Subject: PostgreSQL Session.Open Leak??
Posted by indiocolifa on Wed, 20 Feb 2008 18:52:34 GMT
View Forum Message <> Reply to Message

Anyone tried PostgreSQL/U++? I'm getting leaks detected at the end of program execution, and I've discovered it goes away when isolating the pgsql Session open function in my program.

Note that I'm mantaining a global PostgreSQL session variable in a singleton class (this member variable is not static). Getting the class instance and getting a pointer to that PostgreSQLSession variable works well in my program...

Ideas?

Subject: Re: PostgreSQL Session.Open Leak??
Posted by zsolt on Wed, 20 Feb 2008 21:30:04 GMT
View Forum Message <> Reply to Message

testcase?

Subject: Re: PostgreSQL Session.Open Leak??
Posted by indiocolifa on Wed, 20 Feb 2008 22:30:34 GMT
View Forum Message <> Reply to Message

Tomorrow I'll post the code, it's on the machine at my job running 2003 Server SP1+PostgreSQL 8.3+UPP 2008.1.

Subject: Re: PostgreSQL Session.Open Leak??
Posted by indiocolifa on Thu, 21 Feb 2008 14:35:14 GMT
View Forum Message <> Reply to Message

MAIN.CPP

include "sr2k8.h"
#include "MainWindow.h"

#define SCHEMADIALECT <PostgreSQL/PostgreSQLSchema.h>
#include <Sql/sch_source.h>

bool ConnectToPSQL();
void LoadConfig();

GUI_APP_MAIN

```
{
 ::SetLanguage( ::GetSystemLNG() );

 LoadConfig();
 if (ConnectToPSQL())
 {
  MainWindow mw;
  mw.Run();
 }
}

void LoadConfig()
{
 String cfgfile = ConfigFile();
 if (FileExists(cfgfile))
 {
  VectorMap<String, String> cfg = LoadIniFile(cfgfile);
 }
 else
 {
  // configuracion por default
  String s;
  s << "APPV=1.0\n"
    << "HOST=10.62.200.1\n"
    << "DATABASE=S2K8\n"
    << "USER=sis\n"
    << "PASS=sis\n";

  if (!SaveFile(ConfigFile(),s))
  {

   exit(-1);
  }

 }
}

bool ConnectToPSQL()
{
 if (!G_STATE->GetPsqlSession()->Open("host=localhost user=sis password=sis
dbname=S2K8"))
 {
  Exclamation(Format("Error abriendo base de datos: %s",
   DeQtf(G_STATE->GetPsqlSession()->GetLastError())));
  return false;
 }

 return true;
```

```
}
```

Sr2k8.H

```
#ifndef _sr2k8_sr2k8_h
#define _sr2k8_sr2k8_h

#include <CtrlLib/CtrlLib.h>
#include <SqlCtrl/SqlCtrl.h>

using namespace Upp;

// definiciones para PostgreSQL
#include <PostgreSQL/PostgreSQL.h>
#define SCHEMADIALECT <PostgreSQL/PostgreSQLSchema.h>
#define MODEL <sr2k8/sr2k8db.sch>
#include <Sql/sch_header.h>

#include "globalState.h"
#define G_STATE GlobalState::getInst()

// layout de las ventanas
#define LAYOUTFILE <sr2k8/sr2k8.lay>
#include <CtrlCore/lay.h>

#endif
```

globalState.h (singleton class spec)

```
#ifndef _sr2k8_globalState_h_
#define _sr2k8_globalState_h_

#include <PostgreSQL/PostgreSQL.h>

using namespace Upp;

// clase singleton que mantiene un estado global del sistema

class GlobalState
{
 static GlobalState* pInstance;

 PostgreSQLSession  pgsql;
 String    currentUser;
```

```
 public:
 static GlobalState* getInst()
 {
  if (!pInstance)
   pInstance = new GlobalState;

  return pInstance;
 }

 PostgreSQLSession* GetPsqlSession() { return &pgsql; }

 protected:
 GlobalState() {
  currentUser == "";
 };
 GlobalState(const GlobalState&);
 GlobalState& operator= (const GlobalState&);
};

#endif
```

This is initialized in globalstate.cpp as follows:

```
#include "globalState.h"

GlobalState* GlobalState::pInstance = NULL;
```

Do you see any problem related to the singleton??? Maybe it's that 'new Globalstate' on singleton getInst() method.

Subject: Re: PostgreSQL Session.Open Leak??
Posted by indiocolifa on Thu, 21 Feb 2008 14:49:36 GMT
View Forum Message <> Reply to Message

Yesss,,,, fixed it!

The last line of main is now:

delete G_STATE;

G_STATE is a macro so, it's equal to:

delete globalState::getInst();

since getInst returns a pointer, it's all ok.

Subject: Re: PostgreSQL Session.Open Leak??
Posted by mirek on Thu, 21 Feb 2008 15:14:56 GMT
View Forum Message <> Reply to Message

indiocolifa wrote on Thu, 21 February 2008 09:49Yesss,,,, fixed it!

The last line of main is now:

delete G_STATE;

G_STATE is a macro so, it's equal to:

delete globalState::getInst();

since getInst returns a pointer, it's all ok.

Well, well, that is the price of NOT using U++ way od programming:)

We have nice Single template.

Remove these statics and just use Single<GlobalState>() instead of GlobalState::getInst. Also, your version is MT broken (depends on compiler); Single is guaranteed to work.

No well-behaved program should contain similar "delete"

Mirek

Subject: Re: PostgreSQL Session.Open Leak??
Posted by indiocolifa on Thu, 21 Feb 2008 15:26:36 GMT
View Forum Message <> Reply to Message

Fixed!

Subject: Re: PostgreSQL Session.Open Leak??
Posted by indiocolifa on Thu, 21 Feb 2008 15:28:46 GMT
View Forum Message <> Reply to Message

luzr wrote on Thu, 21 February 2008 13:14indiocolifa wrote on Thu, 21 February 2008 09:49Yesss,,,, fixed it!

The last line of main is now:

delete G_STATE;

G_STATE is a macro so, it's equal to:

delete globalState::getInst();

since getInst returns a pointer, it's all ok.

Well, well, that is the price of NOT using U++ way od programming:)

We have nice Single template.

Remove these statics and just use Single<GlobalState>() instead of GlobalState::getInst. Also, your version is MT broken (depends on compiler); Single is guaranteed to work.

No well-behaved program should contain similar "delete"

Mirek

Hey Mirek, great!

Should I define Globalstate as a normal class (I mean, not using statics like in the Singleton pattern)?

Discovering U++ is fun!

Subject: Re: PostgreSQL Session.Open Leak??
Posted by mirek on Fri, 22 Feb 2008 09:09:09 GMT
View Forum Message <> Reply to Message

indiocolifa wrote on Thu, 21 February 2008 10:28
Should I define Globalstate as a normal class (I mean, not using statics like in the Singleton pattern)?

Discovering U++ is fun!

Depends. I would either use Single directly or created a global function to return Single:

```
GlobalState& TheGlobalState() {
  return Single<GlobalState>();
}
```

but you can also stay a bit closer to "exact" singleton pattern:

```
class GlobalState {
   static GlobalState& getInst() { return Single<GlobalState>(); }
};
```

or if you insist on pointer (which IMO is stupid here, as the result can never be NULL):

```
class GlobalState {
   static GlobalState *getInst() { return &Single<GlobalState>(); }
};
```

In either case, GlobalState will be destructed at exit, avoiding memory leak.

Mirek
