# Accès aux bases de données par programmation

## 1. Introduction :

Le but de ce cours est de présenter l'accès aux bases de données dans un cadre d'application et non de langages scripts inclus dans des pages.

## 2. Accès en c++ :

### *2.1 Accès en.net*

Son inconvénient est bien sur d'exiger le **framework** et son avantage est une relative simplicité grâce à l'utilisation d'objets.
La technologie utilisée ici est **ADO** (technologie de connexion microsoft). **ADO** utilise pour former les requêtes SQL, des drivers **ODBC**, **OLEDB** ou autre , par exemple mySql. **ODBC** tend à être abandonné sur les formats de bases de données récents (par exemple access 2007).

**Remarque**s : les classes accédant aux bases de données faisant partie du **framework** sont des classes managées ; il en découle les corollaires suivants :

- Pour créer un objet de telles classes, la syntaxe est la suivante :

```
OleDbConnection ^connexion = gcnew    OleDbConnection(strCon);
```

(à partir de visual 2005 et du framework 3.5 , le **gcnew** remplace le **new** et le caractère **^** remplace * pour faire le distingo entre des classes génériques et des classes managées) .

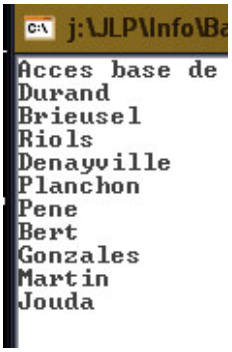
- Si on écrit une classe pour encapsuler ces accès aux bases de données, celle-ci devra elle même être managée, c'est à dire avoir le mot clef **ref** devant le mot clef **class**.

### 2.1.1 Accès par OLEDB :

```
#include <iostream>
using namespace std;
using namespace System;
using namespace System::Data::OleDb;
int main() {
   cout << "Acces base de données access en Oledb" << endl;
   String ^strCon = gcnew String("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=D:/TP/Livre.mdb");
   OleDbConnection ^connexion = gcnew    OleDbConnection(strCon);

   connexion->Open();

   String ^cmdText = gcnew String ("select * from personne;");
   OleDbCommand ^commande = gcnew OleDbCommand(cmdText, connexion );
   OleDbDataReader ^resultat = commande->ExecuteReader();

   while (resultat->Read())  {
       Console::WriteLine (resultat["nom"]);
   }
   resultat->Close();
    connexion->Close();
    return 0;
}
```

### 2.1.2 Accès par Odbc :

2 cas sont possibles, avec et sans DSN. (création d'un **odbc** dans le panneau de configuration).
On peut accéder à toute sorte de base de donnée, y compris mysql.




```
#include <iostream>
using namespace std;
using namespace System;
using namespace System::Data::Odbc;

int main(){
        cout << "Acces base de données access en odbc" << endl;
//      String ^strCon = gcnew String ("DRIVER={Microsoft Access Driver
(*.mdb)};DBQ=D:/TP/Livre.mdb");
        String ^strCon = gcnew String ("DSN=Livre");  // access par dsn

        OdbcConnection ^connexion = gcnew OdbcConnection(strCon);
        connexion->Open();
        String ^cmdText = gcnew String ("select * from personne;");

         OdbcCommand ^commande = gcnew OdbcCommand (cmdText, connexion );
```

```
        OdbcDataReader ^resultat = commande->ExecuteReader();
        while (resultat->Read())      {
                Console::WriteLine (resultat["nom"]);

        }
        resultat->Close();
        connexion->Close();
        return 0;
}
```

### 2.1.3 Accès par Mysql :

Un package doit être téléchargé et installé (**MySql.Data.msi**). Evidement, seules les bases Mysql sont exploitables avec ce driver.

```
#include <iostream>
using namespace std;
using namespace System;

using namespace System::Data::SqlClient;

int main() {
        String ^strCon = gcnew String ("Database=Badgeuse;Data Source=localhost;User Id=root;Password=");

        MySqlConnection ^connexion = gcnew MySqlConnection(connexion);
        String ^ cmdText = gcnew String ("select * from personne;");

        MySqlCommand ^commande = gcnew MySqlCommand(cmdText, connexion );
        MySqlDataReader ^resultat = commande->ExecuteReader();
        while (resultat->Read())      {
                Console::WriteLine (resultat["nom"]);
        }
        resultat->Close();
        connexion->Close();
        return 0;
}
```

## *2.2 Accès par l'API windows (librairie odbc32.lib)*

Accède à n'importe quelle base de donnée sur laquelle un lien **odbc** a été crée.

