

COM实用入门教程

第七讲

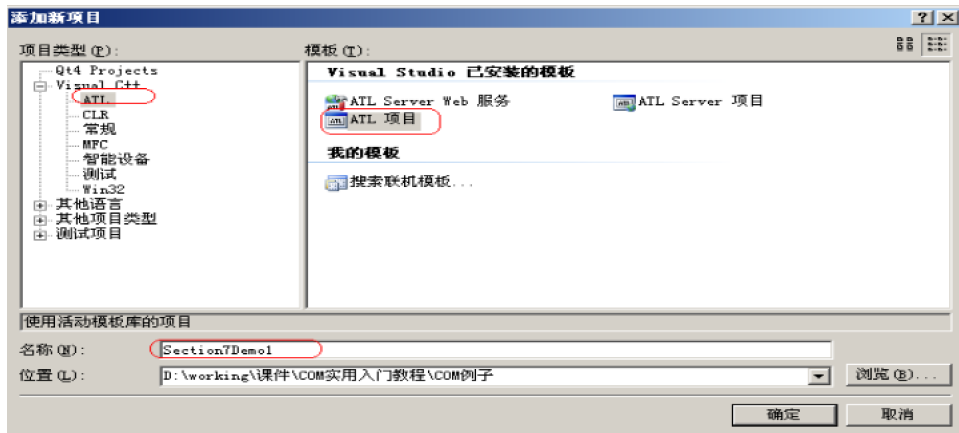
主讲人：阚海忠

VC知识库网站 (www.vckbase.com) 拍摄制作

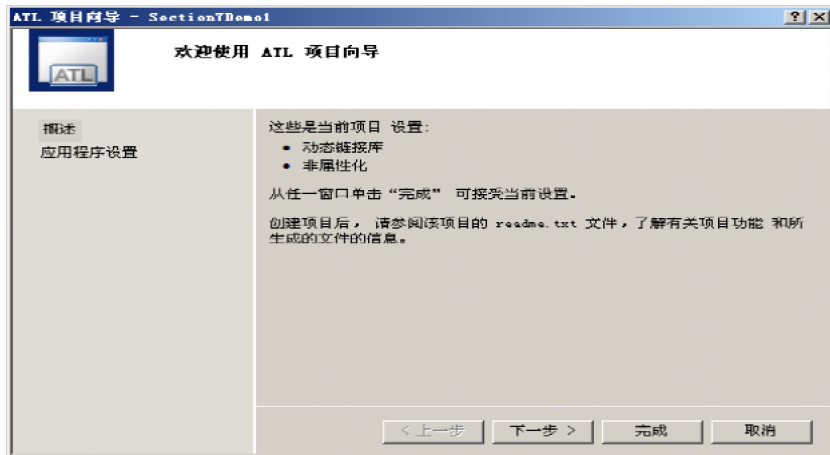
本讲要点：

- 一、为简单对象添加方法与事件；
- 二、在MFC中实现事件接收器；
- 三、在MFC中测试简单对象的方法与事件；
- 四、测试例子的改进。

创建简单对象的向导



创建简单对象的向导



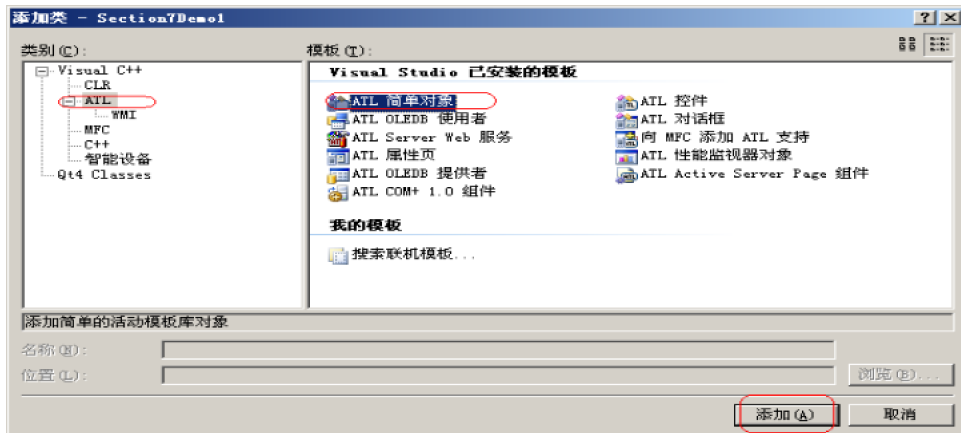
创建简单对象的向导



创建简单对象的向导



创建简单对象的向导



创建简单对象的向导

ATL 简单对象向导 - Section7Demo1

欢迎使用 ATL 简单对象向导

名称
选项

C++

简称(S): HelloSOE .h 文件(H): HelloSOE.h

类(C): CHelloSOE .cpp 文件(F): HelloSOE.cpp

