wxWidgets et MySql sous Windows avec Code::Blocks

Bonjour à tous.

Nous allons voir, dans ce tutoriel, quels sont les outils et les manipulations nécessaires pour obtenir une application wxWidgets qui puisse communiquer avec un serveur MySql sous Windows.

Pour la compilation, nous utiliserons Code::Blocks et MinGW.

1) Préparation et téléchargements

Avant toute chose, vous devez posséder une installation de Code::Blocks et wxWidgets fonctionnelle, c'est-à-dire que vous devez être en mesure de pouvoir compiler une application wxWidgets sous Code::Blocks.

Nous allons dans un premier temps procéder à deux petits téléchargements :

Il vous faut d'abord les libs permettant la communication directe avec le serveur MySql. Le moyen le plus simple de les obtenir est de trouver le DevPak adéquat. Vous aurez ainsi les libs utilisables directement avec MinGW.

Rendez-vous sur le site http://devpaks.org/ dans la rubrique « database », vous trouverez deux DevPaks nommés « libmysql » (un destiné à la version 4 de MySql, l'autre à la version 5). Cliquez sur celui qui vous intéresse, en fonction de la version du serveur MySql avec lequel vous voulez communiquer.

Vous devez normalement arriver sur une page donnant les détails du DevPak correspondant et contenant un lien pour le télécharger.

Enregistrez le fichier obtenu dans un coin de votre disque dur, nous nous en occuperons un peu plus loin.

Pendant que nous sommes dans les téléchargements, nous allons récupérer un petit « Add-On » des libs wxWidgets nommé DatabaseLayer, permettant de faire l'interface entre votre programme et les libs MySql, le tout en utilisant les classes wxWidgets.

Pour cela, rendez-vous sur le site http://wxcode.sourceforge.net qui est un site regroupant un grand nombre de mini-projets relatifs à wxWidgets. Dans la partie de gauche, vous trouverez un lien « Component search » qui vous permettra de rechercher le composant « databaselayer ». Téléchargez le fichier zip contenant les sources de la dernière version, ainsi que celui contenant la documentation, et enregistrez-le au même endroit que le fichier précédent.

Maintenant, nous allons créer l'arborescence destinée à recevoir les fichiers libs et le projet pour compiler le composant « databaselayer ».

Pour ce tutoriel, je vais choisir de placer les libs dans le dossier « ***C:\LibMySql*** ». Je vous laisse bien entendu adapter à votre configuration, sachant qu'il y a malgré tout une règle à respecter : il faut savoir que les chemins contenant des espaces ont tendance à poser problème lors de la compilation et l'édition des liens. Je vous recommande donc d'utiliser un chemin n'en contenant pas.

Donc, dans le dossier « ***C:\LibMySql*** », nous allons créer 6 répertoires :

- le répertoire « ***bin*** » qui va recevoir les fichiers « ***dll*** », et dont il faudra ajouter le chemin à la variable système « PATH ».
- le répertoire « ***include*** » qui va recevoir tous les headers dont l'application finale aura besoin pour être compilée.
- le répertoire « ***lib*** » qui va contenir les fichiers libs pour MinGW (fichiers portant les extensions « ***.a*** » et « ***.def*** »).
- le répertoire « ***tmp*** » dans lequel nous allons décompresser les fichiers téléchargés.
- Le répertoire « ***dblayer*** » qui nous servira à compiler le projet « ***DatabaseLayer*** ».
- Le répertoire « ***doc*** » qui contiendra la documentation du composant « ***DatabaseLayer*** », ainsi que celle présente dans le DevPak mysql.

Lorsque ces répertoires sont créés, placez les deux fichiers téléchargés précédemment dans le répertoire temporaire : « ***C:\LibMySql\tmp*** »

