How do I invoke a verb on an IShellItemArray?

devblogs.microsoft.com/oldnewthing/20120920-00

September 20, 2012

Raymond Chen

A customer wanted to invoke a command on multiple items at once.

> I have an `IShellItemArray`, and I want to invoke a verb with that array as the parameter. I know that I can invoke a verb on a single `IShellItem` by the code below, but how do I pass an entire array?

```
void InvokeVerbOnItem(__in IShellItem *psi,
                      __in_opt PCWSTR pszVerb)
{
 PIDLIST_ABSOLUTE pidl;
 HRESULT hr = SHGetIDListFromObject(psi, &pidl);
 if (SUCCEEDED(hr)) {
  SHELLEXECUTEINFO sei = { sizeof(sei) };
  sei.fMask = SEE_MASK_UNICODE |
              SEE_MASK_INVOKEIDLIST |
              SEE_MASK_IDLIST;
  sei.lpIDList = pidl;
  sei.lpVerb = pszVerb;
  sei.nShow = SW_SHOWNORMAL;
  ShellExecuteEx(&sei);
  CoTaskMemFree(pidl);
 }
}
```

The function `InvokeVerbOnItem` invokes the command by extracting the pidl, then asking `ShellExecuteEx` to invoke the command on the pidl. A limitation of `ShellExecute*` is that it can invoke on only one pidl. What if you want to invoke it on a bunch of pidls at once? (Doing it all at once gives the target program the opportunity to optimize the multi-target invoke.)

As noted in the documentation, passing `SEE_MASK_INVOKEIDLIST` flag tells the `ShellExecuteEx` to "use the **IContextMenu** interface of the selected item's shortcut menu handler."

So if you are frustrated by the limitations of the middle man, then cut out the middle man!

```
void InvokeVerbOnItemArray(__in IShellItemArray *psia,
                           __in_opt PCWSTR pszVerb)
{
 IContextMenu *pcm;
 HRESULT hr = psia->BindToHandler(BHID_SFUIObject,
                                  IID_PPV_ARGS(&pcm));
 if (SUCCEEDED(hr)) {
  ... context menu invoke incorporated by reference ...
  pcm->Release();
 }
}
```

If you think about it, the original `InvokeVerbOnItem` function could've avoid the middle man too. It converted an `IShellItem` (a live object which encapsulates an `IShellFolder` and a child pidl) into an absolute pidl (a dead object), which then passed it to `ShellExecuteEx`, which had to reanimate the object back into an `IShellFolder` and child pidl so it could call `GetUIObjectOf`.

Raymond Chen

Follow