	Ministère de l'Enseignement Supérieur et de la Recherche Scientifique Direction Générale des Etudes Technologiques Institut Supérieur des Etudes Technologiques de Djerba
Module : Atelier Java	
Enseignant : A. ASSES	**Public cible :** L3-MDW ; L3-DSI

TRAVAUX PRATIQUES N°5 : Manipulation d'une base de données en Java : l'API JDBC

Notes de cours :

Pour se connecter à la base de données, on utilise L'API JDBC qui est définie dans le package java. Sql.

Le processus de la connexion à la base de données passe principalement par les étapes suivantes :

1. Charger le driver JDBC
2. Etablir la connexion à la base de données
3. Créer une zone de description de requête
4. Exécuter la requête
5. Traiter les données retournées
6. Fermer les différents espaces

Côté SGBD :

En vérifiant l'installation de WampServer et sous PhpMyAdmin, créez :

- Une nouvelle base de données intitulée 'jdbcDB' comportant une table 'Personne' possédant les colonnes suivantes : 'code, 'nom' et 'prenom' du type String.

Côté environnement de développement Eclipse :

- Sous le package com.isetjb.tpjdbc, créez une nouvelle classe DemoJdbc.
- Téléchargez et ajouter la librairie mysql-connector-java-8.0.11.jar (le driver connecteur Java/MySql) au niveau de « Java Build Path ».
- Editez alors le code suivant comme exemple de connexion à la base de données créée à partir de cette classe Java :

```
package com.isetjb.tpjdbc;
import java.sql.*;
public class DemoJdbc {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		inserePersonne("P1","Ben Salah","Salah");
	}
	public static void inserePersonne(String code, String nom, String prenom) {
		String url = "jdbc:mysql://localhost/jdbcDB";
		String login="root";
		String passwd="";
		Connection cn=null;
		Statement st=null;
		ResultSet rs=null;
		try {
			// Chargement du driver
			Class.forName("com.mysql.jdbc.Driver");

			// Récupération de la connexion
			cn=DriverManager.getConnection(url,login, passwd);
```

```
                // Création d'un statement
                st=cn.createStatement();

                // Exécution des requêtes : Insertion de données
                st.executeUpdate("insert into `Personne` values ('"+code+
                "','"+nom+ "','"+prenom+"')");

                // Récupération et affichage de données
                rs=st.executeQuery("select * from Personne");
                while(rs.next()) {
                        System.out.print(rs.getString("code"));
                        // Passer comme paramètre le nom ou le numéro de colonne
                        System.out.print(" "+rs.getString(2));
                        System.out.println(" "+rs.getString(3));
                }
        }
        catch(SQLException e){
                e.printStackTrace();
        } catch(ClassNotFoundException e) {
                e.printStackTrace();
        }
        finally{
                try {
                        // Libérer les ressources de la mémoire
                        cn.close();
                        st.close();
                        rs.close();
                }
                catch(SQLException e) {
                        e.printStackTrace();
                }
        }
    }
}
```

- Ajouter la table « Ville » au niveau de la base de données contenant les deux colonnes « identifiant » et « nomVille ».
- Faites introduire d'autres requêtes SQL d'insertion de nouvelles données, de modification et de suppression au niveau des deux tables.

Il est à noter qu'à part **Statement** (description d'une requête SQL normale), on peut utiliser un autre type de requête. Il s'agit de **PreparedStatement** : une requête SQL précompilée, qui peut être paramétrée et exécutée plusieurs fois.

A titre d'exemple, ceci peut être traduit comme suit :

```
        PreparedStatement pst = cn.prepareStatement("select * from Personne where
nom=? and prenom=?");
        pst.setString(1, "Ben Salah"); // 1 relatif au premier ?
        pst.setString(2, "Salah"); // 2 relatif au deuxième ?
        rs=pst.executeQuery();
```