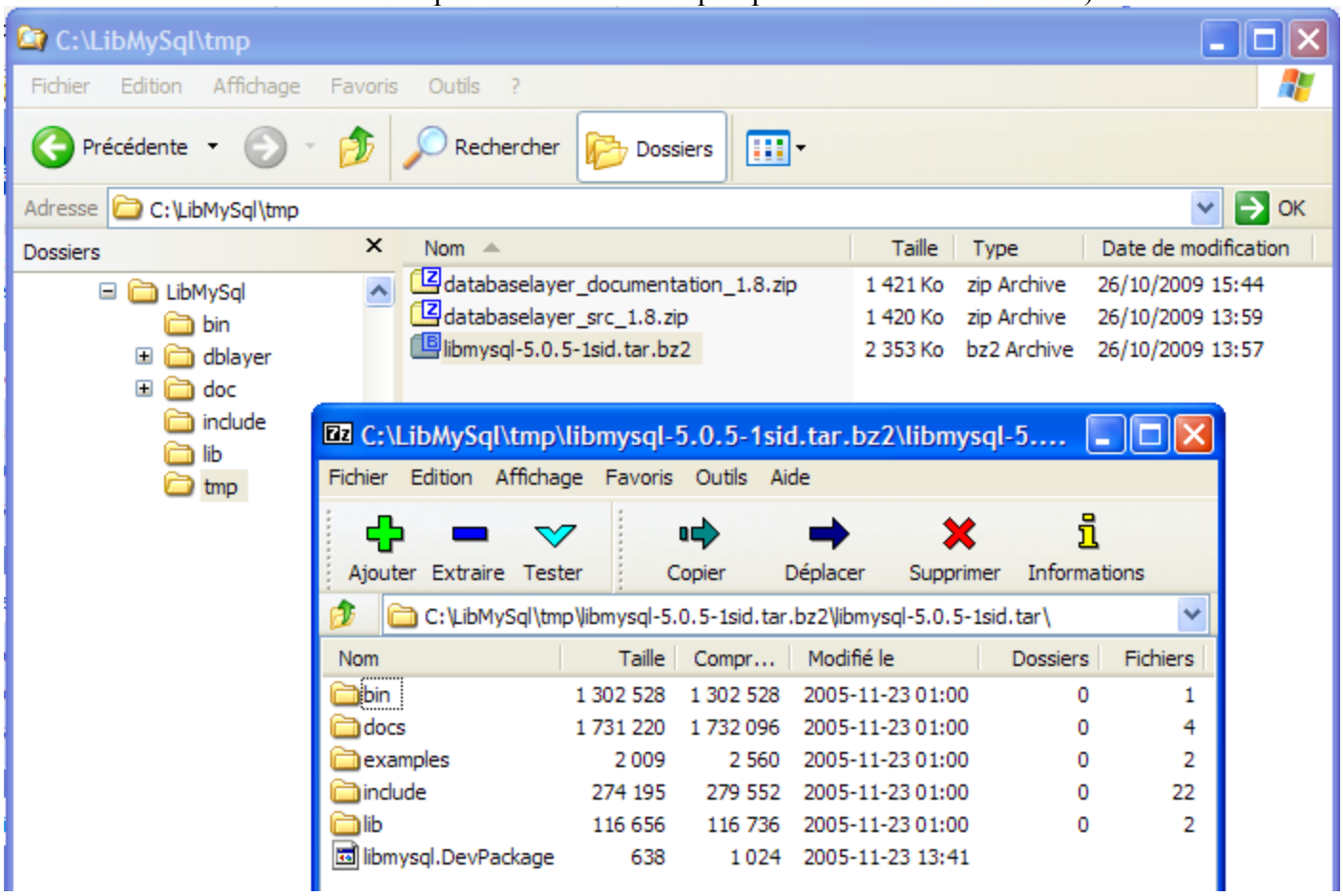
2) Installation du DevPak mysql

Pour ceux d'entre vous qui n'auraient pas jeté un coup d'œil à mon tutoriel sur l'installation des DevPaks wxWidgets pour Code::Blocks, voici quelques explications sur ce format de fichier.

Les fichiers « DevPak » sont en fait des archives « tarball » compressées avec la méthode « bzip2 ».

Il vous suffit donc de renommer votre fichier « ***libmysql-5.0.5-1sid.DevPak*** » (ou ***libmysql-4.1.13a-1sid.DevPak*** si vous avez opté pour la version MySql 4) en « ***libmysql-5.0.5-1sid.tar.bz2*** » (***ou libmysql-4.1.13a-1sid.tar.bz2***).

Vous pouvez désormais ouvrir ce fichier avec WinRar ou 7Zip (avec 7Zip, lorsque vous l'ouvrez, vous avez accès au fichier « .tar » sur lequel il faut double-cliquer pour accéder à son contenu).



