Invoking commands on items in the Recycle Bin

devblogs.microsoft.com/oldnewthing/20110901-00

September 1, 2011

Raymond Chen

Once you've found the items you want in the Recycle Bin, you may want to perform some operation on them. This brings us back to our old friend, `IContextMenu`. At this point, you're just snapping two blocks together. You have one block called *Retrieving properties from items in the Recycle Bin* and you have another block called *Invoking verbs on items*.

For the first block, let's assume you've written a function called `WantToRestoreThisItem` which studies the properties of a Recycle Bin item and determines whether you want to restore it. I leave this for you to implement, since I don't know what your criteria are. Maybe you want to restore files only if they were deleted from a particular directory. Maybe you want to restore files that were deleted while you were drunk. (This assumes you have some other computer program that tracks when you're drunk.)[1] Whatever. It's your function.

For the second block, we have a helper function which should look awfully familiar.

```
void InvokeVerb(IContextMenu *pcm, PCSTR pszVerb)
{
 HMENU hmenu = CreatePopupMenu();
 if (hmenu) {
  HRESULT hr = pcm->QueryContextMenu(hmenu, 0, 1, 0x7FFF, CMF_NORMAL);
  if(SUCCEEDED(hr)) {
   CMINVOKECOMMANDINFO info = { 0 };
   info.cbSize = sizeof(info);
   info.lpVerb = pszVerb;
   pcm->InvokeCommand(&info);
  }
  DestroyMenu(hmenu);
 }
}
```

And now we snap the two blocks together.

```
int __cdecl _tmain(int argc, PTSTR *argv)
{
 HRESULT hr = CoInitialize(NULL);
 if (SUCCEEDED(hr)) {
  IShellItem *psiRecycleBin;
  hr = SHGetKnownFolderItem(FOLDERID_RecycleBinFolder, KF_FLAG_DEFAULT,
                            NULL, IID_PPV_ARGS(&psiRecycleBin));
  if (SUCCEEDED(hr)) {
   IEnumShellItems *pesi;
   hr = psiRecycleBin->BindToHandler(NULL, BHID_EnumItems,
                                     IID_PPV_ARGS(&pesi));
   if (hr == S_OK) {
    IShellItem *psi;
    while (pesi->Next(1, &psi, NULL) == S_OK) {
     if (WantToRestoreThisItem(psi)) {
      IContextMenu *pcm;
      hr = psi->BindToHandler(NULL, BHID_SFUIObject,
                              IID_PPV_ARGS(&pcm));
      if (SUCCEEDED(hr)) {
       InvokeVerb(pcm, "undelete");
       pcm->Release();
      }
     }
     psi->Release();
    }
   }
   psiRecycleBin->Release();
  }
  CoUninitialize();
 }
 return 0;
}
```

One annoyance of the Recycle Bin is that, at least up until Windows 7, it ignores the `CMIC_MASK_FLAG_NO_UI` flag. It always displays a confirmation dialog if something dangerous is about to happen (like overwriting an existing file). To mitigate this problem, we can at least reduce the number of confirmations from one-per-file to just one by batching up all the objects we want to operate on into a single context menu. For this, it's easier to go back to the classical version of the program.

```
int __cdecl _tmain(int argc, PTSTR *argv)
{
 HRESULT hr = CoInitialize(NULL);
 if (SUCCEEDED(hr)) {
  IShellFolder2 *psfRecycleBin;
  hr = BindToCsidl(CSIDL_BITBUCKET, IID_PPV_ARGS(&psfRecycleBin));
  if (SUCCEEDED(hr)) {
   IEnumIDList *peidl;
   hr = psfRecycleBin->EnumObjects(NULL,
     SHCONTF_FOLDERS | SHCONTF_NONFOLDERS, &peidl);
   if (hr == S_OK) {
    // in a real program you wouldn't hard-code a fixed limit
    PITEMID_CHILD rgpidlItems[100];
    UINT cpidlItems = 0;
    PITEMID_CHILD pidlItem;
    while (peidl->Next(1, &pidlItem, NULL) == S_OK) {
     if (WantToRestoreThisItem(psfRecycleBin, pidlItem) &&
         cpidlItems < ARRAYSIZE(rgpidlItems)) {
      rgpidlItems[cpidlItems++] = pidlItem;
     } else {
      CoTaskMemFree(pidlItem);
     }
    }
    // restore the items we collected
    if (cpidlItems) {
     IContextMenu *pcm;
     hr = psfRecycleBin->GetUIObjectOf(NULL, cpidlItems,
                    (PCUITEMID_CHILD_ARRAY)rgpidlItems,
                    IID_IContextMenu, NULL, (void**)&pcm);
     if (SUCCEEDED(hr)) {
      InvokeVerb(pcm, "undelete");
      pcm->Release();
     }
     for (UINT i = 0; i < cpidlItems; i++) {
      CoTaskMemFree(rgpidlItems[i]);
     }
    }
   }
   psfRecycleBin->Release();
  }
  CoUninitialize();
 }
 return 0;
}
```

