Brug af MySQL i C++

Denne artikel bygger ovenpå artiklen "MySQL C API" og forklarer hvordan man kan programmere mere objekt orienteret.

Den forudsætter kendskab til C++ og lidt kendskab til SQL og MySQL C API.

Skrevet den **04. Feb 2009** af **arne_v** I kategorien **Programmering / C/C++** | ★★★☆☆

[vigtigt: artikel 282 er samme artikel som denne - der gik
koks i det i forbindelse med de 2 store nedbrud på Eksperten
i maj måned - jeg vil ikke slette en af dem af hensyn til
dem som har "betalt" for artiklen]

Historie:
V1.0 - 01/04/2004 - original
V1.1 - 26/12/2008 - små ændringer

Indledning

Artiklen vil vise 3 forskellige tilgange til brug af MySQL fra C++.

Den bruger de samme eksempler som artiklen MySQL C API.

Der vil kun blive vist build kommandoer for MS VC++. For andre compilere henvises til førnævnte artikel.

Der vil ikke blive vist MFC/ATL ODBC/OLE DB brug.

Fortsæt med MySQL C API

MySQL C API kan udmærket bruges fra C++.

Eksempel:

query2.cpp

```
#include <cstdio>
#include <cstdlib>

#include <iostream>

using namespace std;

#include "mysql.h"

int main()
{
```

```
    //
    // åben connection til:
    //   server = "localhost"
    //   username = "root"
    //   password = ""
    //   database = "Test"
    //   port = 0 (bliver opfattet som default 3306)
    //
    MYSQL *handle= mysql_init(NULL);
    if(handle == NULL)
    {
        cerr << "MySQL error: " << mysql_error(handle) << endl;
        exit(1);
    }
    if(!mysql_real_connect(handle, "localhost", "root", "", "Test", 0, NULL,
0))
    {
        cerr << "MySQL error: " << mysql_error(handle) << endl;
        exit(1);
    }
    // udfør query
    mysql_query(handle, "SELECT * FROM t1");
    MYSQL_RES *result = mysql_store_result(handle);
    // print resultat af query
    int nfields = mysql_num_fields(result);
    MYSQL_ROW row;
    while ((row = mysql_fetch_row(result))) {
        int *l = (int *)mysql_fetch_lengths(result);
        for (int i=0; i<nfields; i++) {
            char *buf = new char[l[i]+1];
            strncpy(buf,row[i] ? row[i] : "NULL",row[i] ? l[i] : 4);
            buf[row[i] ? l[i] : 4] = '\0';
            cout << " " << buf;
            delete[] buf;
        }
        cout << endl;
    }
    // luk query
    mysql_free_result(result);
    // luk connection
    mysql_close(handle);
    return 0;
}
```

insert2.cpp

```
#include <cstdio>
#include <cstdlib>

#include <iostream>
```

```
using namespace std;

#include "mysql.h"

int main()
{
    //
    // åben connection til:
    //   server = "localhost"
    //   username = "root"
    //   password = ""
    //   database = "Test"
    //   port = 0 (bliver opfattet som default 3306)
    //
    MYSQL *handle= mysql_init(NULL);
    if(handle == NULL)
    {
        cerr << "MySQL error: " << mysql_error(handle) << endl;
        exit(1);
    }
    if(!mysql_real_connect(handle, "localhost", "root", "", "Test", 0, NULL,
0))
    {
        cerr << "MySQL error: " << mysql_error(handle) << endl;
        exit(1);
    }
    // indsæt 10 rækker
    for(int i=0; i<10; i++)
    {
        char sqlcmd[200];
        sprintf(sqlcmd, "INSERT INTO t1 VALUES (%d,'%s')", i, "Dette er en
test");
        mysql_query(handle, sqlcmd);
    }
    // luk connection
    mysql_close(handle);
    return 0;
}
```

Build MS VC++ 6:

```
cl /GX /DSOCKET=int /I\mysql\include query2.cpp \mysql\lib\opt\libmysql.lib
cl /GX /DSOCKET=int /I\mysql\include insert2.cpp \mysql\lib\opt\libmysql.lib
```

Fordele:
* stabilt og kendt API

Ulemper:
* ikke særligt objekt orienteret

MySQL++ API

MySQL har også lavet et specielt C++ API.

