Compte rendu Situation 4

Sommaire

1- La base : Le Pattern Singleton

Le Pattern Singleton sert à faire la connexion entre la base de données et l'ide qui nous permettra de coder nos classes en Java, il permet de faire l'instance de toutes nos classes en un seul objet. Il nous permet de gagner un temps et nous fait faire une économie de ligne de code non négligeable.

untitled

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package BDDSingleton;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/**
 *
 * @author Formation
 */
public class BDDSingleton {

    private static final String DB_URL = "jdBc:mysql://localhost:3306/airafpa";
    private static final String DB_JDBC_DRIVER = "com.mysql.jdbc.Driver";
    private static final String DB_USER = "airafpa1";
    private static final String DB_PASSWORD = "blurp31";

    public Connection connect = null;


    // Connection to my database & user

    private BDDSingleton() {

        try {
            Class.forName(BDDSingleton.DB_JDBC_DRIVER);


        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
            System.exit(1);
        }
    }


    public static BDDSingleton getInstance() {
        return BddSingletonHolder.INSTANCE;
    }

    private static class BddSingletonHolder {

```

untitled

```
        private static final BDDSingleton INSTANCE = new BDDSingleton();
    }

    public boolean connect() {

        if (this.connect == null) {


            try {
                this.connect = DriverManager.getConnection(BDDSingleton.DB_URL, BDDSingleton.DB_USER, BDDSingleton.DB_PASSWORD);

            } catch (SQLException ex) {
                ex.printStackTrace();
                return false;
            }
        }else {
                try {

                    Statement st = this.connect.createStatement();
                    String requete = "SELECT 1";
                    ResultSet rs = st.executeQuery(requete);

                } catch (SQLException ex) {
                    ex.printStackTrace();

                    try {
                        this.connect = DriverManager.getConnection(BDDSingleton.DB_URL, BDDSingleton.DB_USER, BDDSingleton.DB_PASSWORD);

                    } catch (SQLException ex1) {
                        ex1.printStackTrace();
                        return false;
                    }

                }

        } return true;

    }

    public Connection getConnectonManager() {
        return this.connect;
    }

}
```

2- Phase 2 : Le Pattern DAO

Le Pattern DAO nous permet nous permet de regrouper l'accès aux données et de faire de répétition de code dans toutes nos classes, il a aussi la praticité lors de l'apport de modification de ne pas avoir à devoir faire la modification dans toutes nos classes.

untitled

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package BDDDAO;

import BDDSingleton.BDDSingleton;
import java.sql.Connection;
import java.util.ArrayList;

/**
 *
 * @author Formation
 */
public abstract class DAO<T,S> {

    protected BDDSingleton bddmanager = null;
     Connection connect = BDDSingleton.getInstance().connect;

     public DAO (){
         this.bddmanager = BDDSingleton.getInstance();
     }

    public abstract T create(T obj);
    // INSERT INTO

    public abstract T update(T obj);
    // Modify


    public abstract void delete(S id);
    // supress


    public abstract T find(S id);
    // Select

    public abstract boolean isValid(T obj);
    // just in case to test all proprieties


    public abstract ArrayList<T>getAll();
    // Test the id if it exist
}
```

3- Classe de référence : Access_Site

Nous trouvons ci-dessous notre classe de base avec ses getters et setters, qui vont nous servir à construire notre classeDAO.

