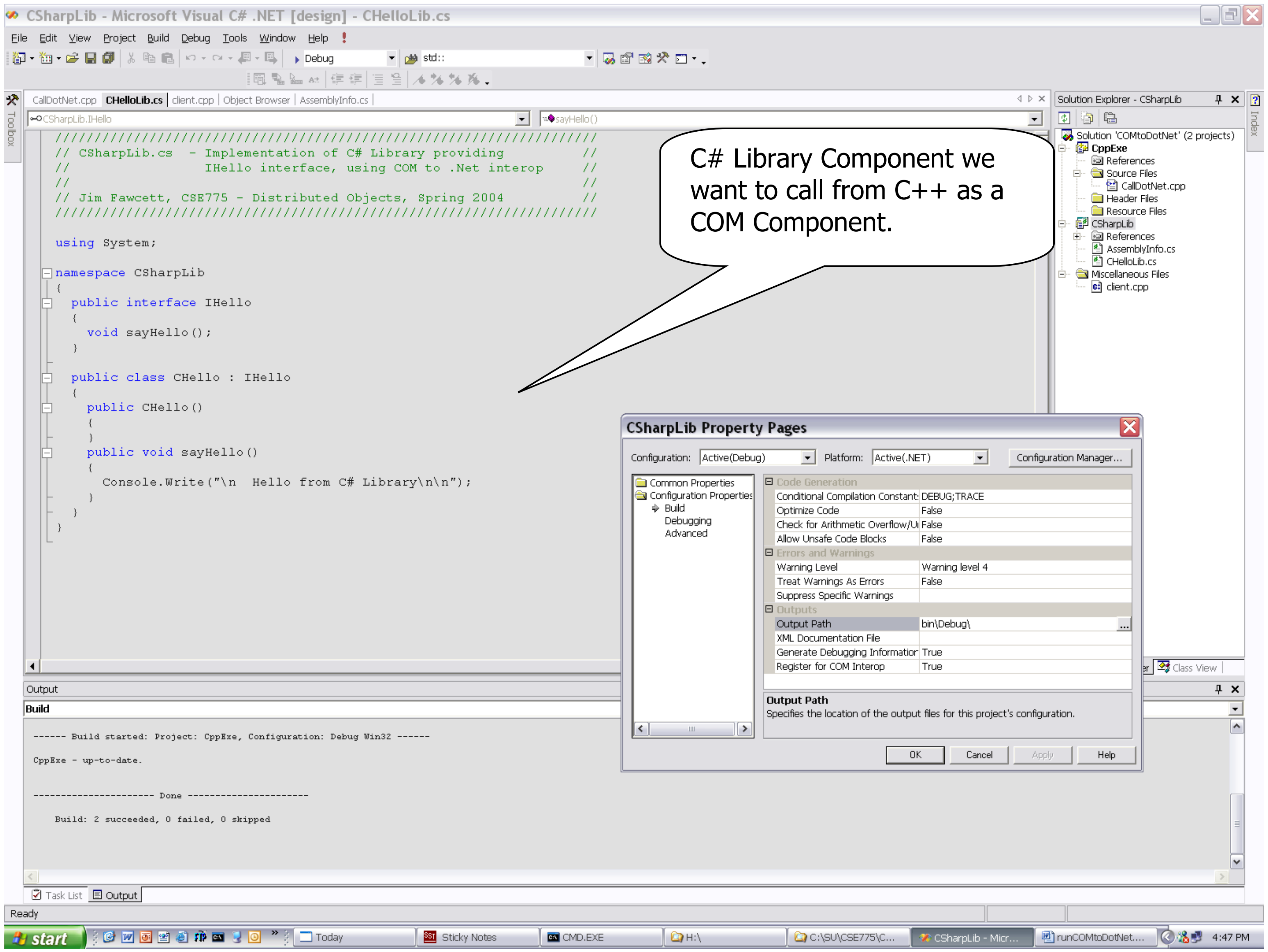
C++ calls .Net Library using COM to .Net Interop

Calling a .Net Component from C++

Jim Fawcett
CSE775 – Distributed Objects
Spring 2004

C++ calls .Net Library using COM to .Net Interop



C# Library Component we want to call from C++ as a COM Component.

```
////////////////////////////////////////////////////////////////////
// CSharpLib.cs  - Implementation of C# Library providing         //
//                 IHello interface, using COM to .Net interop     //
//                                                                //
// Jim Fawcett, CSE775 - Distributed Objects, Spring 2004         //
////////////////////////////////////////////////////////////////////

using System;

namespace CSharpLib
{
  public interface IHello
  {
    void sayHello();
  }

  public class CHello : IHello
  {
    public CHello()
    {
    }
    public void sayHello()
    {
      Console.Write("\n  Hello from C# Library\n\n");
    }
  }
}
```