Idag vedligeholdes det ikke længere af MySQL og skal hentes herfra:
http://tangentsoft.net/mysql++/

Eksempel:

query3.cpp

```
#include <iostream>

using namespace std;

#include "sqlplus.hh"

int main()
{
    try
    {
        //
        // åben connection til:
        //   database = "Test"
        //   server = "localhost"
        //   username = "root"
        //   password = ""
        //
        Connection con("Test", "localhost", "root", "");
        // udfør query
        Query q = con.query();
        q << "SELECT * FROM t1";
        Result res = q.store();
        // print resultat af query
        for (Result::iterator it = res.begin(); it != res.end(); it++)
        {
            Row r = *it;
            for(int i = 0; i < res.columns(); i++)
            {
                cout << " " << (!r[i].is_null() ? r[i] : "NULL");
            }
            cout << endl;
        }
    }
    catch(BadQuery er)
    {
        cerr << "MySQL error: " << er.error << endl;
    }
    return 0;
}
```

insert3.cpp

```
#include <iostream>
#include <sstream>

using namespace std;

#include "sqlplus.hh"

int main()
{
    try
    {
        //
        // åben connection til:
        //   database = "Test"
        //   server = "localhost"
        //   username = "root"
        //   password = ""
        //
        Connection con("Test", "localhost", "root", "");
        // indsæt 10 rækker
        Query q = con.query();
        for(int i=0; i<10; i++)
        {
            ostringstream s;
            s << "INSERT INTO t1 VALUES (" << i << ",'Dette er en test')";
            q.exec(s.str());
        }
    }
    catch(BadQuery er)
    {
        cerr << "MySQL error: " << er.error << endl;
    }
    return 0;
}
```

Build MS VC++ 6:

```
cl /MD /GX /DSOCKET=int /I\mysql++\include /I\mysql\include query3.cpp
\mysql++\lib\mysql++.lib \mysql\lib\opt\mysqlclient.lib
cl /MD /GX /DSOCKET=int /I\mysql++\include /I\mysql\include insert3.cpp
\mysql++\lib\mysql++.lib \mysql\lib\opt\mysqlclient.lib
```

Fordele:
* objekt orienteret
* stort kraftfuldt API
* meget C++'sk

Ulemper:
* meget compiler specifikt og det kan drille rigtigt meget at få det til at virke

* nogen af de avancerede features er implementeret via en 31000 linier header file med #define's
* ligner ikke de objekt orienterede API'er man kender fra C#, Java etc.

Hjemmelavet C++ API

Er man utilfreds med MySQL's C++ API kan man jo lave sit eget.

Her er starten på et simpelt som forsøger at ligne de kendte
objekt orienterede database API'er.

mysqlobj.h

```
#ifndef MYSQLOBJ_H
#define MYSQLOBJ_H

#include <cstdlib>

#include "mysql.h"

class ResultSet
{
    private:
        MYSQL_RES *result;
        MYSQL_ROW row;
        int *len;
        ResultSet();
        void init(MYSQL *handle, char *sqlcmd);
        friend class Connection;
    public:
        ~ResultSet();
        bool Next();
        int GetNoFields();
        bool IsNull(int ix);
        char *Get(int ix);
        int GetInt(int ix) { return atoi(Get(ix)); };
        double GetDouble(int ix) { return atof(Get(ix)); };
        char *GetString(int ix) { return Get(ix); };
};

class Connection
{
    private:
        MYSQL *handle;
        bool connected;
        ResultSet rs;
    public:
        Connection(char *database, char *host = "localhost", char
*user="root", char *pw="", int port=3306);
        ~Connection();
        ResultSet ExecuteQuery(char *sql,...);
        int ExecuteNonQuery(char *sql,...);
        char *GetError();
        inline bool IsConnected() { return connected; };
```

```
};

#endif // MYSQLOBJ_H
```

