[**关于如何记笔记**](onenotes/导航.one)

**在线学术交流系统**

**固定窗口大小** CMainFrame::PreCreateWindow(){

//这是错误的！ 创建向导时选择了”MFC标准”，这就可以，而我不是，从vs2012开始，派生自CFrameWndEx的MainFrame都会记住上次打开时的窗口状态，这些都写进注册表了！而你的代码并没有改变注册表！所以在PreCreateWindow中只是改了style，在InitInstance中加了下面这句：

m\_pMainWnd->SetWindowPos(nullptr, -1, -1, 640, 480, SWP\_NOMOVE | SWP\_NOACTIVATE | SWP\_NOZORDER);

}

**将一个发送对话框停靠在底部**

CDockablePane，显示对话框

**为发送对话框添加控件的响应函数**

颜色对话框、发送、。。。

问题是：数据都存在CDoc中，在dialog的响应函数中拿不到CDoc或太麻烦，所以书中在CView中响应，但我在向导中创建之后并不管用(因为它是WM\_COMMAND的，参加[windows消息](#windows消息))，既然我用了对话框，暂时只能将CDoc传进来了。

**添加用户列表停靠栏**

CDockablePane [创建停靠栏](#CDockablePane——创建停靠栏)

class CUserList : public CListCtrl

[CImageList](#CImageList) m\_imgList;

OnCreate() create imagelist & setimagelist

**点击菜单，连接服务器**

登录对话框

**系统托盘的创建**

[**系统托盘**](#系统托盘)

**设置MainFrame标题**

cs.style &= ~FWS\_ADDTOTITLE;

cs.lpszName = \_T("ChatClient");

[**状态栏信息**](#状态栏)

**创建服务器监听套接字类**

**CDockablePane**

停靠栏，可以：

挂到main frame上；自动隐藏；挂到tabctrl上；浮动工具条；停靠栏中的停靠栏；加载和保存状态(注册表)；等等。

常用操作：

[创建停靠栏](#CDockablePane——创建停靠栏)

[去掉自动隐藏](#CDockablePane——去掉自动隐藏)

[去掉关闭，不可拖动](#CDockablePane——去掉关闭，不可拖动)

[显示对话框](#CDockablePane——显示对话框)

[添加工具栏](#CDockablePane——添加工具栏)

[停靠多个pane](#CDockablePane——停靠多个pane)

[含CTabCtrl的停靠栏](#CDockablePane——含CTabCtrl的停靠栏)

[属性停靠栏](#CDockablePane——属性停靠栏)

[问题记录](#CDockablePane——问题记录)

**­创建停靠栏：**

MainFrm.h 中 #include "FileView.h" #include "ClassView.h"

protected: // control bar embedded members

CFileView m\_wndFileView;

CClassView m\_wndClassView; //class CClassView : public CDockablePane

CMainFrame::OnCreate() {

if (!m\_wndClassView.Create(strClassView, this, CRect(0, 0, 200, 200), TRUE, ID\_VIEW\_CLASSVIEW, // ID\_VIEW\_CLASSVIEW stringtable中添加即可

WS\_CHILD | WS\_VISIBLE | WS\_CLIPSIBLINGS | WS\_CLIPCHILDREN | CBRS\_LEFT | CBRS\_FLOAT\_MULTI))

}

调用OnCreate，ClassView.h 中：#include "ViewTree.h"

成员：CViewTree m\_wndClassView; class CViewTree : public CTreeCtrl

CClassView::OnCreate(){

const DWORD dwViewStyle = WS\_CHILD | WS\_VISIBLE | TVS\_HASLINES | TVS\_LINESATROOT | TVS\_HASBUTTONS | WS\_CLIPSIBLINGS | WS\_CLIPCHILDREN;

if (!m\_wndClassView.Create(dwViewStyle, rectDummy, this, 2))

}

**去掉自动隐藏**

注释 CMainFrame::OnCreate //EnableAutoHidePanes(CBRS\_ALIGN\_ANY);

**去掉关闭，不可拖动**

virtual BOOL FloatPane(CRect rectFloat, AFX\_DOCK\_METHOD dockMethod = DM\_UNKNOWN, bool bShow = true) { return FALSE; }

virtual BOOL CanFloat() const { return FALSE; }

virtual BOOL CanBeClosed() const { return FALSE; }

**显示对话框**

**OnCreate**

m\_dlg.Create(IDD\_DIALOG1, this);

m\_dlg.ShowWindow(SW\_SHOW);

