IUT de NANTES

TD4 : JDBC
R3.07 SQL dans un langage de programmation

L'objectif de ce TD est d'exécuter les différentes requêtes à partir de **Kotlin** via JDBC.

1- Créer un nouveau projet en ***Kotlin*** sous ***IntelliJ****(Choisir IntelliJ et non Gradle lors de la création du projet)*.

2- Créer un ***package BD*** qui contient la classe SessionOracle(voir fichier sur Madoc) qui nous permet à se connecter à la base de données Oracle. Cette classe contient un attribut de type ***Connection plus le login et mot de passe oracle***

N'oubliez pas de charger les fichiers « jar » qui se trouvent aussi sur madoc. Ajoutez ces fichiers aux librairies de votre projet.

Modules Setting….Dependencies...+...Jars or Directories….(choisir les deux jar et faire apply)

3- Créer un ***package Application*** qui contient la ***function*** main et de tester la connexion à la base de données avec votre login et mot de passe oracle(c.a.d. Créer une instance de la classe sessionOracle). Lancer l'exécution sous **IntelliJ**.

4- Créer un package *Bean*. Dans ce package, vous développez la classe *employe*,qui correspond à la table employe sous oracle et contient ***les mêmes attributs que la table***.

5- Créer un package *DAO*. Ce dernier package correspond à la couche accès aux données. Nous faisons correspondre une classe DAO pour chaque table. Cette classe contient des méthodes qui font appels aux différentes requêtes. Elle contient aussi un attribut ou une instance de *sessionOracle*.

Pour la classe *DAOEmploye*, les méthodes *Create, Update et Delete ont* un paramètre de type ***employe.***

Ecrivez cette classe *avec* les opérations suivantes :

- le constructeur avec un paramètre de type SessionOracle.
- *create*, *delete et update, read(compléter le fichier qui se trouve sur madoc)*

La méthode *read* utilise des requêtes de type ***Statement*** et contient la requête suivante :

Select * from employe;

Les requêtes SQL sont exécutées en kotLin avec ***executeQuery*** pour les *select* et ***executeUpdate*** pour les opérations *update*, *insert* et *delete.*

Des valeurs peuvent être injectés à la requête Sql de la manière suivante en kotlin :

```
val requete: String="INSERT INTO EMPLOYE VALUES($em.nuempl},
'${em.nomempl}',${em.hebdo}, ${em.affect},${em.salaire})"
```

où **em** est **instance** de la classe employé, passée en paramètre à la fonction ***create***.

6- Compléter la foction main qui permet de tester la connexion à la base de données, de créer une ***instance de la classe DAOEmploye et d'exécuter les différentes opérations.***

Afin de vérifier l'exécution des méthodes **Kotlin**(create, update et delete) sur la BD, vous pouvez encapsuler par un appel à la méthode read() avant et après.

8- *Remplacez* les ***« Statement »*** par des ***« PreparedStatement »*** pour les méthodes *Create*, *Update* et D*elete* dans une classe ***DAOEmployeBis.***

Exemples pour les trois méthodes :

- Insert into employe values(?,?,?,?)
- Update employe set nomempl=?, hebdo=?, affect=? Where nuempl=?
- Delete from employe where nuempl=?

Chaque paramètre est initialisé avec les méthodes ***stmt.setInt(position du paramètre, valeur)***.

Exemple pour l'opération *insert* : ***stmt.setInt(1,ee.getNumempl() ;***
stmt.setString(2, ee.getNomempl()), etc....;

9- Créer les classe ***DAOMAJ*** qui font appels aux procédures stockées(du package MAJ). Dans ce cas vous utilisez des **CallableStatement.** Les paramètres de la requête sont identiques au prepareStatement.

La requête SQL est sous forme : ‘’call Nom_Package.Nom_Procedure(?, ?, ?….)”