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package BDDAIRAFPA;

import java.util.Objects;

/**
 *
 * @author Formation
 */
public class AccessSite {

    private long user_id = -1;
    private String nickname;
    private String password;

    public AccessSite(long user_id, String nickname, String password) {
        this.user_id = user_id;
        this.nickname = nickname;
        this.password = password;
    }

    // Empty constructor

    public AccessSite (){

    }

    // Getters & Setters

    public long getUser_id() {
        return user_id;
    }

    public void setUser_id(long user_id) {
        this.user_id = user_id;
    }

    public String getNickname() {
        return nickname;
    }

    public void setNickname(String nickname) {
        this.nickname = nickname;
    }
```

```
49
50      public String getPassword() {
51          return password;
52      }
53
54      public void setPassword(String password) {
55          this.password = password;
56      }
57
58      // Modified override of my methods
59
60      @Override
61      public String toString() {
62          return "AccessSite{" + "user_id=" + user_id + ", nickname=" + nickname + ", password=" + password + '}';
63      }
64
65
66      @Override
67      public int hashCode() {
68          int hash = 5;
69          hash = 37 * hash + Objects.hashCode(this.user_id);
70          hash = 37 * hash + Objects.hashCode(this.nickname);
71          hash = 37 * hash + Objects.hashCode(this.password);
72          return hash;
73      }
74
75      @Override
76      public boolean equals(Object obj) {
77          if (this == obj) {
78              return true;
79          }
80          if (obj == null) {
81              return false;
82          }
83          if (getClass() != obj.getClass()) {
84              return false;
85          }
86          final AccessSite other = (AccessSite) obj;
87          if (this.user_id != other.user_id) {
88              return false;
89          }
90          if (!Objects.equals(this.nickname, other.nickname)) {
91              return false;
92          }
93          if (!Objects.equals(this.password, other.password)) {
94              return false;
95          }
96          return true;
97      }
98
99
100 }
101
```

4- Classe DAO : Access_Site_DAO

Voici ci-dessous notre classeDAO avec son CRUD qui nous servira à faire des insertions, modification ou suppressions dans notre base de données.

untitled

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package BDDDAO;

import BDDAIRAFPA.AccessSite;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

/**
 *
 * @author Formation
 */
public class AccessSiteDAO extends DAO<AccessSite, Long> {

    public AccessSiteDAO() {
        super();
    }

    //Override of my methods and application of my CRUD

    @Override
    public AccessSite create(AccessSite obj) {

        AccessSite accessite = new AccessSite();

        if(this.bddmanager.connect()) {
        try {


            PreparedStatement createst = this.bddmanager.getConnectonManager().prepareStatement("INSERT INTO Access_Site values(?,?,?)");


            createst.setLong(1, obj.getUser_id());
            createst.setString(2, obj.getNickname());
            createst.setString(3, obj.getPassword());
            createst.executeUpdate();

            accessite = this.find(obj.getUser_id());

        } catch (SQLException ex) {
            ex.printStackTrace();

