部分 1_ COM 原理

哈尔滨顺时针电脑学校 于凯

01_03_用 C++来解读 COM

哈尔滨顺时针电脑学校

地址：哈尔滨市南岗区复兴街 16 号

电话：0451-86220686 86228969 86220769

网址：http://www.sszkj.com

EMAIL: sszkj@tom.com

一、 用 C++类来实现一个项目

1、案例：

假如有一个工程，我们实现组件化。并可复用。这个工程要求计算两个数的和。一个人实现逻辑，一个人实现界面。做逻辑的人做了一个类，代码如下：

```
//My.h----------------------------------------------------------------------------------------------
#pragma once

class CMy
{
    int m_iResult;
public:
    CMy(void);
    ~CMy(void);
    void Add(int x1,int x2);
    int Get()
    {
        return m_iResult;
    }
};

 //My.cpp-------------------------------------------------------------------------
#include "StdAfx.h"
#include ".\my.h"

CMy::CMy(void)
{
}

CMy::~CMy(void)
```

```
{
}

void CMy::Add (int x1,int x2)
{
   m_iResult=x1+x2;
}
```

他把这两个文件交给了第二个人去用，第二个人拿到后，做界面，然后把第一个人做的两个类包进来，使它们成了他的代码的一部分，然后应用。

```
//CXXXDlg.cpp
#include "My.h"
void CCPlusPlusClassDlg::OnBnClickedButton1()
{
    // TODO: 在此添加控件通知处理程序代码
    CMy obj;
    obj.Add (1,2);
    int i=obj.Get ();
    CString str;
    str.Format ("%d",i);
    MessageBox(str);
}
```

2、问题：

（a）如果第一个人不想让第二个人看到实现代码怎么办？（他想卖钱!）

(b) 如果第一个人改了实现的代码怎么办？原来的程序只能重新集成，重新编译。对已编译好的程序将不能享受到好处。

（c）如果想给一个 VB 程序员或 JAVA 程序员用怎么办？不是 C++程序根本用不了。

（d）每一个应用这个类的程序都要占用一个内存空间来应用这个类。如果起动了两个这样的程序，就占用了两倍的空间。对于大型类。这会很浪费。

(e)想远程调用这个类怎么办？

3、解决问题的方法：

能不能把这个类编成一个 Dll 呢？

二、 用 DLL 实现一个项目

1、 案例：

现在用 DLL 来实现上面的类。

a、建一个空白解决方案

b、制作 DLL

(1)新建项目，选“添入解决方案”，Win32 项目->Dll ，点击“Finish”。

(2)加入一个新类

```
//My.h----------------------------------------------------------------------------------------------
#pragma once

class __declspec(dllexport) CMy
{
```

```
    int m_iResult;
public:
    CMy(void);
    ~CMy(void);
    void Add(int x1,int x2);
    int Get()
    {
        return m_iResult;
    }
};

 //My.cpp-----------------------------------------------------------------
#include "StdAfx.h"
#include ".\my.h"

CMy::CMy(void)
{
}

CMy::~CMy(void)
{
}

void CMy::Add (int x1,int x2)
{
   m_iResult=x1+x2;
}
```

提示：

如果想导出函数，加入一个头文件，如 dd.h,加入导出函数的声明：

```
extern "C" __declspec(dllexport) void Show();
```

在实现文件（cpp）中加入：

```
extern "C" __declspec(dllexport) void Show()
{
        AFX_MANAGE_STATE(AfxGetStaticModuleState());
        CTest test;
        test.DoModal ();
}
```

别忘了在文件中添加：#include "Test.h"和#include "dd.h"）

大功告成,编译吧！

c、制作使用程序

新建项目，选“添入解决方案”，对话框工程。

代码如下：

```
//记得把刚做的DLL拷到testDll项目目录下
#ifdef _DEBUG
#pragma comment(lib,"..\\dll\\Debug\\dll.lib")
#else
#pragma comment(lib,"..\\dll\\Release\\dll.lib")
#endif
#include "..\\dll\\My.h"
void CtestDllDlg::OnBnClickedButton1()
{
    // TODO: 在此添加控件通知处理程序代码
    CMy obj;
    obj.Add (1,2);
    int i=obj.Get ();
    CString str;
    str.Format ("%d",i);
    MessageBox(str);
}
```

2、 问题：

解决了用 C++类来实现的问题的 a、d，其它的没解决。

问题：

（a） 不能跨编译器：

不能所有 C++编译器都能使用，因为 DLL 的导出符号对不同编译器是不同的。对于导出函数的情况是可以用 extern “C”来解决，不过对类内的成员函数就不行了。

