一、在 VS2005 中，C#编写 DLL 并使用 C++调用

（1）C#编写 DLL 程序
建立 C#编写的 DLL 程序 AddDll，
<1>项目类型为：类库

程序代码：

```
using System;
using System.Collections.Generic;
using System.Text;

namespace AddDll
{
    public class Add     //类必须为 public
    {
        public int iadd(int a, int b)   //方法也必须为 public
        {
            int c = a + b;
            return c;
        }
    }
}
```

（2）C++编写调用程序

建立 C++的 Win32 控制台应用程序 UseDll，
<1>项目类型为：Win32 控制台应用程序.
<2>配置：右键点击解决方案资源管理器中的 UseDll，选择“属性”，将公共语言运行库支持设置为“公共语言运行库支持(/clr)”

程序代码：

```
#include "stdafx.h"
#include "stdio.h"

#using "..\debug\AddDll.dll"   //注意，要让程序找到 dll 文件
using namespace AddDll;

int _tmain(int argc, _TCHAR* argv[])
{
        int result;
        Add ^add = gcnew Add();     //注意此处的托管指针
        result = add->iadd(10,90);
        printf("%d",result);
```

```
        scanf("%s");
        return 0;
```

如果操作步骤正确的话，就基本上没有什么问题.

二、在 VS2005 中 C#编写的 COM 组件，使用 VC6.0 或 vc2005 调用

（1）VS2005 中使用 C#编写 COM 组件

<1>建立 C#编写的 COM 组件，项目类型为类库

<2>配置：右键点击解决方案资源管理器中的 AddCom，选择“属性”，选择“生成”，选择“为 COM Interop 注册(_P)”

<3>打开 AssemblyInfo.cs 文件，设置[assembly: ComVisible(true)]
这样就可以生成 AddCom.tlb 文件

程序代码：

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;

//如果对 C#比较熟悉的读者可能很快就明白，不熟悉的读者可能要多摸索几遍
namespace AddCom
{
    //可以通过//菜单的 “工具/guid 生成”。
    //注意要选择 Define Guid{….}格式，并全//部保存下来。
    //因为在做 VC 程序/////////的时候要用到的。

    [Guid("298D881C-E2A3-4638-B872-73EADE25511C")]
    public interface AddComInterface
    {
        [DispId(1)]    //至于此处是什么作用，我也不太明白
        int iadd(int a, int b);
        [DispId(2)]
        string stradd(string   strA, string   strB);
    }

    [Guid("2C5B7580-4038-4d90-BABD-8B83FCE5A467")]
    [ClassInterface(ClassInterfaceType.None)]
    public class AddComService : AddComInterface
    {
```

```
        public AddComService()
        {
        }
        public int iadd(int a, int b)
        {
            int c = 0;
            c = a + b;
            return c;
        }
        public string stradd(string   strA, string   strB)
        {
            return strA+strB ;
        }
    }
}
```

（2）VC6.0 编写调用程序

<1>使用 VC6.0 编写建立 MFC 应用程序 UseCom，项目类型为 MFC AppWizard(exe)

<2>在 stdafx.h 添加：

```
#import "AddCom.tlb"
using namespace AddCom;
```

程序代码：

```
void CUseComDlg::OnButtonUse()
{
        // TODO: Add your control notification handler code here
        int dresult;
        CString strResult;

        CoInitialize(NULL);//NULL 换成 0 也可以

        AddCom::AddComInterfacePtr
p_Add(__uuidof(AddComService));
        dresult = p_Add->iadd(1,2);
        _bstr_t bstr = p_Add->stradd("hello,","world!");  //进入调试
即可查看，
                                                          //这里不再
显示了
        strResult.Format("int:%d ",dresult);
        MessageBox(strResult,"计算结果",MB_OK);
```

```
        CoUninitialize();

}
```

三、在 VC6.0 中编写 COM 组件，使用 VS2005 C#调用
（1）VC6.0 编写 COM,使用 VC6.0 建立 COM 组件，

工程类型：ATL COM AppWizard
程序代码：
接口：interface IAdd：IDispatch

```
        {
                [id(1), helpstring("method iadd")] HRESULT
iadd([in]int a, [in]int b, [out]int * c);
                [id(2), helpstring("method fadd")] HRESULT
fadd([in]float a, [in]float b, [out]float * c);
                [id(3), helpstring("method isub")] HRESULT
isub([in]int a, [in]int b, [out]int * c);
        };
```

实现：STDMETHODIMP CAdd::iadd(int a, int b, int *c)