```

```
49
50          }
51        }return accessite;
52      }
53      @Override
54      public AccessSite update(AccessSite obj) {
55          AccessSite accessite = new AccessSite();
56
57          try {
58
59              PreparedStatement updeatest = this.connect.prepareStatement("UPDATE Access_Site SET nickname = ?, password = ? WHERE user_id = ?");
60
61              updeatest.setLong(3, obj.getUser_id());
62              updeatest.setString(1, obj.getNickname());
63              updeatest.setString(2, obj.getPassword());
64              updeatest.executeUpdate();
65
66              accessite = this.find(obj.getUser_id());
67
68          } catch (SQLException ex) {
69              ex.printStackTrace();
70
71          }
72
73          return accessite;
74
75      }
76
77      @Override
78      public AccessSite find(Long id) {
79          AccessSite accessite = new AccessSite();
80
81          if (this.bddmanager.connect()){
82
83              try {
84              Statement st;
85              st = this.bddmanager.getConnectonManager().createStatement();
86              String requete = "SELECT * FROM access_site WHERE user_id = " + id;
87
88              ResultSet rs = st.executeQuery(requete);
89              if(rs.next()){
90                  accessite.setUser_id(rs.getInt("user_id"));
91                  accessite.setNickname(rs.getString("nickname"));
92                  accessite.setPassword(rs.getString("password"));
93              };
94
95
96              } catch (SQLException ex) {
```

```
95
96              } catch (SQLException ex) {
97              ex.printStackTrace();
98              }
99
100         } else {
101             return accessite;
102
103         }   return accessite;
104     }
105
106
107     @Override
108     public void delete(Long id) {
109
110         try {
111
112             PreparedStatement deletest = this.connect.prepareStatement("DELETE FROM access_site WHERE user_id = " + id);
113             deletest.executeUpdate();
114
115         } catch (SQLException ex) {
116             ex.printStackTrace();
117         }
118
119     }
120
121     @Override
122     public boolean isValid(AccessSite obj) {
123         if(obj.getUser_id() == -1 || obj.getNickname() == null || obj.getPassword() == null){
124             return false;
125         }
126         return true;
127     }
128
129     @Override
130     public ArrayList<AccessSite> getAll() {
131
132     ArrayList<AccessSite> listeAccessSite = new ArrayList<>();
133
134         if(this.bddmanager.connect()) {
135
136             try {
137                 Statement st = this.bddmanager.getConnectonManager().createStatement();
138                 String requete = "SELECT * FROM Access_Site";
139                 ResultSet rs = st.executeQuery(requete);
140
141                 while(rs.next()){
142                     AccessSite accessite = new AccessSite(rs.getInt("user_id"), rs.getString("nickname"), rs.getString("password"));
143                     listeAccessSite.add(accessite);
144                 }
145
146                 } catch (SQLException ex) {
147                 ex.printStackTrace();
148                 return listeAccessSite;
149              }
150         } else {
151             return listeAccessSite;
152
153         }return listeAccessSite;
154     }
155
156 }
157
```

5- Phase finale de Test : Access_Site_Dao_Test

Voici ci-dessous notre phase finale de test, cette classe teste nos méthodes CRUD savoir si elles sont fonctionnelles ou s'il y a encore des soucis. Une fois la phase de test validée, le programme est validé comme fonctionnel à 100%.

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package BDDDAO;

import BDDAIRAFPA.AccessSite;
import java.util.ArrayList;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 *
 * @author Formation
 */
public class AccessSiteDAOTest {

    public AccessSiteDAOTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }

```

```
42      /**
43       * Test of create method, of class AccessSiteDAO.
44       */
45      @Test
46      public void testCreate() {
47          System.out.println("create");
48          AccessSite accessite = new AccessSite(97, "Komaji", "blurps31");
49          AccessSiteDAO instance = new AccessSiteDAO();
50          String result = instance.create(accessite).toString();
51          String expResult = accessite.toString();
52
53
54          assertEquals(expResult, result);
55      }
56
57      /**
58       * Test of update method, of class AccessSiteDAO.
59       */
60      @Test
61      public void testUpdate() {
62          System.out.println("update");
63          Long keyss = 94;
64          AccessSite obj = new AccessSite(keyss, "toto", "password");
65          AccessSiteDAO instance = new AccessSiteDAO();
66          String unexpResult = instance.find(keyss).toString();
67          String result = instance.update(obj).toString();
68          assertNotEquals(unexpResult, result);
69      }
70
71      /**
72       * Test of find method, of class AccessSiteDAO.
73       */
74      @Test
75      public void testFind() {
76          System.out.println("find");
77          Long id = 20;
78          AccessSiteDAO instance = new AccessSiteDAO();
79          String expResult = "AccessSite{user_id=20, nickname=accumsan, password=4b6cfa12441197190186 9dc6e1b00e5d3de5f1cc}";
80          String result = instance.find(id).toString();
81          assertEquals(expResult, result);
```

6- Conclusion

Nous avons lors de cette évaluation, mis en pratique toutes nos connaissances théoriques en JDBC qui nous permet de lier notre base de données à notre code en Java afin de pouvoir apporter des modifications directement via notre IDE.

7- Remerciements

Je remercie nos professeurs qui se sont donnés du mal (comme toujours) pour nous fournir une évaluation de qualité, ainsi que pour l'aide qu'ils nous ont apporté dans les moments difficiles de cette évaluation.