Extend Python Using C++ and ArcObjects

Mark Cederholm UniSource Energy Services

Esri Developer Summit 2012

Using ArcObjects in Python Esri Developer Summit 2010

ArcMap and Python: Closing the VBA Gap Tuesday 4:30 Mesquite C

Extend Python Using C++ and ArcObjects Wednesday 11:15 Mesquite B

Download presentations and code: http://www.pierssen.com/arcgis10/python.htm

COM Interop: relative speed



Benchmark = 500+K ShapeCopy operations

2:26:42 PM

Don't forget arcpy.da (at 10.1)

Querying polygon geometry:



Querying SHAPE_AREA:



Options to extend Python:

 COM/ATL + comtypes: No longer recommended
 Python/C API: Create C Python module (.pyd) SWIG, Cython: Automatically generate wrapper code to create a pyd

ctypes: Create wrapper code on the Python side

Extending Python with C or C++: http://docs.python.org/extending/extending.html ArcObjects in Visual C++:

http://help.arcgis.com/en/sdk/10.0/arcobjects_net/ conceptualhelp/index.html#/Appendix_ArcObject s_in_Visual_C/000100000n8q000000/

[All sample code and walkthroughs assume ArcGIS 10.1 pre-release with Python 2.7]

2:26:42 PM

Free Visual C++ downloads:

Visual C++ 2008 Express (required for Cython installation)

http://www.microsoft.com/download/en/details.aspx?id=20682

- Visual C++ 2010 Express
 - http://www.microsoft.com/visualstudio/en-us/products/2010editions/visual-cpp-express
- Sample code and walkthroughs assume VC 2008E
- All walkthroughs tested successfully in VC 2010E (with minor modifications involving project variables)
- Express versions do not have ATL or MFC

Testing for memory leaks in VC Express:

```
#ifdef _DEBUG
       _CrtMemState s1, s2, s3;
       _CrtMemCheckpoint(&s1);
#endif
       char * sResult = MyFunction(sMyArg);
#ifdef DEBUG
       _CrtMemCheckpoint(&s2);
       if (_CrtMemDifference(&s3, &s1, &s2))
               OutputDebugString(L"Allocated blocks found:\n");
               _CrtMemDumpAllObjectsSince(&s1);
       }
#endif
          In this example, only one block should be allocated
```

2:26:42 PM

Walkthrough 1: Creating a Python module in Visual C++

[Sample Code: PythonC/standalonedemo]

2:26:42 PM

An example C++ class:

```
// demo. h
#include <string>
using namespace std;
class demo
private:
       bool _bInitialized;
       string _sResult;
publ i c:
       demo();
       bool Init();
       char * Inventory(char * Workspace, char * FeatureClass);
private:
       void DoIt(char * ws, char * fc);
};
```

Step 1: Set up Python development package

- Download and unzip Python 2.7 source code package:
 - http://www.python.org/ftp/python/2.7.2/Python-2.7.2.tgz
 - [Download actually has a ".tar" extension which must be changed to ".tgz" for WinZip to handle it properly]
- Create a folder called "src" adjacent to where you intend to create Python extension projects
- Move contents of "Python-2.7.2" folder to src folder
 Folder



2:26:42 PM

Open src\PCbuild\pcbuild.sln in Visual C++

 Build solution in both "Debug" and "Release" configurations (ignore errors), creating python27_d.lib and python27.lib



Step 2: Create a new Visual C++ project

- Project type: Win32 Project
- Name: standalonedemo

src)

2:26:42 PM

- Uncheck "Create directory for solution"
- Location: desired project folder (same level as

| Project types: | Т | 'emplates: | | | |
|---|-----------------------------|-----------------------------|-------------------------------|--|--|
| ☑ Visual C++ □ CLR □ Win32 | | Visual Studio installed ten | nplates | | |
| ····· General | | My Templates | | | |
| | | Search Online Templates | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| A project for creating a Win32 application, console application, DLL, or static library | | | | | |
| Name: | standalonedemo | | | | |
| Location: | C:\apps\Development\PythonC | | | | |
| Solution Name: | standalonedemo | | Create directory for solution | | |
| | | | | | |