**OnSize**

if (m\_dlg.GetSafeHwnd()) {

m\_dlg.SetWindowPos(nullptr, -1, -1, cx, cy, SWP\_NOMOVE | SWP\_NOACTIVATE | SWP\_NOZORDER); SWP\_NOMOVE 忽略x，y；SWP\_NOZORDER忽略 this；

m\_dlg.ShowWindow(SW\_SHOW);

}

[设置对话框背景](#对话框——设置对话框背景)

**添加工具栏**

class CTestDockToolBar : public CMFCToolBar {

virtual void OnUpdateCmdUI(CFrameWnd\* /\*pTarget\*/, BOOL bDisableIfNoHndler)

{

CMFCToolBar::OnUpdateCmdUI((CFrameWnd\*)GetOwner(), bDisableIfNoHndler);

}

};

protected:CTestDockToolBar m\_toolbar;

OnCreate()

m\_toolbar.Create(this, AFX\_DEFAULT\_TOOLBAR\_STYLE, IDR\_TOOLBAR\_TESTDOCK);

m\_toolbar.LoadToolBar(IDR\_TOOLBAR\_TESTDOCK, 0, 0, TRUE);

//m\_toolbar.SetPaneStyle(m\_toolbar.GetPaneStyle() | CBRS\_TOOLTIPS | CBRS\_FLYBY);

//m\_toolbar.SetPaneStyle(m\_toolbar.GetPaneStyle() & ~(CBRS\_GRIPPER | CBRS\_SIZE\_DYNAMIC | CBRS\_BORDER\_TOP | CBRS\_BORDER\_BOTTOM | CBRS\_BORDER\_LEFT | CBRS\_BORDER\_RIGHT));

m\_toolbar.SetOwner(this);

// FALSE：在owner处处理消息，TRUE：父框架处理消息

m\_toolbar.SetRouteCommandsViaFrame(FALSE);

**在OnSize中：**

int cyTlb = m\_wndToolBar.CalcFixedLayout(FALSE, TRUE).cy;

m\_wndToolBar.SetWindowPos(NULL, rectClient.left, rectClient.top, rectClient.Width(), cyTlb, SWP\_NOACTIVATE | SWP\_NOZORDER);

**工具栏的消息响应就在pane类中处理，在向导中通过ID添加。**

[**下拉工具栏制作**](#工具栏——下拉工具栏制作)

**停靠多个pane**

添加另一个pane

CMainFrame::OnCreate：

CDockablePane\* pTabbedBar = NULL;

m\_dock2.AttachToTabWnd(&m\_testDock, DM\_SHOW, FALSE, &pTabbedBar);

**含CTabCtrl的停靠栏**

class COutputWnd : public CDockablePane

CMFCTabCtrl m\_wndTabs;

COutputList m\_wndOutputBuild;

COutputList m\_wndOutputDebug;

class COutputList : public CListBox //为什么经常要继承，因为可能有自己的数据和方法

{public:

COutputList();

public:

virtual ~COutputList();

protected:

afx\_msg void OnContextMenu(CWnd\* pWnd, CPoint point);

afx\_msg void OnEditCopy();

afx\_msg void OnEditClear();

afx\_msg void OnViewOutput();

DECLARE\_MESSAGE\_MAP()

};COutputList m\_wndOutputFind;

afx\_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);

afx\_msg void OnSize(UINT nType, int cx, int cy);

**OnCreate**

CRect rectDummy;

rectDummy.SetRectEmpty();

// 创建选项卡窗口:

if (!m\_wndTabs.Create(CMFCTabCtrl::STYLE\_FLAT, rectDummy, this, 1)) //这里的ID范围内唯一即可，而且CRect为空

// 创建输出窗格:

const DWORD dwStyle = LBS\_NOINTEGRALHEIGHT | WS\_CHILD | WS\_VISIBLE | WS\_HSCROLL | WS\_VSCROLL;

if (!m\_wndOutputBuild.Create(dwStyle, rectDummy, &m\_wndTabs, 2) ||

!m\_wndOutputDebug.Create(dwStyle, rectDummy, &m\_wndTabs, 3) ||

!m\_wndOutputFind.Create(dwStyle, rectDummy, &m\_wndTabs, 4))

// 将列表窗口附加到选项卡:

m\_wndTabs.AddTab(&m\_wndOutputBuild, strTabName, (UINT)0);