Nécessite les inclusions :

```
#include <sql.h>
#include <sqlext.h>
#include <sqltypes.h>
```

Toutes les fonctions retournent SQL_SUCCESS en cas de succès

Phase d'initialisation et de connexion :

```
SQLHENV    henv;
SQLHDBC    hdbc;
SQLRETURN  retcode;

Retcode = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
Retcode = SQLSetConnectAttr(henv, SQL_LOGIN_TIMEOUT, (void*)5, 0);
Retcode = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION,(void*)SQL_OV_ODBC3,0);
Retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
Retcode = SQLConnect(hdbc, (SQLCHAR*)LienOdbc,strlen(LienOdbc),
                        (SQLCHAR*)Login, strlen(Login),(SQLCHAR*)Pwd, strlen(Pwd));
```

Exemple d'une requête :

```
SQLHSTMT   hstmt;
Retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);
Retcode = SQLExecDirect(hstmt, (SQLCHAR*)requete, SQL_NTS);
```

Dans le cas d'une sélection (select * from livre where noLivre = 5):

```
double prix, char titre[50] ;
SQLBindCol(hstmt, 2, SQL_C_CHAR, titre,    80, NULL);
SQLBindCol(hstmt, 5, SQL_C_DOUBLE, &prix, sizeof(double), NULL);
```

Dans le cas d'une requête modifiante (insert into emprunt values (8, 2, '01/01/01', null) ; )

Uniquement tester le retour de la fonction **SQLExecDirect**

```
SQLFreeStmt(hstmt, SQL_DROP);
```

Phase de libération  et de déconnexion :

```
SQLDisconnect (hdbc);
SQLFreeConnect(hdbc);
SQLFreeHandle  (SQL_HANDLE_ENV, henv);
```

# 3. Accès en C# :

- La technologie est forcément en .NET , oledb ou odbc.
- Le langage c# est souvent utilisé avec la base de donnée SqlServer.

```
using System;
using System.Data.OleDb;

namespace accesOledb   {
       class Class1        {
              [STAThread]
              static void Main(string[] args)
              {
                     Console.WriteLine ("Acces base de donnees access en Oledb" );
                     OleDbConnection   m_connexion;
                     string  m_strCon = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=J:/JLP/Info/BaseDeDonnees/Badgeuse/Access/Badgeuse.mdb";  // access
                     m_connexion       = new OleDbConnection(m_strCon);
                     m_connexion.Open();
                     string nom, prenom;
                     int noPers = 2;
                     string requete = "select * from personne where noPers=" + noPers;
                     OleDbCommand    m_commande = new OleDbCommand(requete, m_connexion );
                     OleDbDataReader m_resultat = m_commande.ExecuteReader();

                     if (m_resultat.Read())                  {
                            nom        = Convert.ToString (m_resultat["nom"]);
                            prenom     = Convert.ToString (m_resultat["prenom"]);

                            Console.WriteLine ("nom=" + nom + " prenom " + prenom);
                     }
                     Console.Read();
              } // fin main
       } // fin classe
}
```

# 4. Accès en java (JDBC) :

Accède à n'importe quelle base de donnée sur laquelle un lien **odbc** a été crée.

```
package baselivre;
import java.sql.*;

class accesLivre {
  public static void main(String[] args) {
      System.out.println ("Acces base de donnée Livre");
     try {
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); // Charger le pilote JDBC-ODBC
     }
     catch(Exception e){
         System.out.println("exception due au driver(1)");
      }

     String odbc = "jdbc:odbc:Livre";
     Connection connexion = null;
     try {  // Création d'une instance de la classe Connection
         connexion = DriverManager.getConnection(odbc, "", "");
     }
         catch(Exception e){
             System.out.println("exception due au driver(2)");
      }
     String query = "SELECT * FROM livre;";
      ResultSet results = null;

      try {
         Statement stmt = connexion.createStatement();
         results        = stmt.executeQuery(query);
       }
      catch(Exception e){
       System.out.println("exception due a la requete");
      }
      try {
         while (results.next())         {
           String titre = results.getString("titre");
           String auteur = results.getString("auteur");
           int  noLivre = results.getInt("noLiv");
           System.out.println("noLivre " +noLivre + " titre " + titre + " auteur " + auteur);
         }
      }
         catch(Exception e){
         System.out.println("exception du a la lecture");
      }

  } // fin main
}
```

```
C:\JBuilder7\jdk1.3.1\bin\javaw -classpath "J:\JLP\Info\
baselivre.accesLivre
Acces base de donnée Livre
noLivre 1 titre Les Chouans auteur Balzac
noLivre 2 titre Germinal auteur Zola
noLivre 3 titre L'assomoir auteur Zola
noLivre 4 titre La bête humaine auteur Zola
noLivre 5 titre Les misérables auteur Hugo
noLivre 6 titre La peste auteur Camus
noLivre 7 titre Les lettres persanes auteur Montesquieu
noLivre 8 titre Bel ami auteur Maupassant
noLivre 9 titre Les lettres de mon moulin auteur Daudet
noLivre 10 titre Cesar auteur Pagnol
noLivre 11 titre Marius auteur Pagnol
noLivre 12 titre Fanny auteur Pagnol
noLivre 13 titre Les fleurs du mal auteur Baudelaire
```