☐ 屈性化(A)

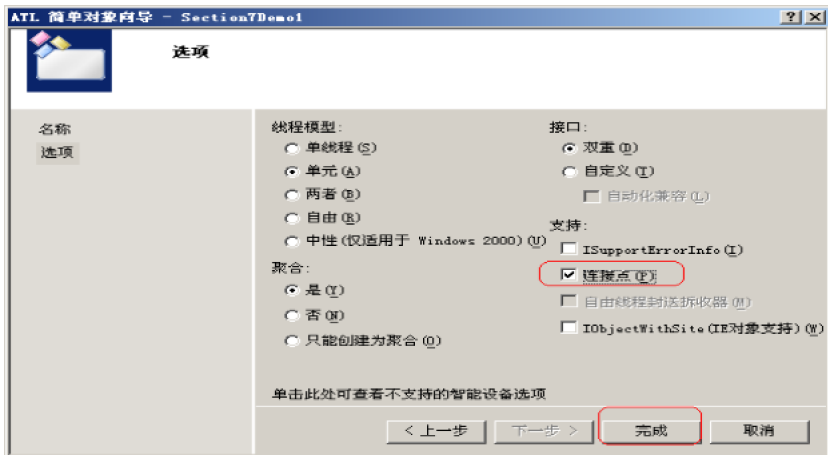
COM

Class(C): HelloSOE 类型(T): HelloSOE Class

接口(I): IHelloSOE ProgID(P): Section7Demo1.HelloSOE

< 上一步 下一步 > 完成 取消

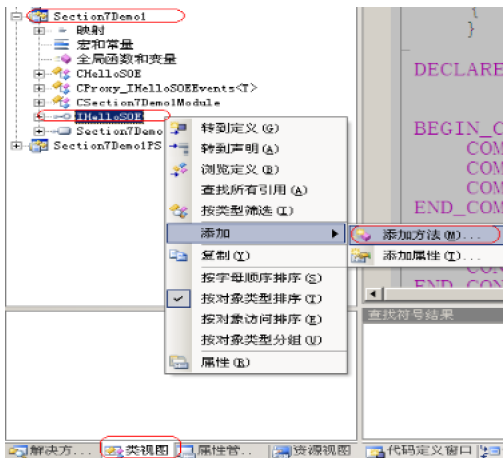
创建简单对象的向导



简单对象的方法与事件

- 接下来我们要添加简单对象的加法方法Add(LONG lA, LONG lB), 返回值以事件方式通知简单对象的调用者。

简单对象的方法



简单对象的方法

添加方法向导 - Section7Demo1

欢迎使用添加方法向导

名称
IDL 属性

返回类型 (R):

方法名 (M):

参数属性:

☐ in (I) ☐ out (O) ☐ retval (V)

参数类型 (P):

参数名 (N):

