The dialog manager, part 3: Creating the controls

devblogs.microsoft.com/oldnewthing/20050331-00

March 31, 2005

Raymond Chen

This is actually a lot less work than creating the frame, believe it or not.

For each control in the template, the corresponding child window is created. The control's sizes and position is specified in the template in DLUs, so of course they need to be converted to pixels.

```
int x = XDLU2Pix(ItemTemplate.x);
int y = YDLU2Pix(ItemTemplate.y);
int cx = XDLU2Pix(ItemTemplate.cx);
int cy = YDLU2Pix(ItemTemplate.cy);
```

The class name and caption also come from the template. There are also the optional extra bytes `pExtra` which nobody uses but which remain in the template definition for historical reasons. Once that information has been collected, it's time to make the donuts.

```
HWND hwndChild = CreateWindowEx(
            ItemTemplate.dwExStyle | WS_EX_NOPARENTNOTIFY,
            pszClass, pwzCaption, ItemTemplate.dwStyle,
            x, y, cx, cy, hdlg, ItemTemplate.dwId,
            hinst, pExtra);
```

Notice that the `WS_EX_NOPARENTNOTIFY` style is forced on for dialog controls.

This next part often trips people up. "When I try to create my dialog, it fails and I don't know why." It's probably because one of the controls on the dialog could not be created, usually because you forgot to register the window class for that control. (For example, you forgot to call the `InitCommonControlsEx` function or you forgot to `LoadLibrary` the appropriate version of the RichEdit control.)

```
if (!hwndChild) {
  DestroyWindow(hdlg);
  return NULL;
}
```

The `DS_NOFAILCREATE` style suppresses the failure check above.

But if the control did get created, then it needs to be initialized.

```
  SetWindowContextHelpId(hwndChild, ItemTemplate.dwHelpID);
  SetWindowFont(hwndChild, hf, FALSE);
```

Repeat once for each item template, and you now have a dialog box with all its child controls. Tell the dialog procedure that it can initialize its child windows, show the (now-ready) dialog box if we deferred the `WS_VISIBLE` bit when constructing the frame, and return the dialog box to our caller, ready for action.

```
  // The default focus is the first item that is a valid tab-stop.
  HWND hwndDefaultFocus = GetNextDlgTabItem(hdlg, NULL, FALSE);
  if (SendMessage(hdlg, WM_INITDIALOG, hwndDefaultFocus, lParam)) {
     SetDialogFocus(hwndDefaultFocus);
  }
  if (fWasVisible) ShowWindow(hdlg);
  return hdlg;
}
```

The `SetDialogFocus` function we saw last year.

So there you have it: You have now seen how dialog box sausages are made.

(Actually, reality is much sausagier, since I skipped over all the app compat hacks! For example, there's a program out there that relies on the subtle placement and absence of the `WS_BORDER` style to decide whether a control is a combo box or a listbox. I guess the `GetClassName` function was too much work?)

I hope this helps you understand a little better how dialog templates fit into the big picture.

Raymond Chen

Follow