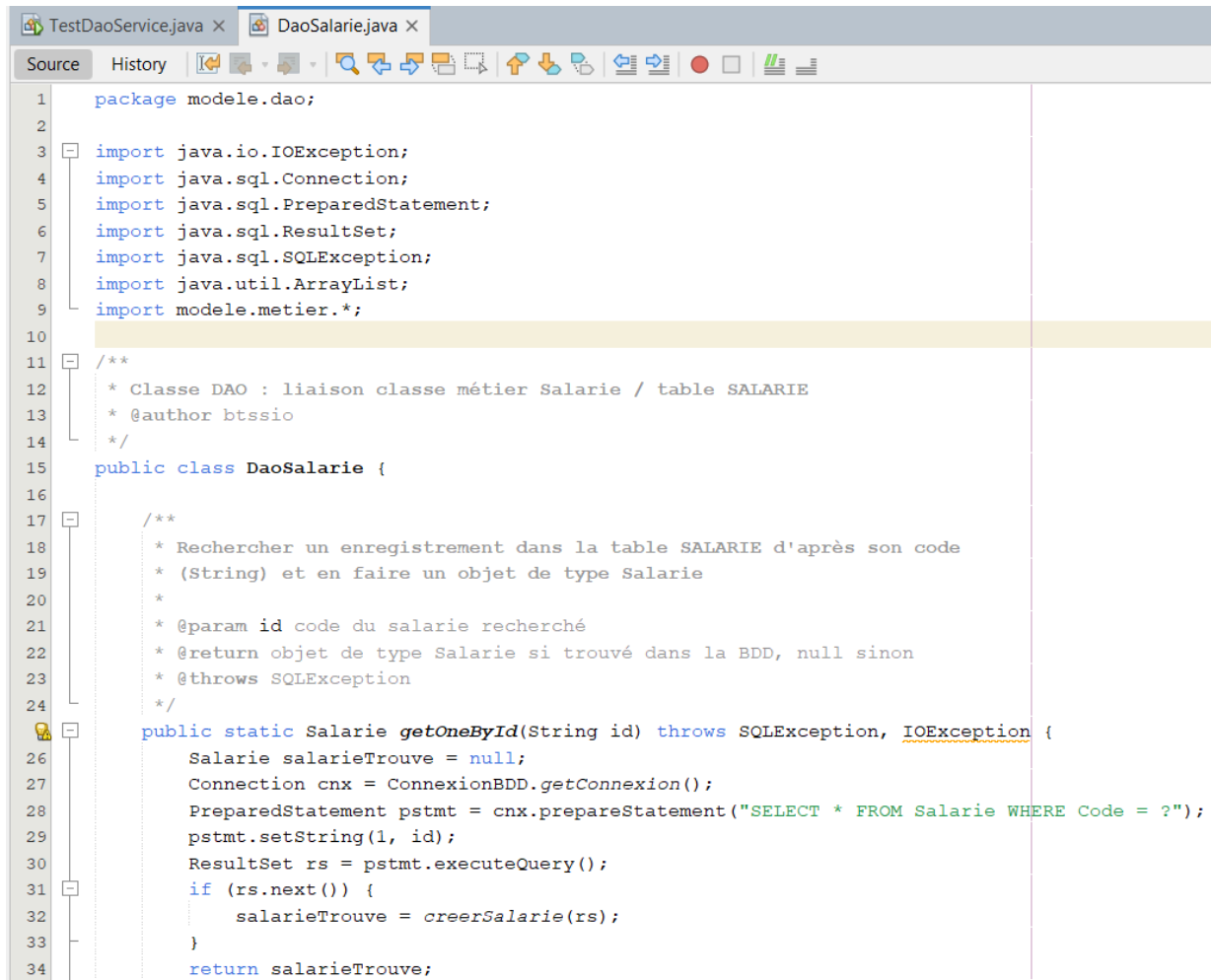
A chaque classe DAO, correspond une classe de tests unitaires montrant le fonctionnement normal de ses méthodes sur un jeu d'essai (paquetage test.dao). Les classes DaoService, DaoSalarie, et TestDaoService existent déjà.

Travail à faire : Testez la classe DaoService. Codez et mettez au point les classes DaoCategorie et TestDaoCategorie. Si nécessaire, adaptez la classe DaoSalarie pour tenir compte de la catégorie du salarié.

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

Codez et mettez au point la classe TestDaoSalarie .

Voici le Code de DaoSalarie :



```

1 package modele.dao;
2
3 import java.io.IOException;
4 import java.sql.Connection;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7 import java.sql.SQLException;
8 import java.util.ArrayList;
9 import modele.metier.*;
10
11 /**
12  * Classe DAO : liaison classe métier Salarie / table SALARIE
13  * @author btssio
14  */
15 public class DaoSalarie {
16
17     /**
18      * Rechercher un enregistrement dans la table SALARIE d'après son code
19      * (String) et en faire un objet de type Salarie
20      *
21      * @param id code du salaire recherché
22      * @return objet de type Salarie si trouvé dans la BDD, null sinon
23      * @throws SQLException
24      */
25     public static Salarie getOneById(String id) throws SQLException, IOException {
26         Salarie salarieTrouve = null;
27         Connection cnx = ConnexionBDD.getConnexion();
28         PreparedStatement pstmt = cnx.prepareStatement("SELECT * FROM Salarie WHERE Code = ?");
29         pstmt.setString(1, id);
30         ResultSet rs = pstmt.executeQuery();
31         if (rs.next()) {
32             salarieTrouve = creerSalarie(rs);
33         }
34         return salarieTrouve;
35     }
36 }