（b） 还是不能实现跨编程语言。

（c） 如果改进了实现，然后用新 DLL 替换了原 DLL，而且如果新类大小和原来的不同。则原来的用户程序如果不重新编译就会出错了。可以用给新的 DLL 起个别的名来解决这个问题，不过这样系统会越来越大。

3、 解决方案：

能不能做个句柄类把接口从实现中分离出来呢？

三、用句柄类来改进上面的 DLL。

1、 案例：

把上例的 CMy 的__declspec(dllexport)去掉。因为我们不想导出它。

在上面的 DLL 程序中加入新类做为句柄类：

```
//HandleClass.h
#pragma once
#include "My.h"

class CMy;
class __declspec(dllexport) CHandleClass
{

    CMy* m_pThis;
public:
```

```
    CHandleClass(void);
    ~CHandleClass(void);
    void Add(int x1,int x2);
    int Get();

};

//HandleClass.cpp
#include "StdAfx.h"
#include ".\handleclass.h"

CHandleClass::CHandleClass(void):m_pThis(new CMy)
{
}

CHandleClass::~CHandleClass(void)
{
}

void CHandleClass::Add (int x1,int x2)
{
   m_pThis->Add(x1,x2);
}

int CHandleClass::Get ()
{
   return m_pThis->Get();
}
```

2、 问题：

解决了以前的问题 c

(a) 还是不能跨语言。

(b) 还是不能跨编译器。

(c) 新的问题，接口类要把每一个方法显示的传递给实现类，首先如果方法很多，那将非常麻烦。其次，这样做一个方法要调用两次函数，增加了开销。

3、 解决方案

因为“某个给定平台上的所有 C++编译器都实现了同样的虚函数调用机制”，那么用抽象基类做二进制接口，就能解决前面的问题 b，而且能解决问题 c。

四、 用抽象基类做二进制接口方案 1

1、 案例：

这样做行不行呢？

```
//IMy.h
#pragma once

class __declspec(dllexport) IMy
```

```
{
public:
    virtual void Add(int x1,int x2)=0;
    virtual int Get()=0;
};

//My.h
#pragma once
#include "IMy.h"

class  CMy:public IMy
{
    int m_iResult;
public:
    CMy(void);
    ~CMy(void);
    void Add(int x1,int x2);
    int Get()
    {
        return m_iResult;
    }
};

//My.cpp
#include "StdAfx.h"
#include ".\my.h"

CMy::CMy(void)
{
}

CMy::~CMy(void)
{
}

void CMy::Add (int x1,int x2)
{
   m_iResult=x1+x2;
}
```

应用中：

```
//记得把dll.dll拷到testDll项目目录下
#ifdef _DEBUG
#pragma comment(lib,"..\\dll\\Debug\\dll.lib")
```

```
#else
#pragma comment(lib,"..\\dll\\Release\\dll.lib")
#endif
#include "..\\dll\\My.h"
void CtestDllDlg::OnBnClickedButton1()
{
    // TODO: 在此添加控件通知处理程序代码
    IMy* p=new CMy;
    p->Add (1,2);
    int i=p->Get ();
    delete p;
    CString str;
    str.Format ("%d",i);
    MessageBox(str);
}
```

2、问题：

编译不过去，因为用户没有 CMy 的实现代码，链接不上。而我们就是不想给用户暴露实现代码。

3、解决方案：

在 DLL 中做个全局函数，让它代表客户调用 new 操作符。

五、 用抽象基类做二进制接口方案 2

1、 解决方案：

把 IMy 的__declspec(dllexport)去了，因为我们不用导出它了。

```
//IMyCreate.h
#pragma once
#include "My.h"

extern "C" __declspec(dllexport) IMy * CreateMy();

//IMyCreate.cpp
#include "StdAfx.h"
#include "IMyCreate.h"
IMy * CreateMy()
{
   return new CMy;
}
```

用户程序：

```
//记得把dll.dll拷到testDll项目目录下
#ifdef _DEBUG
#pragma comment(lib,"..\\dll\\Debug\\dll.lib")
#else
#pragma comment(lib,"..\\dll\\Release\\dll.lib")
```

```
#endif

#include "..\\dll\\IMyCreate.h"
void CtestDllDlg::OnBnClickedButton1()
{
    // TODO: 在此添加控件通知处理程序代码
    IMy* p=CreateMy();
    p->Add (1,2);
    int i=p->Get ();

    CString str;
    str.Format ("%d",i);
    MessageBox(str);
    //下面这句加上就会出错：
    //delete p;
}
```