Output

Build

```
------ Build started: Project: CppExe, Configuration: Debug Win32 ------

CppExe - up-to-date.


---------------------- Done ----------------------

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

CSharpLib Property Pages

Configuration: Active(Debug)   Platform: Active(.NET)   Configuration Manager...

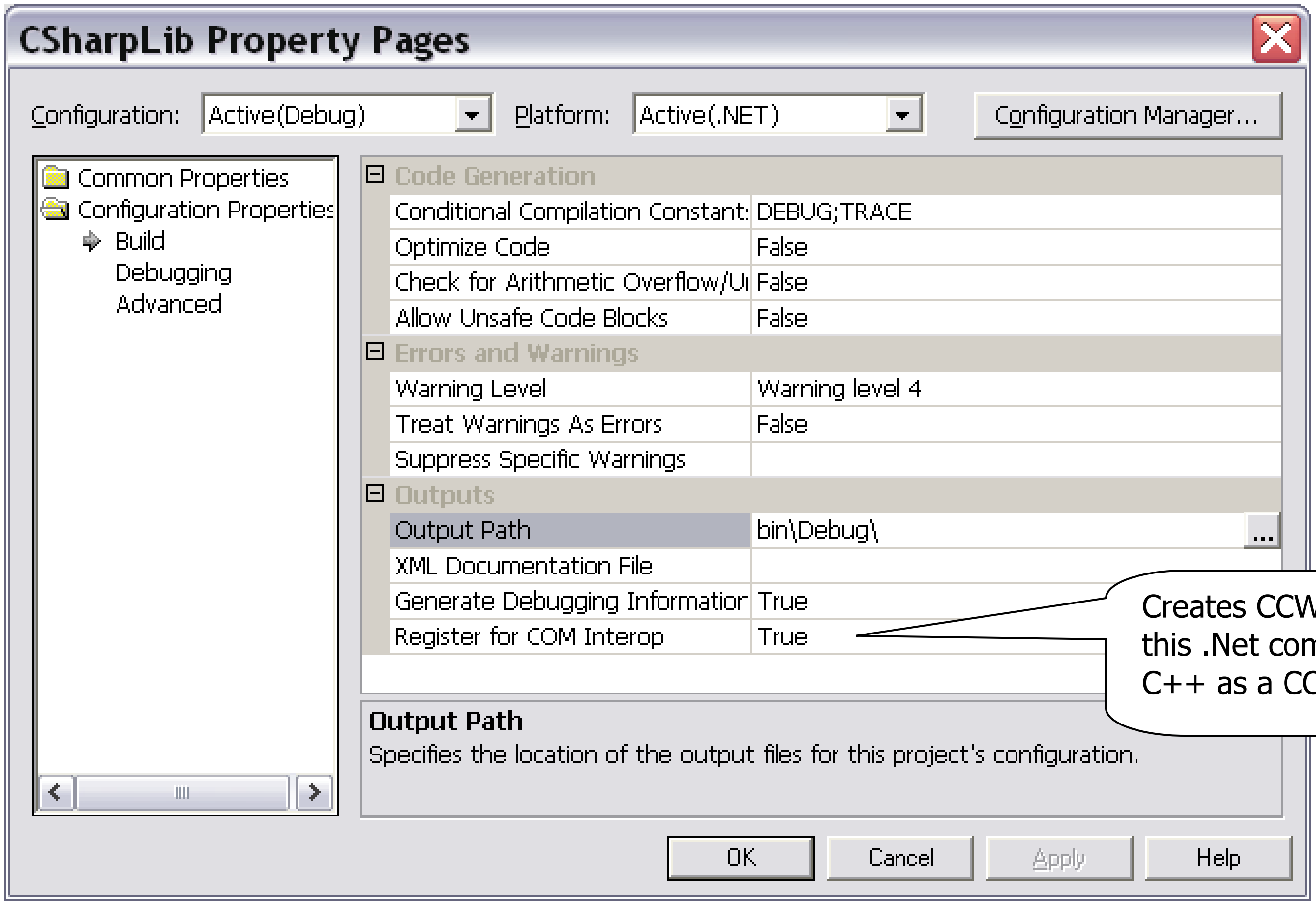
Common Properties
Configuration Properties
- Build
- Debugging
- Advanced

Code Generation	
Conditional Compilation Constant:	DEBUG;TRACE
Optimize Code	False
Check for Arithmetic Overflow/U	False
Allow Unsafe Code Blocks	False
Errors and Warnings	
Warning Level	Warning level 4
Treat Warnings As Errors	False
Suppress Specific Warnings	
Outputs	
Output Path	bin\Debug\
XML Documentation File	
Generate Debugging Informatior	True
Register for COM Interop	True

Output Path
Specifies the location of the output files for this project's configuration.

OK   Cancel   Apply   Help

C++ calls .Net Library using COM to .Net Interop



CSharpLib Property Pages

Configuration: Active(Debug) Platform: Active(.NET) Configuration Manager...

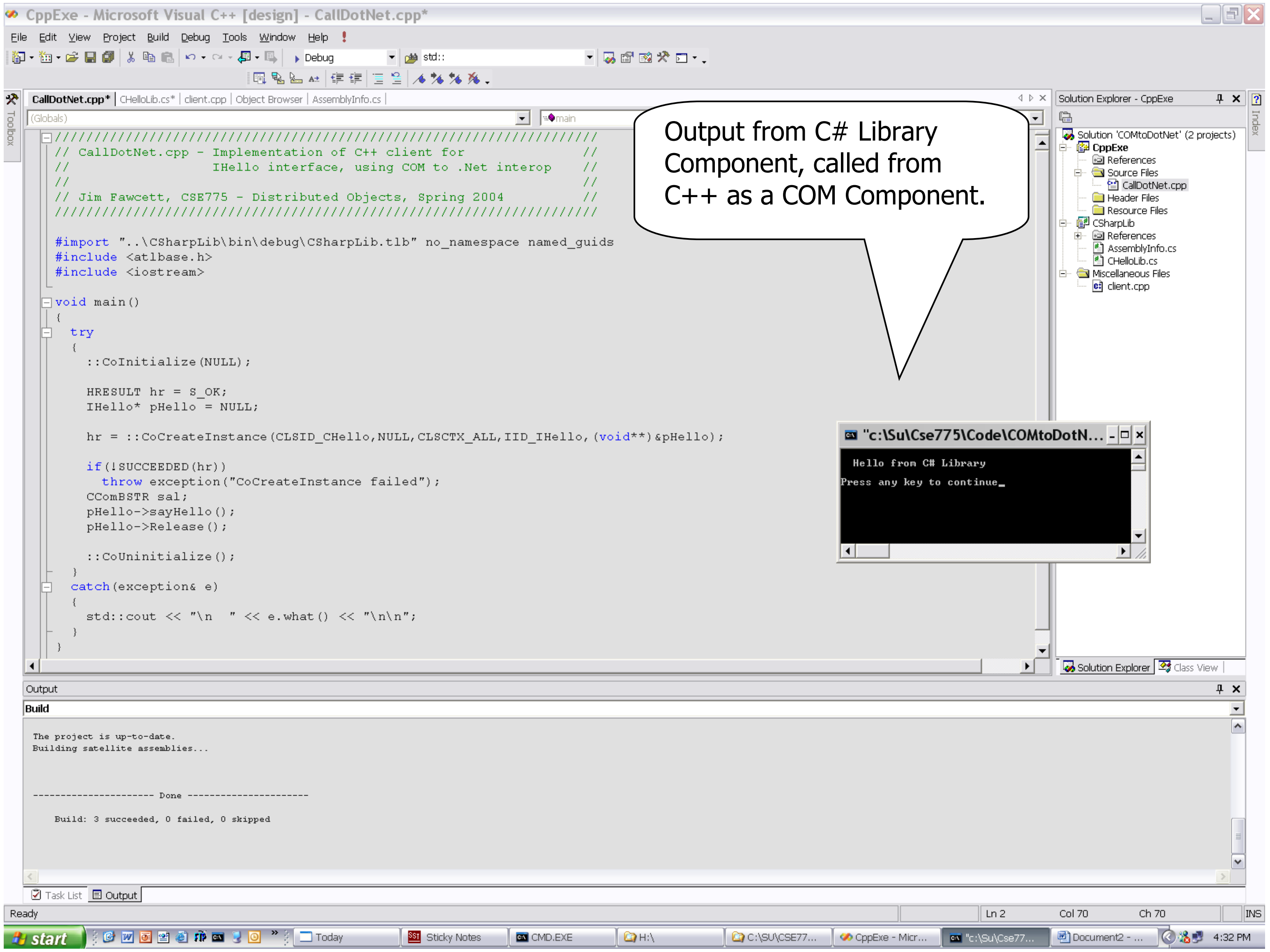
- Common Properties
- Configuration Properties