< 上一步 下一步 > 完成 取消

简单对象的方法

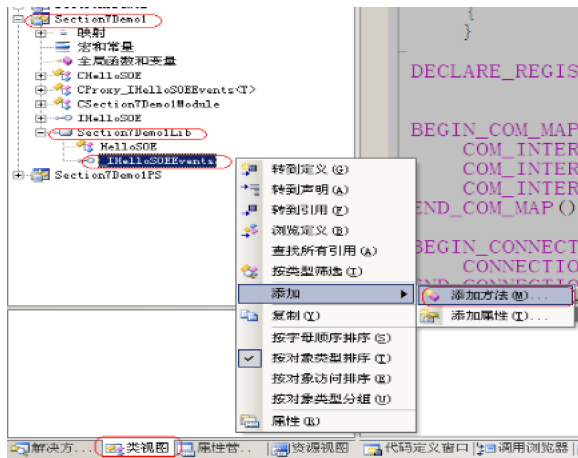
添加方法向导 - Section7Demo1

 IDL 属性

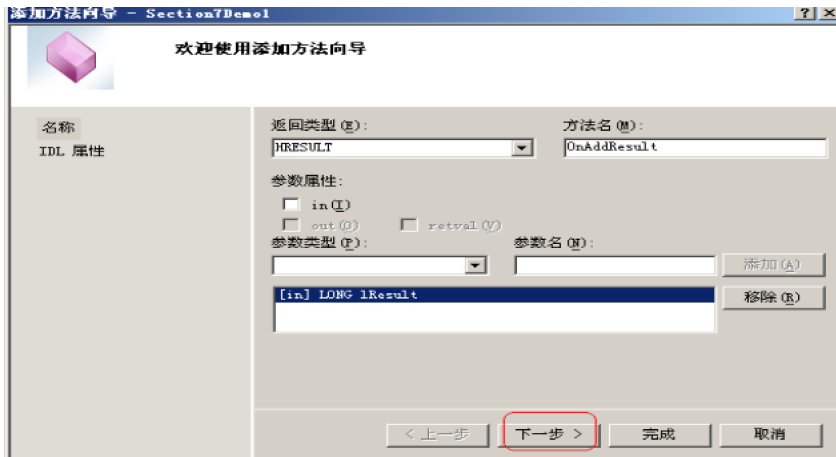
| | | |
|--------|---|------------------|
| 名称 | ID (I) | helpcontext (H): |
| IDL 属性 | 1 | |
| | call_as (A): | helpstring (S): |
| | | 方法Add |
| | <input type="checkbox"/> hidden (H) | |
| | <input type="checkbox"/> source (S) | |
| | <input type="checkbox"/> local (L) | |
| | <input type="checkbox"/> restricted (R) | |
| | <input type="checkbox"/> vararg (V) | |

