PYTHON | WINDOWS

Using Python to Create Shortcuts

At my job, I do a fair amount of system administration scripting in Python. For example, almost all the login scripts are written in Python (with some of them ported from Kixtart). Over the years, I've been tasked with creating shortcuts to new applications that need to be placed on the user's desktop or in their Start Menu or both. In this article, I will show you how to accomplish this task.

Note: This is a Windows-only article, so if you don't use that OS, then this will probably bore you to tears. Heck, it might do that anyway!

The first thing you'll need to do much of anything on Windows is Mark Hammond's PyWin32 package (AKA: Python for Windows extensions). I also recommend Tim Golden's winshell module as it can make finding the user-specific folders much easier. I'll be using Python 2.5 for this article, but as I understand it, PyWin32 is compatible with Python 3, so this tutorial could apply to those of you who use that too.

The simplest task I've had to do is to create a URL link on the user's desktop. Unless you need to specify a specific browser, this is a really simple task:

```
import os, winshell

desktop = winshell.desktop()
path = os.path.join(desktop, "myNeatWebsite.url")
target = "http://www.google.com/"

shortcut = file(path, 'w')
shortcut.write('[InternetShortcut]\n')
shortcut.write('URL=%s' % target)
shortcut.close()
```

In the code above, we import the os and winshell modules. We use winshell to grab the current user's desktop path and then use the os.path's join() function to connect the path and the name of the url shortcut. Since this is Windows only, you really don't need to do it this way. A string concatenation would work just as well. Note that we need to supply a "url" extension so Windows knows what to do. Then we write a file using the path we created before and write *[InternetShortcut]* as the first line. On the second line, we write the url target and then we close the file. That's it!

The next example will be a little more complex. In it, we will use the win32com module from the PyWin32 package to create a shortcut to Media Player Classic, which is a nice open-source media player. Let's take a look at some code so we can see how to do this thing:

```
import os, winshell
from win32com.client import Dispatch
```

```

desktop = winshell.desktop()
path = os.path.join(desktop, "Media Player Classic.lnk")
target = r"P:\Media\Media Player Classic\mplayerc.exe"
wDir = r"P:\Media\Media Player Classic"
icon = r"P:\Media\Media Player Classic\mplayerc.exe"

shell = Dispatch('WScript.Shell')
shortcut = shell.CreateShortCut(path)
shortcut.Targetpath = target
shortcut.WorkingDirectory = wDir
shortcut.IconLocation = icon
shortcut.save()
```

The main takeaway here is the shell part. First you import *Dispatch* from *win32com.client*, then you call *Dispatch('WScript.Shell')* to get a shell object (sort of) which you then use to create a shortcut object. Once you have that, you can assign values to the shortcut's attributes, which are Targetpath, WorkingDirectory and IconLocation. Note that WorkingDirectory corresponds to the "Start In" field in a normal shortcut's property dialog. In the script above, the IconLocation can be an icon file or it can extract the icon directly from the executable. One small got'cha here is that if you don't call *save*, then the icon will not be created. In the previous example, we don't explicitly have to call *close* on the file object because Python will take care of that for us when the script finishes.

Let's take these two examples and make a reusable function out of them so we can use them in any login script we want:

```
from win32com.client import Dispatch

def createShortcut(path, target='', wDir='', icon=''):
    ext = path[-3:]
    if ext == 'url':
        shortcut = file(path, 'w')
        shortcut.write('[InternetShortcut]\n')
        shortcut.write('URL=%s' % target)
        shortcut.close()
    else:
        shell = Dispatch('WScript.Shell')
        shortcut = shell.CreateShortCut(path)
        shortcut.Targetpath = target
        shortcut.WorkingDirectory = wDir
        if icon == '':
            pass
        else:
            shortcut.IconLocation = icon
        shortcut.save()
```

One obvious improvement we could add to this would be to use the *os.path.dirname* method to extract the Working Directory from the target and eliminate the need to pass that information in. Of course, I've seen some screwy shortcuts that don't specify the target or the Working Directory at all! Anyway, I hope you will find this article helpful in your scripting. Until next time!