  - Build
  - Debugging
  - Advanced

Code Generation	
Conditional Compilation Constant:	DEBUG;TRACE
Optimize Code	False
Check for Arithmetic Overflow/U	False
Allow Unsafe Code Blocks	False
Errors and Warnings	
Warning Level	Warning level 4
Treat Warnings As Errors	False
Suppress Specific Warnings	
Outputs	
Output Path	bin\Debug\
XML Documentation File	
Generate Debugging Informatior	True
Register for COM Interop	True

Output Path
Specifies the location of the output files for this project's configuration.

Creates CCW so we can call this .Net component from C++ as a COM component.

C++ calls .Net Library using COM to .Net Interop


CppExe - Microsoft Visual C++ [design] - CallDotNet.cpp*
File Edit View Project Build Debug Tools Window Help
Debug
CallDotNet.cpp* | CHelloLib.cs* | client.cpp | Object Browser | AssemblyInfo.cs
Solution Explorer - CppExe
Solution 'COMtoDotNet' (2 projects)
CppExe
References
Source Files
CallDotNet.cpp
Header Files
Resource Files
CSharpLib
References
AssemblyInfo.cs
CHelloLib.cs
Miscellaneous Files
client.cpp
Solution Explorer
Class View
Task List
Output
Ready
Ln 2
Col 70
Ch 70
INS

Output from C# Library Component, called from C++ as a COM Component.

