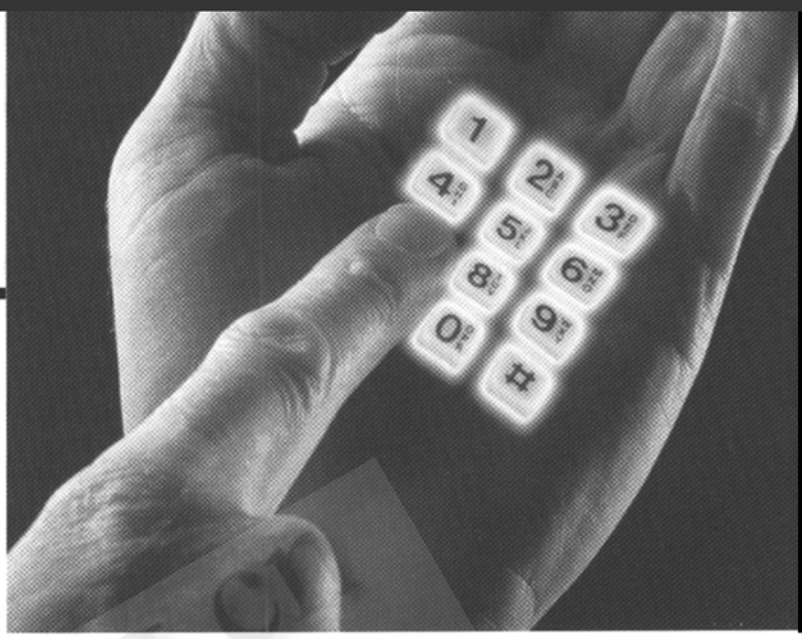
ATL 快速编写 COM 组件



罗月新

(中科院上海技术物理研究所 200083)

摘要：通过ATL可以方便快速地建立小型的，基于COM的组件。本文简要介绍IDL语言编写COM接口的基本概念，并通过实例展示通过ATL开发COM组件的基本过程。

关键词：ATL(活动模板库) COM（组件对象模型） IDL（接口定义语言）

COM（Component Object Model）是Microsoft研制的一项系统级别的面向对象的技术，在COM构架下，可以开发出各种各样地功能专一的软件"积木块（组件）"，将它们按照需要"搭"起来，就能构成复杂的应用系统，COM组件是以Win32动态链接库（DLLs）或可执行文件（EXEs）形式发布的可执行的代码组成，它用来实现面向对象的、与语言无关的、且位置透明的组件或软件模块，COM标志着未来程序开发的方向。

ATL（Active Template Library）是一个基于C++的框架，主要用在基于COM的软件开发上，使用它可以大大简化组件开发的过程。对于刚刚接触COM、具备C++基本知识的程序员，如果想快速建立简单的、基于COM的组件，那么Microsoft的ATL将是您最好的选择。

1 COM接口

COM的主要工作是处理接口（Interface）的实现和使用上的问题，刚刚接触COM的程序员会觉得很深奥，其实接口只不过是一组函数，是一些有明确的输入、输出参数并按照IDL（Interface Definition Language，接口定义语言）语言格式编写的函数。

ATL工程中的IDL文件非常重要，它用IDL以一种与语言无关的方法来定义一个组件接口，从而使组件的客户可以使用该定义，接口是COM的精髓，熟悉IDL是最基本的要求。（对于初学者，可以暂时跳过这段与C语言有差别的代码，直接看标题2中的内容），下面以MathServer.idl为例详细阐述：

```
// MathServer.idl : IDL source for MathServer.dll
import "oaidl.idl";
import "ocidl.idl";
[
  object,
  uuid(88BBB422-CBC0-46D2-B508-503A73C4EEE2),
  helpstring("IMath Interface"),
  pointer-default(unique)
]
```

这是一段属性代码（中括号[]内），它们提供了关于后面定义语句的附加信息。关键字uuid指明了接口的GUID；关键字helpstring则指明了一些对象浏览器可以显示的文本，pointer_default属性为任何一个在接口内定义的指针设定了默认的指针属性。

```
interface IMath : IUnknown
{
  [helpstring("method Add")] HRESULT Add([in] long lParam1,[in] long lParam2,[out,retval] long* plResult);
  [helpstring("method Subtract")] HRESULT Subtract([in] long lParam1,[in] long lParam2,[out,retval] long* plResult);
  [helpstring("method Multiply")] HRESULT Multiply([in] long lParam1,[in] long lParam2,[out,retval] long* plResult);
  [helpstring("method Divide")] HRESULT Divide([in] long lParam1,[in] long lParam2,[out,retval] long* plResult);
};
```

上面的接口描述了组件里一个基于COM的接口，这些代码几乎和C++类的声明语句完全相同，主要差别在于参数的列写上。在IDL中，in和out关键字指定了参数的传递方向，retval关键字指明了参数应该被当作方法的返回值。

```
[
  uuid(934FFC70-C68F-4AEF-BEC2-015D4F8FB52E),
  version(1.0),
  helpstring("MathServer 1.0 Type Library")
]
library MATHSERVERLib
{
  importlib("stdole32.tlb");
```

