Compte Rendu du TD4 d'SQL dans un langage de programmation

Alexandre Clénet / Groupe 2

Il existe avec Kotlin 3 types d'appelle de la base de données :

Methode DAOEmploye :

Ici on créer une requête et un statement qu'on exécute directement ensemble

```
class DAOEmploye(val ss: SessionOracle) {
    var session: SessionOracle? = null
    init {
        this.session=ss
    }

    fun read(){


        --var essai = SessionOracle();
        var conn: Connection? = null
        conn= session?.getConnectionOracle()
        val requete: String="SELECT * FROM employe"
        try {
            val stmt: Statement = conn!!.createStatement()-- Création d'une
requete de type Statemen
            val result: ResultSet= stmt.executeQuery(requete) --Le contenu du
select est dans ResultSet

            /* Parcourir le résultat du select avec la fonction next();*/
            while (result!!.next()) {

                -- getting the value of the id column
                val id = result.getInt("nuempl")
                val nom=result.getString("nomempl")
                val hebdo = result.getInt("hebdo")
                val affect = result.getInt("affect")
                val salaire = result.getInt("salaire")
                println("$id $nom $hebdo $affect $salaire")

            }
            result.close()
        }

        catch(e: SQLException){
            e.printStackTrace()
        }
    }
```

```
    fun create(e: employe){
        var conn: Connection? = null
        conn= session?.getConnectionOracle()
        val requete="INSERT INTO employe
values(${e.getNuempl()},'${e.getNomempl()}',${e.getHebdo()},${e.getAffect()},${e.g
etSalaire()})"
        try {
            val stmt: Statement = conn!!.createStatement()-- Création d'une
requete de type Statemen
            val result = stmt.executeUpdate(requete) --Le contenu du select est
dans ResultSet

        }

        catch(e: SQLException){
            print("${e.errorCode} : ${e.message}")
        }
    }
    fun delete(e: employe){
        var conn: Connection? = null
        conn= session?.getConnectionOracle()
        val requete="Delete from employe where nuempl =${e.getNuempl()}"
        try {
            val stmt: Statement = conn!!.createStatement()-- Création d'une
requete de type Statemen
            val result = stmt.executeUpdate(requete) --Le contenu du select est
dans ResultSet

        }
        catch(e: SQLException){
            print("${e.errorCode} : ${e.message}")
        }
    }
    fun update(e: employe){
        var conn: Connection? = null
        conn= session?.getConnectionOracle()
        val requete="Update employe SET nomempl='${e.getNomempl()}'," +
                "hebdo=${e.getHebdo()}," +
                "affect=${e.getAffect()}," +
                "salaire=${e.getSalaire()}" +
                "where nuempl=${e.getNuempl()}"
        try {
            val stmt: Statement = conn!!.createStatement()-- Création d'une
requete de type Statemen
            val result = stmt.executeUpdate(requete) --Le contenu du select est
dans ResultSet

        }
        catch(e: SQLException){
            print("${e.errorCode} : ${e.message}")
        }
    }
```

```
}
```

Methode DAOEmployeBis :