Capture N°1 : L'arborescence, le fichier DevPak renommé en « .tar.bz2 » et ouvert avec 7Zip

Une fois ouvert avec votre gestionnaire d'archives favori, double-cliquez sur le répertoire « ***bin*** » présent dans l'archive, afin d'accéder à son contenu. Il doit normalement contenir un fichier « ***libmysql.dll*** ». Faites glisser ce fichier dans le répertoire « ***C:\LibMySql\bin*** ». Double-cliquez ensuite sur les deux points dans l'archive, afin de revenir en arrière (comme sur la capture ci-dessus) et procédez de la même manière pour faire glisser le contenu des dossiers « ***docs*** », « ***include*** » et « ***lib*** » respectivement dans les dossiers « ***C:\LibMySql\doc*** », « ***C:\LibMySql\include*** » et « ***C:\LibMySql\lib*** ».

Vous pouvez maintenant fermer le gestionnaire d'archives, nous avons terminé l'installation du DevPak.

3) La compilation de la partie « MySql » de DatabaseLayer

À l'origine, DatabaseLayer est prévu pour fonctionner avec différents moteurs de bases de données (SQLite, MySql, Oracle, Firebird, ProgresSql, …). Afin de ne pas surcharger ce tutoriel et l'installation en résultant, nous n'allons nous occuper que de la partie de cet « Add-On » concernant MySql, que nous allons compiler avec Code::Blocks.

Il faut donc dans un premier temps ouvrir Code::Blocks et créer un projet vide wxWidgets dans le dossier « ***C:\LibMySql\dblayer*** » que l'on nommera « ***wxMySqlDbLayer*** ». Je vous laisse le soin de sélectionner les options avec lesquelles vous avez l'habitude de programmer (pour ce qui est de choisir entre Ansi et Unicode, et entre Statique et Dynamique). De mon côté, le projet sera compilé avec les options « Unicode » et utilisera les libs dynamiques de wxWidgets. Pensez à bien sélectionner l'option « Créer un projet vide », et à ne pas cocher « Créer et utiliser les headers précompilés ».

Nous obtenons donc un répertoire « ***C:\LibMySql\dbLayer*** » dans lequel se trouve uniquement le fichier projet de Code::Blocks (***wxMySqlDbLayer.cbp***).

Nous allons maintenant ajouter les fichiers sources à notre projet. Ouvrez le fichier « ***databaselayer_src_1.8.zip*** » que nous avons placé dans « ***C:\LibMySql\tmp*** ». Entrez dans le dossier « ***databaselayer*** » au sein de cette archive (double-clic), sélectionnez les dossiers « ***src*** » et « ***include*** » et faites-les glisser dans le dossier de notre projet.

Vous pouvez maintenant fermer votre gestionnaire d'archives. Nous allons faire le ménage dans les fichiers que nous venons d'extraire.

Rendez-vous dans le dossier « ***C:\LibMySql\dblayer\src*** » qui a été créé par les manipulations précédentes et supprimez tous les fichiers relatifs aux types de bases de données qui ne nous intéressent pas :

- ***Firebird*.cpp*** : 8 fichiers
- ***Odbc*.cpp*** : 6 fichiers
- ***Oracle*.cpp*** : 5 fichiers
- ***OTL*.cpp*** : 4 fichiers
- ***Progres*.cpp*** : 8 fichiers
- ***Sqlite*.cpp*** : 4 fichiers
- ***Tds*.cpp*** : 5 fichiers

Placez-vous ensuite dans le dossier « ***C:\LibMySql\dblayer\include*** » et répétez l'opération avec les fichiers « en-tête ». Il faut supprimer :

- ***Firebird*.h*** : 8 fichiers
- ***Odbc*.h*** : 6 fichiers
- ***Oracle*.h*** : 5 fichiers
- ***Otl*.h*** : 5 fichiers
- ***Progres*.h*** : 8 fichiers
- ***Sqlite*.h*** : 4 fichiers
- ***Tds*.h*** : 5 fichiers

Au final, il ne doit rester plus que 15 fichiers « ***cpp*** » dans le dossier « ***src*** » et 19 fichiers « ***.h*** » dans le dossier « ***include*** ».

Nous pouvons maintenant ajouter les fichiers « cpp » au projet Code::Blocks (menu « project », « Add files »).

Avant de lancer la compilation, il faut encore faire quelques réglages afin que Code::Blocks puisse trouver les fichiers « en-têtes » et « lib » de mysql. Il faudra également lui indiquer que l'on veut compiler une bibliothèque, et non un exécutable.