```
{
        // TODO: Add your implementation code here
        *c = a + b;

        return S_OK;
}

STDMETHODIMP CAdd::fadd(float a, float b, float *c)
{
        // TODO: Add your implementation code here
        *c = a + b;

        return S_OK;
}

STDMETHODIMP CAdd::isub(int a, int b, int *c)
{
        // TODO: Add your implementation code here
        *c = a - b;

        return S_OK;
}
```

（2）VS2005 使用 C#编写调用程序（网站程序）

使用 VS2005 建立网站 UseCom

<1>配置：在解决方案资源管理器中的主目录点击右键，选择添加引用，选择 COM，添加刚刚建立的 AddCom 1.0 Type Library

<2>在程序中要 using 编写的 COM 组件：using ADDCOMLib;

程序代码：

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using ADDCOMLib;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void ButtonCom_Click(object sender, EventArgs e)
    {
        Add add = new Add();
        int iresult;
        float fresult;
        int sresult;

        add.IAdd(10, 20, out iresult);
        add.fadd((float)1.2,(float)2.3, out fresult);
        add.isub(100, 10, out sresult);

        TextBoxResult.Text = iresult.ToString();
        TextBoxRe2.Text = fresult.ToString();
        TextBoxRe3.Text = sresult.ToString();
    }
}
```

四、在 VC6.0 中编写 COM 组件，使用 VC6.0 调用（

1）VC6.0 编写 COM 组件,使用 VC6.0

建立 COM 组件，

工程类型：ATL COM AppWizard 程序代码：

接口：interface IAdd：IDispatch

```
        {
                [id(1),    helpstring("method    iadd")]    HRESULT
iadd([in]int a, [in]int b, [out]int * c);
                [id(2),    helpstring("method    fadd")]    HRESULT
fadd([in]float a, [in]float b, [out]float * c);
                [id(3),    helpstring("method    isub")]    HRESULT
isub([in]int a, [in]int b, [out]int * c);
        };
```

实现：STDMETHODIMP CAdd::iadd(int a, int b, int *c)

```
{
        // TODO: Add your implementation code here
        *c = a + b;

        return S_OK;
}

STDMETHODIMP CAdd::fadd(float a, float b, float *c)
{
        // TODO: Add your implementation code here
        *c = a + b;

        return S_OK;
}

STDMETHODIMP CAdd::isub(int a, int b, int *c)
{
        // TODO: Add your implementation code here
        *c = a - b;

        return S_OK;
}
```

（2）VC6.0 编写调用程序

<1>使用 VC6.0 建立 MFC 应用程序 UseCOM，调用刚刚建立的 COM 组件

<2>将上面程序 AddCom 生成的 AddCom.dll 放入本程序的工程目录和程序生成目录中

<3>在 StdAfx.h 中加入：
#import "AddCom.dll" no_namespace

程序代码：

```
void CUseComDlg::OnBUTTONUse()
{
	// TODO: Add your control notification handler code here
	CString strResult;
	CoInitialize(NULL);//NULL 换成 0 也可以
	IAddPtr m_add = NULL;
	HRESULT hr = S_OK;
	hr = m_add.CreateInstance(__uuidof(Add));

	int d_a = 90;
	int d_b = 10;
	int d_c;
	int d_d;
	float f_a = 1;
	float f_b = 2;
	float f_c;

	m_add->_IAdd(d_a,d_b,&d_c);
	m_add->fadd(f_a,f_b,&f_c);
	m_add->isub(d_a,d_b,&d_d);

	strResult.Format("返回结果：%d; %f; %d",d_c,f_c,d_d);
	MessageBox(strResult,"结果",MB_OK);

	m_add.Release();
	m_add = NULL;
	CoUninitialize();

}
```

以上内容主要由 csdn 张宇提供.ID: **gisfarmer**
http://blog.csdn.net/gisfarmer/archive/2009/08/06/4418313.aspx

五　C# 动态调用 DLL 方法一

目的:使用两个未公开的 Win32 API 函数来存取控制台窗口，这就需要使用动态调用的方法，动态调用中使用的 Windows API 函数主要有三个,即：Loadlibrary，GetProcAddress 和 Freelibrary。步骤如下(vc 操作)：

1. Loadlibrary: 装载指定 DLL 动态库

2. GetProcAddress：获得函数的入口地址

3. Freelibrary: 从内存中卸载动态库

但是在 C#中是没有函数指针的，无法直接使用 GetProcAddress 返回的入口地址。有以下解决方法，在 .NET 2.0 新增了 Marshal.GetDelegateForFunctionPointer 方法可以满足这个要求，MSDN 里的解释是：将非托管函数指针转换为委托。

