.Net Attributes

Jim Fawcett

CSE681 – Software Modeling and Analysis

Fall 2007

References

- Applied .Net Attributes, Bock and Barnaby, Apress, 2003
- The C# Programming Lanuage, Anders Hejlsberg, et. al., Addison-Wesley, 2004
- Pro C# 2005 and the .Net 2.0 Platfrom, Andrew Troelsen, Apress, 2005
- COM Programming with Microsoft .Net, Templeman, Mueller, Microsoft Press, 2003
- Brown Bag Seminar, Applied .Net Attributes, Mario Tayah, 2005
- MSDN listing of .Net Attribute Hierarchy
- MSDN: Extending Metadata with Attributes

Uses of Attributes in .Net

- Provide custom additions to metadata for managed types
- Support serialization
- Support debugging and tracing
- Set COM+ attributes
  - Activation, queuing, security, events, contexts, object pooling, synchronization, transactions
- Support creation of COM objects
- Support creation of .Net controls
- Support creation of Web Services
- Create ATL Server code – essentially builds ISAPI filters
- Implement performance counters
- Implement OLEDB consumers

Kinds of Attributes

- Custom attributes
  - Add entries to metadata but are not used by run-time
- Distinguished custom attributes
  - These attributes have data stored in the assembly next to the items to which it applies.
  - ***OneWay*** is a distinguished custom attribute that affects marshaling by the run-time
- Pseudo custom attributes
  - Changes, does not extend existing metadata
  - ***Serializable*** is a pseudo custom attribute. It sets or resets the metadata flag tdSerializable

Defining Custom Attributes

- Create a class marked with the AttributeUsage attribute

```
[AttributeUsage(AttributeTargets::All, AllowMultiple=true)]
class myAttribute : System.Attribute { … }
```

- Targets include:
  Assembly, Class, Delegate, Event, Field, Method, …, All

- Typically, the class provides a constructor accepting a value of some type, e.g., string, and a property to retrieve that value.

- The value is stored in the metadata of the assembly that implements the attributed target.

- It is retrieved using the Reflection API.

Using Custom Attributes

- Decorate the target code with the custom attribute:

```
[myAttribute(args)]
public class decoratedClass { … }
```

  This serializes member data of the myAttribute class into metadata for the assembly holding class decoratedClass.

- Now other programs can access this information from the assembly's metadata using reflection:

```
Type t = typeof(target);
object [] obj = Attribute.GetCustomAttributes(t);
```

  Now cast the elements of the obj array to the types of the stored metadata, e.g., the member data of class myAttribute.

Some .Net Provided Attributes

- [CLSCompliant(true)] - class fails to compile if not compliant
- [Conditional("Debug")] - won't get called unless Debug defined
- [Assembly: AssemblyTitle("…")] - assembly descriptions
- [Assembly: AssemblyVersion("1.2")]
- [DllImport("kernel32.dll")] - accessing unmanaged global function
  public static extern int Beep(int freq, int dur);
- [Serializable()] - enabling serialization
  public class myClass { … }
- [OneWay()] - marshal only to remote object
  public void myFunc(string msg) { … }
- [Synchronization()] - allow access by one thread at a time
  class SomeClass : ContextBoundObject { … }

Design-Time and Security Attributes

Attributes used with user defined controls

[Category("Custom Properties")]	- makes property page category
[DefaultEvent(myEvent)]	- double click on control to wire up
[Description("myPropertDesc")]	- description shown when selected
[ToolBoxBitmap("myBitMap.bmp")]	– defines bitmap used in toolbox

Declarative security settings

- [FileIOPermission(SecurityAction.Deny,
  Read=@"c:\Windows\System32")]
  public in ReadFile(string path) { … }

