Filtering the Browse for Folder dialog so it shows only drive letters

devblogs.microsoft.com/oldnewthing/20170424-00

April 24, 2017

Raymond Chen

Today, we're going to customize the Browse for Folder dialog so it shows only drive letters.

Start with our previous Browse for Folder customization program, and make these changes:

```
// Lazy global variable
PIDLIST_ABSOLUTE g_pidlMyComputer;

class CFunnyFilter :
    public RuntimeClass<
    RuntimeClassFlags<RuntimeClassType::ClassicCom>,
    IFolderFilter>
{
public:
  // *** IFolderFilter ***
  IFACEMETHODIMP ShouldShow(
        IShellFolder* psf,
        PCIDLIST_ABSOLUTE pidlFolder,
        PCUITEMID_CHILD pidlItem)
  {
    int compare = CompareDepth(pidlFolder);
    if (compare < 0) return S_OK;
    if (compare > 0) return S_FALSE;

    STRRET str;
    psf->GetDisplayNameOf(pidlItem, SHGDN_FORPARSING, &str);
    wchar_t buf[4];
    if (SUCCEEDED(StrRetToBuf(&str, pidlItem, buf, ARRAYSIZE(buf))) &&
        PathIsRoot(buf)) return S_OK;
    return S_FALSE;
  }

  IFACEMETHODIMP GetEnumFlags(
      IShellFolder* psf,
      PCIDLIST_ABSOLUTE pidlFolder,
      HWND *phwnd,
      DWORD *pgrfFlags) {
    if (CompareDepth(pidlFolder) > 0) *pgrfFlags = 0;
    return S_OK;
  }

private:
  static int CompareDepth(PCIDLIST_ABSOLUTE pidl)
  {
    if (pidl == nullptr) return -1;
    if (ILIsEqual(pidl, g_pidlMyComputer)) return 0;
    if (ILIsParent(pidl, g_pidlMyComputer, FALSE)) return -1;
    return 1;
  }
};

int WINAPI WinMain(HINSTANCE hinst, HINSTANCE hinstPrev,
                   LPSTR lpCmdLine, int nShowCmd)
{
  CCoInitialize init;
  BROWSEINFO bi = { };
  TCHAR szDisplayName[MAX_PATH];
```

```
  SHGetSpecialFolderLocation(nullptr, CSIDL_DRIVES, &g_pidlMyComputer);
  bi.pidlRoot = g_pidlMyComputer;
  bi.pszDisplayName = szDisplayName;
  bi.lpfn = BrowseCallbackProc;
  bi.ulFlags = BIF_NEWDIALOGSTYLE | BIF_RETURNONLYFSDIRS;
  PIDLIST_ABSOLUTE pidl = SHBrowseForFolder(&bi);
  CoTaskMemFree(pidl);
  CoTaskMemFree(g_pidlMyComputer);
  return 0;
}
```

Okay, let's see what we've got.

First, we declare a global variable to remember the location of what was once called *My Computer* but nowadays goes by the name *This PC*. Whatever it is, it's the thing that contains your drive letters.

The real work happens in the filter. Starting at the bottom, we have a method called `CheckDepth` which determines whether the passed-in folder is an ancestor of, equal to, or a descendant of *My Computer*. Actually, we treat anything that isn't a parent or equal to *My Computer* as if it were a descendant.

The `CheckDepth` method is method is a bit tricky for a few reasons. First, it treats the null pointer as equivalent to the desktop, so that it is the ancestor of everything. For whatever reason, that's what `IFolderFilter` gives you, so we accommodate it.

Second, if you pass `FALSE` to `ILIsParent`, it means that the function will return a nonzero value if the first ID list is an ancestor of *or is equal to* the second ID list. Therefore, we have to do the equality test first.

Okay, working upward, the next method is `GetEnumFlags`. This is called when the Browse for Folder dialog wants to enumerate the children of a folder, and it's our chance to influence what gets enumerated. We don't want to expand the drives themselves, so if we have something that is a child of *My Computer*, we set the enumeration flags to zero, which means that nothing gets enumerated.

The first method is `ShouldShow`. This is where most of the excitement is. You are given a folder and an item in that folder, and your job is to decide whether that item should be shown in the Browse for Folder dialog.

First, we say that folders which are ancestors of *My Computer* can show all of their children. This ensures that the Browse for Folder dialog can reach *My Computer* in the first place.

Second, we say that descendants of *My Computer* do not show any children. This is technically redundant because our `GetEnumFlags` prevented those children from being enumerated, but we'll block them here just to be sure they don't show up.

Finally, if we are showing children of *My Computer* itself, we ask for the parsing name of the item and see if a drive root comes back. If the parsing name is longer than four characters, then the `StrRetToBuf` function will fail with an insufficient-buffer error, in which case we know that we don't have a drive root.

The handy `StrRetToBuf` function deals with the kooky `STRRET` structure so we don't have to.

So that's the filtering. The last changes are to `WinMain`: We obtain the item ID list for *My Computer* and set it as the root for the Browse for Folder dialog. (Remember that Little Programs do little to no error checking.) We also tell the Browse for Folder dialog that we require the user to select a file system object. That ensures that the *OK* button is disabled when the user is sitting at *My Computer*. And after the excitement is done, we clean up.

There you have it. A Browse for Folder dialog that shows only drive letters.

I'm not sure how useful this is, but I never claimed that this was useful.

Raymond Chen