Dans la boîte de propriétés du projet (menu « Project », « Properties »), activez l'onglet « Build Targets », et pour chaque configuration dans la liste de gauche (debug, release), modifiez, à droite, la valeur de la zone déroulante (changez « GUI Application » en « Dynamic library »). Quand c'est fait, fermez la boite de dialogue « Project / Targets options ».

Ouvrez ensuite la boite de dialogue des options de compilation (menu « Project », « Build options »), sélectionnez le projet dans la liste de gauche (de cette façon, les réglages que nous allons faire seront appliqués à chaque configuration).

Pour indiquer à Code::Blocks où se trouvent les fichiers « en-têtes » supplémentaires dont il aura besoin, sélectionnez l'onglet « Search directories », et le sous-onglet « Compiler ». Ajoutez le dossier « ***C:\LibMySql\include*** » à la liste actuelle (bouton « Add »).

Ensuite, faites la même chose pour ajouter le dossier « ***C:\LibMySql\lib*** » au « linker ».

Il reste à indiquer le fait que notre bibliothèque sera liée à la lib « mysql » : dans l'onglet « Linker settings », ajoutez « ***libmysql.a*** » à la liste « link libraries ». Vous pouvez maintenant fermer la boite de dialogue d'options du projet.

Voilà, il ne reste plus qu'à sélectionner la bonne configuration (debug ou release), et à lancer la compilation qui devrait normalement se dérouler sans problème.

La compilation a dû vous créer 3 fichiers dans le dossier « ***C:\LibMySql\dblayer*** » :

- Un fichier « .a » et un « .def » qu'il faut copier dans le dossier « ***C:\LibMySql\lib*** »
- Un fichier « .dll » qu'il faut copier dans le dossier « ***C:\LibMySql\bin*** »

Voilà, il ne reste plus qu'à copier le contenu entier du dossier « ***C:\LibMySql\dblayer\include*** » dans le dossier « ***C:\LibMySql\include*** » afin de regrouper les fichiers « en-têtes » de cette nouvelle lib avec ceux de mysql, et nous avons terminé l'installation des libs.

Pendant que nous y sommes, nous allons mettre en place la documentation de DatabaseLayer.

Ouvrez le fichier « ***databaselayer_documentation_1.8.zip*** », sélectionnez le dossier « ***html*** » présent dans cette archive et faites-le glisser dans le dossier « ***doc*** » de notre arborescence («***C:\LibMySql\doc*** »). Pour la consulter, il suffit d'ouvrir le fichier « ***index.html*** ».

Nous allons maintenant passer à la partie « utilisation de ces libs » dans une petite application.

4) Utilisation des libs créées

Nous allons maintenant créer une petite application wxWidgets qui utilise ces libs toutes fraîches.

Commençons par créer un projet wxWidgets (placez-le dans votre répertoire de développement habituel, ou dans un nouveau sous-dossier « test » de « ***C:\LibMySql*** ». Peu importe le nom que vous donnerez à ce projet, le seul impératif est que vous sélectionniez les mêmes options de compilation que pour la création des libs.

Il faut ensuite renseigner les chemins vers les emplacements de nos libs. Pour cela, reportez-vous à la fin du chapitre précédent, en ajoutant également le fichier « ***.a*** » que nous avons créé à la liste des libs à lier au projet (***libwxMySqlDbLayer.a***).

Ajoutez ensuite un bouton (ou une entrée de menu) à la fenêtre de cette nouvelle application, et associez lui une méthode événementielle afin que nous puissions réaliser les tests.

Je ne vais pas détailler l'ajout de ce bouton, ni la mise en place de la méthode événementielle avec son insertion dans la table des événements de la fenêtre, vous devez normalement être capable de le faire tout seul.

Premier test : récupérer la liste des tables d'une base de données

La première chose que nous allons voir est le fait que notre application communique bien avec le serveur mysql. Pour cela, nous allons simplement récupérer la liste des tables d'une base de données (il faudra bien entendu que ce serveur soit en marche et qu'au moins une base de données ait été créée).