< 上一步 下一步 > **完成** 取消

简单对象的事件




简单对象的事件



简单对象的事件

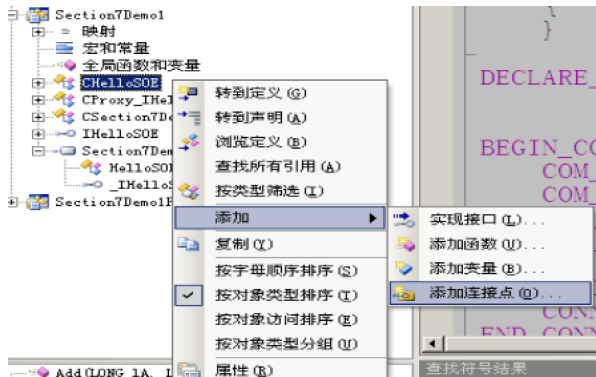
添加方法向导 - Section7Demo1

 IDL 属性

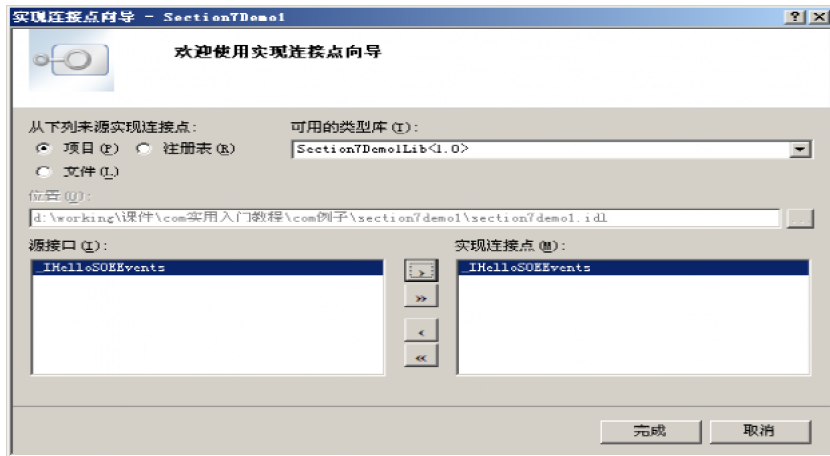
| | | |
|--------|---|------------------|
| 名称 | ID (I) | helpcontext (X): |
| IDL 属性 | 1 | |
| | call_as (A): | helpstring (G): |
| | | 方法OnAddResult |
| | <input type="checkbox"/> hidden (E) | |
| | <input type="checkbox"/> source (S) | |
| | <input type="checkbox"/> local (L) | |
| | <input type="checkbox"/> restricted (R) | |
| | <input type="checkbox"/> vararg (V) | |

< 上一步 下一步 > 完成 取消

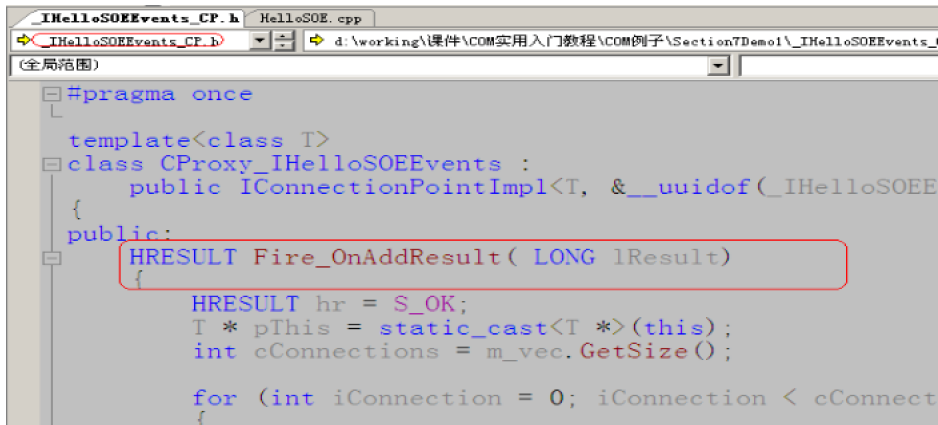
简单对象的事件



简单对象的事件



查看生成的部分代码

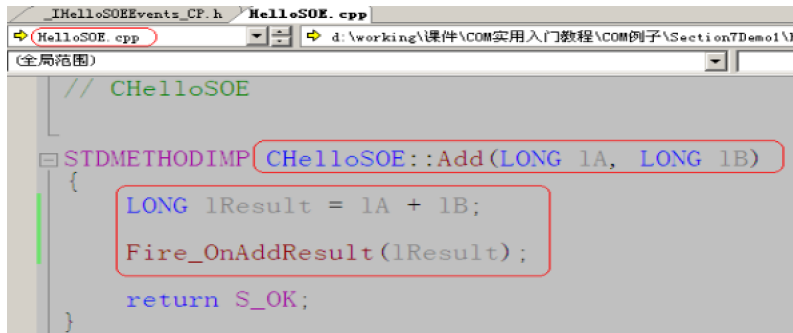


```
#pragma once

template<class T>
class CProxy_IHelloSOEEvents :
    public IConnectionPointImpl<T, &__uuidof(_IHelloSOEEvents), T>
{
public:
    HRESULT Fire_OnAddResult( LONG lResult)
    {
        HRESULT hr = S_OK;
        T * pThis = static_cast<T *>(this);
        int cConnections = m_vec.GetSize();

        for (int iConnection = 0; iConnection < cConnections; iConnection++)
        {
            CProxy_IHelloSOEEvents * pConn = m_vec[iConnection];
            pConn->Fire_OnAddResult(lResult);
        }
    }
};
```