Application type: DLL

| | S. | | Applicatio | n Settings | | | | | | | | | | |
|------|-----------------|------------------------|------------|--|---|--------------|--------------|---|--|------------------------------|--|---|---|--|
| | Overv Applic | view ation Settings | | Application ty <u>W</u> indow Console <u>D</u> LL Static lift Additional op <u>Empty p</u> Export s <u>Precomp</u> | vpe: s application application prary tions: project symbols piled header | Add common f | header files | for: | | | | | | |
| | | | | | | | | C:\apps apps T apps T ad Da Da T ad T ad T ad Da T ad T ad | Development\Py Idin_maker ata evelopment ArcMap_Pytho PythonC PythonC C Standalone | ythonC\s × • • • | standalone Mame Mame dilim E Rea Star star star star star star star star | edemo ain.cpp dMe.txt ndalonedemo.cp ndalonedemo.si ndalonedemo.si ndalonedemo.vi ndalonedemo.vi ndalonedemo.vi ndalonedemo.vi ndalonedemo.vi | op tb n uo tproj tproj.L | |
| 2:26 | 5:42 P | M | | | | | | | eldEdit acle | | h stda h tarç | afx.h jetver.h | | |

 Properties (All Configurations) C/C++|General|Additional Include Directories: ..\src\Include ..\src\PC [C:\Program Files*]\ArcGIS\Desktop10.1\com *wherever ArcGIS is installed

| tion: All Configurations | Platform: Ac | tive(Win32) | |
|--|--|-------------------------------|----------------------------|
| mon Properties iguration Properties | Additional Include Resolve #using R | Directories "C: | \Program Files\ArcGIS\Desk |
| Seneral Debugging | Debt Additional Supp | Include Directories | ?× |
| | Use | n Files\ArcGIS\Desktop10.1\co | |

 Properties (All Configurations)
 Linker|General|Additional Library Directories: ..\src\PCbuild

| ration: All Configurations | Platform: Active(Win32) | |
|--|---|--|
| Optimization Preprocessor Code Generation Language Precompiled Headers Output Files Browse Information Advanced Command Line | Cutput File Show Progress Version Enable Incremental Linking Suppress Startup Banner Ignore Import Library Register Output Per-user Redirection Additional Library Directories Link Library Dependencies | \$(OutDir)\\$(Proje Not Set Yes (/NOLOGO) No No No No So Yes |
| General | Use Library Dependency Inputs | No |

 Properties (Debug)
 Debugging | Command: ..\..\src\PCbuild\python_d.exe
 Debugging | Working Directory: .\Debug

| ration: Debug | Platform: Active(Win32) | • C |
|--|---|----------------------------|
| ommon Properties onfiguration Properties General | Debugger to launch: Local Windows Debugger | |
| Debugging 3 C/C++ 3 Linker General | Command Command Arguments | \\src\PCbuild\python_d.exe |
| | Working Directory Attach | .\Debug No |
| | | |
| | | |

 Properties (Debug)
 Linker|General|Output File: \$(OutDir)\\$(ProjectName)_d.pyd
 Linker|Input|Additional Dependencies: python27_d.lib

| ration: Debug | Platform: Active(Win32) | | Configu | |
|------------------------|----------------------------|----------------|------------------------------|----------------|
| mmon Properties | Output File | \$(OutDir)\\$(| ProjectName)_d.pyd | |
| nfiguration Properties | Show Progress | Not Set | | |
| - General | Version | | | |
| - Debugging | Enable Incremental Linking | Yes (/INCREN | MENTAL) | |
| | Suppress Startup Banner | Yes (/NOLOGO |) | |
| Linker | Ignore Import Library | No | | |
| General | Register Output | No | | |
| | vation. Debug | | Platform (Active(Wip32)) | |
| | ration: pebug | | | |
| | mmon Properties | s | Additional Dependencies | python27_d.lib |
| | nfiguration Prop | erties | Ignore All Default Libraries | No |
| | General | | Ignore Specific Library | |
| | ··· Debugging | | Module Definition File | |
| | ·C/C++ | | Add Module to Assembly | |
| | | | Embed Managed Resource File | |
| 2:26:43 PM | General | | Force Symbol References | |
| | Input | | Delay Loaded DLLs | |