Ce que notre méthode événementielle devra faire :

- Créer un objet de type « MysqlDatabaseLayer » qui va nous servir à communiquer avec le serveur. Lors de la création de cet objet, nous lui donnerons en paramètres l'adresse du serveur, le nom de la base de données, le nom d'utilisateur et le mot de passe à utiliser pour la connexion.
- Créer un objet de type « wxArrayString » qui nous servira à récupérer la liste des tables disponibles.
- Demander la liste des tables
- Afficher une wxMessageBox avec les résultats obtenus.
- Fermer la connexion avec le serveur mysql et détruire l'objet MysqlDatabaseLayer.

Et voici ce que ça donne :

Code wxWidgets :

```
#include "MysqlDatabaseLayer.h"
// Création de l'objet MysqlDatabaseLayer
MysqlDatabaseLayer *dbLayer=new MysqlDatabaseLayer(
    _T("localhost"), // Hôte sur lequel tourne le serveur mysql
    _T("test_db"),   // Nom de la base de données
    _T("user"),      // Nom d'utilisateur
    _T("pwd"));      // Mot de passe
// Création de l'objet wxArrayString pour récupérer les résultats
wxArrayString arrResult;
// Récupération de la liste des tables de la base de données
arrResult=dbLayer->GetTables();
// Préparation du message à afficher
wxString sMsg=_T("Liste des tables :");
for (int i=0;i<(int)arrResult.Count();i++)
{
    sMsg << _T("\n") << arrResult[i];
}
// Affichage du résultat avec une wxMessageBox
wxMessageBox(sMsg);
// Fermeture de la connexion avec le serveur mysql
dbLayer->Close();
// Libération de la mémoire occupée par l'objet MysqlDatabaseLayer
delete dbLayer;
```

Et voilà, c'est aussi simple que cela.

Bien sûr, la simplicité du code est en rapport avec celle de l'action effectuée. Nous verrons plus loin comment effectuer des actions plus complexes sur une base de données mysql.

Une petite précision malgré tout : suivant votre installation, il se peut que le serveur mysql ne soit pas joignable sur le port habituel (3306). Dans ce cas, vous pouvez spécifier le numéro de port à la suite du nom de l'hôte sur lequel tourne ce serveur. Par exemple, si vous avez installé **UsbWebServer** (http://www.usbwebserver.com), mysql utilise par défaut le port 3307. Il aurait dans ce cas fallut mettre « **_T("localhost:3307")** » lors de la construction de l'objet MysqlDatabaseLayer.

Autre petite précision, qui a son importance : par défaut, les erreurs lors de requêtes, ou lors de connexion au serveur, sont gérées avec des exceptions. Comme vous pouvez le constater, le code ci-dessus ne les gère pas, ce qui fait que dans le cas d'un serveur injoignable, vous risquez d'obtenir un joli message d'erreur.

Nous allons donc modifier le code afin d'en tenir compte :

```
#include "MysqlDatabaseLayer.h"
// Création de l'objet MysqlDatabaseLayer
MysqlDatabaseLayer *dbLayer=new MysqlDatabaseLayer();
try
{
    // Ouverture de la connexion avec le serveur
    dbLayer->Open(
    _T("localhost"), // Hôte sur lequel tourne le serveur mysql
    _T("test_db"),   // Nom de la base de données
    _T("user"),      // Nom d'utilisateur
    _T("pwd"));      // Mot de passe
}
catch (DatabaseLayerException& e)
{
    wxString sErr;
    sErr.Printf(_T("Erreur %0d\n%s"), e.GetErrorCode(), e.GetErrorMessage().c_str());
    wxMessageBox(sErr, _T("Exception wxDatabaseLayer"), wxICON_ERROR);
    delete dbLayer;
    return;
}
// Création de l'objet wxArrayString pour récupérer les résultats
wxArrayString arrResult;
// Récupération de la liste des tables de la base de données
try
{
    arrResult=dbLayer->GetTables();
}
catch (DatabaseLayerException& e)
{
    wxString sErr;
    sErr.Printf(_T("Erreur %0d\n%s"), e.GetErrorCode(), e.GetErrorMessage().c_str());
    wxMessageBox(sErr, _T("Exception wxDatabaseLayer"), wxICON_ERROR);
    dbLayer->Close();
    delete dbLayer;
    return;
}
// Préparation du message à afficher
wxString sMsg=_T("Liste des tables :");
for (int i=0;i<(int)arrResult.Count();i++)
{
    sMsg << _T("\n") << arrResult[i];
}
// Affichage du résultat avec une wxMessageBox
wxMessageBox(sMsg);
// Fermeture de la connexion avec le serveur mysql
dbLayer->Close();
// Libération de la mémoire occupée par l'objet MysqlDatabaseLayer
delete dbLayer;
```

Deuxième test : exécuter une requête simple

Nous venons de voir qu'il était très simple de communiquer avec le serveur mysql.

