API mySQL

Table des matières

L'API C fournit un accès de bas niveau pour le protocole client / serveur MySQL et permet aux programmes C d'accéder au contenu de base de données. Le code de l'API C est distribuée avec MySQL et mis en œuvre dans la bibliothèque libmysqlclient.

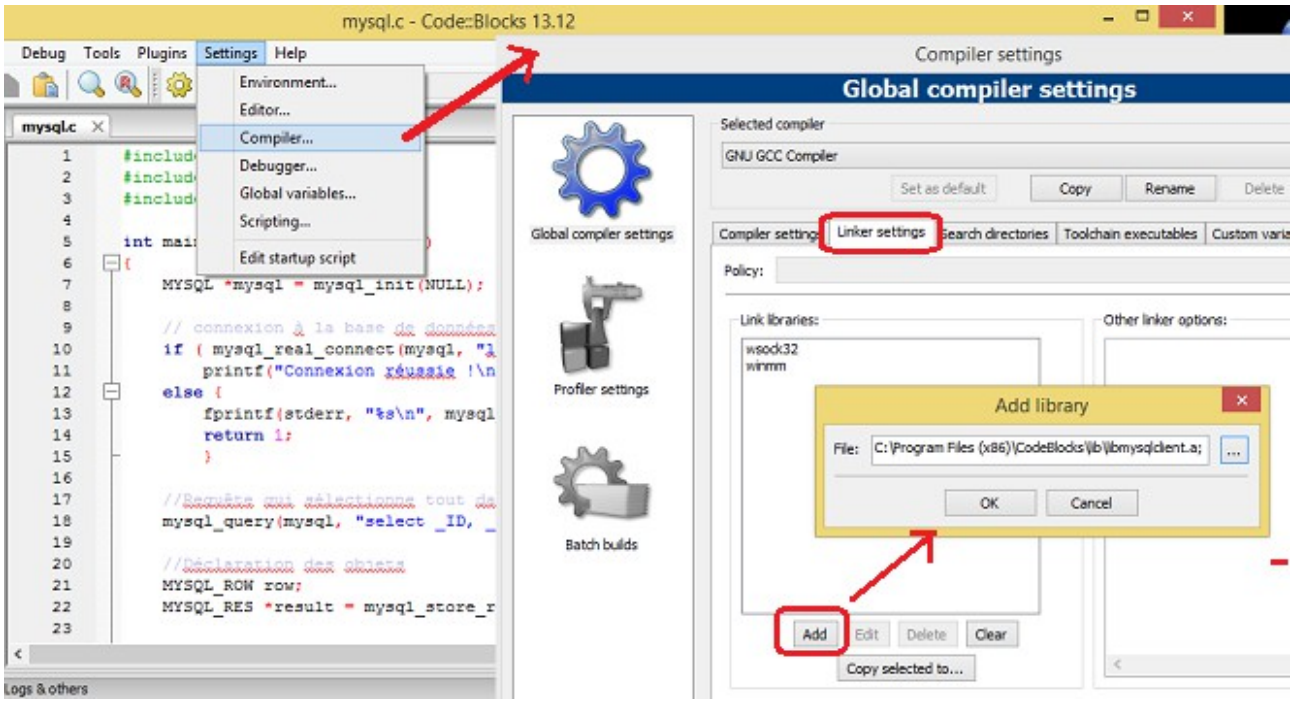
1. Installation

Télécharger ce fichier compressé.

Il contient :

- le fichier libmysql.dll, à mettre dans le dossier de tous vos programmes utilisant l'API MySQL.
- le fichier libmysqlclient.a, à mettre dans le dossier lib du dossier mingw32 de Code::Blocks.
- un dossier MYSQL où se trouvent tous les fichiers d'en-têtes. Copier ce dossier dans le dossier include du dossier mingw32 de Code::Blocks.

Déclarer le fichier libmysqlclient.a dans le linker : menu Settings/Compiler... puis onglet « Linker settings », bouton « Add ».



2. Fonctions principales de l'API

2.1. Connexion/déconnexion

Tout d'abord, créez et configurez un nouveau projet en console.

Pour pouvoir utiliser l'API MySQL, il faut commencer par inclure deux fichiers d'en-têtes :

```
#include <winsock.h>
#include <MYSQL/mysql.h>
```

Avant de pouvoir faire des requêtes dans le programme, il faut d'abord déclarers un objet de type MYSQL, l'initialiser et se connecter à une base de données.

```
MYSQL mysql;              // déclaration
mysql_init(MYSQL *mysql); // initialisation
```

Pour se connecte, nous allons utiliser deux fonctions :

```
mysql_options(MYSQL *mysql, enum mysql_option option, const char *arg);
mysql_real_connect(MYSQL *mysql,
       const char *host, const char *user, const char *passwd, const char *db,
       unsigned int port, const char *unix_socket, unsigned long client_flag);
```

mysql_options sert à spécifier des options de connexion, et mysql_real_connect sert à se connecter à une base de données.