Developing COM Quickly by ATL

```
    importlib("stdole2.tlb");
    [
      uuid(4979EC68-2DCB-4EB8-8BAD-A7F69E751419),
      helpstring("Math Class")
    ]
    coclass Math
    {
    [default] interface IMath;
    };
};
```

这段代码与宿主文件及宿主文件所包含的组件有关。属性代码段从整体上描述了类型库，它包含一个GUID、版本号和帮助字符串，关键字 library 指定了库的名称，关键字 coclass 是 component class 的缩写，用来指定独立组件以及它们支持的接口。

2 使用 ATL 实现 Math 组件

ATL提供了实现基于COM组件内核的支持，利用ATL向导程序可以生成组件的大部分代码，ATL向导程序隐藏了很多实现细节，这有助于学习ATL，它提供的功能包括：

- ATL COM AppWizard，负责建立ATL工程；
- ATL Object Wizard，为基本的COM组件创建代码；
- ATL对低级别的COM功能提供内置式支持，如IUnknown、类工厂和自注册功能；
- ATL支持Microsoft的接口定义语言IDL等。

2.1 创建 MathServer 工程

启动Visual C++并创建新工程，选择ATL COM AppWizard，并输入工程名为MathServer，如图1，点击OK进入下一步。

2.2 ATL COM AppWizard

在ATL COM AppWizard Step 1 of 1向导页中选择 Server Type 为 Dynamic Link Library（动态链接库），如图2。点击Finish后并按下OK，即可生成MathServer工程，为Math组件提供基本的宿主支持。

2.3 添加 Math 组件

点击 Visual C++ 中的 Insert->New ATL Object...菜单项调用 ATL Object Wizard 向导程序，如图3，选择Simple Object，点击Next，显示图4对话框（已填入参数）。

2.4 ATL Object Wizard Names 选项卡

我们只需把ATL Object Wizard属性对话框下Names中的Short Name设置为Math，其他可修改也可接受其默认值。

2.5 ATL Object Wizard Attributes 选项卡

ATL Object Wizard 属性对话框下 Attributes 选项卡中的内容可使你为组件指定基本的COM支持选项，如图5，在这里对默认值的唯一更改是在Interface中选择Custom（自定义）接口。当设置完这些选项后，点击OK。

2.6 IMath 接口的实现

Object Wizard为MathServer工程添加了CMath类和IMath接口的实现文件，接下来就是要为Math组件添加接口方法。在Visual C++中的Class View选项卡下，右键点击IMath接口并选择Add Method...，如图6所示，在Return Type中选择HRESULT，在Method Name中输入Add，在Parameters中输入双引号中的内容"[in] long lParam1,[in] long lParam2,[out,retval] long* plResult"，点击OK就为Math组件添加了Add（加法）接口，同样只需改变Method Name为Subtract、Multiply或Divide（减法、乘法和除法），Parameters的内容仍输入上面双引号中的内容即可。

打开Math.cpp文件，并添加黑体显示的代码段：

```
STDMETHODIMP CMath::Add(long lParam1, long lParam2, long *plResult)
{
    *plResult=lParam1+lParam2;
    return S-OK;
}
```