In the course of the enumeration, we save the `ITEMIDLIST`s of all the items we want to restore, then create one giant context menu for all of them. This is the programmatic equivalent of multi-selecting the items from the Recycle Bin and then right-clicking. We then invoke the undelete verb on the entire group.

Okay, so now suppose you want to restore the files, but instead of restoring them to their original locations, you want to restore them to a special folder. Like, say, *C:\Files I deleted while I was drunk*.[1] No problem. We just need a different block to snap into: The drag/drop block.

```
void DropOnRestoreFolder(IDataObject *pdto)
{
 IDropTarget *pdt;
 if (SUCCEEDED(GetUIObjectOfFile(NULL,
        L"C:\\Files I deleted while I was drunk",
        IID_PPV_ARGS(&pdt)))) {
  POINTL pt = { 0, 0 };
  DWORD dwEffect = DROPEFFECT_MOVE;
  if (SUCCEEDED(pdt->DragEnter(pdto, MK_LBUTTON,
                               pt, &dwEffect))) {
   dwEffect &= DROPEFFECT_MOVE;
   if (dwEffect) {
    pdt->Drop(pdto, MK_LBUTTON, pt, &dwEffect);
   } else {
    pdt->DragLeave();
   }
  }
  pdt->Release();
 }
}
```

And now it's just a matter of snapping out the undelete block and snapping in the drag/drop block.

```
int __cdecl _tmain(int argc, PTSTR *argv)
{
 HRESULT hr = CoInitialize(NULL);
 if (SUCCEEDED(hr)) {
  IShellFolder2 *psfRecycleBin;
  hr = BindToCsidl(CSIDL_BITBUCKET, IID_PPV_ARGS(&psfRecycleBin));
  if (SUCCEEDED(hr)) {
   IEnumIDList *peidl;
   hr = psfRecycleBin->EnumObjects(NULL,
     SHCONTF_FOLDERS | SHCONTF_NONFOLDERS, &peidl);
   if (hr == S_OK) {
    // in a real program you wouldn't hard-code a fixed limit
    PITEMID_CHILD rgpidlItems[100];
    UINT cpidlItems = 0;
    PITEMID_CHILD pidlItem;
    while (peidl->Next(1, &pidlItem, NULL) == S_OK) {
     if (WantToRestoreThisItem(psfRecycleBin, pidlItem) &&
         cpidlItems < ARRAYSIZE(rgpidlItems)) {
      rgpidlItems[cpidlItems++] = pidlItem;
     } else {
      CoTaskMemFree(pidlItem);
     }
    }
    // restore the items we collected
    if (cpidlItems) {
     IDataObject *pdto;
     hr = psfRecycleBin->GetUIObjectOf(NULL, cpidlItems,
                    (PCUITEMID_CHILD_ARRAY)rgpidlItems,
                    IID_IDataObject, NULL, (void**)&pdto);
     if (SUCCEEDED(hr)) {
      DropOnRestoreFolder(pdto);
      pdto->Release();
     }
     for (UINT i = 0; i < cpidlItems; i++) {
      CoTaskMemFree(rgpidlItems[i]);
     }
    }
   }
   psfRecycleBin->Release();
  }
  CoUninitialize();
 }
 return 0;
}
```

Footnotes

[1] If being drunk isn't your thing, then substitute some other form of impaired judgment.

Raymond Chen

Follow