1. mysql_options :
   - Le premier argument est un pointeur de structure, que nous avons vu juste avant.
   - Le deuxième argument est l'option que nous voulons configurer, nous utiliserons toujours MYSQL_READ_DEFAULT_GROUP, qui lie les options spécifiées dans le fichier my.cnf.
   - Enfin, le troisième argument est la valeur de cette option ; vous pouvez mettre le nom que vous souhaitez.
2. mysql_real_connect :
   - Le premier argument est toujours un pointeur vers la structure.
   - Le deuxième argument est le nom de domaine.
   - Le troisième argument est l'identifiant de connexion.
   - Le quatrième argument est le mot de passe.
   - Le cinquième argument est le nom de la base de données.
   - Le sixième argument est le port (0 par défaut).
   - Le septième argument est le socket à utiliser (NULL par défaut).
   - Et le huitième argument est le flag (0 par défaut).

Pour travailler en local (avec EasyPHP par exemple), il faut mettre localhost comme 2ème argument, puis root en identifiant, et le mot de passe.

Remarque : certains hébergeurs bloquent le port pour la connexion, il vous est donc impossible d'utiliser l'API avec ces hébergeurs.

Pour terminer, il faut fermer la connexion MySQL à l'aide de la fonction :

mysql_close(MYSQL *mysql);

Exemple :

```
#include <stdio.h>
#include <winsock.h>
#include <MYSQL/mysql.h>

int main(int argc, char **argv)
{
    MYSQL *mysql = mysql_init(NULL);

    mysql_options(mysql,MYSQL_READ_DEFAULT_GROUP, "option");

    if ( mysql_real_connect(mysql, "localhost", "root", "", "ecoserre", 0, NULL, 0) )
```

```
        printf("Connexion réussie !");
    else
        fprintf(stderr, "%s\n", mysql_error(mysql));

    mysql_close(mysql);

    return 0;
}
```

2.1. Les requêtes n'attendant aucun jeu de résultats

Il s'agit de : INSERT, DELETE, CREATE, UPDATE par exemple qui n'attendent aucune réponse. Au contraire, des requêtes comme SELECT ou SHOW attendent une réponse.

La fonction permettant de faire des requêtes MySQL est la suivante :

```
mysql_query(MYSQL *mysql, const char *query);
```

Il faut spécifier pour cette fonction le pointeur de structures ainsi que la requête. Voici un exemple de son utilisation:

```
mysql_query(&mysql, "INSERT INTO ma_table VALUES('valeur 1', 'valeur 2', 'etc')");
```

Cette requête va enregistrer dans la table ma_table les valeurs indiquées dans l'exemple. Maintenant, pour savoir combien de lignes ont été affectées par la requête, il suffit d'appeler la fonction :

```
mysql_affected_rows(MYSQL *mysql);
```

2.2. Les requêtes attendant un jeu de résultats

Pour la requête en elle même ça ne change pas, il faut toujours utiliser la fonction mysql_query. Cependant, pour récupérer le résultat il y a deux façons de faire :

- mysql_use_result(MYSQL *mysql)
- mysql_store_result(MYSQL *mysql)

2.2.1. mysql_use_result

Cette fonction récupère le jeu de résultat, mais ne l'enregistre pas dans le client. Son principal avantage réside dans sa rapidité et dans sa faible utilisation de la mémoire. Toutefois, vous ne pourrez pas faire d'opérations comme revenir au résultat n°2.

Il faut stocker la valeur de retour de cette fonction dans un pointeur de structure de type MYSQL_RES, qui est fait spécialement pour stocker le jeu de résultats, et appeler une autre fonction :

mysql_fetch_row(MYSQL_RES *result);

La valeur de retour de cette fonction devra quant à elle être stockée dans un objet de type MYSQL_ROW (correspondant généralement à un tableau de chaînes de caractères) mysql_fetch_row stocke les valeurs de la ligne suivante dans le jeu de résultats. Il est nécessaire, de libérer la mémoire allouée par la fonction :

mysql_free_result(MYSQL_RES *result);

Exemple :

```
#include <stdio.h>
```

```c
#include <winsock.h>
#include <MYSQL/mysql.h>

int main(int argc, char **argv)
{
    MYSQL *mysql = mysql_init(NULL);

    // options de connexion
    mysql_options(mysql, MYSQL_READ_DEFAULT_GROUP, "option");

    // connexion à la base de données
    if ( mysql_real_connect(mysql, "localhost", "root", "", "ecoserre", 0, NULL, 0) )
        printf("Connexion réussie !\n");
    else {
        fprintf(stderr, "%s\n", mysql_error(mysql));
        return 1;
        }

    //Requête qui sélectionne tout dans ma table scores
    mysql_query(mysql,
       "select _ID, _ident from actuateurs_i10n where _lang = 'fr' order by _ID");

    //Déclaration des objets
    MYSQL_ROW row;
    MYSQL_RES *result = mysql_use_result(mysql);

    //Tant qu'il y a encore un résultat ...
    while ( (row = mysql_fetch_row(result)) )
        printf("%s : %s\n", row[0], row[1]);

    //Libération du jeu de résultat
    mysql_free_result(result);

    // fermeture
    mysql_close(mysql);

    return 0;
}
```

Ce qui donne :

```
Connexion réussie !
1 : Eclairage
2 : Ventilateur
3 : Pompe à eau
4 : Trappe
```

Le jeu de résultats est mis dans result, puis on fait une boucle pour lister toutes les lignes. Mais cela n'indique toujours pas la valeur de la ligne : deux nouvelles fonctions vont le permettre.