Properties (Release) C/C++|Code Generation|Runtime Library: Multithreaded (/MT)

| ration: Release | Platform: Active(Win32) | |
|-------------------------|---|----------------------|
| | | |
| mmon Properties | Enable String Pooling | No |
| infiguration Properties | Enable Minimal Rebuild | No |
| General | Enable C++ Exceptions | Yes (/EHsc) |
| Debugging | Smaller Type Check | No |
| ŀC/C++ | Basic Runtime Checks | Default |
| General | Runtime Library | Multi-threaded (/MT) |
| Optimization | Struct Member Alignment | Default |
| Preprocessor | Buffer Security Check | Yes |
| Code Generation | Epable Euroction-Level Linking | Vec (/Cu) |
| | Linable Function-Level Linking | 105 (/ 47) |

Properties (Release)
 Linker|General|Output File:
 \$(OutDir)\\$(ProjectName).pyd
 Linker|Input|Additional Dependencies:
 python27.lib

| ration: Release | Platform: Active(Win32) | | |
|-------------------------|---|---|--------------------|
| ommon Properties | Output File | \$(OutDir)\\$(ProjectName).pyd | |
| onfiguration Properties | Show Progress | Not Set | |
| General | Version | | |
| ···· Debugging | Enable Incremental Linking | No (/INCREMENTAL:NO) | |
|]·· C/C++ | Suppress Startup Banner | Yes (/NOLOGO) | |
|] Linker | Ignore Import Library | No | |
| General | Register Output | No | |
| | mmon Properties nfiguration Properties | Additional Dependencies Ignore All Default Libraries | python27.lib No |
| | ·· Debugging | Module Definition File | |
| | - C/C++ | Add Module to Assembly | |
| | Linker | Embed Managed Resource File | |
| 2.24.42.04 | General | Force Symbol References | |
| 2:26:43 PIVI | Input | Delay Loaded DLLs | |

Step 3: Run Debug configuration

 Copy demo.h, demo.cpp, demo_wrap.h, demo_wrap.cpp into project and add as existing items



Add sample code for stdafx.h

// TODO: reference additional headers your program requires here #import "libid:6FCCEDE0-179D-4D12-B586-58C88D26CA78" no namespace raw : rename ("esriProductCode", "esriVersionProductCode") rename ("VersionP #import "esriFramework.olb" raw interfaces only raw native types no nar #import "esriSystem.olb" raw interfaces only raw native types no names; #import "esriSystemUI.olb" raw interfaces only raw native types no name #import "esriEditor.olb" raw interfaces only raw native types no namesp #import "esriGeodatabaseUI.olb" raw interfaces only raw native types no #import "esriArcMapUI.olb" raw interfaces only raw native types no name #import "esriGeometry.olb" raw interfaces only raw native types no name #import "esriDisplay.olb" raw interfaces only raw native types no names #import "esriGeoDatabase.olb" raw interfaces only raw native types no : #import "esriDataSourcesFile.olb" raw interfaces only raw native types #import "esriDataSourcesRaster.olb" raw interfaces only raw native type #import "esriDataSourcesGDB.olb" raw interfaces only raw native types : #import "esriCarto.olb" raw interfaces only raw native types no namespa #import "esriEditorExt.olb" raw interfaces only raw native types no nar #import "esriNetworkAnalysis.olb" raw interfaces only raw native types #import "esriOutput.olb" raw interfaces only raw native types no names #import "esriOutputUI.olb" raw_interfaces_only raw_native_types no_name

Add sample code for standalonedemo.cpp

```
#include "stdafx.h"
#include <Python.h>
#include "demo wrap.h"
static PyMethodDef module methods[] = {{NULL}}; // No module-level methods
PyMODINIT FUNC
initstandalonedemo(void)
ł
    PyObject* m;
   if (PyType Ready(&DemoType) < 0)
        return:
   m = Py_InitModule3("standalonedemo", module_methods,
                       "Example module that creates an extension type.");
    if (m == NULL)
      return:
    Py INCREF(&DemoType);
    PyModule AddObject(m, "demo", (PyObject *)&DemoType);
```

Have a shapefile or file geodatabase handyBuild and run:

>>> from standalonedemo import demo >>> d = demo()>>> d.Init()>>> S =d. Inventory("C: /apps/Data/AZHi ghways/AZHi ghways. gdb", "AZ_Highways") c:\apps\Development\PythonC\src\PCbuild\python_d.exe Python 2.7.2 (default, Mar 14 2012, 15:10:23) [MSC v.1500 32 bit (Intel)] on w >>> print s Type "help", "copyright", "credits" or "license" for more information. >>> from standalonedemo import demo [42599 refs] >>> d = demo<> [42601 refs] >>> d.Init() True [42603 refs] >>> s = d.Inventory("C:/apps/Data/AZHighways/AZHighways.gdb", "AZ_Highways"> [42605 refs] >>> print s Demo: Count = 1067Average length = 0.100254[42605 refs] >>> _ - 🔋 🖉 🕒 🛒 🖓 Show output from: Debug 'python_d.exe': Loaded 'C:\Program Files\ArcGIS\Desktopl0.l\bin\SchematicLib.dll' Allocated blocks found: Dumping objects -> (1976) normal block at 0x00BB73A8, 64 bytes long. 2:26:43 PM Data: <Demo: Count => 44 65 6D 6F 3A 0A 20 20 20 43 6F 75 6E 74 20 3D Object dump complete.

demo_wrap.h:

```
#include "demo.h"
typedef struct
{
        Py0bject_HEAD;
        demo *demo;
} Demo;
```

extern PyTypeObject DemoType;

demo_wrap.cpp:

#include "stdafx.h"
#include <Python.h>
#include "demo_wrap.h"

PyMethodDef Demo_methods[] = {
 {"Init", (PyCFunction)Demo_Init, METH_NOARGS,
 "demo.Init(): Initialize COM and ArcObjects"},
 {"Inventory", (PyCFunction)Demo_Inventory,
METH_VARARGS,
 "demo.Inventory(string ws, string fc):
inventory feature class fc in workspace ws"},
 {NULL}
};

demo_wrap.cpp (continued):

Py_TPFLAGS_DEFAULT | Py_TPFLAGS_BASETYPE, /*tp_flags*/
"demo object", /*tp_doc*/

Demo_methods,/*tp_methods*/ // Methods only0,/*tp_members*/ // No properties

Demo_new,

/*tp_new*/

....

...

};

demo_wrap.cpp (continued):

```
Py0bject * Demo_new(PyType0bject *type, Py0bject
*args, Py0bject *kwds)
```

```
Demo *self:
self = (Demo *)type->tp_alloc(type, 0);
if (self != NULL)
       self->demo = new demo();
       if (self->demo == NULL)
              Py_DECREF(sel f);
              return NULL;
ł
return (Py0bject *) sel f;
          void Demo_dealloc(Demo* self)
          {
                 delete self->demo;
                 self->ob_type->tp_free((PyObject*)self);
          }
```

}

demo_wrap.cpp (continued): PyObject * Demo_Init(Demo *self) if (self->demo == NULL) return NULL; bool bResult; try bResult = self - >demo - >Init();catch(...) bResult = false; if (!bResult) Py_INCREF(Py_False); return Py_False; Py_INCREF(Py_True); return Py_True; }

demo_wrap.cpp (continued):

```
PyObject * Demo_Inventory(Demo *self, PyObject *args)
      if (self->demo == NULL)
             return NULL:
      const char *ws, *fc;
      if (!PyArg_ParseTuple(args, "ss", &ws, &fc))
             printf("Usage: demo. Inventory((string) workspace,
(string) featureclass) \n");
             return NULL;
      char *s = self->demo->Inventory((char *)ws, (char *)fc);
      PyObject * pyResult = Py_BuildValue("s", s);
      return pyResult;
```

Debugging with python_d

Cannot use site packages
 Cannot debug inside ArcMap
 However, you can use automation for debugging

[Sample Code: PythonC/arcmapdemo]

2:26:43 PM

| - 🗈 🐟 🗳 🗉 | |
|---|--|
| | 🚺 🔤 c:\apps\Development\PythonC\src\PCBuild\python_d.exe |
| Arizona Highways Highways HWY_TYPE Interstate Hwy US Hwy State Hwy State Hwy ✓ Highways I've Driven ✓ State Boundary ✓ County Boundaries ✓ State Background | <pre>>>> execfile("/demo.py") STATUS: Querying Highways STATUS: Querying State Boundary STATUS: Querying State Boundary STATUS: Querying County Boundaries STATUS: Querying State Background Map Inventory: Highways: Count = 478 Average length = 0.099274 Highways I've Driven: Count = 318 Average length = 0.136567 State Boundary: Count = 1 Average length = 23.257268 Average area = 28.859090 County Boundaries: Count = 1 Average length = 7.373075 Average area = 28.859090 Iderage length = 23.257268 Average area = 28.859090 Iderage area = 28.859090 I</pre> |
| | |

Can you incorporate .NET (CLR) forms?