图1 创建MathServer工程

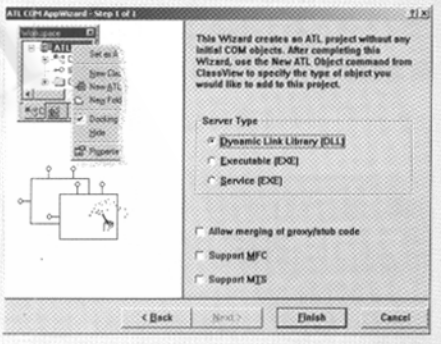


图2 ATL COM AppWizard，Step 1 of 1 向导页

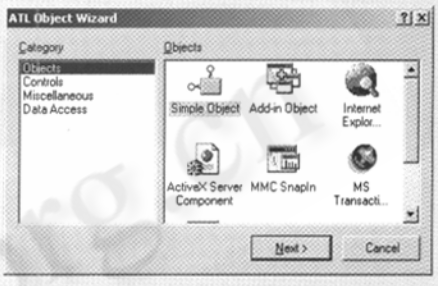


图3 ATL Object Wizard 向导

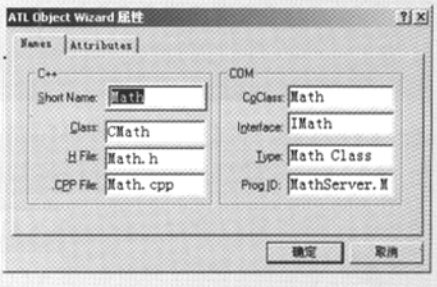


图4 ATL Object Wizard 属性下 Names 选项卡

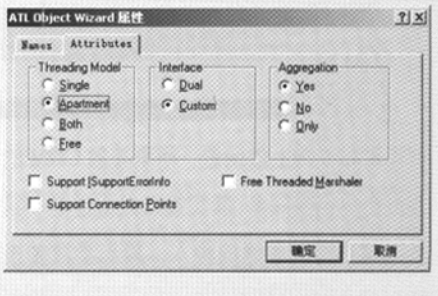


图5 ATL Object Wizard 属性下 Attributes 选项卡

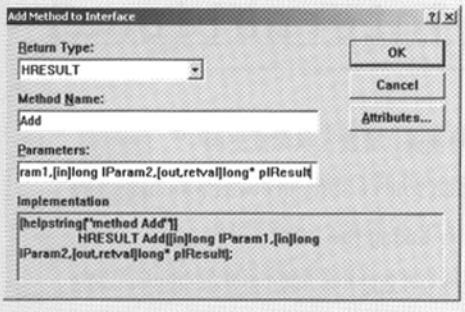


图 6 Add Method 添加接口

```
STDMETHODIMP CMath::Subtract(long lParam1, long lParam2, long *plResult)
{
    *plResult=lParam1-lParam2;
    return S-OK;
}
STDMETHODIMP CMath::Multiply(long lParam1, long lParam2, long *plResult)
{
    *plResult=lParam1*lParam2;
    return S-OK;
}
STDMETHODIMP CMath::Divide(long lParam1, long lParam2, long *plResult)
{
    *plResult=lParam1/lParam2;
    return S-OK;
}
```

这样通过使用ATL内建的COM功能完成一些简单功能的添加。编译结束后就生成具备简单功能的Math组件的DLL文件：MathServe.dll。

3 建立简单的COM客户程序

组件和使用组件服务的软件实体之间的关系可以理解为服务器和客户机的关系。一个COM组件通常要为其他软件实体提供一些功能，因此它也被成为服务器（Server），比如刚才建立的Math组件。一个使用组件功能的软件实体则被成为客户或客户机（Client）。

在这里建立一个简单的Win32控制台应用程序。启动Visual C++，使用AppWizard，选择Win32 Console Application（Win32控制台应用程序），并命名为MathClient，建立空的应用程序（A Empty Application），并创建一个名为MathClient.cpp的文件，添加代码修改如下：

```
#include "windows.h"
#include "iostream.h"
#include "initguid.h"
#include "..\MathServer\MathServer-i.c"
//包含自动生成的组件接口文件
#include "..\MathServer\MathServer.h"
//包含组件的头文件
int main(int argc,char* argv [])
{
    cout<<"Initializing COM"<<endl;
    if(FAILED(CoInitialize(NULL)))
// 初始化 COM 库
    {
      cout<<"Unable to Initialize COM"
<<endl;
      return -1;
    }

    IMath* pMath; // 定义 COM 组件对象指针
    //建立COM实例，CoCreateInstance函数中各参数的意义请阅读VC帮助信息
    HRESULT hr=CoCreateInstance(CLSID-Math,
    NULL,
    CLSCTX-INPROC,
    IID-IMath,
    (void**)&pMath);
    if(FAILED(hr)) // 如果不能建立 COM 实例，则调用 C++ 输出函数输出错误信息
    {
      cout.setf(ios::hex,ios::basefield);
      cout<<"Failed to create server instance.
HR="<<hr<<endl;
      return -1;
    }
    cout<<"Instance created"<<endl;
    long result;
    pMath->Multiply(100,8,&result); // 调用COM组件的乘法函数，result存储计算结果
    cout<<"100*8 is"<<result<<endl; // 输出计算结果
    pMath->Subtract(1000,333,&result); //调用COM组件的减法函数，result存储计算结果
    cout<<"1000-333 is"<<result<<endl; // 输出计算结果
    cout<<"Releaseing Math interface"
<<endl;
    pMath->Release(); // 释放COM对象占用的内存
    cout<<"Shutting down COM"<<endl;
    CoUninitialize(); // 关闭 COM
    return 0;
    }
```

编译运行程序，你会发现实现了Math组件，使用ATL编写COM原来如此简单。

4 结束语

ATL的框架为组件开发提供了最基本的实现代码，类似于MFC，大大提高了开发效率。当然上面实现的只是一个很简单的组件，目的只是为了给刚刚接触COM编程的朋友们一点感性认识。其实COM还涉及到很多内容，比如多接口、进程调度、自动化和线程管理等。

参考文献

1 Dale Rogerson（美）著，杨秀章译，COM技术内幕：微软组件对象模型，清华大学出版社，1998.12。

2 Tom Armstrong，Ron Patton（美）著，董梁，丁杰等译，ATL开发指南，电子工业出版社2000.11。