1. `mysql_num_fields(MYSQL_RES *result);`

   Elle retourne un entier non signé qui correspond au nombre de champs de la table sélectionnée. Donc, pour pouvoir avoir les valeurs de la requête, il faut stocker dans une variable le nombre de champs, puis faire une boucle pour avoir chaque valeur, une par une.

2. `mysql_fetch_lengths(MYSQL_RES *result);`

Elle retourne un tableau d'entiers non signés qui correspondent à la taille de la valeur de chaque champ du résultat. Avec ces informations, voici un exemple de requête complète :

```
#include <stdio.h>
#include <winsock.h>
#include <MYSQL/mysql.h>

int main(int argc, char **argv)
{
    MYSQL *mysql = mysql_init(NULL);

    // options de connexion
    mysql_options(mysql, MYSQL_READ_DEFAULT_GROUP, "option");

    // connexion à la base de données
    if ( mysql_real_connect(mysql, "localhost", "root", "", "ecoserre", 0, NULL, 0) )
        printf("Connexion réussie !\n");
    else {
        fprintf(stderr, "%s\n", mysql_error(mysql));
        return 1;
        }

    //Requête qui sélectionne tout dans ma table scores
    mysql_query(mysql,
       "select _ID, _ident from actuateurs_i10n where _lang = 'fr' order by _ID");

    //Déclaration des objets
    MYSQL_ROW row;
    MYSQL_RES *result = mysql_use_result(mysql);

    //On récupère le nombre de champs
    unsigned int i, num_champs = mysql_num_fields(result);

    //Tant qu'il y a encore un résultat ...
    while ( (row = mysql_fetch_row(result)) ) {
        //On déclare un pointeur long non signé pour y stocker la taille des valeurs
        unsigned long *lengths = mysql_fetch_lengths(result);

        //On fait une boucle pour avoir la valeur de chaque champs
        for(i = 0; i < num_champs; i++)
            printf("[%.*s] ", (int) lengths[i], row[i] ? row[i] : "NULL");

        printf("\n");
        }

    //Libération du jeu de résultat
    mysql_free_result(result);

    // fermeture
    mysql_close(mysql);

    return 0;
}
```

Ce qui donne :

```
Connexion réussie !
```

```
[1] [Eclairage]
[2] [Ventilateur]
[3] [Pompe à eau]
[4] [Trappe]
```

2.2.2. mysql_store_result

Le principe est le même, à la différence que cette fonction stocke le jeu de résultat dans une mémoire tampon, ce qui rend possible des éventuelles opérations à partir de ce jeu. Notez néanmoins que la fonction peut se révéler plus lente que mysql_use_result.

Pour réaliser une requête similaire à celle de l'exemple précédent il suffit juste de changer mysql_use_result par mysql_store_result.

La fonction suivante permet d'accéder aux éléments du jeu de résultat :

```
mysql_data_seek(MYSQL_RES *result, long offset);
```

Elle prend en argument le jeu de résultat ainsi que le numéro de la ligne qu'on veut obtenir.

NB : offset est un indice qui comence à 0 !

```
#include <stdio.h>
#include <winsock.h>
#include <MYSQL/mysql.h>

int main(int argc, char **argv)
{
    MYSQL *mysql = mysql_init(NULL);

    // connexion à la base de données
    if ( mysql_real_connect(mysql, "localhost", "root", "", "ecoserre", 0, NULL, 0) )
        printf("Connexion réussie !\n");
    else {
        fprintf(stderr, "%s\n", mysql_error(mysql));
        return 1;
        }

    //Requête qui sélectionne tout dans ma table scores
    mysql_query(mysql,
       "select _ID, _ident from actuateurs_i10n where _lang = 'fr' order by _ID");

    //Déclaration des objets
    MYSQL_ROW row;
    MYSQL_RES *result = mysql_store_result(mysql);

    //On choisit une ligne.
    mysql_data_seek(result, 2);

    //On récupère le nombre de champs
    unsigned int i, num_champs = mysql_num_fields(result);

    //Tant qu'il y a encore un résultat ...
    while ( (row = mysql_fetch_row(result)) ) {
        //On déclare un pointeur long non signé pour y stocker la taille des valeurs
        unsigned long *lengths= mysql_fetch_lengths(result);

        //On fait une boucle pour avoir la valeur de chaque champs
```

```
        for(i = 0; i < num_champs; i++)
            printf("[%.*s] ", (int) lengths[i], row[i] ? row[i] : "NULL");

        printf("\n");
        }

    //Libération du jeu de résultat
    mysql_free_result(result);

    // fermeture
    mysql_close(mysql);

    return 0;
}
```

Ce qui donne :

```
Connexion réussie !
[3] [Pompe à eau]
[4] [Trappe]
```