Yes, but there are several potential pitfalls
 Keep ArcObjects in unmanaged code!!!
 #pragma managed(push, off) #pragma managed(pop)
 RECOMMENDED: Put forms in a separate pyd
 Better yet, why not use wxPython instead?

[Sample Code: PythonC/formdemo]

| 🔤 c:\apps\Development\PythonC\src\PCbu | ild\python_d.e> | ke 🛛 | |
|--|-----------------|-----------------|---------------|
| Python 2.7.2 (default, Mar 14 2 | 012, 15:10:2 | 23> [MSC v.1500 | 32 bit (Inte] |
| 52 Type "help", "copyright", "cred | its" or "lie | cense" for more | information. |
| >>> import formdemo [42617 refs] | 🛃 ResultForm | | |
| <pre>/// formuemo.lestResultrorm()</pre> | OK | | |
| | | | <u></u> |
| | | | |
| | | | |
| | | | |
| | | | |

Why use SWIG or Cython?

Pros:

- Useful for wrapping existing code
- Save time creating wrapper code for new code

Cons:

- Create baggage
- Still require Python development package
- Still must debug in python_d
- Cannot directly consume and return ArcObjects

Why use SWIG?

Pros:

- Supports multiple languages: same C++ can be wrapped for Python and C#
- No compilation required just unzip and use
- Fast, easy setup: for many cases requires just a few lines of configuration code

Cons:

- Produces a python wrapper along with the pyd
- Requires creating typemap wrapper code for Python objects
- Does not support callbacks to Python

Walkthrough 2: Creating a Python module using SWIG

[Sample Code: PythonC/swigdemo]

2:26:43 PM

Step 1: Set up SWIG

- Set up Python development package as in walkthrough #1 step 1
- Download swigwin-2.0.4.zip http://www.swig.org/download.html
- Unzip swigwin-2.0.4.zip to temporary folder
- Create a folder called "swig" adjacent to where you intend to create Python extension projects
- Move contents of swigwin-2.0.4 folder to swig folder



Step 2: Create project

- Create a new project as in walkthrough #1 step 2, called swigdemo
- Properies (debug)
 Linker|General|Output File: \$(OutDir)_\$(ProjectName)_d.pyd
- Properties (release)
 Linker|General|Output File: \$(OutDir)_\$(ProjectName).pyd
- Add new items: swigdemo.i, and swigdemo_wrap.cpp
- Right-click swigdemo.i and bring up Properties (All Configurations) Custom Build Setup|General|Command Line:

...\swig\swig.exe -c++ -python -o
\$(ProjectDir)\\$(InputName)_wrap.cpp \$(InputPath)

Custom Build Setup|General|Outputs: \$(ProjectDir)\\$(InputName)_wrap.cpp

 Right-click swigdemo_wrap.cpp and bring up Properties (All Configurations)

C/C++|Precompiled Headers|Create/Use Precompiled Header: Not Using Precompiled Headers

2:26:44 PM

Step 3: Build and run Debug configuration

- Copy demo.h and demo.cpp into project and add as existing items Add sample code for stdafx.h and swigdemo.i
- Build Debug configuration
- Copy swigdemo.py to Debug folder
- Run python_d:
 - >>> from swigdemo import demo
 - >>> d = demo()
 - >>> d. Init()
 - >>> s =

d. Inventory("C:/apps/Data/AZHi ghways/AZHi ghways.gdb", "AZ_Hi ghways")

>>> print s

Why use Cython?

