How can I get the list of programs the same way that Programs and Features gets it?

devblogs.microsoft.com/oldnewthing/20131230-00

December 30, 2013

Raymond Chen

A customer wanted to get the list of programs the same way that the Programs and Features folder gets it.

Here, here's an idea: Instead of trying to mimic the Programs and Features folder, just ask the Programs and Features folder for its contents! That way, no matter what changes are made to how the Programs and Features folder obtains its contents (and those changes occur pretty often), your program will always match it, because you're just showing the same thing.

Here's the basic idea, in scripting language since it's quicker:

```
var shell = new ActiveXObject("Shell.Application");
var programsFolder = shell.Namespace(
    "::{26EE0668-A00A-44D7-9371-BEB064C98683}\\8\\" +
    "::{7B81BE6A-CE2B-4676-A29E-EB907A5126C5}");
var items = programsFolder.Items();
for (var i = 0; i < items.Count; i++) {
    var item = items.Item(i);
    WScript.StdOut.WriteLine(item);
    WScript.StdOut.WriteLine("Size = " + item.ExtendedProperty("System.Size"));
    WScript.StdOut.WriteLine("------------");
}
```

Okay, first of all, how did I get that magic string for the Programs and Features folder? I opened the Control Panel and dragged the *Uninstall a program* link onto the program from a few weeks ago.

The program itself is pretty straightforward. It's the standard *enumerate everything in a folder and print it* program we've seen before. The only trick was finding the folder.

As for the C++ version, it should also look familiar, because we've done it before more than once. The only difference is the way we create the folder and the details we choose to display. (For extra credit: Change this program to bind to the persisted pidl instead of the parsing name.)

```
int __cdecl wmain(int argc, wchar_t **argv)
{
 CCoInitialize init;
 CComPtr<IShellItem> spPrinters;
 SHCreateItemFromParsingName(
   L"::{26EE0668-A00A-44D7-9371-BEB064C98683}\\8\\"
   L"::{7B81BE6A-CE2B-4676-A29E-EB907A5126C5}", nullptr,
   IID_PPV_ARGS(&spPrograms));
 CComPtr<IEnumShellItems> spEnum;
 spPrograms->BindToHandler(nullptr, BHID_EnumItems,
                              IID_PPV_ARGS(&spEnum));
 for (CComPtr<IShellItem> spProgram;
      spEnum->Next(1, &spProgram, nullptr) == S_OK;
      spProgram.Release()) {
  CComHeapPtr<wchar_t> spszName;
  spProgram->GetDisplayName(SIGDN_NORMALDISPLAY, &spszName);
  wprintf(L"%ls\n", spszName);
  PrintDetail(CComQIPtr<IShellItem2>(spProgram), &PKEY_Size, L"Size");
 }
 return 0;
}
```

Bonus script: You can even see what verbs are available.

```
var shell = new ActiveXObject("Shell.Application");
var programsFolder = shell.Namespace(
    "::{26EE0668-A00A-44D7-9371-BEB064C98683}\\8\\" +
    "::{7B81BE6A-CE2B-4676-A29E-EB907A5126C5}");
var items = programsFolder.Items();
for (var i = 0; i < items.Count; i++) {
    var item = items.Item(i);
    WScript.StdOut.WriteLine(item);
    WScript.StdOut.WriteLine("Size = " + item.ExtendedProperty("System.Size"));
    var verbs = item.Verbs();
    for (var j = 0; j < verbs.Count; j++) {
       var verb = verbs.Item(j);
       WScript.StdOut.WriteLine("Action: " + verb.Name);
    }
    WScript.StdOut.WriteLine("------------");
}
```

And if you're really ambitious, you can even call `verb.DoIt` to carry out the action. Don't use this power for evil.

Note: Since we are working with the Programs and Features folder, we are necessarily targeting Windows Vista and later, since that was the version of Windows in which the Programs and Features folder was introduced. Therefore, I am free to use functionality introduced in Windows Vista.

I've been doing Little Programs for a year now. I kind of like it, so I'm going to continue for another year, but I'm going to relax the rules a bit: The Little Programs are now just programs that I think are interesting for whatever reason. They don't need to actually solve a problem.

Raymond Chen

Follow