Subject: Separate Database Access code
Posted by jibe on Mon, 21 Jan 2013 14:33:14 GMT
View Forum Message <> Reply to Message

Hi,

I need the same code to access a database in several applications. I was thinking about a library providing this code, but I'm unable to have something working as soon as I try to separate the database access code from main application code...

Is it possible ?

I tried the joined code, but I get an "invalid memory access" when I execute it...

This error occurs while the database is created. There is no error if I don't put any table in the database (empty .sch file).

What is wrong ? Am I blind and unable to see a big mistake in my code ?

Thanks.

File Attachments
1) TestDB.zip, downloaded 265 times

Subject: Re: Separate Database Access code
Posted by lectus on Mon, 21 Jan 2013 14:57:47 GMT
View Forum Message <> Reply to Message

Hi!

Add Quote:Sqlite3Session lDB; as a member of libDB and after you open the database add:
Quote:SQL = lDB;:

```
class libDB {
typedef libDB CLASSNAME;

public:
 Sqlite3Session lDB;
 libDB();
 ~libDB();
 void InitDB();
};
```

```
void libDB::InitDB()
{
 LOG("Open DB");
 if(!lDB.Open(ConfigFile("libDB.db"))) {
  Exclamation(t_("Cannot create or open libDB database file\n"));
  return;
 }
 LOG("SqlSchema");
 SqlSchema sch(SQLITE3);

 SQL = lDB;
.
.
.
```

It fixes the problem. Actually the fix was just "SQL = lDB". I recommend putting lDB as a member of libDB just so your library has a better design.

Subject: Re: Separate Database Access code
Posted by jibe on Mon, 21 Jan 2013 16:33:28 GMT
View Forum Message <> Reply to Message

Thanks for the help !

Yes, this fixes the problem. I should have test this !

But I don't like this solution, as - if I understand well - SQL is global and can be used anywhere. As the applications that will use the lib could use a second database, and want to use SQL global variable for this second database, I didn't want to use it.

Instead of using SQL, I tried :

```
void libDB::InitDB()
{
 LOG("Open DB");
 if(!lDB.Open(ConfigFile("libDB.db"))) {
  Exclamation(t_("Cannot create or open libDB database file\n"));
  return;
 }
 LOG("SqlSchema");
 SqlSchema sch(SQLITE3);

 Sql sql(lDB);
```

but I always get an invalid memory access error...

Is there a way to have this working without using SQL ?

Subject: Re: Separate Database Access code
Posted by lectus on Tue, 22 Jan 2013 16:49:15 GMT
View Forum Message <> Reply to Message

I'm no U++ expert, but I don't see the problem with global SQL.

You can set SQL variable and then later have the program set it to something else.

And then set it back with no problem.

Also, consider building this library as DLL. So, this way the global SQL will be internal to the DLL and thus different from the application using it.

Subject: Re: Separate Database Access code
Posted by jibe on Tue, 22 Jan 2013 20:46:08 GMT
View Forum Message <> Reply to Message

Hi,

Yes, you are right. It will work using one of your solutions. But:

1 - Re-affacting the value of SQL is dangerous. Developing an application, you will have to re-affect the right value before any use of SQL, as you will not be sure if it has been used by the library. Not very clean and bugs could be numerous !

2 - Making a DLL is a better solution. But as I want this library to be multi-platform, it will not be very easy. A simple U++ library that you add in a project is a lot easier to do !

The U++ documentation says :
Quote:Most applications need to work with just single database backend, therefore repeating SqlSession parameter in all Sql declarations would be tedious.

To this end U++ supports concept of "main database" which is represented by SQL variable. SQL is of Sql type. When any other Sql variable is created with default constructor (no session parameter provided), it uses the same session as the one the SQL is bound to. To assign session to global SQL, use operator=:
If I understand well :
- It is possible to work with several databases,
- To work with another database than the "main database" (the one in my library is a secondary one), I must create sql variables with session parameter provided.

It's just what I'm trying to do... with no succes !

Subject: Re: Separate Database Access code
Posted by lectus on Tue, 22 Jan 2013 21:08:39 GMT
View Forum Message <> Reply to Message

jibe wrote on Tue, 22 January 2013 15:462 - Making a DLL is a better solution. But as I want this library to be multi-platform, it will not be very easy. A simple U++ library that you add in a project is a lot easier to do !

You can build .DLL in Windows and .SO in Linux. It's still multi-platform.

Another way: build a static library and let GCC link it on Linux and Mingw or VS on Windows.

Subject: Re: Separate Database Access code
Posted by jibe on Thu, 24 Jan 2013 08:33:15 GMT
View Forum Message <> Reply to Message

Hi,

Thanks for the help

Probably you are right. But my problem is less to find a solution for this particular case, but more to understand how to have several databases in the same application.

It's possible that a dynamic library is the best solution in this case, anyway it's surely a good solution. But somewhere, it's working around the true question : is it possible to have several databases with U++, and how to do ? Knowing that, I'll be able to decide by myself to use or not a dynamic library, without having the impression to make a workaround by ignorance