# COM实用入门教程　　第三讲

主讲人：阙海忠

VC知识库网站 (www.vckbase.com) 拍摄制作

# 本讲要点：

- 一、IDispatch的简介与作用；
- 二、常用的COM数据类型；
- 三、COM数据类型的转换；
- 四、VARIANT类型的派生类。

# IDispatch的简介与作用

- 我们之前讲过COM组件是跨应用的。到目前为止，我们都只是通过C++来调用COM组件的接口。
- 而vbscript，Javascript这些解释性语言跟Word,Excel等提供的宏语言如何认识开发者为COM组件提供的自定义接口呢？
- 如果最终不能调用我们为COM组件提供的自定义接口，那我们开发出来的COM组件，将不是跨应用的。

# IDispatch的简介与作用

- 如何解决这个问题呢？这就需要自动化技术，让解释性语言可以调用到我们的自定义接口。
- 具有自动化功能的组件，是支持IDispatch接口的COM组件。

# IDispatch的简介与作用

- IDispatch接口能够接收一个函数的字符串名称，并执行了这个函数。
- 解释性语言跟宏语言，要调用 COM组件的自定义接口时，都是通过自动化控制程序把自定义接口中的函数名称的字符串跟函数参数传递给IDispatch，让IDispatch间接地去执行自定义接口中的函数。

# IDispatch的简介与作用

- IDispatch中最令人感兴趣的两个函数是GetIDsOfNames和Invoke。
- GetIDsOfNames函数将读取一个函数名称，并返回其调度ID。
- Invoke函数接收一个调度ID，跟函数参数，执行调度ID所对应的函数的功能。
- 所以，解释性语言跟宏语言不必直接认识我们的自定义接口，只要我们的COM组件支持了IDispatch接口，它们就能通过自动化控制程序调用我们的IDispatch接口，通过传入函数名字的字符串，跟函数参数来执行调用我们的自定义接口。

# IDispatch的简介与作用

- 由于篇幅有限，加之目前的ATL开发平台都能自动实现IDispatch接口，COM的开发人员一般不需要手动实现该接口。我们这边就先不讲IDispatch的手动实现，只讲解IDispatch的简介与支持自动化技术的作用。

# IDispatch的简介与作用

- 在后续的讲解中，我们会通过ATL制作一个支持IDispatch的自动化COM组件，并进行跨应用的测试。

# 这一讲，主要讲解如下要点：

- 一、IDispatch的简介与作用；
- 二、常用的COM数据类型；
- 三、COM数据类型的转换；
- 四、VARIANT类型的派生类。

# COM数据类型讲解大纲

- 常用的COM数据类型有：

  CHAR, CHAR*, BYTE, BYTE*, SHORT, SHORT*, USHORT, USHORT*, INT, INT*, UINT, UINT*, LONG, LONG*, ULONG, ULONG*, FLOAT, FLOAT*, DOUBLE, DOUBLE*, VARIANT_BOOL, VARIANT_BOOL*, BSTR, BSTR*, IUnknown*, IUnknown**, IDispatch*, IDispatch**, VARIANT, VARIANT*

# 常用的COM数据类型

- CHAR: typedef char CHAR;
- CHAR*, CHAR的指针
- BYTE, typedef unsigned char BYTE;
- BYTE*, BYTE的指针
- SHORT, typedef short SHORT;
- SHORT*, SHORT的指针
- USHORT, typedef unsigned short USHORT;
- USHORT*, USHORT的指针

# COM数据类型讲解大纲

- 常用的COM数据类型有：

  CHAR, CHAR*, BYTE, BYTE*, SHORT, SHORT*, USHORT, USHORT*, INT, INT*, UINT, UINT*, LONG, LONG*, ULONG, ULONG*, FLOAT, FLOAT*, DOUBLE, DOUBLE*, VARIANT_BOOL, VARIANT_BOOL*, BSTR, BSTR*, IUnknown*, IUnknown**, IDispatch*, IDispatch**, VARIANT, VARIANT*

# 常用的COM数据类型

- INT, typedef int INT;
- INT*, INT的指针
- UINT, typedef unsigned int UINT;
- UINT*, UINT的指针
- LONG, typedef long LONG;
- LONG*, LONG的指针
- ULONG, typedef unsigned long ULONG;
- ULONG*, ULONG的指针

