Why do some shortcuts not support editing the command line and other properties?

devblogs.microsoft.com/oldnewthing/20121210-00

December 10, 2012

Raymond Chen

Ben L observed that some shortcuts do not permit the command line and other options to be edited. "Where is this feature controlled? Is there a way to override this mode?" This question is echoed by "Anonymous (Mosquito buzzing around)" (and don't think we don't know who you are), who in a huge laundry list of questions adds, "Why does the Game Explorer limit customizing command line, target, etc?" These questions are looking at the situation backwards. The issue is not "Why do these shortcuts block editing the command line?" The issue is "Why do some shortcuts *allow* editing the command line?" Recall that shortcuts are references to objects in the shell namespace. Shell namespace objects are abstract. Some of them refer to files, but others refer to non-file objects, like control panels, printers, and dial-up networking connectoids. And in the abstract, these objects support verbs like *Open* and *Rename*. But there is no requirement that a shell namespace object support "Run with command line argument". If you have a shortcut to *an executable*, then the LNK file handler says, "Okay, this is a special case. Executables support command line arguments, so I will run the executable with the command line arguments set by the `IShellLink::SetArguments` method. Note that the shortcut target and arguments are separate properties. The LNK file property sheet hides this from you by calling `IShellLink::GetPath` and `IShellLink::GetArguments`, then taking the two strings and combining them into a single *Target* field for display. When you save the changes, the LNK file property sheet takes the *Target*, figures out which part is the executable and which part is the arguments, and calls `IShellLink::SetPath` and `IShellLink::SetArguments` on the two parts. In other words, the command line is all a ruse. This special action is performed only for executable targets, because those are the only things that accept arguments. If you create a shortcut to a control panel, you'll find that the *Target* is not editable. If you create a shortcut to a printer, you'll find that the *Target* is not editable. If you create a shortcut to a dial-up networking connectoid, you'll find that the *Target* is not editable. Having a non-editable command line is the normal case. *The file system is the weirdo*. Shortcuts to advertise applications and shortcuts to items in the Games folder are not shortcuts to executables. They are shortcuts into the shell namespace for various types of virtual data. An advertised application is a shell namespace object that represents "an installed application". It is not a pointer directly to the executable, but rather a reference to an entry in the MSI database,

which in turn contains information about how to install the program, repair it, update it, and run it. The shell doesn't even know what the command line is. To launch an advertised shortcut, the shell asks the MSI database for the command line, and it then executes that command line that MSI returns. The value set by `IShellLink::SetArguments` never enters the picture. Similarly, the entries in the Games Folder are not executables; they are entries in the games database.

I can see how this can be confusing, because when you click on these shortcuts, *a program runs*, but these shortcuts are not shortcuts directly to programs. As a result, the code that takes a *Target* and *Arguments* and combines them into a command line does not get a chance to run.

Raymond Chen

Follow