2、 问题：

因为 IMy 的析构函数不是虚函数，因而 delete p，所以销毁的是 IMy，而不是 CMy，一方面导致内存泄露，另一方面，在 VS.NET2003 下将出错。

3、 解决方案：

把接口类的析构函数做成虚函数行不行？不行！因为虚函数在 vtbl 中的位置因编译器不同而不同。这样又不能跨编译器了。

可以这样做，在接口类中再做个纯虚函数 Delete()，然后在实现类中利用这个方法删除自身。

六、 用抽象基类做二进制接口方案 3

1、 案例：

```
//IMy.h
#pragma once

class  IMy
{
public:
    virtual void Add(int x1,int x2)=0;
    virtual int Get()=0;
    virtual void Delete()=0;
};

//My.h
#pragma once
#include "IMy.h"

class  CMy:public IMy
{
    int m_iResult;
```

```
public:
    CMy(void);
    ~CMy(void);
    void Add(int x1,int x2);
    int Get()
    {
        return m_iResult;
    }
    void Delete();
};

//My.cpp
#include "StdAfx.h"
#include ".\my.h"

CMy::CMy(void)
{
}

CMy::~CMy(void)
{
}

void CMy::Add (int x1,int x2)
{
   m_iResult=x1+x2;
}

void CMy::Delete()
{
  delete this;
}
```

用户程序：

```
//记得把dll.dll拷到testDll项目目录下
#ifdef _DEBUG
#pragma comment(lib,"..\\dll\\Debug\\dll.lib")
#else
#pragma comment(lib,"..\\dll\\Release\\dll.lib")
#endif

#include "..\\dll\\IMyCreate.h"
void CtestDllDlg::OnBnClickedButton1()
{
```

```
    // TODO: 在此添加控件通知处理程序代码
    IMy* p=CreateMy();
    p->Add (1,2);
    int i=p->Get ();

    CString str;
    str.Format ("%d",i);
    MessageBox(str);

    p->Delete();
}
```

2、 问题：

已经非常好了，因为虚函数是跨编译器的，所以解决了在不同编译器中使用这一 DLL 的问题。

现在的问题是，客户程序总是需要那个引出库 dll.lib，太不好了，而且启动了这个 DLL 就不能想什么时候关就什么时候关，这样浪费内存。能不能不用这个.lib 呢，可以！

3、 解决方案：

用动态加载 DLL。

七、 用抽象基类做二进制接口方案 4

1、 案例：

改进一下客户程序，新做个函数，不用.lib 了：

```
//记得把dll.dll拷到testDll项目目录下

#include "..\\dll\\IMyCreate.h"
IMy * CallCreateMy(HMODULE&  h,const TCHAR* szDLL,const char* szFn);

void CtestDllDlg::OnBnClickedButton1()
{
    // TODO: 在此添加控件通知处理程序代码
    HMODULE h; //DLL句柄
    const TCHAR szDLL[]=TEXT("dll.dll"); //DLL文件名
    const char szFn[]="CreateMy";  //函数名

    IMy* p=CallCreateMy(h,szDLL,szFn);
    if(!p)
    {
       MessageBox("是不是没把DLL文件拷过来啊？");
       return;
    }
    p->Add (1,2);
    int i=p->Get ();

    CString str;
    str.Format ("%d",i);
```

```
    MessageBox(str);

    p->Delete();
    //关了DLL
    FreeLibrary(h);
}

IMy * CallCreateMy(HMODULE&  h,const TCHAR* szDLL,const char* szFn)
{
   static IMy *(*pfn)()=0;

   //如果是初次就初始化ptr
   if(!pfn)
   {
      //启动DLL
      h=LoadLibrary (szDLL);
      if(h)
          *(FARPROC*) &pfn= GetProcAddress (h,szFn); //得到函数的地址
   }
   return pfn? pfn():0;

}
```

2、 问题：

到目前为止已经太他妈酷了！改进原来的方法，没问题，不影响现在的用户，他们不用重新编译。而且我们的 DLL 是跨编译器的。而且我们可以不用让用户看到我们如何实现的，（可以卖钱了!），而且不用像一般 C++类一样每个使用这些功能的程序都要包含这些代码以占空间和内存。

