

TD 3 : Implémentation des Web Services MySQL

Dans ce TD, nous allons nous intéresser à terminer les services de **création d'utilisateurs** et de **login**.

3.1 Rappel de Cours

Question 1. Rappelez les intérêts d'utilisation d'une base de donnée relationnelle.

Question 2. Donnez un exemple de requête SQL.

Question 3. Quel est l'intérêt de JDBC ?

Question 4. Qu'est ce qu'un curseur ?

3.2 Tables SQL

Question 1. Donnez un exemple de programme JAVA pour traiter et afficher les résultats de votre requête en JAVA.

On s'intéresse ici à définir les tables nécessaires pour notre site Web.

Question 2. Comment sera faite la création des tables ?

Question 3. Quelles sont les informations que nous souhaitons stocker ? Quelles seront les informations stockées en MySQL ?

Question 4. Donnez une définition des différentes tables à créer.

Question 5. Comment insérer des valeurs pour tester nos services ?

3.3 MySQL et JAVA/TOMCAT

Question 1. Ecrire la fonction `getMySQLConnection()` permettant de renvoyer une nouvelle connexion à une base des données.

Pour des raisons de concurrences, en TOMCAT, les connections ne sont pas créées manuellement, mais à l'aide d'un `DataSource` (package `javax.sql`).

Le `Datasource` est spécifié ainsi dans un fichier `context.xml` stocké dans le répertoire **META-INF**.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <Resource
    name="jdbc/db" type="javax.sql.DataSource"
    maxActive="100" maxIdle="30" maxWait="10000"
    url="jdbc:mysql://[host]/[database]"
    driverClassName="com.mysql.jdbc.Driver"
    username="[username]" password="[password]"
  />
</Context>
```

La création s'effectue alors à l'aide d'un objet `Database`.

```
package tools;

import java.sql.Connection;
import java.sql.SQLException;

import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;

public class Database {
  private DataSource dataSource;
```

1
2
3
4
5
6
7
8
9
10
11

```

12 public Database(String jndiname) throws SQLException {
13     try {
14         dataSource = (DataSource) new InitialContext().lookup("java:comp/env/" + jndiname);
15     } catch (NamingException e) {
16         // Handle error that it's not configured in JNDI.
17         throw new SQLException(jndiname + " is missing in JNDI! : "+e.getMessage());
18     }
19 }
20
21 public Connection getConnection() throws SQLException {
22     return dataSource.getConnection();
23 }
24
25 }

```

et la création de connexion s'effectue ainsi :

```

1 public static Connection getMySQLConnection() throws SQLException {
2     if (DBStatic.mysql_pooling==false) {
3         return(DriverManager.getConnection("jdbc:mysql://" + DBStatic.mysql_host + "/" +
4             DBStatic.mysql_db, DBStatic.mysql_username, DBStatic.mysql_password));
5     }
6     else {
7         if (database==null) {
8             database=new Database("jdbc/db");
9         }
10        return(database.getConnection());
11    }
12 }

```

3.4 Service de création d'un utilisateur

On rappelle l'organisation du service de création d'un utilisateur :

Fonction : JSONObject createUser(String login, String password, String nom, String prenom)

1. Vérification des paramètres (il ne faut pas que les paramètres valent null).
2. Vérifier si un utilisateur avec le même login existe déjà. Si oui => erreur 1.
3. Ajouter l'utilisateur à la base de données.
4. Renvoyer OK.

Question 1. Ecrire la fonction permettant de vérifier si l'utilisateur existe déjà dans la base de données.

Question 2. Ecrire la fonction permettant d'insérer un utilisateur dans la base de données.

3.5 Service de login

Pour rappel, le service de login s'organise ainsi :

```

1 if ((login==null) || (password==null))
2     return(ServicesTools.error("Wrong Arguments",0));
3
4 try {
5     //Verifie que l'utilisateur existe sinon ERROR 1
6     boolean is_user=AuthenticationTools.userExists(login);
7     if (!is_user) return(ServicesTools.error("Unknown user "+login,1));
8
9     //Verifie que le password et l'utilisateur sont OK sinon ERROR 2
10    boolean password_ok=AuthenticationTools.checkPassword(login,password);
11    if (!password_ok) return(ServicesTools.error("Bad password "+login,2));
12 }

```

```
//Récupère l'id de l'utilisateur
int id_user=AuthenticationTools.getIdUser(login);

JSONObject retour=new JSONObject();
//Insère une nouvelle session dans la base de données
String key=AuthenticationTools.insertSession(id_user,false);
retour.put("key",key);

return(retour);
}
catch(JSONException e) {
    return(ServicesTools.error("JSON Problem "+e.getMessage(),100));
}
catch (SQLException e) {
    return(ServicesTools.error("Problem while generating session key",1000));
}
catch (Exception e) {
    return(ServicesTools.error("Problem...",10000));
}
```

Question 1. Ecrire les fonctions manquantes.