# COM数据类型

## COM数据类型

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\*

• 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

• 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

• 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

• 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

## BSTR类型

• 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。

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 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 bstrA = SysAllocString(L"Hello BSTR");

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

SysFreeString(bstrA);

SysFreeString(bstrB);

## VARIANT类型

### 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

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 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。

### 用VARIANT保存FLOAT类型。

VARIANT var;

VariantInit(&var);

var.vt = VT\_R4;

var.fltVal = 1.23f;

VariantClear(&var);

### 用VARIANT保存BSTR类型。

VARIANT var;

VariantInit(&var);

var.vt = VT\_BSTR;

var.bstrVal = SysAllocString(L"Hello UIPower");

VariantClear(&var);

### 用VARIANT保存VARIANT\_BOOL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;

}

### 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);

#### 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);

### 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。

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)

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。

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。