Support Creation of COM Objects

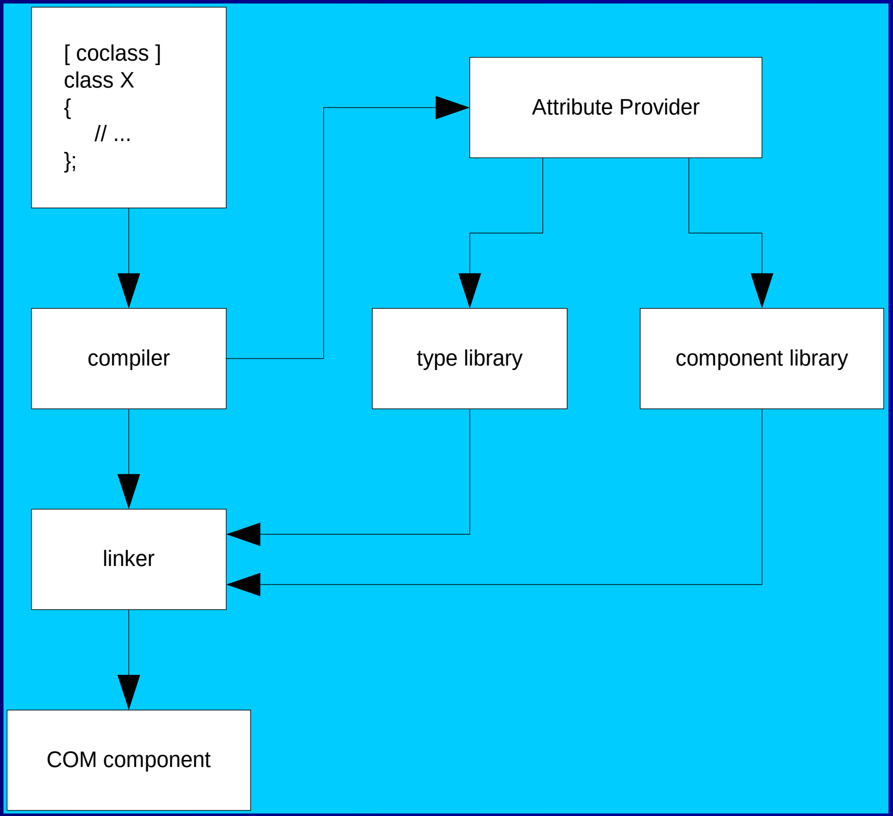
COM Attributes

-
  ```
  [ coclass,  // an implementing class plus COM infrastructure
    threading("apartment"),
    vi_progid("AttribATL.Test"),
    progid("AttribATL.Test.1"),
    version(1.0),
    uuid("CC51A06F-70D1-4113-A821-3756FC45ADF9"),
    helpstring("Test Class")
  ]
  ```

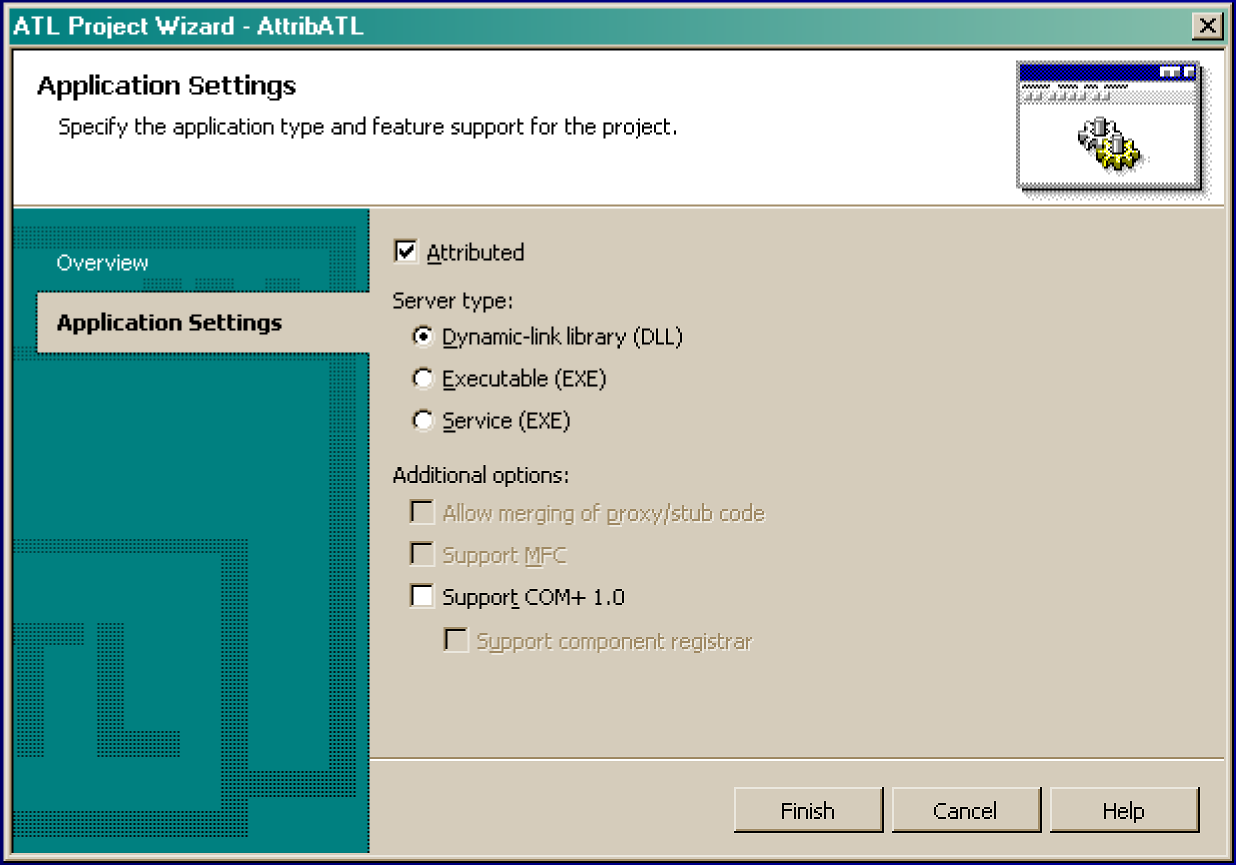
-
  ```
  [ module  // defines a COM server
    (
      dll,
      uuid = "{E9944495-22AF-422F-A011-AE3FD9E17644}",
      name = "AttribATL",
      helpstring = "AttribATL 1.0 Type Library",
      resource_name = "IDR_ATTRIBATL“
     )
  ];
  ```
- object attribute identifies an interface, events , IDL attributes, …