Pros:

Produces a standalone pyd

Produces cleaner C wrappers

Cons:

To install, Cython uses distutils to compile pyds
Distutils expects VC 2008 unless you tweak it
Requires writing some wrapper code (in Pyrex)

Walkthrough 3: Creating a Python module using Cython

[Sample Code: PythonC/cythondemo]

2:26:44 PM

Step 1: Set up Cython

- Set up Python development package as in walkthrough #1 step 1
- Download Cython-0.15.1.zip http://cython.org/release/Cython-0.15.1.zip
- Unzip to temporary folder
- Verify that Visual C++ 2008 is installed
- Open a command prompt and cd to folder containing setup.py python setup.py install

Step 2: Create project

Create a new project as in walkthrough #1 step 2, called cythondemo

- Make sure python is in the PATH environment
- Add new items: cythondemo.pyx, and cythondemo_wrap.cpp
- Right-click cythondemo.pyx and bring up Properties (All Configurations) Custom Build Setup|General|Command Line: python -c "from Cython.Compiler.Main import compile; compile('cythondemo.pyx', output_file='\$(InputName)_wrap.cpp', cplus=True)"

Custom Build Setup|General|Outputs: \$(ProjectDir)\\$(InputName)_wrap.cpp

 Right-click cythondemo_wrap.cpp and bring up Properties (All Configurations)

C/C++|Precompiled Headers|Create/Use Precompiled Header: Not Using Precompiled Headers

Step 3: Build and run Debug configuration

- Copy demo.h and demo.cpp into project and add as existing items Add sample code for stdafx.h and cythondemo.pyx
- Build Debug configuration
- Run python_d:
 - >>> from cythondemo import PyDemo
 - >>> d = PyDemo()
 - >>> d. Init()
 - >>> s =
 - d. Inventory("C:/apps/Data/AZHi ghways/AZHi ghways.gdb", "AZ_Hi ghways")
 - >>> print s

"And now for something completely different..."

2:26:44 PM

Why use ctypes?

Pros:

- No Python development package required
- Code can be debugged inside an ArcMap session
- Can easily consume and return comtypeswrapped ArcObjects
- The DLL produced can be used in both Python and .NET

Cons:

Only C exported functions can be used by ctypes:
 C++ objects cannot be directly exposed

Installing comtypes and wxPython and creating Python add-ins Presentation: ArcMap and Python: Closing the VBA Gap

ctypes – A foreign function library for Python

http://docs.python.org/library/ctypes.html

Walkthrough 4: Creating a helper module for ArcMap using ctypes

[Sample Code: PythonC/ctdemo]

2:26:44 PM

The maphelper class:

```
// maphelper.h
#include <string>
using namespace std;
typedef void *StatusCallback(char * msg);
class maphelper
private:
        StatusCallback *_pStatusCall;
       string _sResult;
public:
       maphelper();
       void Init(StatusCallback *pStatusCall);
        IMap * GetFocusMap();
       char * Inventory(IMap * ipMap, IEnvelope * ipExtent);
private:
       void IntMapInventory(IMap * ipMap, IEnvelope * ipExtent);
       void Status(string sMsg);
};
```

Step 1: Create project

Project type: Win32 Project

Name: ctdemo

Uncheck "Create directory for solution"

Application type: DLL

Properties (all configurations)

C/C++|General|Additional Include Directories:

[C:\Program Files*]\ArcGIS\Desktop10.1\com

- *wherever ArcGIS is installed
- Properties (debug)
 Debugging Command:

Debugging|Command:

[C:\Program Files*]\ArcGIS\Desktop10.1\bin\ArcMap.exe *wherever ArcGIS is installed

Properties (release)
 C/C++|Code Generation|Runtime Library: Multi-threaded (/MT)

Step 2: Run Debug configuration

- Copy the following into project and add as existing items: maphelper.h, maphelper.cpp
- Add sample code for stdafx.h and ctdemo.cpp
- Copy ctdemo.py and demo.py to project folder
- Build and run:
 - Open an mxd
 - Open ArcMap's Python window
 - >>> sProjPath = "c:/apps/Development/PythonC/ctdemo"
 - >>> execfile(sProjPath + "/demo.py")

| 2 🔍 | | Python | | | |
|----------------------|----------|---|--|--|--|
| zona Highways | | >>> sProjPath = "c:/appg/Davalopment/DythenC/ctdome" | | | |
| Highways | | <pre>>>> areafile (aProiDath "/dama nu")</pre> | | | |
| HWY_TYPE | | >>> exectile(sprojpath + "/demo.py") | | | |
| 💳 Interstate Hwy | | STATUS: Querying Highways | | | |
| - US Hwy | | STATUS: Querying Highways I've Driven | | | |
| — State Hwy | | STATUS: Querying State Boundary | | | |
| Highways I've Driven | | STATUS: Querying County Boundaries | | | |
| | y | STATUS: Querying State Background | | | |
| Major Roads | 1 | Map Inventory: | | | |
| State Boundary | <u>/</u> | Highways: | | | |
| County Boundaries | | Count = 675 | | | |
| State Background | | Average length = 0.103588 | | | |
| | | Highways I've Driven: | | | |
| | | Count = 421 | | | |