Voyons maintenant comment exécuter une requête, mais nous devons dans un premier temps créer la base de données servant de support à ce tutoriel.

En effet, il n'est pas possible, avec le composant MysqlDatabaseLayer, d'exécuter des actions telles que la création ou la suppression de bases de données. Ce genre d'action est bien entendu possible en utilisant directement l'API mysql, mais nous ne nous étendrons pas sur ce sujet).

Nous allons donc créer une base de données (que nous nommerons « ***wxdb-test*** ») dans laquelle nous allons créer une table « ***utilisateurs*** » possédant les champs « ***id***, ***nom***, ***prenom*** » et nous y insérerons quelques données afin de pouvoir continuer nos tests.

Voici les requêtes MySql à exécuter (depuis phpMyAdmin, par exemple) :

Code MySql

```
CREATE DATABASE `wxdb-test` DEFAULT CHARACTER SET latin1 COLLATE latin1_general_ci;
USE `wxdb-test`;
CREATE TABLE `wxdb-test`.`utilisateurs` (`id` INT NOT NULL AUTO_INCREMENT PRIMARY
KEY, `nom` VARCHAR(20) NOT NULL, `prenom` VARCHAR(20) NOT NULL) ENGINE = MyISAM;
INSERT INTO `utilisateurs` (`nom`, `prenom`) VALUES
('Dupont', 'Jean'),
('Durand', 'Jean'),
('Quiroule', 'Pierre'),
('Hissofesses', 'Paul');
```

Voilà, nous avons maintenant ce qu'il nous faut pour travailler.

La première requête que nous allons exécuter sur cette base de données va nous permettre d'ajouter un enregistrement. Nous n'en attendons aucun résultat, si ce n'est la confirmation que tout s'est bien passé.

La classe « ***MysqlDatabaseLayer*** » possède deux méthodes permettant d'exécuter une requête sur le serveur :

- **RunQuery(const wxString &strQuery, bool bParseQuery)** pour une requête simple, avec pour seule valeur de retour un booléen indiquant si cette exécution de requête s'est bien passée (ne me demandez pas à quoi sert le deuxième paramètre, ce n'en sais rien pour l'instant).
- **RunQueryWithResults(const wString &strQuery)** pour une requête dont on attend une série de résultats en retour.

Vous l'avez sans doute deviné : pour insérer un enregistrement dans la table, nous allons utiliser la première méthode.

Voici donc le code correspondant (je ne rentrerais pas plus dans les détails, car cette opération ne le nécessite pas) :

Code wxWidgets :

```
#include "MysqlDatabaseLayer.h"
// Création de l'objet MysqlDatabaseLayer
MysqlDatabaseLayer *dbLayer=new MysqlDatabaseLayer();
try
{
    // Ouverture de la connexion avec le serveur
    dlLayer->Open(_T("localhost"), _T("wxdb-test"), // Hôte + Nom base de données
                  _T("user"), _T("pwd"));           // Utilisateur + Mot de passe
}
catch (DatabaseLayerException& e)
{
    wxString sErr;
    sErr.Printf(_T("Erreur %0d\n%s"), e.GetErrorCode(), e.GetErrorMessage().c_str());
    wxMessageBox(sErr, _T("Exception wxDatabaseLayer"), wxICON_ERROR);
    delete dbLayer;
    return;
}
bool bRes; // Variable pour le stockage du résultat
// Ajout d'un enregistrement dans la base de données
try
```

```
{
    // Exécution de la requête
    bRes=dbLayer->RunQuery(_T("INSERT INTO `utilisateurs` (`nom`,`prenom`) VALUES
('Dupont','Marie');"), false);
}
catch (DatabaseLayerException& e)
{
    wxString sErr;
    sErr.Printf(_T("Erreur %0d\n%s"), e.GetErrorCode(), e.GetErrorMessage().c_str());
    wxMessageBox(sErr, _T("Exception wxDatabaseLayer"), wxICON_ERROR);
    dbLayer->Close();
    delete dbLayer;
    return;
}
// On informe éventuellement l'utilisateur s'il y a eut un problème
if (bRes==false)
    wxMessageBox(_T("Erreur lors de l'exécution de la requête")) ;
// Fermeture de la connexion avec le serveur mysql
dbLayer->Close();
// Libération de la mémoire occupée par l'objet MysqlDatabaseLayer
delete dbLayer;
```

Vous pouvez compiler et exécuter, il n'y a pas de raison pour que ça ne marche pas : avec PhpMyAdmin, vous devriez être en mesure de voir que notre nouvel enregistrement a bel et bien été ajouté à la table.