# COM数据类型讲解大纲

- 常用的COM数据类型有：

  CHAR, CHAR*, BYTE, BYTE*, SHORT, SHORT*, USHORT, USHORT*, INT, INT*, UINT, UINT*, LONG, LONG*, ULONG, ULONG*, FLOAT, FLOAT*, DOUBLE, DOUBLE*, VARIANT_BOOL, VARIANT_BOOL*, BSTR, BSTR*, IUnknown*, IUnknown**, IDispatch*, IDispatch**, VARIANT, VARIANT*

# 常用的COM数据类型

- FLOAT, typedef float FLOAT;
- FLOAT*, FLOAT的指针
- DOUBLE, typedef double DOUBLE;
- DOUBLE*, DOUBLE的指针
- VARIANT_BOOL, COM中的布尔类型
  typedef short VARIANT_BOOL;
  /* 0 == FALSE, -1 == TRUE */
- VARIANT_BOOL*, VARIANT_BOOL的指针

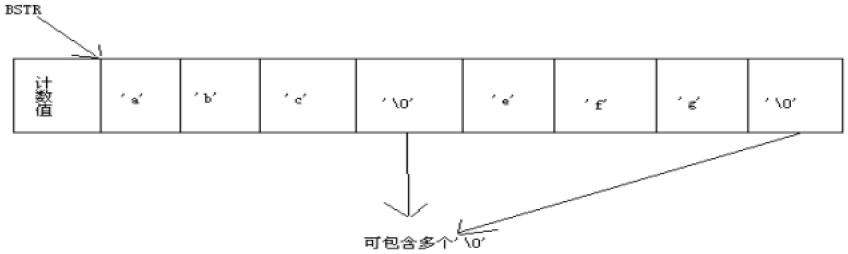
# COM数据类型讲解大纲

- 常用的COM数据类型有：

  CHAR, CHAR*, BYTE, BYTE*, SHORT, SHORT*, USHORT, USHORT*, INT, INT*, UINT, UINT*, LONG, LONG*, ULONG, ULONG*, FLOAT, FLOAT*, DOUBLE, DOUBLE*, VARIANT_BOOL, VARIANT_BOOL*, BSTR, BSTR*, IUnknown*, IUnknown**, IDispatch*, IDispatch**, VARIANT, VARIANT*

# BSTR

- BSTR,　　COM中的字符串类型
- BSTR*,　　BSTR的指针
- BSTR是指向的是宽字符串的指针，是一个带有字符计数值的字符串，且这个计数值是保存在字符数组的开头。



# BSTR

- 错误的赋值：

```
BSTR bstr = L"Hello UIPower";
```

之所有错误，是因为这样子的字符串前面没有计数值。

# BSTR

- BSTR SysAllocString(const OLECHAR *)是COM中申请BSTR字符串的方法。
- BSTR SysAllocStringLen(const OLECHAR *, UINT) 根据字符串指针与字符个数构造BSTR字符串。
- UINT SysStringLen(BSTR) 获取字符串前面的计数值。
- void SysFreeString(BSTR)释放字符串，当COM中的字符串（BSTR）不再使用时，调用该函数。

# BSTR

```
BSTR bstrA = SysAllocString(L"Hello BSTR");

BSTR bstrB = SysAllocStringLen(bstrA, SysStringLen(bstrA));

SysFreeString(bstrA);

SysFreeString(bstrB);
```

# COM数据类型讲解大纲

- 常用的COM数据类型有：

  CHAR, CHAR*, BYTE, BYTE*, SHORT, SHORT*, USHORT, USHORT*, INT, INT*, UINT, UINT*, LONG, LONG*, ULONG, ULONG*, FLOAT, FLOAT*, DOUBLE, DOUBLE*, VARIANT_BOOL, VARIANT_BOOL*, BSTR, BSTR*, IUnknown*, IUnknown**, IDispatch*, IDispatch**, VARIANT, VARIANT*

# 常用的COM数据类型

- IUnknown*, COM的最基本的接口指针
- IUnknown**, IUnknown*的指针
- IDispatch*, 支持组件自动化的接口
- IDispatch**, IDispatch*的指针

