

转 MFC窗口最小化到托盘

2013年10月13日 16:29:10 阅读量：3128

把程序放到托盘上的本质就是先在托盘区绘制一个图标，然后把程序隐藏不见，再对托盘的图标进行消息处理，就可以了。绘制图标以及确定图标所传递消息的函数只有一个，那就是 `WINSHELLAPI BOOL WINAPI Shell_NotifyIcon(DWORD dwMessage, NOTIFYICONDATA pnid);` 这个函数负责向系统传递消息，以添加、修改或删除托盘区的图标。它的返回值是个布尔类型的。就是说，如果返回0就是不成功，非0才成功。参数 `dwMessage` 是表示这个函数的应用功能是哪一方面，是添加、删除，还是修改图标。如果是添加，则它的值为 `NIM_ADD`；删除则是 `NIM_DELETE`；而修改是 `NIM_MODIFY`。参数 `pnid` 就是具体的和程序在托盘区的图标有关的结构了。它的定义如下：

```
[cpp]
1. typedef struct _NOTIFYICONDATA { DWORD cbSize;
2.   HWND hWnd;
3.   UINT uID;
4.   UINT uFlags;
5.   UINT uCallbackMessage;
6.   HICON hIcon;
7.   char szTip[64];
8. } NOTIFYICONDATA, *PNOTIFYICONDATA;
```

`cbSize`：结构的长度，用“位”来做单位。一般在程序中，我们用 `(DWORD)sizeof(NOTIFYICONDATA)` 给它赋值。

`hWnd`：一个句柄，如果对托盘中的图标进行操作，相应的消息就传给这个句柄所代表的窗口。大多数情况下是 `this->m_hWnd`。

`uID`：在工程中定义的图标ID

`uFlags`：这个成员标志着其他哪些成员的数据是有效的，分别为 `NIF_ICON`, `NIF_MESSAGE`, `NIF_TIP`，分别代表着数据有效的成员是 `hIcon`, `uCallbackMessage`, `szTip`。当然，三个值可以用“|”联系在一起。

下面分别对涉及到的成员进行阐述：

`hIcon`：要增加，删除或修改的图标句柄。如果只知道个 `uID`，一般可能会用函数 `LoadIcon` 来得到句柄。例如 `LoadIcon (AfxGetInstanceHandle() ,MAKEINTRESOURCE (IDR_MAINFRAME))`。

`uCallbackMessage`：这在对托盘区的操作中，是比较重要的数据成员。这是个消息标志，当用鼠标对托盘区相应图标进行操作的时候，就会传递消息给 `hWnd` 所代表的窗口。所以说，在 `uFlags` 中，一般都得标志它有效。这里一般都是自定义的消息。

`szTip`：鼠标移动到托盘图标上时的提示文字。

托盘编程例子：

1、将程序最小化到系统托盘区的函数 `toTray()`。

```
[cpp]
1. void CTimeWakeDlg::toTray()
2. {
3.     NOTIFYICONDATA nid;
4.     nid.cbSize=(DWORD)sizeof(NOTIFYICONDATA);
5.     nid.hWnd=this->m_hWnd;  nid.uID=IDR_MAINFRAME;
6.     nid.uFlags=NIF_ICON|NIF_MESSAGE|NIF_TIP ;
7.     nid.uCallbackMessage=WM_SHOWTASK;//自定义的消息名称
8.     nid.hIcon=LoadIcon(AfxGetInstanceHandle(),MAKEINTRESOURCE(IDR_MAINFRAME));
9.     strcpy(nid.szTip,"计划任务提醒");//信息提示条为“计划任务提醒”
10.    Shell_NotifyIcon(NIM_ADD,&nid);//在托盘区添加图标
11.    ShowWindow(SW_HIDE);//隐藏主窗口
12. }
```

2、程序已经最小化到托盘区了，但是对托盘图标的操作如何进行呢？这就体现了结构 `NOTIFYICONDATA` 的成员 `uCallbackMessage` 的作用了。它所提供的作用就是，当用户用鼠标点击托盘区的图标的时候（无论是左键还是右键），会向 `hWnd` 所代表的窗口传递消息，如果是上例，消息的名称就是 `WM_SHOWTASK`。

根据VC的消息机制，对自定义消息增加消息响应函数。

在头文件的 `//{{AFX_MSG和//}}AFX_MSG` 之间声明消息响应函数：

```
afx_msg LRESULT onShowTask(WPARAM wParam,LPARAM lParam);
```

然后在CPP文件中添加消息映射。在 `BEGIN_MESSAGE_MAP` 和 `END_MESSAGE_MAP` 之间加入：

```
ON_MESSAGE(WM_SHOWTASK,onShowTask)将消息和消息响应函数映射起来。
```

然后就是在CPP文件中加入函数 `onShowTask` 的实现了：

```

1. LRESULT CTimeWakeDlg::onShowTask(WPARAM wParam,LPARAM lParam)
2. //wParam接收的是图标ID,而lParam接收的是鼠标的行为
3. {
4.     if(wParam!=IDR_MAINFRAME)
5.         return 1;
6.     switch(lParam)
7.     {
8.         case WM_RBUTTONDOWN://右键起来时弹出快捷菜单,这里只有一个“关闭”
9.         {
10.             LPPPOINT lpoint=new tagPOINT;
11.             ::GetCursorPos(lpoint);//得到鼠标位置
12.             CMenu menu;
13.             menu.CreatePopupMenu();//声明一个弹出式菜单
14.
15.
16.             //增加菜单项“关闭”,点击则发送消息WM_DESTROY给主窗口(已隐藏),将程序结束。
17.             menu.AppendMenu(MF_STRING,WM_DESTROY,"关闭");
18.             //确定弹出式菜单的位置
19.             menu.TrackPopupMenu(TPM_LEFTALIGN,lpoint->x,lpoint->y,this);
20.             //资源回收
21.             HMENU hmenu=menu.Detach();
22.             menu.DestroyMenu();
23.             delete lpoint;
24.         } break;
25.         case WM_LBUTTONDBLCLK://双击左键的处理
26.         {
27.             this->ShowWindow(SW_SHOW);//简单的显示主窗口完事儿
28.         } break;
29.     } return 0;
30. }

```

3. 调用

在窗口添加WM_SIZE的消息OnSize ()

```

1. void CTimeWakeDlg::OnSize(UINT nType, int cx, int cy)
2. {
3.     CDialog::OnSize(nType, cx, cy);
4.
5.
6.     // TODO: 在此处添加消息处理程序代码
7.     if(nType==SIZE_MINIMIZED) //判断是最小化按钮时,执行最小化到托盘函数
8.         ToTray();
9. }

```

注意：

定义消息名称以消息号,并注册消息,该步很重要!我就是因为没有注册消息,导致调试了很久都找不到问题所在。该步都是在Dlg.cpp(Dlg的实现中)中操作。定义消息名称和消息号:#define WM_SHOWTASK (WM_USER+1001),1001只是用于指定一个消息号,可以随便指定。注册则是在BEGIN_MESSAGE_MAP(Dlg,CDialog)和END_MESSAGE_MAP()之间添加ON_MESSAGE(WM_SHOWTASK, onShowTask)。

文章标签：MFC 窗口 最小化 托盘

个人分类：纯编程

此PDF由spygg生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com