Accès aux bases de données

Contexte

Il s'agit du même contexte que dans le TP précédent, il est question cette fois de s'intéresser à la persistance des données : utilisateurs et groupes.
Les données relatives à ces objets seront stockées dans une base de données Mysql.

Contraintes fonctionnelles

L'application Web doit charger à sont démarrage les utilisateurs, puis les groupes, en répartissant les utilisateurs dans leurs groupes respectifs.
A la fermeture de l'application, ou à la demande, les objets (utilisateurs et groupes) sont mis à jour dans la base de données (ajout, modification, suppression).

Contraintes techniques

La base de données sera stockée sur un serveur mysql au format innodb.
Lors de la modification d'un objet (utilisateur ou groupe), celui ci sera tagué par son membre recordStatus, permettant ensuite de déterminer quelle mise à jour devra être effectuée sur l'objet au moment de la sauvegarde dans la base.
Il sera nécessaire d'ajouter un membre **recordStatus** sur la classe Bo, de type **RecordStatus**.
Le type RecordStatus sera définit en tant qu'énumération et proposera les valeurs suivantes :

- rsNone : par défaut, aucune modification
- rsUpdate : objet modifié
- rsNew : objet nouveau
- rsDelete : objet supprimé

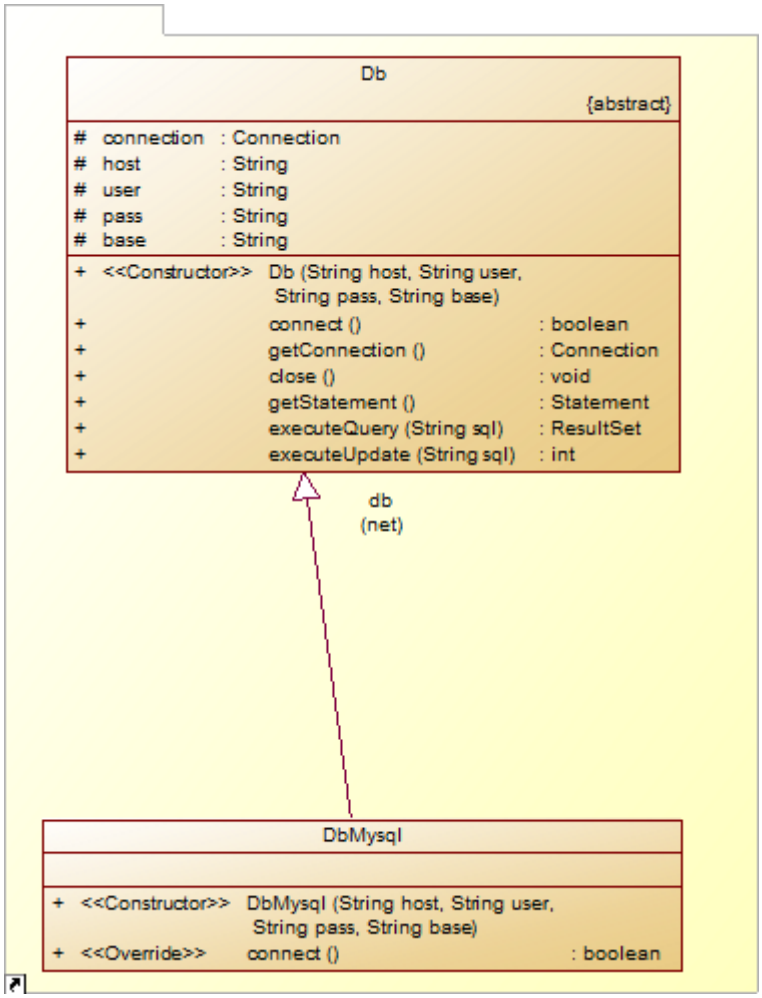
Missions

- Créer la base de données nommée **mUsers**, et saisissez des données de test
- Créer les classes d'accès aux bases de données et à Mysql (**Db** et **DbMysql**)
- Créer l'énumération RecordStatus et ajouter le membre recordStatus dans la classe Bo, mettre à jour les méthodes devant modifier le recordStatus
- Créer et implémenter les méthodes de la classe **DbGateway**
- Intégrer la persistance des données dans le projet web, au chargement et à la fermeture de l'application, et à la demande.