# COM数据类型讲解大纲

- 常用的COM数据类型有：

  CHAR, CHAR*, BYTE, BYTE*, SHORT, SHORT*, USHORT, USHORT*, INT, INT*, UINT, UINT*, LONG, LONG*, ULONG, ULONG*, FLOAT, FLOAT*, DOUBLE, DOUBLE*, VARIANT_BOOL, VARIANT_BOOL*, BSTR, BSTR*, IUnknown*, IUnknown**, IDispatch*, IDispatch**, VARIANT, VARIANT*

# VARIANT简介

- c++, vbscript, javascript……计算机语言多种多样，COM产生的目的之一就是要跨语言，而VARIANT数据类型就具有了跨语言的特性，同时它可能存储任何的数据类型，说夸张一点，它是“万能数据类型”。
- VARIANT*,　　VARIANT的指针

# VARIANT简介

- 为实现“万能类型”的功能，在c++中，VARIANT是一个结构体。该结构体内部又有联合体（联合了多种基本的数据类型），又有变量类型标志VARTYPE vt。可见VARIANT被设置得多么巧妙，合理。
- VARIANT结构体的定义，太长了，我们在这边不写出来了，我们可以查看Visual Studio 安装目录\VC\PlatformSDK\Include\oaidl.h文件中关于VARIANT结构体的定义。

# VARIANT的初始化与清除

- VariantInit()函数，用来初始化一个VARIANT 变量，把vt域设置成VT_EMPTY，表示空类型。vt域指示VARIANT结构体中的联合体所保存的数据类型。
- VariantClear()函数，用来清除一个VARIANT变量。

# VARIANT的初始化与清除

```
VARIANT var;
VariantInit(&var);
//此时  var.vt == VT_EMPTY;
//...其它操作
VariantClear(&var);
```

# VARIANT的使用

- 用VARIANT保存LONG类型。

```
VARIANT var;
VariantInit(&var);

var.vt = VT_I4;  //为什么vt是VT_I4呢？
var.lVal = 100;
VariantClear(&var);
```

//查看VARIANT结构体定义之前的注释可以查看到VT_I4与数据类型的对应关系，同时也可以查看到VT_I2，VT_R4等与相应数据类型的对应关系。

# VARIANT的使用

- 用VARIANT保存FLOAT类型。

```
VARIANT var;
VariantInit(&var);

var.vt = VT_R4;
var.fltVal = 1.23f;
VariantClear(&var);
```

# VARIANT的使用

- 用VARIANT保存BSTR类型。

```
VARIANT var;
VariantInit(&var);

var.vt = VT_BSTR;
var.bstrVal = SysAllocString(L"Hello UIPower");
VariantClear(&var);
```

# VARIANT的使用

- 用VARIANT保存类型。

```
VARIANT var;
VariantInit(&var);

var.vt = VT_BOOL;
var.boolVal = VARIANT_FALSE;
VariantClear(&var);
```

# 从VARIANT读取相应类型的值。

```
if (var.vt == VT_I4)
{
        long lValue = var.lVal;
}
else if (var.vt == VT_R4)
{
        float fValue = var.fltVal;
}
else if (var.vt == VT_BSTR)
{
        BSTR bstrValue = var.bstrVal;
}
else if (var.vt == VT_BOOL)
{
        VARIANT_BOOL varbValue =
var.boolVal;
}
```

# 这一讲，主要讲解如下要点：

- 一、IDispatch的简介与作用；
- 二、常用的COM数据类型；
- 三、COM数据类型的转换；
- 四、VARIANT类型的派生类。

# COM数据类型的转换

```
//LONG转换成FLOAT
VARIANT var;
VariantInit(&var);

var.vt = VT_I4;
var.lVal = 100;

VariantChangeType(&var,&var,0, VT_R4);

if (var.vt == VT_R4)
{
                float fValue = var.fltVal;
}

VariantClear(&var);
```

# COM数据类型的转换

```
//LONG转换成BSTR
VARIANT var;
VariantInit(&var);

var.vt = VT_I4;
var.lVal = 100;

VariantChangeType(&var,&var,0, VT_BSTR);

if (var.vt == VT_BSTR)
{
        BSTR fValue = var.bstrVal;
}

VariantClear(&var);
```