What they do

- Metadata attributes provide data that is used at compile time and/or runtime in an assembly's metadata.
- COM attributes cause code to be generated and injected into the MSIL stream.
- C++ uses two providers:
  - clxx.dll used for type generation and marshaling
  - Atlprov.dll for ATL.



Building Attributed ATL Component



ATL Project Wizard - AttribATL

Application Settings

Specify the application type and feature support for the project.

[✓] Attributed

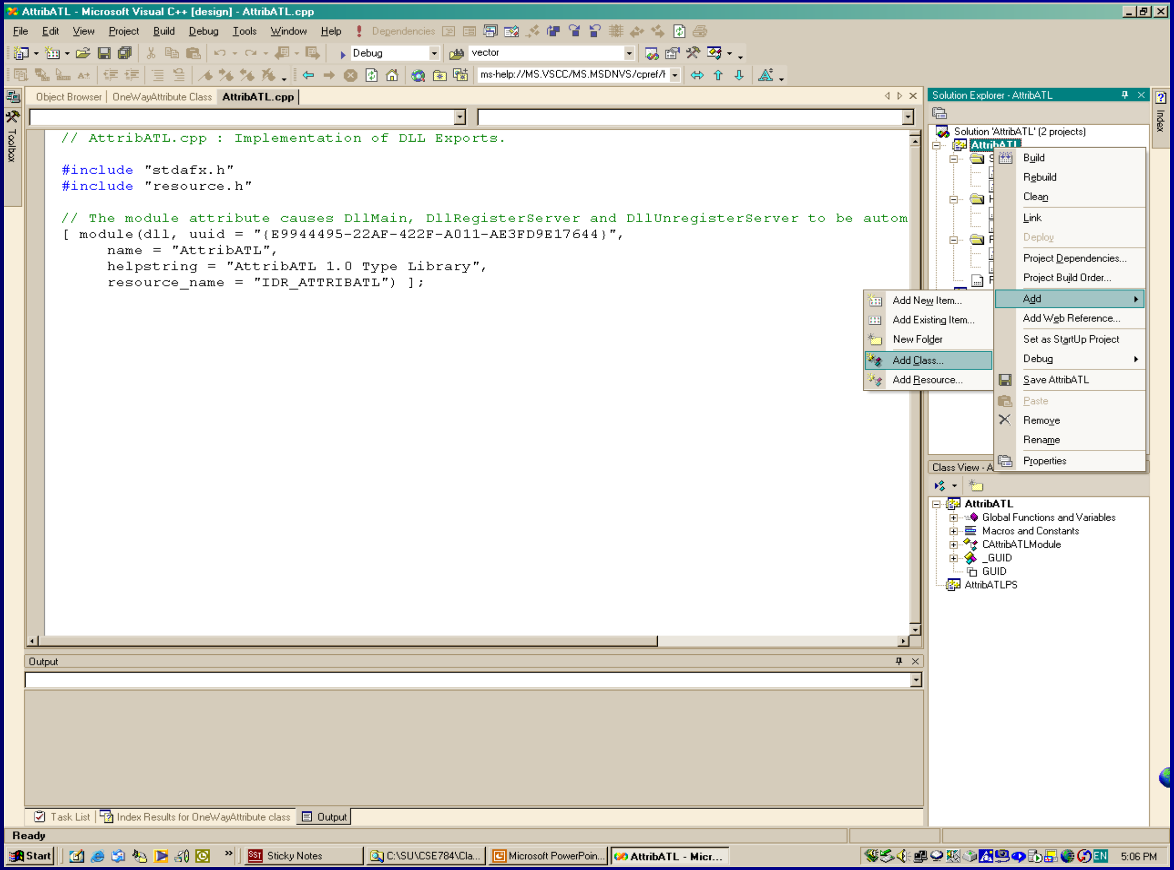
Server type:

- (●) Dynamic-link library (DLL)
- ( ) Executable (EXE)
- ( ) Service (EXE)

Additional options:

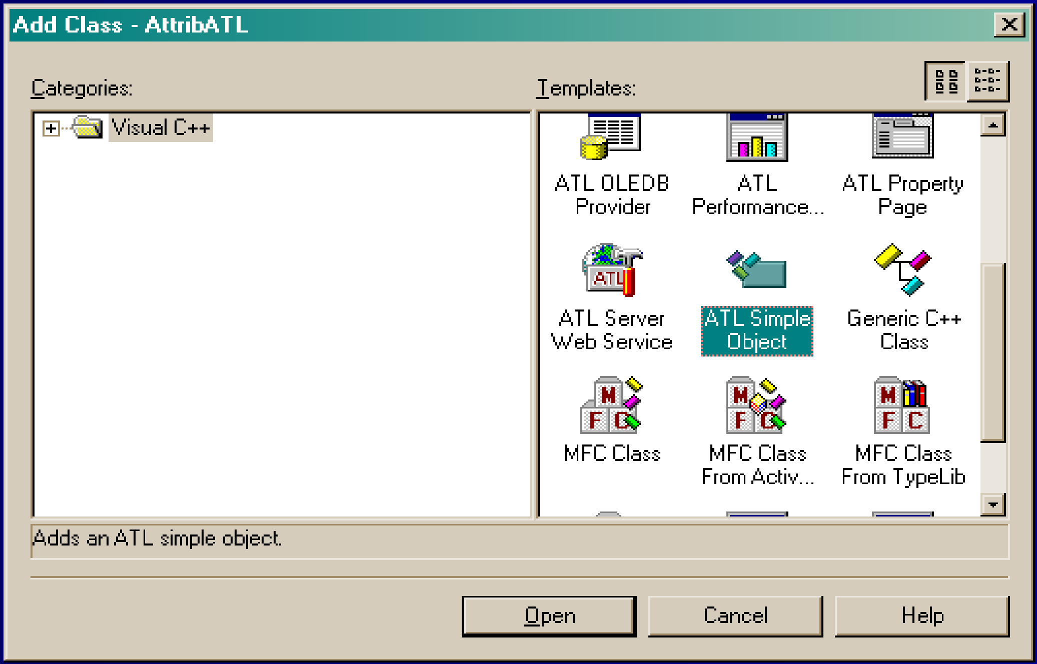
- [ ] Allow merging of proxy/stub code
- [ ] Support MFC
- [ ] Support COM+ 1.0
  - [ ] Support component registrar