Annexes

Utiliser Accès aux bases de données

Classes d'accès aux données



|h Db.java

```
package net.db;

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

/**
 * Représente une connexion à une base de données
 * @author jc
 *
 */
public abstract class Db {
    protected Connection connection;
    protected String host;
    protected String user;
    protected String pass;
    protected String base;

    /**
     * Constructeur
     * @param host url du serveur de base de données
     * @param user nom d'utilisateur
     * @param pass mot de passe
     * @param base base de données
```

```
     */
    public Db(String host, String user, String pass, String base) {
        super();
        this.host = host;
        this.user = user;
        this.pass = pass;
        this.base = base;
        connection=null;
    }

    /**
     * Etablit la connexion à la base de données
     * @return vrai si la connexion est établie
     */
    public abstract boolean connect();

    /**
     * Retourne l'objet connection
     * @return connection
     */
    public Connection getConnection() {
        return connection;
    }

    /**
     * Ferme la connexion à la base
     * @throws SQLException
     */
    public void close() throws SQLException{
        // TODO à implémenter;
    }

    /**
     * Retourne un statement sur la connexion en cours
     * @return statement
     * @throws SQLException
     */
    public Statement getStatement() throws SQLException{
        // TODO à implémenter;
    }

    /**
     * Exécute une instruction sql renvoyant un résultat
     * @param sql Instruction à exécuter
     * @return resultSet
     * @throws SQLException
     */
    public ResultSet executeQuery(String sql) throws SQLException{
        // TODO à implémenter;
    }

    /**
     * Exécute une instruction sql de mise à jour ne renvoyant pas de résultat
     * @param sql
     * @return le dernier autoIncrément généré en cas d'insertion
     * @throws SQLException
```

```
     */
    public int executeUpdate(String sql) throws SQLException{
        // TODO à implémenter;
    }
}
```

|h DbMysql.java

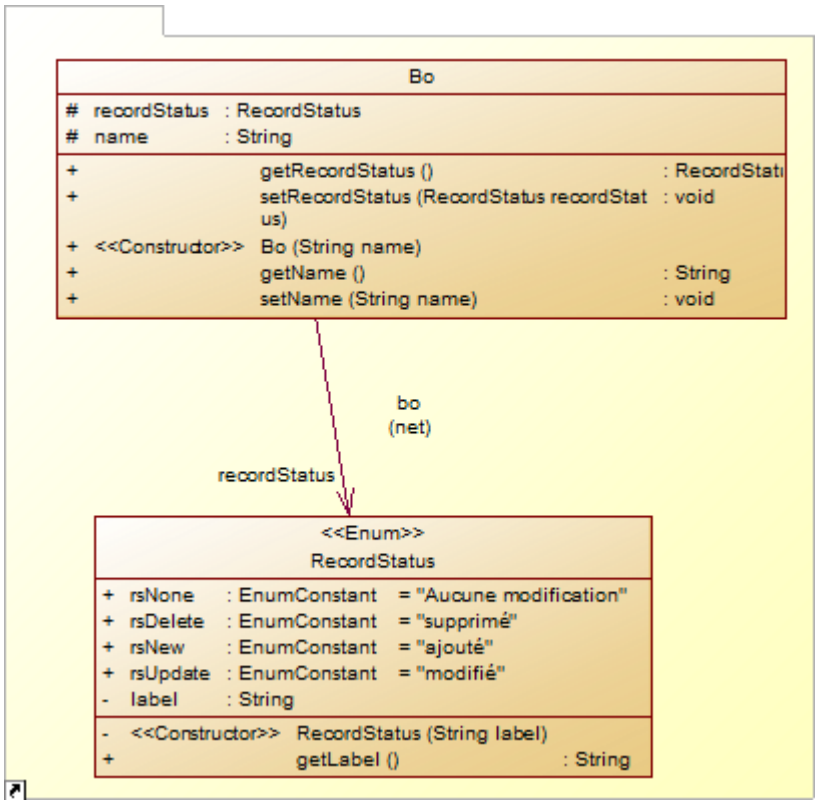
```
package net.db;

/**
 * Représente une connexion à une base de données Mysql
 * @author jc
 *
 */
public class DbMysql extends Db {

    public DbMysql(String host, String user, String pass, String base) {
        super(host, user, pass, base);
    }

    @Override
    public boolean connect() {
        return false;
    }
}
```