Cette deuxième methode consiste à d'abord préparer la requete ensuite y associé les atributs sql/kotlin puis enfin executer.

```
class DAOEmployeBis(val ss: SessionOracle) {
    var session: SessionOracle? = null
    init {
        this.session=ss
    }

    fun read(){


        --var essai = SessionOracle();
        var conn: Connection? = null
        conn= session?.getConnectionOracle()
        val requete: String="SELECT * FROM employe"
        try {
            val stmt: Statement = conn!!.createStatement()-- Création d'une
requete de type Statemen
            val result: ResultSet= stmt.executeQuery(requete) --Le contenu du
select est dans ResultSet

            /* Parcourir le résultat du select avec la fonction next();*/
            while (result!!.next()) {

                -- getting the value of the id column
                val id = result.getInt("nuempl")
                val nom=result.getString("nomempl")
                val hebdo = result.getInt("hebdo")
                val affect = result.getInt("affect")
                val salaire = result.getInt("salaire")
                println("$id $nom $hebdo $affect $salaire")

            }
            result.close()
        }

        catch(e: SQLException){
            e.printStackTrace()
        }
    }
    fun create(e: employe){
        var conn: Connection? = null
        conn= session?.getConnectionOracle()
        val requete="INSERT INTO employe values(?,?,?,?,?)"
        try {
```

```
            val stmt: PreparedStatement = conn!!.prepareStatement(requete)
            stmt.setInt(1,e.getNuempl())
            stmt.setString(2,e.getNomempl())
            stmt.setInt(3,e.getHebdo())
            stmt.setInt(4,e.getAffect())
            stmt.setInt(5,e.getSalaire())
            val result = stmt.executeUpdate() --Le contenu du select est dans
ResultSet

        }

        catch(e: SQLException){
            print("${e.errorCode} : ${e.message}")
        }
    }
    fun delete(e: employe){
        var conn: Connection? = null
        conn= session?.getConnectionOracle()
        val requete="Delete from employe where nuempl =?"
        try {
            val stmt: PreparedStatement = conn!!.prepareStatement(requete)--
Création d'une requete de type Statemen
            stmt.setInt(1,e.getNuempl())
            val result = stmt.executeUpdate() --Le contenu du select est dans
ResultSet

        }
        catch(e: SQLException){
            print("${e.errorCode} : ${e.message}")
        }
    }
    fun update(e: employe){
        var conn: Connection? = null
        conn= session?.getConnectionOracle()
        val requete="Update employe SET nomempl=?," +
                "hebdo=?," +
                "affect=?," +
                "salaire=?" +
                "where nuempl=?"

        try {
            val stmt: PreparedStatement = conn!!.prepareStatement(requete)--
Création d'une requete de type Statemen
            stmt.setInt(5,e.getNuempl())
            stmt.setString(1,e.getNomempl())
            stmt.setInt(2,e.getHebdo())
            stmt.setInt(3,e.getAffect())
            stmt.setInt(4,e.getSalaire())
            val result = stmt.executeUpdate() --Le contenu du select est dans
ResultSet

        }
        catch(e: SQLException){
            print("${e.errorCode} : ${e.message}")
```

```
        }
    }
}
```

Methode DAOEmployeTer :

Cette troisième méthode consiste a cette fois appeller des Procédures créer au préalable avec Oracle, on appelle la procédure on associe les attributs puis on execute.

```
class DAOEmployeTer(val ss: SessionOracle) {
    var session: SessionOracle? = null
    init {
        this.session=ss
    }

    fun read(){


        --var essai = SessionOracle();
        var conn: Connection? = null
        conn= session?.getConnectionOracle()
        val requete: String="SELECT * FROM employe"
        try {
            val stmt: Statement = conn!!.createStatement()-- Création d'une
requete de type Statemen
            val result: ResultSet= stmt.executeQuery(requete) --Le contenu du
select est dans ResultSet

            /* Parcourir le résultat du select avec la fonction next();*/
            while (result!!.next()) {

                -- getting the value of the id column
                val id = result.getInt("nuempl")
                val nom=result.getString("nomempl")
                val hebdo = result.getInt("hebdo")
                val affect = result.getInt("affect")
                val salaire = result.getInt("salaire")
                println("$id $nom $hebdo $affect $salaire")

            }
            result.close()
        }

        catch(e: SQLException){
            e.printStackTrace()
        }
    }
    fun create(e: employe){
        var conn: Connection? = null
        conn= session?.getConnectionOracle()
        val requete="INSERT INTO employe
```

```
values(${e.getNuempl()},'${e.getNomempl()}',${e.getHebdo()},${e.getAffect()},${e.g
etSalaire()})"
        try {
            val stmt: Statement = conn!!.createStatement()-- Création d'une
requete de type Statemen
            val result = stmt.executeUpdate(requete) --Le contenu du select est
dans ResultSet

        }

        catch(e: SQLException){
            print("${e.errorCode} : ${e.message}")
        }
    }
    fun delete(e: employe){
        var conn: Connection? = null
        conn= session?.getConnectionOracle()
        val requete="Delete from employe where nuempl =${e.getNuempl()}"
        try {
            val stmt: Statement = conn!!.createStatement()-- Création d'une
requete de type Statemen
            val result = stmt.executeUpdate(requete) --Le contenu du select est
dans ResultSet

        }
        catch(e: SQLException){
            print("${e.errorCode} : ${e.message}")
        }
    }
    fun update(e: employe){
        var conn: Connection? = null
        conn= session?.getConnectionOracle()
        val requete="Update employe SET nomempl='${e.getNomempl()}'," +
                "hebdo=${e.getHebdo()}," +
                "affect=${e.getAffect()}," +
                "salaire=${e.getSalaire()}" +
                "where nuempl=${e.getNuempl()}"
        try {
            val stmt: Statement = conn!!.createStatement()-- Création d'une
requete de type Statemen
            val result = stmt.executeUpdate(requete) --Le contenu du select est
dans ResultSet

        }
        catch(e: SQLException){
            print("${e.errorCode} : ${e.message}")
        }
    }


}
```