Subject: Re: Is it possible to call a stored procedure with an output parameter?
Posted by BioBytes on Sat, 11 Feb 2012 15:50:18 GMT
View Forum Message <> Reply to Message

Hello papascalientes,

I guess you mean a prepared statement as follows:

Inserting data with a prepared statement
```
PreparedStatement* pStatement = pDatabase->PrepareStatement(_("INSERT INTO table1
(column1, column2) VALUES (?, ?)"));
if (pStatement)
{
 pStatement->SetParamString(1, _("One"));
 pStatement->SetParamString(2, _("Two"));
 pStatement->RunQuery():
 pDatabase->CloseStatement(pStatement);
}
```

or

```
PreparedStatement* pStatement = pDatabase->PrepareStatement(_("SELECT * FROM table1
WHERE column1 - ?)"));
if (pStatement)
{
 pStatement->SetParamString(1, _("One"));
 DatabaseResultSet* pResults = pStatement->RunQueryWithResults():
 if (pResults)
 {
  while (pResults->Next())
  {
   wxString strOne = pResults->GetResultString(_("column1"));
   wxString strTwo = pResults->GetResultString(_("column2"));
  }
  pDatabase->CloseResultSet(pResults);
 }
 pDatabase->CloseStatement(pStatement);
}
```

This is possible when using databaselayer library designed for wxWidgets.

For U++ have a look to:

2. Using global main database, executing statements with parameters, getting resultset info

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=:

```
#include <Core/Core.h>

#include <plugin/sqlite3/Sqlite3.h>
using namespace Upp;

CONSOLE_APP_MAIN

    Sqlite3Session sqlite3;

    if(!sqlite3.Open(ConfigFile("simple.db"))) {

        Cout() << "Can't create or open database file\n";

        return;
    }

#ifdef _DEBUG

    sqlite3.SetTrace();

#endif

    SQL = sqlite3;

    SQL.Execute("drop table TEST");

    SQL.ClearError();

    SQL.Execute("create table TEST (A INTEGER, B TEXT)");

    for(int i = 0; i < 10; i++)

        SQL.Execute("insert into TEST(A, B) values (?, ?)", i, AsString(3 * i));

    Sql sql;

    sql.Execute("select * from TEST");

    for(int i = 0; i < sql.GetColumns(); i++)

        Cout() << sql.GetColumnInfo(i).name << '\n';

    while(sql.Fetch())
```

```
    Cout() << sql[0] << " \"" << sql[1] << "\'\n";
}
```

As global SQL is regular Sql variable too, it can be used to issue SQL statements.

I don't know if this is what you are looking for.

Biobytes