现在要解决的是，如何给这个 DLL 再加一些方法。改变现有接口？不行，那样影响现有用户。接口一旦发布就不要再改了！

3、 解决方案：

再加上一个接口。

八、 用抽象基类做二进制接口方案 5

1、 案例：

这是最后一个用抽象基类做二进制接口的方案了，除了不能跨语言，什么功能都有了。

实现一个 IMyUnkown 的公共接口，其它接口都由它来派生：

```
//IMyUnkown.h
#pragma once

class  IMyUnkown
{
public:
    virtual void * QueryInterface(const char* pszType)=0;
    virtual void AddRef()=0;
    virtual void Release()=0;
```

```
};
```

更改原来的接口：

```
//IMy.h
#pragma once
#include "IMyUnkown.h"

class  IMy:public IMyUnkown
{
public:
    virtual void Add(int x1,int x2)=0;
    virtual int Get()=0;
};
```

新增一个接口：

```
//IMy2.h
#pragma once
#include "IMyUnkown.h"

class  IMy2:public IMyUnkown
{
public:
    virtual const char* SayHello()=0;

};
```

更改实现类：

```
//My.h
#pragma once
#include "IMy.h"
#include "IMy2.h"

class  CMy:public IMy,public IMy2
{
    int m_iResult;
    int m_iRes;
public:
    CMy(void);
    ~CMy(void);
    void Add(int x1,int x2);
    int Get()
    {
        return m_iResult;
    }
```

```
    const char* SayHello(){ return "Hello world";}

    void * QueryInterface(const char* pszType);
    void AddRef();
    void Release();
};

//My.cpp
#include "StdAfx.h"
#include ".\my.h"
#include <cstring>
using namespace std;

//将指针计数初始化为0
CMy::CMy(void):m_iRes(0)
{
}

CMy::~CMy(void)
{
}

void CMy::Add (int x1,int x2)
{
   m_iResult=x1+x2;
}



void * CMy::QueryInterface(const char* pszType)
{
    void * pvResult=0;
  if(strcmp(pszType,"IMy")==0)
      pvResult= static_cast<IMy*>(this);

  else if(strcmp(pszType,"IMy2")==0)
      pvResult= static_cast<IMy2*>(this);

  else if(strcmp(pszType,"IMyUnkown")==0)
      //这里注意，当请求公共接口时，由于IMy与IMy2都继承自IMyUnkown，这样就有二义
性
      pvResult= static_cast<IMy*>(this);
  else
    //未支持的接口
```

```
    return 0;
  ((IMyUnkown *)pvResult)->AddRef ();
  return pvResult;
}

void CMy::AddRef ()
{
   //指针复制了，增加计数
    ++m_iRes;
}
void CMy::Release()
{
    //指针销毁时，将对象销毁
    if(--m_iRes==0)
       delete this;
}
```

用户代码部分：

```
//记得把dll.dll拷到testDll项目目录下

#include "..\\dll\\IMyCreate.h"
IMy * CallCreateMy(HMODULE&  h,const TCHAR* szDLL,const char* szFn);

void CtestDllDlg::OnBnClickedButton1()
{
    // TODO: 在此添加控件通知处理程序代码
    HMODULE h; //DLL句柄
    const TCHAR szDLL[]=TEXT("dll.dll"); //DLL文件名
    const char szFn[]="CreateMy";  //函数名

    IMy* p=CallCreateMy(h,szDLL,szFn);
    if(!p)
    {
       MessageBox("是不是没把DLL文件拷过来啊？");
       return;
    }
    p->Add (1,2);
    int i=p->Get ();

    IMy2* p2=(IMy2*)p->QueryInterface ("IMy2");
    CString s;
    if(p2)
        s=p2->SayHello ();
```

```
    CString str;
    str.Format ("%d--%s",i,s);
    MessageBox(str);

    p2->Release();
    p->Release();
    //关了DLL
    FreeLibrary(h);
}

IMy * CallCreateMy(HMODULE&  h,const TCHAR* szDLL,const char* szFn)
{
   static IMy *(*pfn)()=0;

   //如果是初次就初始化ptr
   if(!pfn)
   {
      //启动DLL
      h=LoadLibrary (szDLL);
      if(h)
          *(FARPROC*) &pfn= GetProcAddress (h,szFn); //得到函数的地址
   }
   return pfn? pfn():0;

}
```

2、 问题：

现在一切都已解决，只有二个问题，那就是，现在还不能实现跨语言应用我们的 DLL，当然还有一个问题就是还不能远程调用我们的 DLL。第一个问题由 COM 来解决，第二个问题由 COM+来解决。

不过，他们的实现内部机理与我们的没有什么大的不同了。第一个问题是用接口定义语言 IDL 来解决的。也就是不用 C++语言来定义接口，而用一种大多数语言都支持的一种公共角本语言。第二个问题，是用网络服务器来实现。我们马上就会学到。