Adding a Class



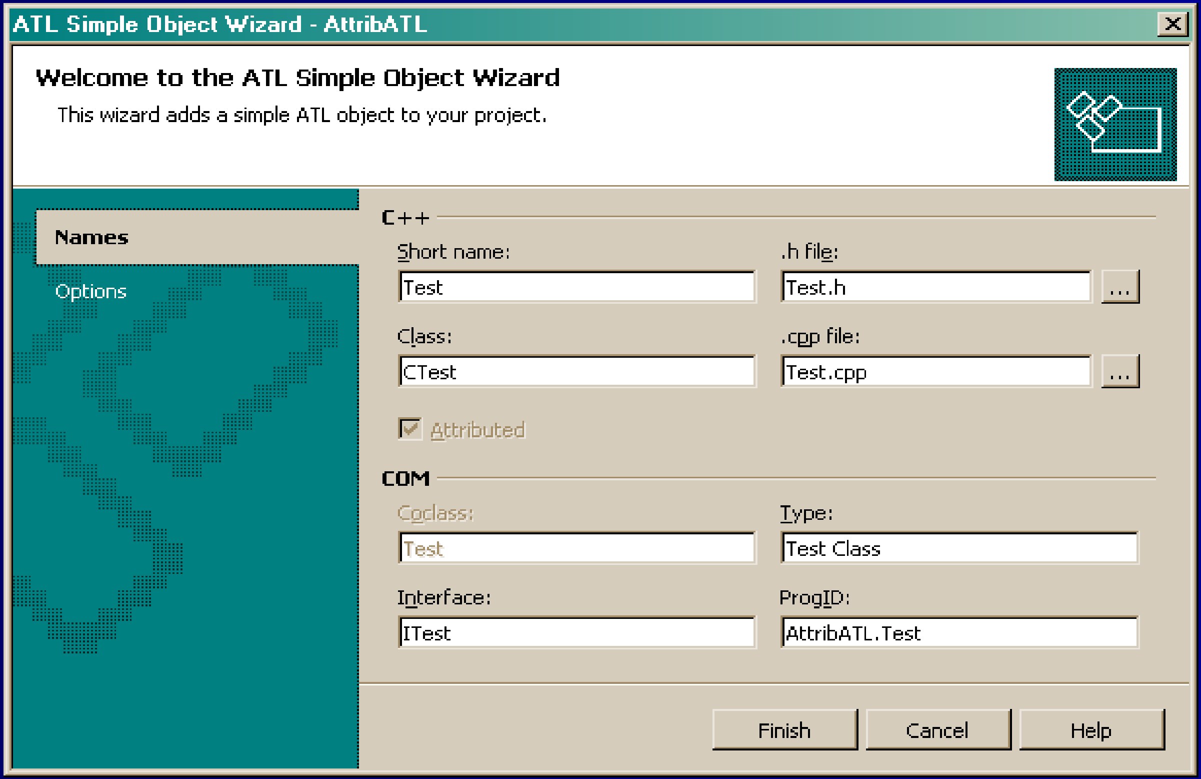
```
// AttribATL.cpp : Implementation of DLL Exports.

#include "stdafx.h"
#include "resource.h"

// The module attribute causes DllMain, DllRegisterServer and DllUnregisterServer to be autom
[ module(dll, uuid = "{E9944495-22AF-422F-A011-AE3FD9E17644}",
     name = "AttribATL",
     helpstring = "AttribATL 1.0 Type Library",
     resource_name = "IDR_ATTRIBATL") ];
```

Adding Simple Object



Adds an ATL simple object.

Test Class


ATL Simple Object Wizard - AttribATL

Welcome to the ATL Simple Object Wizard

This wizard adds a simple ATL object to your project.

C++

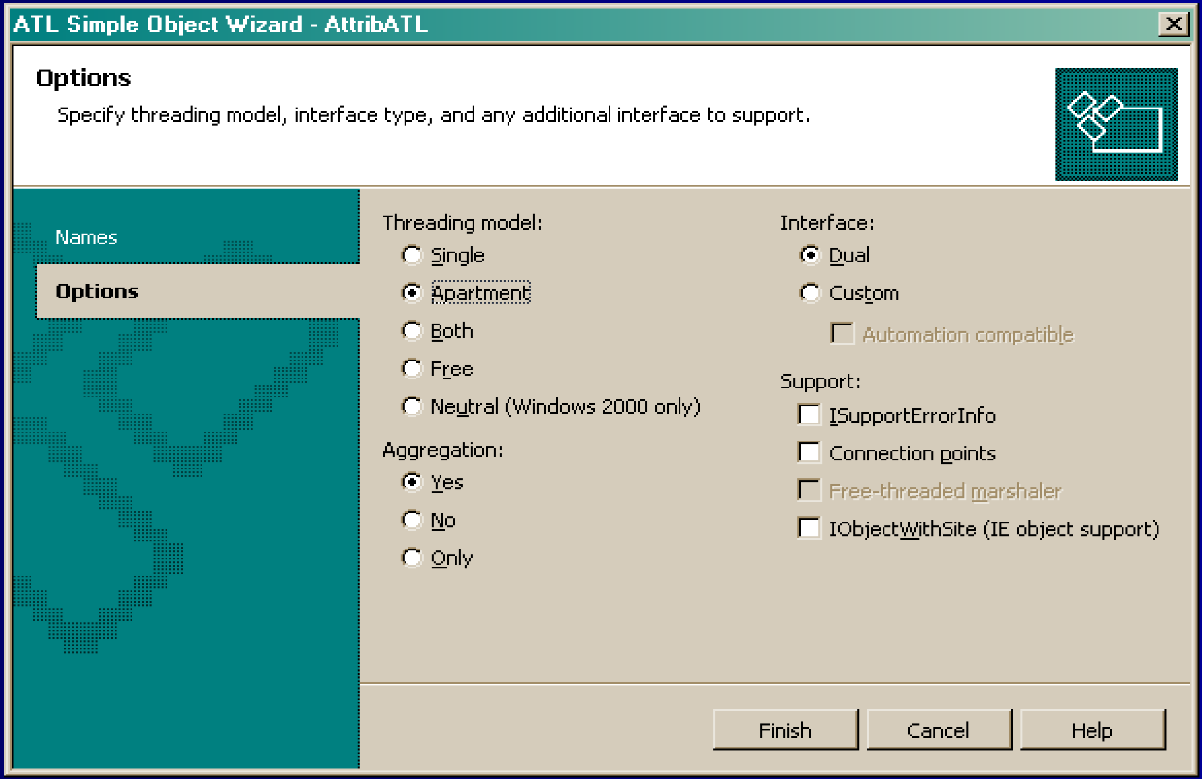
Short name: Test	.h file: Test.h
Class: CTest	.cpp file: Test.cpp