**。。。**

**OnSize**

m\_wndTabs.SetWindowPos (NULL, -1, -1, cx, cy, SWP\_NOMOVE | SWP\_NOACTIVATE | SWP\_NOZORDER); //只要改变CTabCtrl即可

**属性停靠栏**

CMFCPropertyGridCtrl 参加PropertiesWnd.h

**问题记录**

点击dockablepane的叉后，下次启动就不显示了，因为写入注册表了。

拖动崩溃，rebuild一下。

拖动残影，因为只是dockpane，还没放置东西进去。

下拉工具栏图标如何关联？

**CTreeView**

class CTreeView : public CCtrlView

内含CTreeCtrl，通过GetTreeCtrl()获取

使用：

1. 向导创建class CUserTreeView : public CTreeView
2. 在这里面对[CTreeCtrl](#CTreeCtrl)操作

备注：为什么要attach在CView中呢？因为mfc是mainframe-view-doc架构的，它们可以互相配合！

**CTreeCtrl**

class CTreeCtrl : public CWnd

方法参加《帮助文档》

**常用操作：**

[添加节点](#CTreeCtr——添加节点)

[给节点显示图标](#CTreeCtr——给节点显示图标)

[修改控件风格](#CTreeCtrl——修改风格)

插入标记

[展开/收缩节点](#CTreeCtrl——展开收缩节点)

[遍历](#CTreeCtrl——遍历)

[鼠标在CTreeCtrl的位置](#CTreeCtrl——HitTest)

[消息相应](#CTreeCtrl——消息响应)

[改变标签文本](#CTreeCtrl——改变标签文本)

动态提供节点文本

[**checkbox处理**](#CTreeCtrl——checkbox处理)

[问题记录](#CTreeCtrl——问题记录)

**风格**：在VS中输入TVS\_HASBUTTONS，F12即可看到所有

#define TVS\_HASBUTTONS 0x0001 父项有展开合拢按钮

#define TVS\_HASLINES 0x0002 子项之间存在连线

#define TVS\_LINESATROOT 0x0004 根项之间存在连线

#define TVS\_EDITLABELS 0x0008 鼠标单击可修改数项名称

#define TVS\_DISABLEDRAGDROP 0x0010 禁止树控件发送TVN\_BEGINDRAG消息

#define TVS\_SHOWSELALWAYS 0x0020 选中项即使在窗口失去[输入焦点](http://baike.baidu.com/view/2833479.htm)时仍然保持选中状态

#define TVS\_RTLREADING 0x0040 从右往左显示文本

#define TVS\_NOTOOLTIPS 0x0080 无提示

#define TVS\_CHECKBOXES 0x0100 [在每个项前显示checkbox](#TVS_CHECKBOXES)

#define TVS\_TRACKSELECT 0x0200 允许跟踪

#define TVS\_SINGLEEXPAND 0x0400 只许一个项展开，按ctrl单击，未选中的不会收起

#define TVS\_INFOTIP 0x0800 通过发送TVN\_GETINFOTIP得到功能提示信息

#define TVS\_FULLROWSELECT 0x1000 选取整行，不能与TVS\_HASLINES并存

#define TVS\_NOSCROLL 0x2000 无垂直滚动条

#define TVS\_NONEVENHEIGHT 0x4000 项之间不等距，可用TVM\_SETITEMHEIGHT设置高度

#define TVS\_NOHSCROLL 0x8000 无水平滚动条

**checkbox处理**

这种风格一旦创建，将不能移除。只能destroy后再create一个新的。

参加《帮助文档：Working with state image indexes》

当点击checkbox时，引发NM\_CLICK并传递TVHT\_ONITEMSTATEICON。

注意事项：

GetCheck()返回原始状态； 要使用SelectItem()改变当前项的选择状态；

代码：

void CTreeCtrlTestDlg::OnClickTree1(NMHDR \*pNMHDR, LRESULT \*pResult)

{

// TODO: Add your control notification handler code here

CPoint pt;

UINT uF = 0;

GetCursorPos(&pt);

m\_tree.ScreenToClient(&pt);

HTREEITEM hItem = m\_tree.HitTest(pt, &uF);

m\_tree.SelectItem(hItem); //设置选中

if (TVHT\_NOWHERE & uF)

return;

if (TVHT\_ONITEMSTATEICON & uF) //如果勾选checkbox

{

BOOL bChk = !m\_tree.GetCheck(hItem); //当前状态

//有子项，全勾上

if (m\_tree.ItemHasChildren(hItem))

