Subject: PythonPP - tiny C++ wrappers for Python C API
Posted by Novo on Mon, 04 Apr 2011 04:26:30 GMT
View Forum Message <> Reply to Message

PythonPP is an initial version of a port to UPP of tiny C++ wrappers for Python C API, which I developed a long time ago. They are somewhat similar to the wrappers in the BOOST library, although they are significantly less complicated.

If you are interested in such kind of code I can keep porting/improving it. It basically allows to call Python scripts from C++, or to expose C++ code to Python. The whole UPP can be exposed to Python

EDIT: Update file with source code.

File Attachments
1) PythonPP.zip, downloaded 382 times

Subject: Re: PythonPP - tiny C++ wrappers for Python C API
Posted by nlneilson on Mon, 04 Apr 2011 16:22:03 GMT
View Forum Message <> Reply to Message

It looks interesting.

I looked for some kind of Help or Readme file for an explanation of how this can be used, capabilities or limitations (API?).

I have used sockets for interaction between Python, C++, and Java.

Your PythonPP may be an easier/cleaner/more efficient way in certain situations.

Neil

Subject: Re: PythonPP - tiny C++ wrappers for Python C API
Posted by Novo on Mon, 04 Apr 2011 20:44:15 GMT
View Forum Message <> Reply to Message

nlneilson wrote on Mon, 04 April 2011 12:22It looks interesting.

I looked for some kind of Help or Readme file for an explanation of how this can be used, capabilities or limitations (API?).

I have used sockets for interaction between Python, C++, and Java.

Your PythonPP may be an easier/cleaner/more efficient way in certain situations.

Neil

PythonPP can do pretty much the same as Boost.Python. It serves the same purpose - to make using of Python's API easier. Capabilities are similar. PythonPP is probably missing some advanced features of Boost.Python, but they can be added.

If you want to use raw Python C API you will need to learn how to deal with dozens of functions, internal structures, and reference counting. PythonPP is hiding all this complexity from you, and it is making your code safer and readable.

I'll try to make more examples tonight

Subject: Re: PythonPP - tiny C++ wrappers for Python C API
Posted by Novo on Tue, 05 Apr 2011 00:50:20 GMT
View Forum Message <> Reply to Message

A simple example of calling Python from C++:

```
#include <PythonPP/PythonPP.h>
#include <PythonPP/PythonPP_emb.h>

using namespace Upp;

CONSOLE_APP_MAIN
{
 pythonpp::Engine eng;
 eng.ExecuteStr("print 'Hello World!'");
}
```

Subject: Re: PythonPP - tiny C++ wrappers for Python C API
Posted by nlneilson on Tue, 05 Apr 2011 04:19:55 GMT
View Forum Message <> Reply to Message

```
PythonPP_emb.cpp
c:\myapps\pythonpp\PythonPP_error.h(25) : fatal error C1083: Cannot open include file:
'Python.h': No
  such file or directory
c:\myapps\pythonpp\PythonPP_error.h(25) : fatal error C1083: Cannot open include file:
'Python.h': No
  such file or directory
PythonPP_ext.cpp
PythonPP: 2 file(s) built in (0:01.30), 652 msecs / file, duration = 1344 msecs, parallelization 100%
```

I unzipped your PythonPP.zip

Put PythonPP and PythonPPApp in MyApps
errors
Copied everything from PythonPP into PythonPPApp
same errors as above

In PythonPP.upp
library(LINUX) python2.6;
Apparently this is set up for Linux only with Python 2.x

I tried it on Win XP where I just have Python 3.2
My problem not your set up.

Neil

Subject: Re: PythonPP - tiny C++ wrappers for Python C API
Posted by nlneilson on Tue, 05 Apr 2011 05:02:43 GMT
View Forum Message <> Reply to Message

In Ubuntu 10.10 with the default Python 2.6
----- Core (SSE2 GCC SHARED LINUX POSIX) (1 / 3)
----- PythonPP (SSE2 GCC SHARED LINUX POSIX) (2 / 3)
PythonPP_emb.cpp
PythonPP_ext.cpp
In file included from /home/neil/upp/MyApps/PythonPP/PythonPP_object.h:10,
from /home/neil/upp/MyApps/PythonPP/PythonPP_pdt.h:10,
from /home/neil/upp/MyApps/PythonPP/PythonPP.h:10,
from /home/neil/upp/MyApps/PythonPP/PythonPP_emb.h:23,
from /home/neil/upp/MyApps/PythonPP/PythonPP_emb.cpp:1:
/home/neil/upp/MyApps/PythonPP/PythonPP_error.h:25: fatal error: Python.h: No such file or directory
compilation terminated.
In file included from /home/neil/upp/MyApps/PythonPP/PythonPP_object.h:10,
from /home/neil/upp/MyApps/PythonPP/PythonPP_seq.h:10,
from /home/neil/upp/MyApps/PythonPP/PythonPP_ext.h:10,
from /home/neil/upp/MyApps/PythonPP/PythonPP_ext.cpp:1:
/home/neil/upp/MyApps/PythonPP/PythonPP_error.h:25: fatal error: Python.h: No such file or directory
compilation terminated.
PythonPP: 2 file(s) built in (0:00.20), 104 msecs / file, duration = 249 msecs, paralleliz
ation 90%

There were errors. (0:00.43)

Subject: Re: PythonPP - tiny C++ wrappers for Python C API

Posted by Novo on Tue, 05 Apr 2011 14:44:04 GMT
View Forum Message <> Reply to Message

On Linux you need to install python-dev (it will install v2.6 on my system). After that add /usr/include/python2.6 to include directories for GCC build method.