[✓ Attributed]

COM

Coclass: Test	Type: Test Class
Interface: ITest	ProgID: AttribATL.Test

Selecting Attribute Parameters


ATL Simple Object Wizard - AttribATL

Options

Specify threading model, interface type, and any additional interface to support.

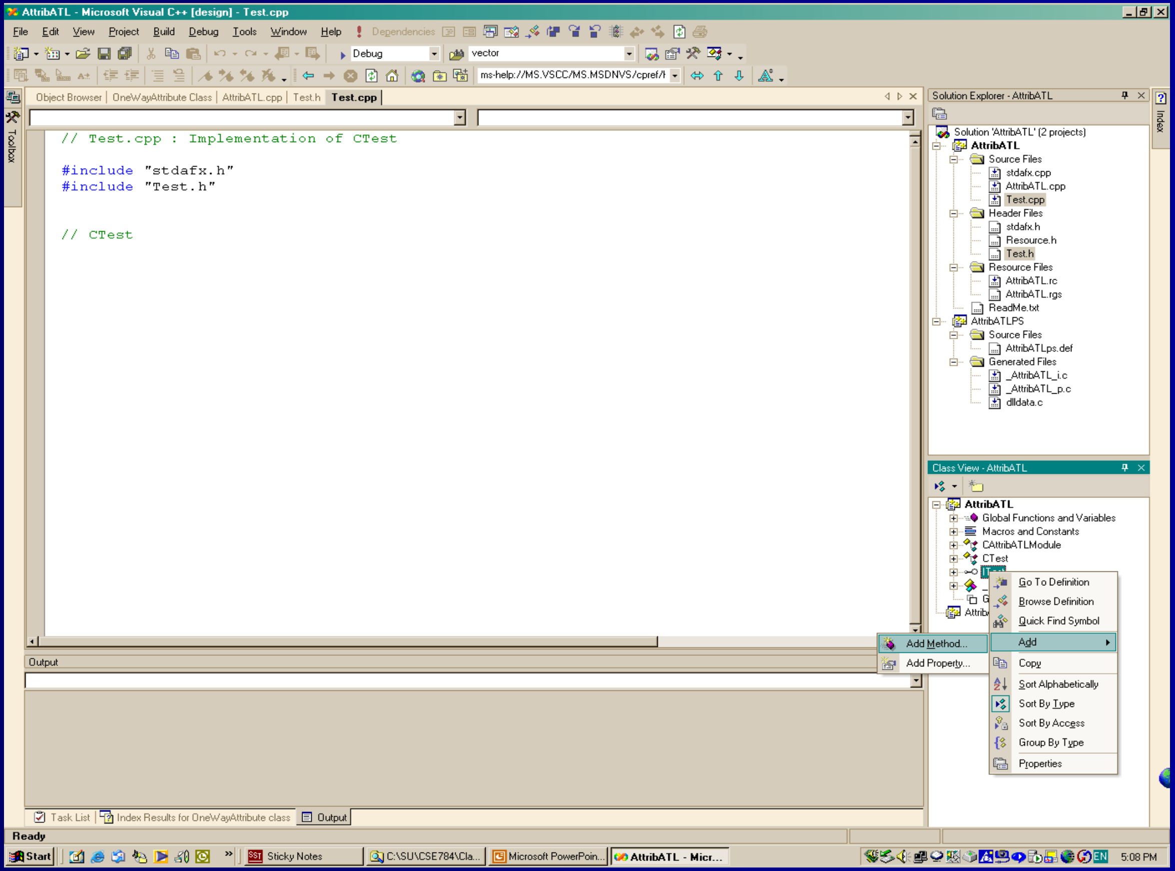
Threading model:
- ( ) Single
- (●) Apartment
- ( ) Both
- ( ) Free
- ( ) Neutral (Windows 2000 only)