{

HTREEITEM hChildItem = m\_tree.GetChildItem(hItem);

while (hChildItem != NULL)

{

m\_tree.SetCheck(hChildItem, bChk);

hChildItem = m\_tree.GetNextItem(hChildItem, TVGN\_NEXT);

}

}

//兄弟项处理

BOOL bFalse;

bFalse = 0;

if (!m\_tree.ItemHasChildren(hItem))//没有子项

{

HTREEITEM hParentItem = m\_tree.GetParentItem(hItem);

if (bChk)

m\_tree.SetCheck(hParentItem, bChk); //父项勾上

if (!bChk)

{

HTREEITEM hChildItem = m\_tree.GetChildItem(hItem);

while (hChildItem != NULL && bFalse == 0)

{

if (hChildItem != hItem)

bFalse = m\_tree.GetCheck(hChildItem);

hChildItem = m\_tree.GetNextItem(hChildItem, TVGN\_NEXT);

}

if (bFalse == 0)

m\_tree.SetCheck(hParentItem, bChk);

}

}

}

else {

//处理普通的单击事件

}

\*pResult = 0;

}

**添加节点**

每一个结点都有一个句柄

HTREEITEM item, sub;

item = m\_tree.InsertItem(\_T("Parent"), TVI\_ROOT); //返回创建的节点，TVI\_ROOT存在不可删除

sub = m\_tree.InsertItem(\_T("Sub1"), item);

sub = m\_tree.InsertItem(\_T("Sub2"), item, sub);

默认状态下：不展开，双击展开，可同时展开

修改控件风格

m\_tree.ModifyStyle(0, TVS\_CHECKBOXES);

**给节点显示图标**

添加[CImageList](#CImageList)成员，绑定图像，CTreeCtrl::SetImageList()

m\_tree.InserIitem(项名, 未选中图像，选中图像，父项，在此项后);

item = m\_tree.InsertItem(\_T("Parent1"),0,0, TVI\_ROOT);

sub = m\_tree.InsertItem(\_T("Sub1"), 1, 1, item);

**展开/收缩节点**

BOOL Expand(HTREEITEM, UINT);

|  |  |  |
| --- | --- | --- |
|  | TVE\_COLLAPSE | 收缩列表。 |
| · | TVE\_COLLAPSERESET | 收缩列表并删除子项。 |
| · | TVE\_EXPAND | 展开列表。 |
| · | TVE\_TOGGLE | 如果列表当前是展开的则收缩列表；反之则展开列表。 |

m\_tree.Expand(item, TVE\_TOGGLE);

**遍历**

GetRootItem() GetChildItem() GetPrevSiblingItem/GetNextSiblingItem GetParentItem()

**鼠标在CTreeCtrl的位置**

HTREEITEM HitTest(CPoint pt, UINT\* pFlag);

pt为要测试的客户坐标，如果pt没有落在item上，返回NULL。

pFlag：详见《帮助文档：HitTest》，如TVHT\_ABOVE，点在客户区上方。

CPoint pt;

UINT flag;

GetCursorPos(&pt);

m\_tree.ScreenToClient(&pt);

HTREEITEM h = m\_tree.HitTest(pt, &flag);

if(h){//... }

**CTreeCtrl消息相应**

[windows消息](#windows消息)

向导搜索控件ID，添加消息处理，可看到增加了：

ON\_NOTIFY(NM\_RCLICK, IDC\_TREE1, &CTreeCtrlTestDlg::OnRclickTree1)

OnRclickTree1(NMHDR \*pNMHDR, LRESULT \*pResult)

[NMHDR](#NMHDR)

一般将pNMHDR 转成 NMTREEVIEW\* ，也有NMTVDISPINFO\*等

**改变标签文本**

添加TVN\_ENDLABELEDIT消息响应，添加代码：

if (pTVDispInfo->item.pszText == nullptr) {

return; //用户放弃

}

m\_tree.SetItemText(pTVDispInfo->item.hItem, pTVDispInfo->item.pszText);

}

问题记录

1. 按回车，dialog退出。

原因：当用户按下Enter键时，Windows就会自动去找输入焦点落在了哪一个按钮上。如果对话框没有默认按钮，那么即使对话框中没有OK按 钮，OnOK函数也会自动被调用。