简单对象的方法实现与事件触发



The screenshot shows a code editor with two tabs: `_IHelloSOEEvents_CP.h` and `HelloSOE.cpp`. The `HelloSOE.cpp` tab is active and highlighted with a red circle. The file path in the address bar is `d:\working\课件\COM实用入门教程\COM例子\Section7Demo1\`. The code is written in C++ and implements a method `Add` for the `CHelloSOE` class. The method signature `CHelloSOE::Add(LONG lA, LONG lB)` is highlighted with a red box. The implementation block is enclosed in curly braces and contains the following lines: `LONG lResult = lA + lB;`, `Fire_OnAddResult(lResult);`, and `return S_OK;`. The entire implementation block is also highlighted with a red box. The scope is set to `(全局范围)` (Global Scope).

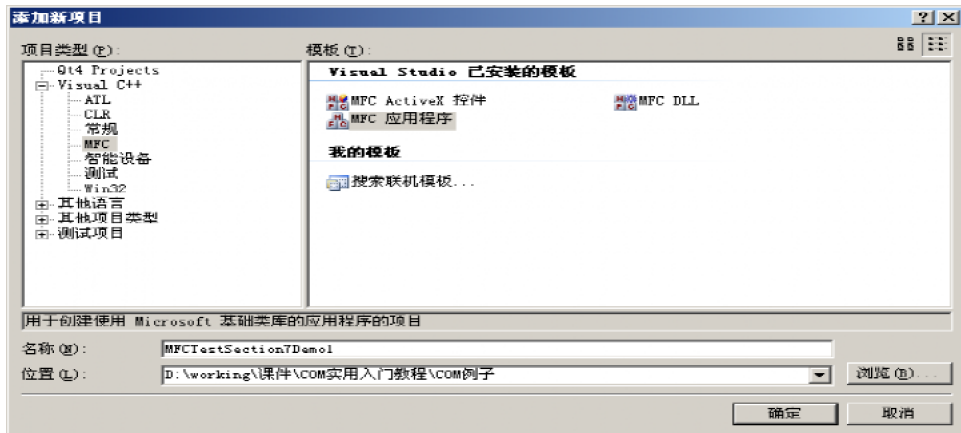
```
// CHelloSOE

STDMETHODIMP CHelloSOE::Add(LONG lA, LONG lB)
{
    LONG lResult = lA + lB;
    Fire_OnAddResult(lResult);
    return S_OK;
}
```