Aggregation:
- (●) Yes
- ( ) No
- ( ) Only

Interface:
- (●) Dual
- ( ) Custom
  - [ ] Automation compatible

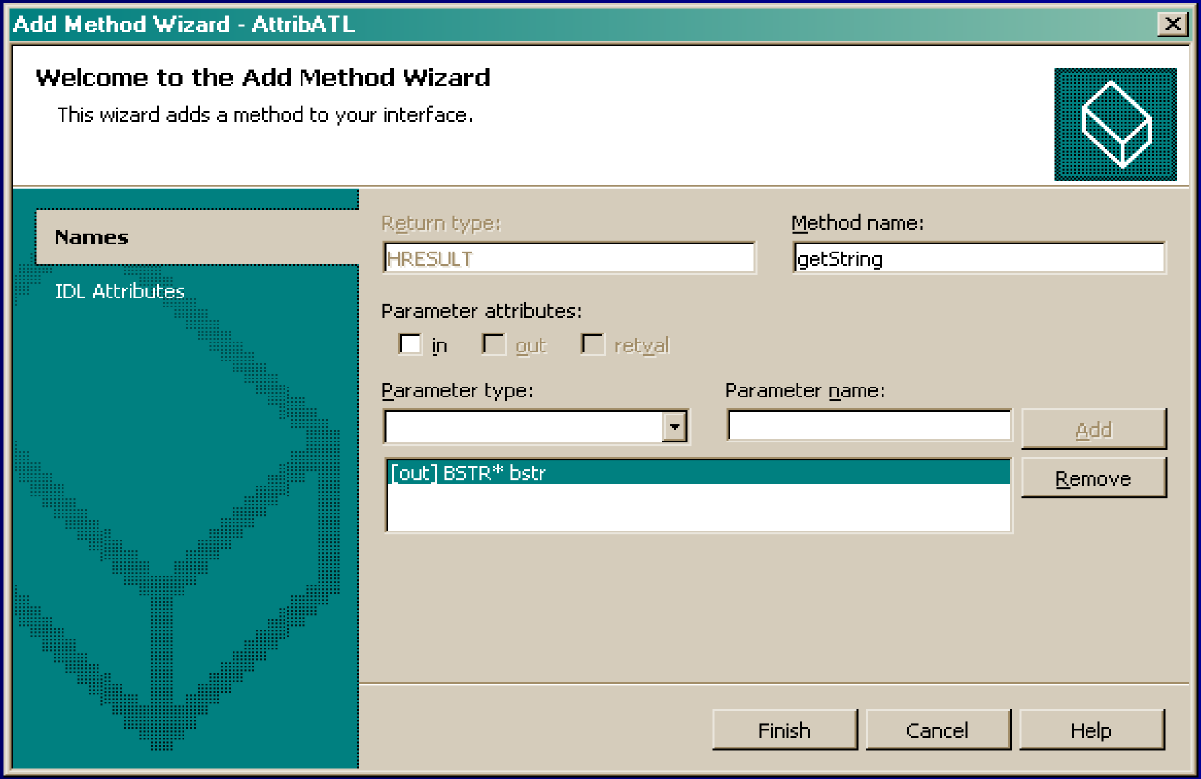
Support:
- [ ] ISupportErrorInfo
- [ ] Connection points
- [ ] Free-threaded marshaler
- [ ] IObjectWithSite (IE object support)

Adding Methods to Interface



```
// Test.cpp : Implementation of CTest

#include "stdafx.h"
#include "Test.h"


// CTest
```

Sending and Receiving Strings


Add Method Wizard - AttribATL

Welcome to the Add Method Wizard

This wizard adds a method to your interface.