解决：让OnOk为空，或者重写。或者添加NM\_RETURN响应（控件获得焦点时，按Enter）

2，如果你在dialog或者view中使用，则CTreeCtrl会自动创建，不能再用Create。在CDockablePane中可以。

**CImageList**

class CImageList : public CObject

相同大小图像集合，索引从0开始。

配合控件（CListCtrl，CTreeCtrl等），SetImageList(&CImageList, LVSIL\_NORMAL)方法。

LVSIL\_NORMAL 大图标

LVSIL\_SMALL 小图标

LVSIL\_STATE 状态图标

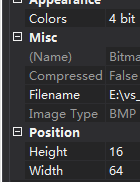
绑定到对象

BOOL Create(资源ID，宽pix，nGrow可增加的图像数量，COLORREF 图像类型)

图像类型：ILC\_COLOR、ILC\_COLOR4、ILC\_COLOR8、ILC\_COLOR16、ILC\_COLOR24、ILC\_COLOR32、 ILC\_COLORDDB和ILC\_MASK & RGB(r,g,b)

m\_images.Create(IDB\_BITMAP1, 16, 0, ILC\_COLOR4);

m\_tree.SetImageList(&m\_images, LVSIL\_NORMAL);

**windows消息**

3种：窗口消息、命令消息和控件通知消息。

窗口消息：WM\_PAINT, WM\_CREATE。。。

命令消息：WM\_COMMAND

控件通知消息：一个窗口内的子控件发生了一些事情，需要通知父窗口。

消息路由：因为MFC程序中有CMyView,CMyDoc,CMyFrameWnd,CMyApp等，MFC框架要做的工作是给用户提供一个机会，让用户可以选择这些类当中的任意一个来处理我们的命令消息。

消息路由主要是针对WM\_COMMAND的。

创建自定义消息响应：

1）#define WM\_XX WM\_USER + n

2) afx\_msg LRESULT MyHandle(WPARAM w, LPARAM l);

3) ON\_MESSAGE(WM\_XX, MyHandle)

4) LRESULT CMainFrame::MyHandle(...){}

**NMHDR**

win32引进的将控件通知消息统一，并可附带附加消息的结构体。

struct NMHDR{

HWnd hWndFrom  相当于原WM\_COMMAND传递方式的lParam

UINT idFrom  相当于原WM\_COMMAND传递方式的wParam（low-order）

UINT code  相当于原WM\_COMMAND传递方式的Notify Code(wParam"s high-order) }；

}

对于CTreeCtrl通知消息，有几种附加消息的结构，如：

typedef struct \_NM\_TREEVIEW

 {

NMHDR hdr; //通知消息句柄 ，这就像继承！

UINT action; //通知消息标志

TV\_ITEM itemOld; //原来的数据结构

TV\_ITEM itemNew; //新的数据结构 ，这就是附加数据！

POINT ptDrag; //拖动指针

} NM\_TREEVIEW;

typedef struct \_TV\_ITEM {

UINT mask; //结构成员有效性屏蔽位

HTREEITEM hItem; //数据项控制句柄

UINT state; //数据项状态

UINT stateMask; //状态有效性屏蔽位

 LPSTR pszText; //数据项名称字符串

 int cchTextMax; //数据项名称的最大长度

 int iImage; //数据项图标索引号

int iSelectedImage;//选中数据项图标索引号

int cChildren; //子项标识

LPARAM lParam; //程序定义的32位数据

 } TV\_ITEM, FAR \*LPTV\_ITEM;

对话框

常用操作：

[设置对话框背景](#对话框——设置对话框背景)

**设置对话框背景**

void CTestDockPaneDlg::OnPaint()

{

CPaintDC dc(this);

CRect rc;

GetClientRect(rc);

dc.FillSolidRect(rc, RGB(255, 255, 255));

dc.FillPath();

}

**工具栏**

常用操作

[下拉工具栏制作](#工具栏——下拉工具栏制作)

**下拉工具栏制作**

添加资源 IDR\_POPUP\_SORT 一个弹出菜单

TestDock.cpp中

class CTestDockMenuButton : public CMFCToolBarMenuButton

{

public:

CTestDockMenuButton(HMENU hMenu = NULL) : CMFCToolBarMenuButton((UINT)-1, hMenu, -1) {}

};