```
/////////////////////////////////////////////////////////////////////
// CallDotNet.cpp - Implementation of C++ client for              //
//                  IHello interface, using COM to .Net interop    //
//                                                                 //
// Jim Fawcett, CSE775 - Distributed Objects, Spring 2004          //
/////////////////////////////////////////////////////////////////////

#import "..\CSharpLib\bin\debug\CSharpLib.tlb" no_namespace named_guids
#include <atlbase.h>
#include <iostream>

void main()
{
  try
  {
    ::CoInitialize(NULL);

    HRESULT hr = S_OK;
    IHello* pHello = NULL;

    hr = ::CoCreateInstance(CLSID_CHello,NULL,CLSCTX_ALL,IID_IHello,(void**)&pHello);

    if(!SUCCEEDED(hr))
      throw exception("CoCreateInstance failed");
    CComBSTR sal;
    pHello->sayHello();
    pHello->Release();

    ::CoUninitialize();
  }
  catch(exception& e)
  {
    std::cout << "\n  " << e.what() << "\n\n";
  }
}
```

"c:\Su\Cse775\Code\COMtoDotN...

```
 Hello from C# Library
Press any key to continue_
```

Output

Build

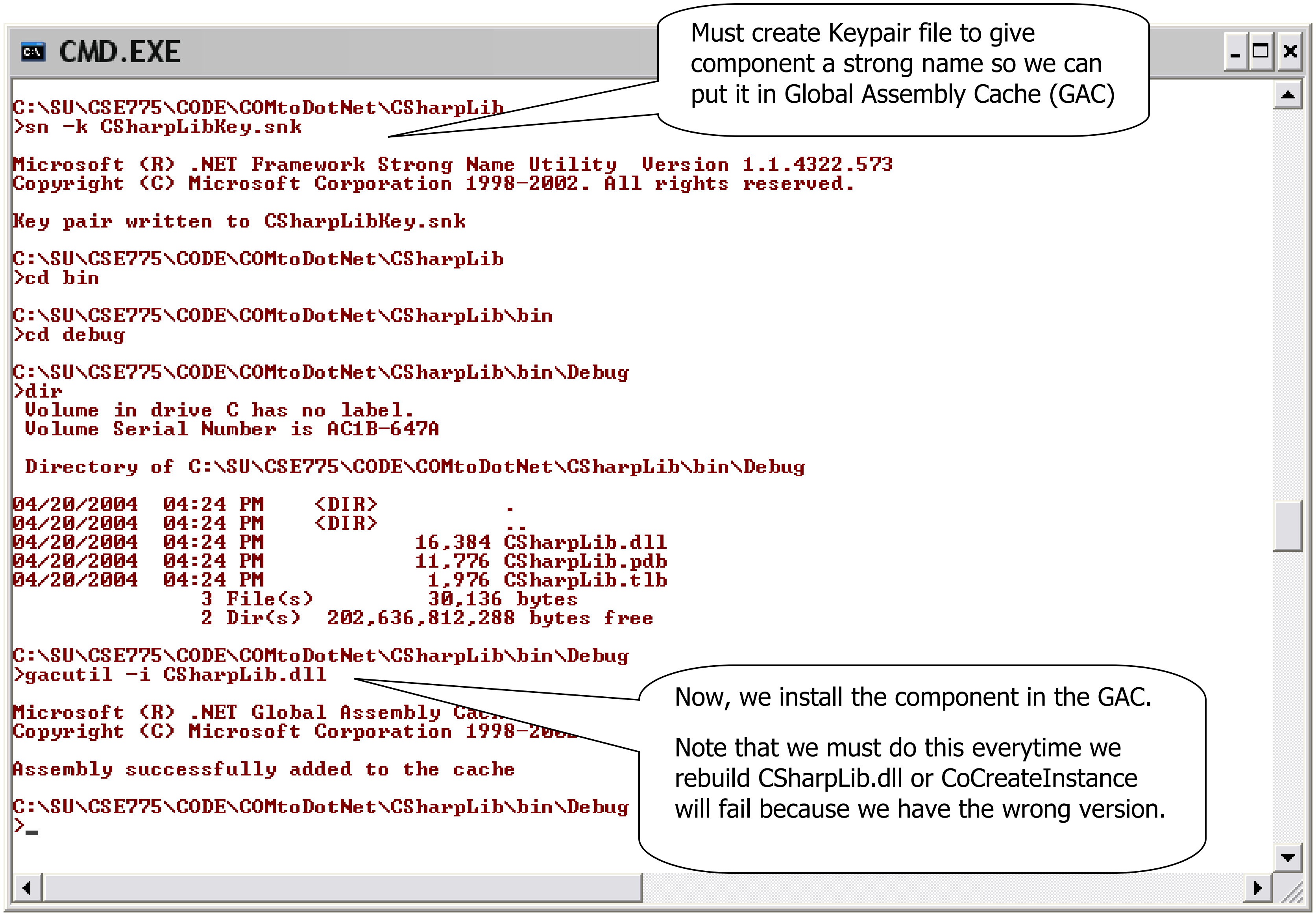
```
The project is up-to-date.
Building satellite assemblies...