Return type: HRESULT

Method name: getString

Parameter attributes:

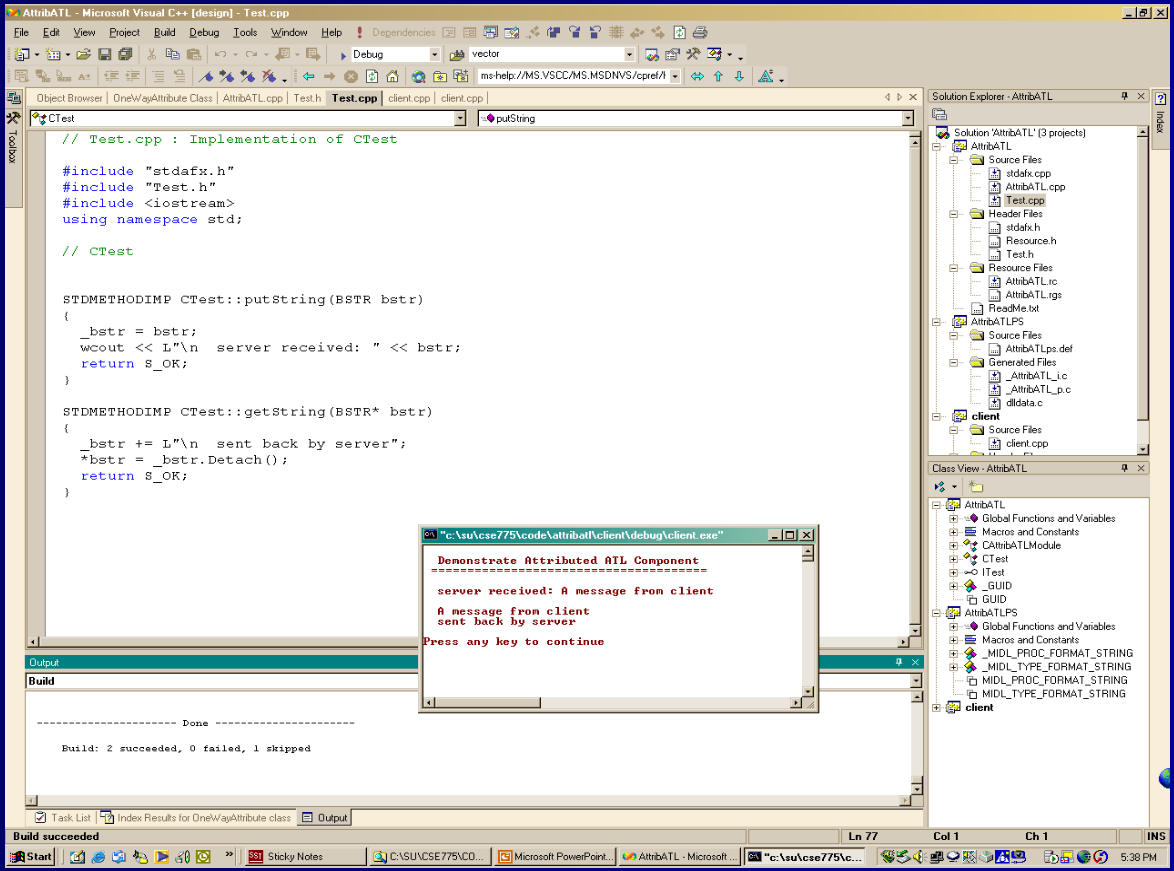
[ ] in [ ] out [ ] retval

Parameter type:

Parameter name:

[out] BSTR* bstr

Client Running Attributed ATL Component


AttribATL - Microsoft Visual C++ [design] - Test.cpp
Object Browser | OneWayAttribute Class | AttribATL.cpp | Test.h | Test.cpp | client.cpp | client.cpp
CTest
putString
Solution Explorer - AttribATL
Class View - AttribATL
Output
Build succeeded
Ln 77 Col 1 Ch 1 INS

```
// Test.cpp : Implementation of CTest

#include "stdafx.h"
#include "Test.h"
#include <iostream>
using namespace std;

// CTest


STDMETHODIMP CTest::putString(BSTR bstr)
{
  _bstr = bstr;
  wcout << L"\n  server received: " << bstr;
  return S_OK;
}

STDMETHODIMP CTest::getString(BSTR* bstr)
{
  _bstr += L"\n  sent back by server";
  *bstr = _bstr.Detach();
  return S_OK;
}
```

"c:\su\cse775\code\attribatl\client\debug\client.exe"

```
 Demonstrate Attributed ATL Component
=======================================

 server received: A message from client

 A message from client
 sent back by server

Press any key to continue
```

```
--------------------- Done ---------------------

    Build: 2 succeeded, 0 failed, 1 skipped
```

End of Presentation