Troisième test : exécuter une requête et récupérer un résultat simple

Nous allons maintenant voir comment récupérer un simple résultat (une valeur entière).

Il s'agit cette fois-ci d'obtenir le nombre d'utilisateurs enregistrés dans notre base de données.

La requête sera :

Code MySql

```
SELECT COUNT(1) FROM utilisateurs;
```

Comme cette requête ne fourni qu'un seul résultat (et non une liste de résultats comme c'est fréquemment le cas lors de l'utilisation d'une base de données), nous allons utiliser la méthode ***GetSingleResultInt*** de la classe ***MysqlDatabaseLayer***.

Le premier paramètre à fournir à cette méthode est un ***wxString*** contenant la requête à exécuter.

Le deuxième est le numéro d'index du champ à récupérer. Dans notre cas, le résultat ne contiendra qu'un seul champ. Il faudra donc lui donner la valeur « **1** » pour récupérer le 1er champ.

Ce qui donne comme code :

Code wxWidgets :

```
#include "MysqlDatabaseLayer.h"
// Création de l'objet MysqlDatabaseLayer
MysqlDatabaseLayer *dbLayer=new MysqlDatabaseLayer();
try
{
    // Ouverture de la connexion avec le serveur
    dlLayer->Open(_T("localhost"), _T("wxdb-test"), // Hôte + Nom base de données
                  _T("user"), _T("pwd"));           // Utilisateur + Mot de passe
}
catch (DatabaseLayerException& e)
{
    wxString sErr;
    sErr.Printf(_T("Erreur %0d\n%s"), e.GetErrorCode(), e.GetErrorMessage().c_str());
    wxMessageBox(sErr, _T("Exception wxDatabaseLayer"), wxICON_ERROR);
    delete dbLayer;
    return;
}
int iRes=0; // Variable pour le stockage du résultat
// Lecture du nombre d'utilisateurs enregistrés dans la base de données
try
{
    // Exécution de la requête
    iRes=dbLayer->GetSingleResultInt(_T("SELECT COUNT(1) FROM utilisateurs;"), 1);
}
catch (DatabaseLayerException& e)
```

```
{
    wxString sErr;
    sErr.Printf(_T("Erreur %0d\n%s"), e.GetErrorCode(), e.GetErrorMessage().c_str());
    wxMessageBox(sErr, _T("Exception wxDatabaseLayer"), wxICON_ERROR);
    dbLayer->Close();
    delete dbLayer;
    return;
}
// Affichage du résultat
wxString sMsg;
sMsg.Printf(_T("Il y a %0d entrées dans la table utilisateurs."),iRes);
wxMessageBox(sMsg) ;
// Fermeture de la connexion avec le serveur mysql
dbLayer->Close();
// Libération de la mémoire occupée par l'objet MysqlDatabaseLayer
delete dbLayer;
```

Si vous jetez un coup d'œil à la documentation de la classe **`MysqlDatabaseLayer`** (et surtout à celle de la classe **`DatabaseLayer`** dont elle est dérivée), vous vous apercevrez qu'elle contient plusieurs méthodes de ce style, en fonction du type de valeur à récupérer :

- **`GetSingleResultInt (.....)`** dont nous venons de voir l'utilisation.
- **`GetSingleResultString (.....)`** pour récupérer une chaîne de caractères (un wxString).
- **`GetSingleResultLong (.....)`** pour un entier de type « long ».
- **`GetSingleResultBool (.....)`** pour une valeur booléenne.
- **`GetSingleResultDate (.....)`** pour une valeur de type wxDateTime.
- **`GetSingleResultBlob (.....)`** pour des données binaires.
- **`GetSingleResultDouble (.....)`** pour une valeur décimale de type « double ».

Quatrième test : exécuter une requête et récupérer un résultat plus complexe

Nous allons maintenant voir comment récupérer un résultat plus complexe que précédemment.