OnCreate中

CMenu menuSort;

menuSort.LoadMenu(IDR\_MENU\_TEST\_POPUP);

m\_toolbar.ReplaceButton(ID\_BUTTON32772,//替换的按钮

CTestDockMenuButton(menuSort.GetSubMenu(0)->GetSafeHmenu()));

CTestDockMenuButton\* pButton = DYNAMIC\_DOWNCAST(CTestDockMenuButton, m\_toolbar.GetButton(1));

if (pButton) {

pButton->m\_bText = FALSE;

pButton->m\_bImage = TRUE;

pButton->SetImage(GetCmdMgr()->GetCmdImage(m\_nCurrSort));

pButton->SetMessageWnd(this);

}

**CListCtrl**

**风格**

LVS\_ICON: 为每个item显示大图标  
      LVS\_SMALLICON: 为每个item显示小图标  
      LVS\_LIST: 显示一列带有小图标的item  
      LVS\_REPORT: 显示item详细资料

扩展风格： **LVS\_EX\_CHECKBOXES** 、**LVS\_EX\_FULLROWSELECT**、**LVS\_EX\_GRIDLINES**等

**不允许重绘列表控件**

ctl.SetRedraw(FALSE);

数据添加完后，ctl.SetRedraw(TRUE);

**插入表头**

ctl.InsertColumn(i, title[i], LVCFMT\_LEFT, width);

**插入内容**

ctl.InsertItem(i, 内容); //第i行

ctl.SetItemText(i, j, 内容); //第i行的第j列，0就是上面那条

**系统托盘**

BOOL Shell\_NotifyIcon( DWORD dwMessage,PNOTIFYICONDATA lpdata);

dwMessage：

**NIM\_ADD**向托盘区域添加一个图标。

**NIM\_DELETE NIM\_MODIFY**

typedef struct \_NOTIFYICONDATA {

DWORD cbSize;

HWND hWnd;

UINT uID; 图标的标识符

UINT uFlags; 哪些成员起作用

UINT uCallbackMessage; **NIF\_MESSAGE**

HICON hIcon; **NIF\_ICON**

TCHAR szTip[64]; **NIF\_TIP**

DWORD dwState; **NIF\_STATE**

DWORD dwStateMask; **NIF\_STATE**

TCHAR szInfo[256]; **NIF\_INFO** 使用气球提示代替普通的工具提示框。

union {

UINT uTimeout;

UINT uVersion;

};

TCHAR szInfoTitle[64];

DWORD dwInfoFlags;

GUID guidItem;

} NOTIFYICONDATA, \*PNOTIFYICONDATA;

mainframe.h #define WM\_SYSTRAY WM\_USER + 1001

protected: NOTIFYICONDATA pnid;

OnCreate() :

//创建系统托盘

pnid.cbSize = sizeof(NOTIFYICONDATA);

pnid.hIcon = LoadIcon(AfxGetApp()->m\_hInstance,

MAKEINTRESOURCE(IDR\_MAINFRAME));

pnid.hWnd = m\_hWnd;

pnid.uCallbackMessage = WM\_SYSTRAY;

pnid.uFlags = NIF\_ICON | NIF\_MESSAGE;

pnid.uID = IDR\_MAINFRAME;

Shell\_NotifyIcon(NIM\_ADD, &pnid);

至此，托盘可正常显示

托盘消息响应：

[创建自定义消息](#windows消息)

//wParam接收的是图标的ID，lParam接收的是鼠标的行为

switch(lp){

case WM\_RBUTTONUP:

case WM\_LBUTTONDBLCLK:

ShowWindow(SW\_SHOW); //一开始并没有用，因为你还没有重载WindowProc

}

重载

LRESULT CMainFrame::WindowProc(UINT message, WPARAM wParam, LPARAM lParam)

{

LRESULT result = CFrameWndEx::WindowProc(message, wParam, lParam);

if (SC\_MINIMIZE == wParam) {

ShowWindow(SW\_HIDE);

}

return result;

}

问题：虽然最小化任务栏上是没有了，但是双击托盘它只是在任务栏上显示，你还要点一下图标。

**状态栏**

**在string table资源中添加ID，如**ID\_CONNECTIONS , caption”online cnt:00”

在MainFrm.cpp中，

static UINT indicators[] =

{

ID\_SEPARATOR, // status line indicator

ID\_CONNECTIONS,

ID\_MESSAGES,

}



**在向导中添加对**ID\_CONNECTIONS的响应，如void CXXDoc::OnUpdateConnections(CCmdUI\*)