mysqlobj.cpp

```
#include <cstdio>
#include <cstdarg>

#include "mysqlobj.h"

ResultSet::ResultSet()
{
    result = NULL;
}

void ResultSet::init(MYSQL *handle, char *sqlcmd)
{
    if(result != NULL)
    {
        mysql_free_result(result);
        result = NULL;
    }
    mysql_query(handle, sqlcmd);
    result = mysql_store_result(handle);
}

ResultSet::~ResultSet()
{
    if(result != NULL)
    {
        mysql_free_result(result);
        result = NULL;
    }
}

bool ResultSet::Next()
{
    row = mysql_fetch_row(result);
    if(row != NULL)
    {
        len = (int *)mysql_fetch_lengths(result);
        return true;
    }
    else
    {
        return false;
    }
}

int ResultSet::GetNoFields()
```

```
{
    return mysql_num_fields(result);
}

bool ResultSet::IsNull(int ix)
{
    return (row[ix] == NULL);
}

char *ResultSet::Get(int ix)
{
    return row[ix];
}

Connection::Connection(char *database, char *host, char *user, char *pw, int
port) : rs()
{
    handle = mysql_init(NULL);
    if(handle == NULL)
    {
        connected = false;
    }
    else if(!mysql_real_connect(handle, host, user, pw, database, port, NULL,
0))
    {
        connected = false;
    }
    else
    {
        connected = true;
    }
}

Connection::~Connection()
{
    mysql_close(handle);
    connected = false;
}

ResultSet Connection::ExecuteQuery(char *sql,...)
{
    va_list argptr;
    va_start(argptr, sql);
    char buf[10000];
    vsprintf(buf, sql, argptr);
    va_end(argptr);
    rs.init(handle, buf);
    return rs;
}

int Connection::ExecuteNonQuery(char *sql,...)
{
    va_list argptr;
    va_start(argptr, sql);
    char buf[10000];
```

```
    vsprintf(buf, sql, argptr);
    va_end(argptr);
    return mysql_query(handle, buf);
}

char *Connection::GetError()
{
    return mysql_error(handle);
}
```

Eksempel:

query4.cpp

```
#include <iostream>

using namespace std;

#include "mysqlobj.h"

int main()
{
    //
    // åben connection til:
    //   database = "Test"
    //   server = "localhost" (default)
    //   username = "root" (default)
    //   password = "" (default)
    //   port = 3306 (default)
    //
    Connection con("Test");
    if(!con.IsConnected())
    {
        cerr << con.GetError() << endl;
        exit(1);
    }
    // udfør query
    ResultSet rs = con.ExecuteQuery("SELECT * FROM t1");
    // print resultat af query
    while (rs.Next()) {
        for (int i=0; i<rs.GetNoFields(); i++) {
            cout << " " << (!rs.IsNull(i) ? rs.Get(i) : "NULL");
        }
        cout << endl;
    }
    return 0;
}
```

insert4.cpp

```
#include <iostream>

using namespace std;

#include "mysqlobj.h"

int main()
{
    //
    // åben connection til:
    //   database = "Test"
    //   server = "localhost" (default)
    //   username = "root" (default)
    //   password = "" (default)
    //   port = 3306 (default)
    //
    Connection con("Test");
    if(!con.IsConnected())
    {
        cerr << con.GetError() << endl;
        exit(1);
    }
    // indsæt 10 rækker
    for(int i=0; i<10; i++)
    {
        con.ExecuteNonQuery("INSERT INTO t1 VALUES (%d,'%s')", i, "Dette er en
test");
    }
    return 0;
}
```

Build MS VC++ 6:

```
cl /c /GX /DSOCKET=int /I\mysql\include mysqlobj.cpp
cl /GX /DSOCKET=int /I\mysql\include query4.cpp mysqlobj.obj
\mysql\lib\opt\libmysql.lib
cl /GX /DSOCKET=int /I\mysql\include insert4.cpp mysqlobj.obj
\mysql\lib\opt\libmysql.lib
```

Fordele:
* objekt orienteret
* velkendt stil

Ulemper:
* simpelt API
* man skal selv vedligeholde det

Bemærk at mysqlobj.h/mysqlobj.cpp kun er mit forslag. Man kan lave andre varianter.
Men man er meget velkommen til at tage udgangspunkt i mysqlobj.h/mysqlobj.cpp.

MySQL Connector for C++

MySQL har lavet et nye C++ API til erstatning for MySQL++

Det skal hentes separat herfra:
  <a
href="http://dev.mysql.com/downloads/connector/cpp/1.0.html">http://dev.mysql.com/downloads/connector/cpp/1.0.html

Jeg har endnu ikke prøvet dette API.

Det er stadig kun i alpha udgave og jeg har problemer med bare at få buildet det.

Kommentar af ferrari_brian d. 09. Jun 2006 | 1

Nice ... hvis bare jeg havde fundet denne artikel noget før!

Kommentar af mcgoat d. 02. Apr 2004 | 2

God og rimelig simpel at gennemskue

Kommentar af lopper d. 05. Apr 2004 | 3

Tak det giver et godt indblik i hvordan man kan lave C++ med mySQL.

Kommentar af brhino d. 08. Apr 2004 | 4