Enumération RecordStatus

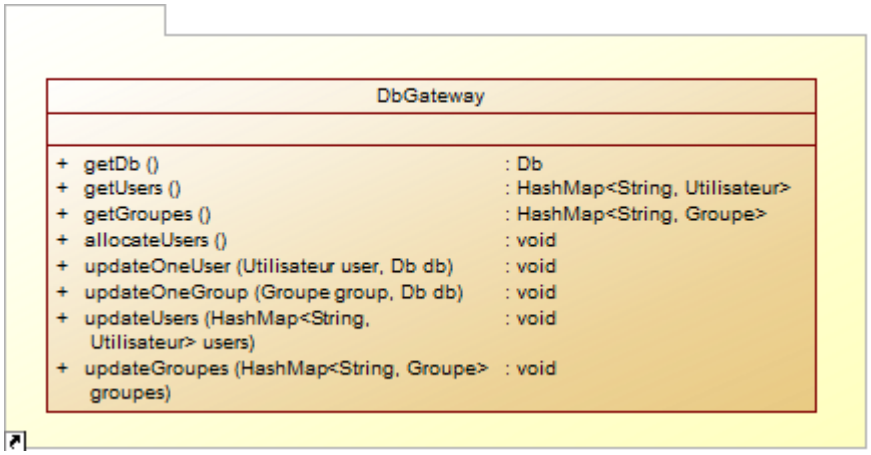


|h RecordStatus.java

```
package net.bo;

/**
 * Enumération des statuts d'un objet permettant ensuite sa sérialistion en
BDD
 * @author jc
 *
 */
public enum RecordStatus {
    rsNone(""),rsDelete("supprimé"),rsNew("ajouté"),rsUpdate("modifié");
    private String label;
    private RecordStatus(String label){
        this.label=label;
    }
    public String getLabel() {
        return label;
    }
}
```

Classe passerelle



|h DbGateway.java

```
package net.technics;

import java.util.HashMap;

import net.bo.Groupe;
import net.bo.Utilisateur;
import net.db.Db;

/**
 * Classe passerelle entre le modèle objet et la base de données
 * @author jc
 *
 */
public class DbGateway {
```

```
    /**
     * Retourne une connexion disponible à la base de données
     * @return Connexion disponible à la BDD
     */
    public static Db getDb(){
        // TODO à implémenter;
        return null;
    }
    /**
     * charge les utilisateurs depuis la connection à la base de données de
l'application
     * @return Hashmap des utilisateurs
     */
    public static HashMap<String, Utilisateur> getUsers(){
        // TODO à implémenter;
        return null;
    }
    /**
     * charge les groupes depuis la connection à la base de données de
l'application
     * @return Hashmap des groupes
     */
    public static HashMap<String, Groupe> getGroupes(){
        // TODO à implémenter;
        return null;
    }

    /**
     * Répartit les utilisateurs dans leur groupe, et remplit les groupes
d'utilisateurs
     */
    public static void allocateUsers(){
        // TODO à implémenter;
    }

    /**
     * Met à jour dans la base de données db l'utilisateur passé en paramètre
     * en utilisant son membre recordStatus
     * @param user utilisateur à mettre à jour
     * @param db connexion à la base de données
     */
    public static void updateOneUser(Utilisateur user,Db db){
        // TODO à implémenter;
    }

    /**
     * Met à jour dans la base de données db le groupe passé en paramètre
     * en utilisant son membre recordStatus
     * @param group groupe à mettre à jour
     * @param db connexion à la base de données
     */
    public static void updateOneGroup(Groupe group,Db db){
        // TODO à implémenter;
    }
    /**
     * Met à jour dans la base de données la hashMap des utilisateurs passée
```

```
en paramètre
     * en utilisant le membre recordStatus de chaque objet
     * @param users hashMap des utilisateurs
     */
    public static void updateUsers(HashMap<String, Utilisateur> users){
        // TODO à implémenter;
    }

    /**
     * Met à jour dans la base de données la hashMap des groupes passée en
paramètre
     * en utilisant le membre recordStatus de chaque objet
     * @param groupes hashMap des groupes
     */
    public static void updateGroupes(HashMap<String, Groupe> groupes){
        // TODO à implémenter;
    }
}
```

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/slam4/tp3?rev=1348563751

Last update: **2019/08/31 14:38**