Unfortunately, there is no generic libpython library. They all have versions.

I haven't tried to compile PythonPP on Windows. It is an initial version.

Subject: Re: PythonPP - tiny C++ wrappers for Python C API
Posted by kohait00 on Tue, 12 Apr 2011 07:25:55 GMT
View Forum Message <> Reply to Message

hi novo

i'd like to compare the features / API invocation of PythonPP to that one of boost.Python. could you help here? since there is no API reference

yesterday i have added a BoostPyTest to bazaar. could you provide a similar one?

Subject: Re: PythonPP - tiny C++ wrappers for Python C API
Posted by Novo on Tue, 12 Apr 2011 14:48:55 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 12 April 2011 03:25hi novo

i'd like to compare the features / API invocation of PythonPP to that one of boost.Python. could you help here? since there is no API reference

yesterday i have added a BoostPyTest to bazaar. could you provide a similar one?

Hi kohait00,
I updated source code of PythonPP. I also added an example of a module, which unfortunately doesn't work yet.

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

////////////////////////////////////////////////////////////////////////////
// Compatibility macros
//
// From Python 2.2 to 2.3, the way to export the module init function
// has changed. These macros keep the code compatible to both ways.
```

```
//
#if PY_VERSION_HEX >= 0x02030000
#  define PYDBAPI_MODINIT_FUNC(name)        PyMODINIT_FUNC name(void)
#else
#  define PYDBAPI_MODINIT_FUNC(name)        DL_EXPORT(void) name(void)
#endif

using namespace Upp;

namespace python
{

 class Test : public pythonpp::ExtObject<Test>
 {
 public:
  Test();

  pythonpp::Object Foo ( const pythonpp::Tuple& args );
  pythonpp::Object Boo ( const pythonpp::Tuple& args );
 };

 Test::Test()
 {
  ROAttr( "__class__", GetTypeObject() );
  PrepareForPython(this);
 }

 pythonpp::Object Test::Foo ( const pythonpp::Tuple& /*args*/ )
 {
  Cout() << "This is a UPP call !" << EOL;
 }

 pythonpp::Object Test::Boo ( const pythonpp::Tuple& /*args*/ )
 {
  Cout() << "This is Boo !" << EOL;
 }
}

//////////////////////////////////////////////////////////////////////////////

static struct PyMethodDef methods[] = {
 { NULL, NULL }
};

//////////////////////////////////////////////////////////////////////////////
// Module initialization
PYDBAPI_MODINIT_FUNC ( initPythonPPModule )
{
```

```
 pythonpp::ModuleExt::Declare ( "PythonPPModule", methods );
 PyObject *module = pythonpp::ModuleExt::GetPyModule();

 // Declare class Test.
 python::Test::
  Def ( "Foot", &python::Test::Foo, "Method #1" ).
  Def ( "Boo", &python::Test::Boo, "Method #2" );
 python::Test::Declare ( "PythonPPModule.Test" );

 if ( PyType_Ready ( &python::Test::GetType() ) == -1 )
 {
  return;
 }

 if ( PyModule_AddObject ( module, const_cast<char*> ( "Test" ), ( PyObject* )
&python::Test::GetType() ) == -1 )
 {
  return;
 }
}
```

I spent a couple of evenings trying to bring it back to life and understood that it is more complicated than I though.

Unless there is some kind of restriction the best way to go is Boost.Python.

Subject: Re: PythonPP - tiny C++ wrappers for Python C API
Posted by kohait00 on Tue, 12 Apr 2011 15:17:34 GMT
View Forum Message <> Reply to Message

if so, i'd love to have some help in 'exposing' upp to python..via boost then.

Subject: Re: PythonPP - tiny C++ wrappers for Python C API
Posted by Novo on Thu, 14 Apr 2011 03:35:25 GMT
View Forum Message <> Reply to Message

further PythonPP development?

My memory is getting back slowly. Besides just bringing PythonPP back to life it requires

complete redesign of marshalling code. Old code never looks good.

To be fair, I already promised a lot to many people. Before PythonPP I promised a firebird driver for UPP. I think I will switch to firebird now.

Quote:
if so, i'd love to have some help in 'exposing' upp to python..via boost then.

Of course, Boost is the best, unless there are political issues related to it. IMHO you shouldn't have any problem using it because it was around for a decade and you should be able to find a lot of examples on internet. This can be an inspirational reading

---

Subject: Re: PythonPP - tiny C++ wrappers for Python C API
Posted by kohait00 on Thu, 14 Apr 2011 05:25:19 GMT
View Forum Message <> Reply to Message

Well there are plenty examples on using python but when it comes to embedding python there is almost but no quality information available.my current boost py test package is evolving slowly.but I am on the way.often problems arise around ownership issues since most info leaves it to python but when it comes to exposing existing upp instances almost no examples.

So far my biggest concern is to choose the right upp representation for the python data types.which is ready for the trivial types but cumbersome for the containers.see my boost python thread
Thanks