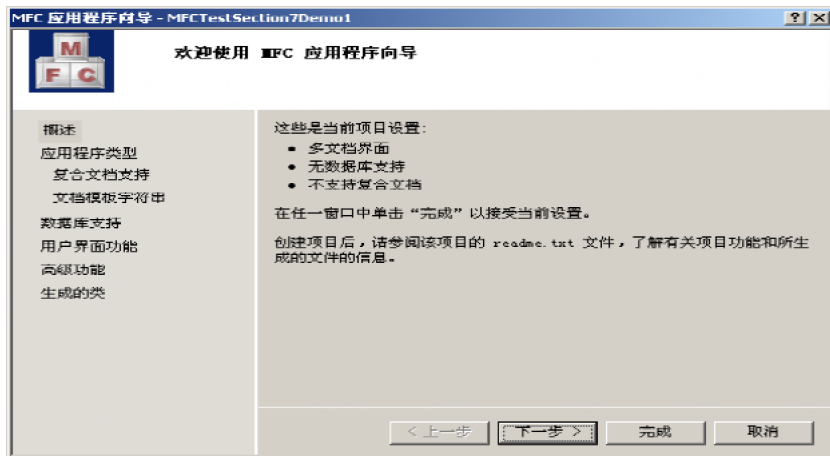
本讲要点：

- 一、为简单对象添加方法与事件；
- 二、在MFC中实现事件接收器；
- 三、在MFC中测试简单对象的方法与事件；
- 四、测试例子的改进。

MFC 向导



MFC 向导



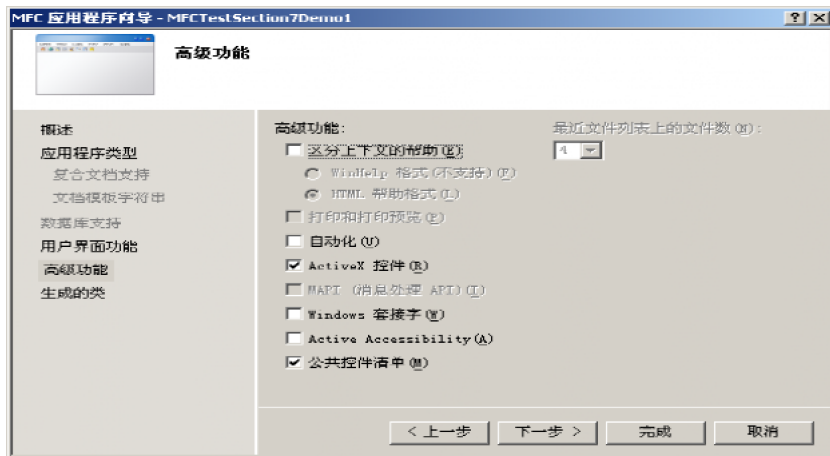
MFC 向导



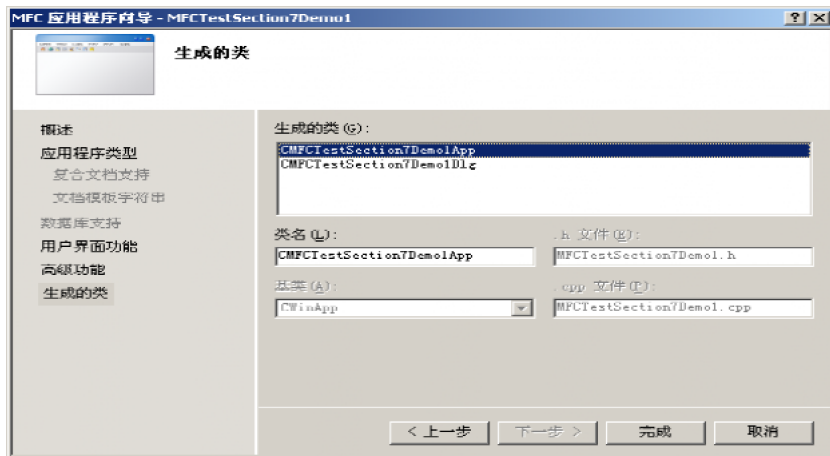
MFC 向导



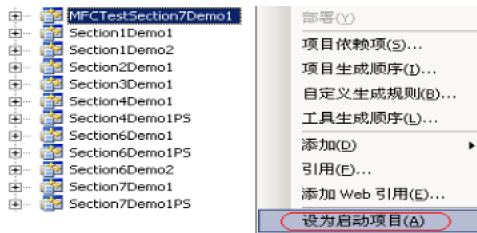
MFC 向导



MFC 向导



MFC 向导



添加代码

- 在stdafx.h中，包含Section7Demo1.h文件。
- 在stdafx.cpp中，包含Section7Demo1_i.c文件。

事件接收器的实现

- 为了使CMFCTestSection7Demo1Dlg能够接收简单对象的事件，我们得让CMFCTestSection7Demo1Dlg成为事件接收器。事件接收器需要继承于IDispatch，才能够被简单对象注册。关于事件注册，等下会讲解，我们现在先看下如何实现CMFCTestSection7Demo1Dlg。

事件接收器的实现

```
class CMFCTestSection7Demo1Dlg :  
    public CDialog, public IDispatch  
{...};
```

