转 引用计数和AddRef、Release

2013年10月14日 23:22:09　阅读数：2019

AddRef和Release实现的是一种名为引用计数的内存管理技术，这种技术是使组件能够自己将自己删除的最简单同时也是效率最高的方法。COM组件将维护一个称作是引用计数的数值。当客户虫组件取得一个接口时，此数值增1，当客户使用完某个接口后，此数值将减1。当此数值为0时，组件即可将自己从内存中删除。

为正确的使用引用计数，需要了解一下三条规则：

（1） 在返回之前调用AddRef。对于哪些返回接口指针的函数，包括QueryInterface和CreateInstance，在返回之前用相应的指针调用AddRef。

（2） 使用完接口之后调用此接口的Release函数。

（3） 在赋值之后调用AddRef。如在将一个接口赋给另外一个接口指针时调用AddRef。

生命期嵌套在引用同一接口的指针的生命期内的指针可以不进行引用计数。在函数中，无需对存在于局部变量的接口指针进行引用计数。因为局部变量的生命期同函数的生命期是一样的，因此也将包含在调用者的生命期内。但当从某个全局变量或向某个全局变量复制一个指针时，则需要对此指针进行引用计数，因为全局变量可以从任意函数中的任意地方被释放。

一般而言，客户必须为每一个接口维护一个单独的引用计数值。

总结引用计数的几条具体规则如下：

（1） **输出参数规则** 。任何在输出参数中（如QueryInterface的void** ppv）或作为返回值返回一个新的接口指针的函数必须对此接口指针调用AddRef。即在返回之前调用AddRef。

（2） **输出参数规则** 。在输入参数（C++的按值传递的参数或常量）传入函数的接口指针，无需调用AddRef和Release。因为函数的生命期嵌套在调用者的生命期内。

（3） **输入－输出参数规则** ，即在函数体中可以使用输入－输出参数的值，然后可以对这些制进行修改并将其返回给调用者，对于具有这种功能的参数传进来的接口指针，必须在给它赋另外一个接口指针值之前调用其Release，并在函数返回之前，对输入参数中所保存的接口指针调用AddRef。

（4） **局部变量规则** 。对于局部复制的接口指针，由于它们只在函数的生命期内才存在，无需调用AddRef和Release。

（5） **全局变量规则** 。对于保存在全局变量中的接口指针，在将其传递给另外一个函数之前，必须调用其AddRef。对于保存在成员变量中的接口指针，也应按此中方式进行处理。因为类中的任何成员函数都可以改变此中接口指针的状态。

（6） **不能确定时的规则** 。对于任何不能确定的情形，都应调用AddRef和Release。

在决定要对引用计数进行优化时，应给哪些没有进行引用计数的指针加上相应的注释，否则，其他程序员在修改代码时，将可能会增大接口指针的生命期，从而使引用计数的优化遭到破坏 。

文章标签： 引用计数　AddRef　Release

个人分类： 纯编程