// ctdemo. cpp

```
#include "stdafx.h"
#include "maphelper.h"
```

```
extern "C" __declspec( dllexport )
void * maphelper_New()
```

```
maphelper *m = new maphelper;
return (void *)m;
```

```
extern "C" __declspec( dllexport )
void maphelper_Init(void * self, StatusCallback *pStatusCall)
{
    maphelper *m = (maphelper*)self;
    m->Init(pStatusCall);
```

}

}

// ctdemo. cpp (continued)

```
extern "C" __declspec( dllexport )
char * maphelper_Inventory(void * self, IMap * ipMap,
IEnvelope * ipExtent)
ł
      maphelper *m = (maphelper*)self;
      return m->Inventory(ipMap, ipExtent);
}
extern "C" __declspec( dllexport )
void maphelper_Delete(void * self)
      maphelper *m = (maphelper*) self;
      delete m;
}
```

class maphelper_factory: $_dl l = None$ def __init__(self, path=None): from os import getcwd from ctypes import cdll # NOT windll! if not (self._dll is None): return Beware: each call if path is None: to LoadLibrary sPath = getcwd() loads a new el se: instance of the dll! sPath = pathsPath += "\\ctdemo. dll" self._dll = cdll.LoadLibrary(sPath) from comtypes. client import GetModule sLibPath = GetLibPath() GetModule(sLibPath + "esriArcMapUI.olb") def new_maphelper(self): return maphelper(self._dll)

2:26:44 PM

```
class maphelper:
   dl l = None
   mh = None
   _cb = None
    def __init__(self, dll=None):
        from ctypes import c_void_p
        self._dll = dll
        self._mh = c_void_p(self._dll.maphelper_New())
    def \_del \_(sel f):
        self._dll.maphelper_Delete(self._mh)
    def Init(self, StatusCallback):
        from ctypes import CFUNCTYPE, c_char_p
        STATUSFUNC = CFUNCTYPE(None, c_char_p)
        try:
            self._cb = STATUSFUNC(StatusCallback)
            self._dll.maphelper_Init(self._mh, self._cb)
        except:
            pass
        return
```

2:26:44 PM

```
def GetFocusMap(self):
        from ctypes import POINTER
        import comtypes.gen.esriCarto as esriCarto
        pMap = None
        try:
             pMap = POINTER(esriCarto.IMap) \setminus
(self._dll.maphelper_GetFocusMap(self._mh))
        except:
             pass
        return pMap
    def Inventory(self, pMap, pEnv):
        from ctypes import c_char_p
        sResult = "'
        try:
             sResult = c_char_p(\setminus
self._dll.maphelper_Inventory(self._mh, pMap, pEnv))
             sResult = sResult.value
        except:
             pass
        return sResult
2:26:44 PM
```

sDebugPath = sProjPath + "/Debug" import sys sys. path. insert(0, sProjPath) from ctdemo import * import comtypes.gen.esriCarto as esriCarto h = demohost()mf = maphelper_factory(sDebugPath) m = mf. new_maphelper() m. Init(h. Status) pMap = m. GetFocusMap()pAV = pMap. QueryInterface(esriCarto.IActiveView) pExtent = pAV. Extent s = m. Inventory(pMap, pExtent) print s del m

ArcMap add-in examples (10.1):

Python add-in (ctypes/wxPython): PythonC/addins/pydemo C# add-in (pinvoke): PythonC/addins/csdemo Python add-in (pyd/CLR): PythonC/addins/clrdemo

Questions?

Mark Cederholm <u>mcederholm@uesaz.com</u>
This presentation and sample code may be downloaded at:

http://www.pierssen.com/arcgis10/python.htm