类CMFCTestSection7Demo1Dlg 继承于CDialog跟IDispatch。

事件接收器的实现

- 在MFCTestSection7Demo1.h中声明IUnknown的三个接口与IDispatch的四个接口：

```
virtual HRESULT STDMETHODCALLTYPE QueryInterface(  
    /* [in] */ REFIID riid,  
    /* [iid_is][out] */ void __RPC_FAR * __RPC_FAR *ppvObject);  
  
virtual ULONG STDMETHODCALLTYPE AddRef( void);  
  
virtual ULONG STDMETHODCALLTYPE Release( void);  
  
virtual HRESULT STDMETHODCALLTYPE GetTypeInfoCount(  
    /* [out] */ UINT *pctinfo);  
  
virtual HRESULT STDMETHODCALLTYPE GetTypeInfo(  
    /* [in] */ UINT iTInfo,  
    /* [in] */ LCID lcid,  
    /* [out] */ ITypeInfo **ppTInfo);  
  
virtual HRESULT STDMETHODCALLTYPE GetIDsOfNames(  
    /* [in] */ REFIID riid,  
    /* [size_is][in] */ LPOLESTR *rgszNames,  
    /* [in] */ UINT cNames,  
    /* [in] */ LCID lcid,  
    /* [size_is][out] */ DISPID *rgDispId);  
  
virtual /* [local] */ HRESULT STDMETHODCALLTYPE Invoke(  
    /* [in] */ DISPID dispIdMember,  
    /* [in] */ REFIID riid,  
    /* [in] */ LCID lcid,  
    /* [in] */ WORD wFlags,  
    /* [out][in] */ DISPPARAMS *pDispParams,  
    /* [out] */ VARIANT *pVarResult,  
    /* [out] */ EXCEPINFO *pExcepInfo,  
    /* [out] */ UINT *puArgErr);
```


事件接收器的实现

- 在MFCTestSection7Demo1.h中声明加法运行结果的接收事件：

```
void OnAddResult(LONG lResult);
```

事件接收器的实现

- 实现IUnknow的三个接口。

```
HRESULT STDMETHODCALLTYPE CMFCTestSection7Demo1Dlg::QueryInterface(  
    /* [in] */ REFIID riid,  
    /* [iid_is][out] */ void __RPC_FAR * __RPC_FAR *ppvObject)  
{  
    if (riid == IID_IDispatch || riid == IID_IUnknown || riid == DIID_IHelloSOEEvents)  
    {  
        *ppvObject = static_cast<IDispatch*>(this);  
    }  
    else  
    {  
        *ppvObject = NULL;  
        return E_NOINTERFACE;  
    }  
  
    AddRef();  
    return S_OK;  
}  
  
ULONG STDMETHODCALLTYPE CMFCTestSection7Demo1Dlg::AddRef( void)  
{  
    return ++m_ulCount;  
}  
  
ULONG STDMETHODCALLTYPE CMFCTestSection7Demo1Dlg::Release( void)  
{  
    if (0 == --m_ulCount)  
    {  
        delete this;  
        return 0;  
    }  
  
    return m_ulCount;  
}
```

事件接收器的实现

- 不具体实现如下接口，因为在事件接收过程中用不到。

```
HRESULT STDMETHODCALLTYPE CMFCTestSection7Demo1Dlg::GetTypeInfoCount(  
    /* [out] */ UINT *pctinfo)  
{  
    return E_NOTIMPL;        //不实现  
}  
  
HRESULT STDMETHODCALLTYPE CMFCTestSection7Demo1Dlg::GetTypeInfo(  
    /* [in] */ UINT iTInfo,  
    /* [in] */ LCID lcid,  
    /* [out] */ ITypeInfo **ppTInfo)  
{  
    return E_NOTIMPL;        //不实现  
}  
  
HRESULT STDMETHODCALLTYPE CMFCTestSection7Demo1Dlg::GetIDsOfNames(  
    /* [in] */ REFIID riid,  
    /* [size_is][in] */ LPOLESTR *rgszNames,  
    /* [in] */ UINT cNames,  
    /* [in] */ LCID lcid,  
    /* [size_is][out] */ DISPID *rgDispId)  
{  
    return E_NOTIMPL;        //不实现  
}
```

