Code::Fox

Author: Flurreh
Date: 26.11.2013

Code::Fox MySQL Class

Because the MySQL C-connector isn't that nice to use when you're coding in C++ using `std::string` and this stuff, I've written a small class which wraps the C-connector with C++-style data types and so on.
I've also implemented the possibility to use prepared statements, which means you don't have to put the values into the query string manually.
This document will describe how to use the Code::Fox MySQL class.

Files

The MySQL class comes with two files, `mysqlquery.h` and `mysqlquery.cpp`.
What you will still need are the MySQL headers and libraries which can be found on the MySQL website or in MySQL server installations.

Classes

It offers two classes, `MySQLConnection` and `MySQLQuery`. The first one is used to establish a connection to a MySQL server and the second one is used to create and execute SQL statements.

Functions

Functions in `MySQLConnection`:

```
bool Connect(std::string sHostname, uint16_t wPort, std::string
sUsername, std::string sPassword, std::string sDB)
```
This function is used to connect to a MySQL server. Pretty self-explaining I guess.

```
bool SelectDB(std::string sSchemaName);
```
Sets the database you want to work on.

```
void Disconnect()
```
Closes the MySQL connection.

```
std::string GetLastError()
```
Returns the last SQL error as a string.

```
MYSQL *getConn()
```
Returns a pointer to the MYSQL object.

```
bool IsConnected()
```
Returns true when connected, false when not connected.

```
std::string EscapeString(std::string value)
```
Returns the input string as an escaped one.

Functions in MySQLQuery:

```
MySQLQuery(MySQLConnection *mConn, std::string sStatement);
~MySQLQuery();

// sets the value of idx to a given string (also adds quotation marks and
escapes the string)
bool setString(unsigned int idx, std::string value);
// sets the value of idx to a given int
bool setInt(unsigned int idx, int value);
// sets the value of idx to a given double
bool setDouble(unsigned int idx, double value);
// sets the value of idx to a given time_t
bool setTime(unsigned int idx, time_t value);
// sets the value of idx to NULL
bool setNull(unsigned int idx);

// executes a SELECT-statement
bool ExecuteQuery();
// executes an UPDATE-statement
bool ExecuteUpdate();
// executes an INSERT-statement and returns the last inserted ID
int ExecuteInsert();

// builds the query string with filled-in arguments and returns it
std::string BuildQueryString();

// returns a field name
std::string getFieldName(unsigned int field);
// gets a string value from the given row and field
std::string getString(unsigned int row, unsigned int field);
std::string getString(unsigned int row, std::string field);
// gets an int value from the given row and field
int getInt(unsigned int row, unsigned int field);
int getInt(unsigned int row, std::string field);
// gets a double value from the given row and field
double getDouble(unsigned int row, unsigned int field);
double getDouble(unsigned int row, std::string field);
// gets a time value from the given row and field
time_t getTime(unsigned int row, unsigned int field);
time_t getTime(unsigned int row, std::string field);

// returns the result row count
unsigned int GetResultRowCount();
unsigned int GetFieldCount();
```

Author: Flurreh
Date: 26.11.2013

Usage

The usage of this class is pretty much self-explaining, though I should tell that all indexes start at 1, not at 0.

```
    // create connection
    MySQLConnection *sqlConn = new MySQLConnection();
    sqlConn->Connect("127.0.0.1", 3306, "root", "1234", "testpw");

    // create query
    MySQLQuery *sqlQuery = new MySQLQuery(sqlConn, "SELECT username, password,
status FROM account");
    sqlQuery->ExecuteQuery();

    // get the values
    for(unsigned int i = 1; i <= sqlQuery->GetResultRowCount(); i++)
    {
        for(int n = 1; n <= sqlQuery->GetFieldCount(); i++)
        {
            std::cout << sqlQuery->getString(i, n) << "\t\t";
        }
        std::cout << std::endl;
    }
```

This example selects all entries of the table „account“ and prints them in the console.

```
    // create connection
    MySQLConnection *sqlConn = new MySQLConnection();
    sqlConn->Connect("127.0.0.1", 3306, "root", "1234", "testpw");

    // create query
    MySQLQuery *sqlQuery = new MySQLQuery(sqlConn, "INSERT INTO account
(username, password, status) VALUES(?, PASSWORD(?), ?");
    sqlQuery->setString(1, "testuser");
    sqlQuery->setString(2, "testpw");
    sqlQuery->setInt(3, 1);
    std::cout << "Last inserted line: " << sqlQuery->ExecuteInsert() <<
std::endl;
```

This example inserts a row into the „account“ table and prints out the last inserted line index.

If you find any bugs, please report them to me here on GitHub or on elitepvpers (nico_w).