后面的事就简单啦，把它编成了一个类来方便调用。

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;

namespace feiyun0112.cnblogs.com
{
    public class DllInvoke
    {

        Win API#region Win API
        [DllImport("kernel32.dll")]
        private extern static IntPtr LoadLibrary(string path);

        [DllImport("kernel32.dll")]
        private extern static IntPtr GetProcAddress(IntPtr lib, string funcName);

        [DllImport("kernel32.dll")]
        private extern static bool FreeLibrary(IntPtr lib);
        #endregion

        private IntPtr hLib;
        public DllInvoke(String DLLPath)
        {
            hLib = LoadLibrary(DLLPath);
        }

        ~DllInvoke()
        {
            FreeLibrary(hLib);
        }

        //将要执行的函数转换为委托
        public Delegate Invoke (string APIName,Type t)
```

```
        {
            IntPtr api = GetProcAddress(hLib, APIName);
            return (Delegate)Marshal.GetDelegateForFunctionPointer(api, t);
        }

    }
}
```

下面是使用的例子：

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;
using feiyun0112.cnblogs.com;

namespace ConsoleApplication1
{
    class Program
    {
        Win API#region Win API
        [DllImport("kernel32.dll")]
        public static extern IntPtr GetStdHandle(int nStdHandle);
        const int STD_OUTPUT_HANDLE = -11;
        #endregion

        public delegate bool SetConsoleDisplayMode(IntPtr hOut, int dwNewMode,
out int lpdwOldMode);

        static void Main(string[] args)
        {
            DllInvoke dll = new DllInvoke("kernel32.dll");

            int dwOldMode;

            //标准输出句柄
            IntPtr hOut = GetStdHandle(STD_OUTPUT_HANDLE);

            //调用 Win API,设置屏幕最大化
            SetConsoleDisplayMode s =
(SetConsoleDisplayMode)dll.Invoke("SetConsoleDisplayMode",
typeof(SetConsoleDisplayMode));
            s(hOut, 1, out dwOldMode);

            Console.WriteLine("**********Full Screen
```

```
Mode********************");
                Console.ReadLine();

            }
        }
}
```

六、C#之 dll 调用方法二

很多时候，windows 平台下的一些底层操作函数（如读写串口或并口函数）都是用 VC++来编制的，而且一般也会以 dll 动态链接库的形式来导出这些操作函数。如果直接在 VC 写的应用程序中调用 dll 导出的函数，是很容易的。然而，如果要在 C#写的应用程序中调用，又该如何来做呢？
综合网上查询的相关资料，自己在 VS .Net 2003 下实现了在 C#中调用 dll 中导出函数，实现过程及代码如下：

（1） 首先将要调用 dll 放到 Asp.net 工程项目的 bin 目录下；
（2） 以示例来说，我是将 wanmei.dll, wanmei.lib 放到自己的工程项目的 bin 目录下。

其中导出的函数原型如下：

```
Bool setcom(int val);       // 设置串口，参数为串口号
Bool findcard(unsigned char mode, unsigned long *num); // 寻卡，返回卡号 num
```

（2）下面就是在 C#中具体如何编写代码来调用了，因为上面 findcard()函数中用到了指针，而 C#中是不使用指针的，这些问题都是要考虑解决的，还是看代码和相关注释：

```
using System.Runtime.InteropServices;            // 调用动态库 dll 时需要使用

namespace CMS
{
    /// <summary>
    /// Operator 的摘要说明。
    /// </summary>
    public class Operator : System.Web.UI.Page
    {
        // 导入 wanmei.dll 动态库中的读卡函数 findcard(), setcom()等函数
        [DllImport("wanmei.dll", EntryPoint="findcard",
            CharSet=CharSet.Ansi,
CallingConvention=CallingConvention.StdCall)]
        public static extern bool findcard(int mode,ref long num);

        [DllImport("wanmei.dll", EntryPoint="setcom",
            CharSet=CharSet.Ansi,
CallingConvention=CallingConvention.StdCall)]
        public static extern bool setcom(int pvar);
```

```
            ...
    }
}
```

注：其中指针是用 C#中的 ref 变量来解决的。

（3）最后我们就可以在 Operator 类中的成员函数中调用以上 dll 导出函数 findcard(),setcom()了。在 Operator 的成员函数 btn_readcard_Click()中相关调用代码如下：

```
        private void btn_readcard_Click(object sender, System.EventArgs e)
        {
            long cardid = 0;
            int mode = 1;
             ...
            if (!setcom(1))
            {
                 lbl_error.Text = "无法设置串口 1";
                return;
            }

             findcard(mode, ref cardid);
             ...
        }
```

注:以上资料均来自互联网，希望对大家有所帮助.