事件接收器的实现

- 实现Invoke方法:

dispIdMember与事件ID对应, 事件ID从1开始。因为我们只有一个事件, 所以该事件的ID是1。

pDispParams->rgvarg是事件的参数数组, 元素用VARIANT类型表示。

```
HRESULT STDMETHODCALLTYPE CMFCTestSection7Demo1Dlg::Invoke(  
    /* [in] */ DISPID dispIdMember,  
    /* [in] */ REFIID riid,  
    /* [in] */ LCID lcid,  
    /* [in] */ WORD wFlags,  
    /* [out][in] */ DISPPARAMS *pDispParams,  
    /* [out] */ VARIANT *pVarResult,  
    /* [out] */ EXCEPINFO *pExcepInfo,  
    /* [out] */ UINT *puArgErr)  
{  
    if (dispIdMember == 1)  
    {  
        OnAddResult(pDispParams->rgvarg[0].lVal);  
    }  
  
    return S_OK;  
}
```

事件接收器的实现

- 在OnAddResult中简单地弹出加法运算结果的信息框。实现代码如下：

```
CString str;  
str.Format(_T("加法运算的结果是： %ld"),  
lResult);  
AfxMessageBox(str);
```

本讲要点：

- 一、为简单对象添加方法与事件；
- 二、在MFC中实现事件接收器；
- 三、在MFC中测试简单对象的方法与事件；
- 四、测试例子的改进。

简单对象的创建

```
CComPtr<IHelloSOE> spHelloSOE;  
hr = spHelloSOE.CoCreateInstance(CLSID_HelloSOE);
```

CLSID_HelloSOE为组件的clsid。

注册事件

- 在创建简单对象后，需要注册事件接收源。事件注册采用Advise方法。

```
DWORD dw = 0;
```

```
    hr = spHelloSOE.Advise(this,  
    DIID__IHelloSOEEvents, &dw);
```

- 第一个参数代表接收源，第二个参数代表事件接口的IID，第三个参数代表注册成功后的cookie。

测试简单对象的方法与事件

- 添加简单对象的方法调用。

`hr = spHelloSOE->Add(9,99);`

运行程序，以上方法运行完后会触发OnAddResult，从而打印出加法运算的结果。



本讲要点：

- 一、为简单对象添加方法与事件；
- 二、在MFC中实现事件接收器；
- 三、在MFC中测试简单对象的方法与事件；
- 四、测试例子的改进。

测试例子的改进

- 在刚才的测试例子中，对话框类既是对话框的管理类又是事件接收器的类。我们可以把事件接收器分离出来。然后使事件接收器作为对话框管理类的一个成员，同时事件接收器存放对话框类的对象指针。

测试例子的改进

- 我们让事件接收器的类名为**CSink**，它必须继承于**IDispatch**，实现方法跟**MFCTestSection7Demo1**项目的实现方法差不多，具体代码查看**MFCTest2Section7Demo1**项目的**sink.h**与**sink.cpp**文件。最为关键的是**CSink**必须保存对话框的指针，以便在接收事件后，通过该指针调用对话框的相关方法。

测试例子的改进

- 对话框的构造函数添加 `m_pSink = new CSink(this);` 以创建事件接收器。
- 析构函数添加 `m_pSink->Release();` 以释放事件接收器。

测试例子的改进

- 在对话框类中，对事件的注册采用

```
hr = spHelloSOE.Advise(m_pSink,  
    DIID__IHelloSOEEvents, &dw);
```

代替

```
hr = spHelloSOE.Advise(this,  
    DIID__IHelloSOEEvents, &dw);
```

回 顾

- 本讲要点：
 - 一、为简单对象添加方法与事件；
 - 二、在MFC中实现事件接收器；
 - 三、在MFC中测试简单对象的方法与事件；
 - 四、测试例子的改进。