# 这一讲，主要讲解如下要点：

- 一、IDispatch的简介与作用；
- 二、常用的COM数据类型；
- 三、COM数据类型的转换；
- 四、VARIANT类型的派生类。

# VARIANT类型的派生类

直接通过VARIANT API,比如VariantInit, VariantClear等来操作VARIANT比较麻烦，不方便。

给VARIANT赋值也需要设置VARIANT.vt，再设置VARIANT.lVal或VARIANT.fltVal等，也比较麻烦，不方便。

于是有人编写了派生类(子类)来继承VARIANT，然后在派生类中拓展VARIANT的使用方法，使VARIANT的使用更方便，更直接。

# VARIANT类型的派生类

- 目前VARIANT的派生类主要有_variant_t，CComVariant，COleVariant。

  _variant_t是VC为支持COM类型而自带的VARIANT派生类，在Visual Studio安装目录\VC\include\comutil.h中定义。

  CComVariant是ATL库为支持COM类型而自带的VARIANT派生类，在Visual Studio 安装目录\VC\atlmfc\include\atlcomcli.h中定义。

  COleVariant是MFC库为支持COM类型而自带的VARIANT派生类，在Visual Studio 安装目录\VC\atlmfc\include\afxdisp.h中定义。
- 它们的功能都类似。下面我们以CComVariant为例讲解VARIANT类型的派生类。

# CComVariant简介

- CComVariant是VARIANT的派生类(子类)，拥有VARIANT的所有用法，可以访问VARIANT的成员。比如CComVariant.lVal或CComVariant.vt或CComVariant.fltVal等。可用CComVariant替代VARIANT。
- CComVariant提供了多种方法(包括构造方法，赋值方法)，方便了用户的使用。

# CComVariant的构造与析构

- 构造时，会对自身调用VariantInit()函数。
- 析构时，会对自身调用VariantClear()函数。

所以使用CComVariant代替VARIANT，就不需要再使用VariantInit()函数与VariantClear()函数。

# CComVariant的构造方法

- CComVariant()
- CComVariant(const VARIANT& varSrc)
- CComVariant(const CComVariant& varSrc)
- CComVariant(const wchar_t *wszSrc)
- CComVariant(const char *szSrc)
- CComVariant(bool bSrc)
- CComVariant(BYTE nSrc)
- CComVariant(short nSrc)
- CComVariant(float fltSrc)
- CComVariant(IDispatch* pSrc)
- CComVariant(IUnknown* pSrc)
- CComVariant(char cSrc)
- .....很多，不一一列举，可以用c++基本类型跟COM基本类型做参数构造一下CComVariant对象。使用例子查看Section3Demo1.cpp文件。

# CComVariant的赋值方法

- CComVariant& operator=(bool bSrc)
- CComVariant& operator=(int nSrc)
- CComVariant& operator=(BYTE nSrc)
- CComVariant& operator=(short nSrc)
- CComVariant& operator=(long nSrc)
- CComVariant& operator=(float fltSrc)
- CComVariant& operator=(double dblSrc)
- CComVariant& operator=(IDispatch* pSrc)
- CComVariant& operator=(IUnknown* pSrc)
- CComVariant& operator=(char cSrc)
- .....很多，不一一列举，可以用c++基本类型跟COM基本类型赋值给CComVariant对象。使用例子查看Section3Demo1.cpp文件。

# CComVariant的清除

- HRESULT Clear()
  //清除CComVariant，调用了VariantClear清除本身。

# CComVariant与VARIANT的关联

- HRESULT Attach(VARIANT* pSrc)
  //关联一个VARIANT，被关联后pSrc->vt=VT_EMPTY，CComVariant获得了关联对象的控制权。*pSrc这时不再使用，同时也无需调用VariantClear(pSrc)函数。
- HRESULT Detach(VARIANT* pDest)
  //Attack的反向过程，把CComVariant控制权交给*pDest对象，CComVariant本身的vt=VT_EMPTY。在不使用*pDest对象时，需要调用VariantClear(pDest)函数。
- 相关例子可以查看Section3Demo1.cpp文件

# 回顾

- 这一讲，主要讲解如下要点：

一、IDispatch的简介与作用；

自动化支持

二、常用的COM数据类型；

特别了解一下BSTR，VARIANT_BOOL，VARIANT类型

三、COM数据类型的转换；

通过VARIANT类型来转换数据类型

四、VARIANT类型的派生类。

继承了VARIANT的一切特性，可用CComVariant替代VARIANT。CComVariant提供了多种构造方法与赋值方法，用户可以直接方便地采用c++基本类型或COM基本类型构造或者赋值给CComVariant。