Il s'agit cette fois-ci d'obtenir la liste complète des utilisateurs enregistrés dans notre base de données.

La requête sera :

Code MySql

```
SELECT * FROM utilisateurs;
```

Cette requête fourni plusieurs résultats (5 dans notre cas). Nous allons utiliser la classe ***`DatabaseResultSet`*** qui va nous permettre de récupérer ces résultats un par un.

Et pour exécuter la requête, nous utiliserons la méthode ***`RunQueryWithResults`*** de la classe ***`MysqlDatabaseLayer`*** qui justement nous renvoie un pointeur vers un élément ***`DatabaseResultSet.`***

Le seul paramètre à fournir à cette méthode est un ***`wxString`*** contenant la requête à exécuter.

En retour, si le pointeur vaut ***`NULL`***, c'est qu'aucun enregistrement n'a été trouvé.

Dans le cas contraire, il pointera vers un élément vide précédant le premier enregistrement obtenu. Ainsi, il suffira d'appeler la méthode ***`Next()`*** de cette classe pour obtenir le premier élément et les suivants.

Quand tous les éléments nécessaires auront été récupérés, il faudra penser à appeler la méthode ***`Close()`*** pour « fermer » la liste des résultats et libérer la mémoire qu'elle occupe.

Ce qui donne comme code :

Code wxWidgets :

```
#include "MysqlDatabaseLayer.h"
// Création de l'objet MysqlDatabaseLayer
MysqlDatabaseLayer *dbLayer=new MysqlDatabaseLayer();
try
{
    // Ouverture de la connexion avec le serveur
    dlLayer->Open(_T("localhost"), _T("wxdb-test"), // Hôte + Nom base de données
                  _T("user"), _T("pwd"));           // Utilisateur + Mot de passe
}
catch (DatabaseLayerException& e)
{
    wxString sErr;
    sErr.Printf(_T("Erreur %0d\n%s"), e.GetErrorCode(), e.GetErrorMessage().c_str());
```

```
    wxMessageBox(sErr, _T("Exception wxDatabaseLayer"), wxICON_ERROR);
    delete dbLayer;
    return;
}
DatabaseResultSet* rSet=NULL;; // Pointeur pour les résultats
// Obtention des utilisateurs enregistrés dans la base de données
try
{
    // Exécution de la requête
    rSet=dbLayer->RunQueryWithResults(_T("SELECT * FROM utilisateurs;"), 1);
}
catch (DatabaseLayerException& e)
{
    wxString sErr;
    sErr.Printf(_T("Erreur %0d\n%s"), e.GetErrorCode(), e.GetErrorMessage().c_str());
    wxMessageBox(sErr, _T("Exception wxDatabaseLayer"), wxICON_ERROR);
    dbLayer->Close();
    delete dbLayer;
    return;
}
if(rSet==NULL) // Y a-t'il eut des résultats ?
{
    wxMessageBox(_T("Aucun résultat trouvé !"), _T("Résultat"), wxICON_INFORMATION);
} else {
    wxString Nom, Prenom, sRes=_T("Liste des enregistrements :");
    int id;
    while(rSet->Next()) // On itère sur tous les résultats
    {
        // Récupération de l'id (champ n°1)
        id=rSet->GetResultInt(1);
        // Récupération du nom (champ n°2)
        Nom=rSet->GetResultString(2);
        // Récupération du prénom (champ n°3)
        Prenom=rSet->GetResultString(3);
        // Ajout de l'enregistrement aux résultats
        sRes << wxString::Format(_T("\n %02d : %s %s"), id, Nom.c_str(),
Prenom.c_str());
    }
    // On libère la mémoire occupée par les résultats
    dbLayer->CloseResultSet(rSet);
    // Et on les affiche
    wxMessageBox(sRes);
    }
// Fermeture de la connexion avec le serveur mysql
dbLayer->Close();
// Libération de la mémoire occupée par l'objet MysqlDatabaseLayer
delete dbLayer;
```

Voilà, vous disposez maintenant du minimum vital pour développer une application communicant avec un serveur MySql.

Dans suite de ce tutoriel, nous verrons comment préparer une liste de requêtes afin d'en exécuter plusieurs en même temps, ce qui permettra de limiter les accès au serveur, mais il faudra pour l'instant vous contenter de ce que vous venez de lire.

@+, et bonne prog.

Xav'