--------------------- Done ---------------------

    Build: 3 succeeded, 0 failed, 0 skipped
```

C++ calls .Net Library using COM to .Net Interop



```
C:\SU\CSE775\CODE\COMtoDotNet\CSharpLib
>sn -k CSharpLibKey.snk

Microsoft (R) .NET Framework Strong Name Utility  Version 1.1.4322.573
Copyright (C) Microsoft Corporation 1998-2002. All rights reserved.

Key pair written to CSharpLibKey.snk

C:\SU\CSE775\CODE\COMtoDotNet\CSharpLib
>cd bin

C:\SU\CSE775\CODE\COMtoDotNet\CSharpLib\bin
>cd debug

C:\SU\CSE775\CODE\COMtoDotNet\CSharpLib\bin\Debug
>dir
 Volume in drive C has no label.
 Volume Serial Number is AC1B-647A

 Directory of C:\SU\CSE775\CODE\COMtoDotNet\CSharpLib\bin\Debug

04/20/2004  04:24 PM    <DIR>          .
04/20/2004  04:24 PM    <DIR>          ..
04/20/2004  04:24 PM            16,384 CSharpLib.dll
04/20/2004  04:24 PM            11,776 CSharpLib.pdb
04/20/2004  04:24 PM             1,976 CSharpLib.tlb
               3 File(s)         30,136 bytes
               2 Dir(s)  202,636,812,288 bytes free

C:\SU\CSE775\CODE\COMtoDotNet\CSharpLib\bin\Debug
>gacutil -i CSharpLib.dll

Microsoft (R) .NET Global Assembly Cac
Copyright (C) Microsoft Corporation 1998-20

Assembly successfully added to the cache

C:\SU\CSE775\CODE\COMtoDotNet\CSharpLib\bin\Debug
>_
```

Must create Keypair file to give component a strong name so we can put it in Global Assembly Cache (GAC)

Now, we install the component in the GAC.

Note that we must do this everytime we rebuild CSharpLib.dll or CoCreateInstance will fail because we have the wrong version.

C++ calls .Net Library using COM to .Net Interop

Steps to build are:

1. Create C# library project and write some code to provide one or more classes that are exposed through .Net interfaces.
2. Set the project properties to register for COM interop on the Properties\Configuration Properties\Build tab.
3. Build the library project to create a dll.
4. Create a C++ client written in the usual way to access a COM component, where CLSID and IID are[1]:
    a. CLSID_ClassName
    b. IID_InterfaceName
5. In the project directory create a Keypair file, as shown above.
6. In the debug directory use the gacutil tool to put the library assembly in the GAC.
7. Now you can build the C++ component and run it to call the .Net component.

[1] Here, ClassName and InterfaceName are the class name and interface name, as used in the library source code.