```

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

```

/**
 * Extraire l'ensemble des enregistrements de la table SALARIE
 *
 * @return liste d'objets de type Salarie
 * @throws SQLException
 */
public static ArrayList<Salarie> getAll() throws SQLException, IOException {
    ArrayList<Salarie> lesSalariesTrouves = new ArrayList<>();
    Connection cnx = ConnexionBDD.getConnexion();
    PreparedStatement pstmt = cnx.prepareStatement("SELECT * FROM Salarie");
    ResultSet rs = pstmt.executeQuery();
    while (rs.next()) {
        Salarie unSalarie = creerSalarie(rs);
        lesSalariesTrouves.add(unSalarie);
    }
    return lesSalariesTrouves;
}

/**
 * Transforme un enregistrement de la table SALARIE en instance de Salarie
 * @param rs jeu d'enregistrements ; l'enregistrement courant est concerné
 * @return instance de Salarie
 * @throws SQLException
 */
private static Salarie creerSalarie(ResultSet rs) throws SQLException, IOException {
    Salarie unSalarie = null;
    // Récupération du service du salarié
    Service unService = DaoService.getOneById(rs.getInt("CodeServ"));
    Categorie uneCategorie = DaoCategorie.getOneById(rs.getString("NumCat"));
    unSalarie = new Salarie(
        rs.getString("Code"),
        rs.getString("Nom"),
        rs.getString("Prenom"),
        (java.util.Date) rs.getDate("DateNaiss"),
        (java.util.Date) rs.getDate("DateEmbauche"),
        rs.getString("Fonction"),
        rs.getDouble("TauxHoraire"),
        rs.getString("situationFamiliale"),
        rs.getInt("NbrEnfants"),
        unService,
        uneCategorie
    );
    return unSalarie;
}

```

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

Ce code Java représente une classe DAO (Data Access Object) pour interagir avec la table SALARIE d'une base de données. Voici une explication fonction par fonction :

`getOneById(String id)`: Cette méthode recherche un enregistrement dans la table SALARIE en fonction de son identifiant (code), passé en paramètre. Elle exécute une requête SQL SELECT avec une clause WHERE pour récupérer l'enregistrement correspondant à l'ID spécifié. Si un enregistrement est trouvé, il est transformé en objet de type Salarie et renvoyé. Sinon, null est renvoyé.

`getAll()`: Cette méthode extrait tous les enregistrements de la table SALARIE. Elle exécute une requête SQL SELECT sans condition pour récupérer tous les enregistrements. Ensuite, elle parcourt le ResultSet résultant, crée un objet Salarie pour chaque enregistrement et les ajoute à une liste. Cette liste est ensuite renvoyée.

`creerSalarie(ResultSet rs)`: Cette méthode privée est utilisée pour transformer un enregistrement de la table SALARIE, représenté par un ResultSet, en instance de l'objet Salarie. Elle récupère les données de l'enregistrement, telles que le code, le nom, le prénom, etc., et crée un nouvel objet Salarie avec ces données.

Voici le test de la classe DaoService :

```

Output - JavaApplication_1Slam_Projet2_Sirh-Infoware (run)

run:

Test 1 : DaoService.getOneById
getConnexion : jdbc:mysql://localhost:3307/SIRH
Service d'id 2 trouvé :
Service{code=2, designation=Administration, email=administration@infoware.com, telephone=0169983210}

Test 2 : DaoService.getAll
Service{code=1, designation=Informatique, email=informatique@infoware.com, telephone=0169983212}
Service{code=2, designation=Administration, email=administration@infoware.com, telephone=0169983210}
Service{code=3, designation=Commercial, email=commercial@infoware.com, telephone=0169983215}
Service{code=4, designation=Comptable, email=comptable@infoware.com, telephone=0169983218}
4 services trouvés

Connexion la BDD fermée
BUILD SUCCESSFUL (total time: 0 seconds)

```

On peut voir que les deux tests ont bien fonctionné, le premier test on arrive bien à se connecter puis il trouve bien l'id 2 et affiche son état. Et dans un deuxième test, il affiche bien tous les états des services trouvés.

Remarques :

Chaque classe DAO fournit (a minima) les méthodes suivantes :

- o une méthode `getOneById` qui retourne un objet métier d'après son identifiant ;
- o une méthode `getAll` qui retourne une collection d'objets métier tirés de l'ensemble des

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

enregistrements de la des table.s correspondante.s.

Ces classes peuvent être complétées en fonction des besoins : vous pourrez ajouter les méthodes qui vous semblent nécessaires à l'occasion des étapes suivantes. Les classes DAO seront regroupées dans un paquetage modele.dao ; leurs noms portent le préfixe « Dao ». Les classes de test unitaire seront situées dans le paquetage test.dao; leurs noms portent le préfixe « Test».

Le code permettant la connexion à la base de données se trouve dans une classe commune à toutes les classes DAO, afin d'éviter de le dupliquer : ConnexionBDD (classe fournie). Les paramètres de connexion sont fixés dans le fichier "sirhJdbc.properties" (fourni). L'application se connecte avec l'utilisateur "sirh_util" qui ne possède de droits que sur ses propres tables.

Bilan de l'étape :

On a pas eu de difficultés particulières sur cette partie. Juste on a passé un peu plus de temps à chercher comment faire.

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 unitaires (trace d'exécution des classes de test commentée) ;
o un bilan de l'étape (fait, non fait, difficultés rencontrées).