Understanding what things mean in context: Dispatch interfaces

devblogs.microsoft.com/oldnewthing/20060116-10

January 16, 2006

Raymond Chen

Remember that you have to understand what things mean in context. For example, the IActiveMovie3 interface has a method called get_MediaPlayer. If you come into this method without any context, you might expect it to return a pointer to an IMediaPlayer interface, yet the header file says that it returns a pointer to an IDispatch interface instead. If you look at the bigger picture, you'll see why this makes sense. IActiveMovie3 is an IDispatch interface. As you well know, the IDispatch interface's target audience is scripting languages, primarily classic Visual Basic (and to a lesser degree, JScript). Classic Visual Basic is a dynamically-typed language, wherein nearly all variables are merely "objects", the precise type of which is not known until run-time. A statically-typed language will complain at compile time that you are invoking a method on an object that doesn't support that method or that you are passing the wrong number or type of operands to a method. A dynamically-typed language, on the other hand, doesn't check until the line of code is actually executed whether the method exists, and if it does, whether you called it correctly. When working with IDispatch and dynamically-typed languages, therefore, the natural unit of currency for objects is the IDispatch. All objects take the form of IDispatch. Objects that produce other objects will produce IDispatch interfaces, because that's what the scripting engine is expecting. That's why the get_MediaPlayer method returns an IDispatch. Because that's what the scripting engine expects. And, if you are familiar with the context, it's also what you should expect.

A tell-tale sign of this context comes from the name "get_MediaPlayer". This name does not follow the COM function naming convention but rather is a constructed name for the C/C++ binding of the "get" property. C/C++ bindings are the assembly language of OLE automation: You're operating with the nuts and bolts of OLE automation, and if you want to play at this level, you're going to have to know how to use a screwdriver.

Raymond Chen

Follow