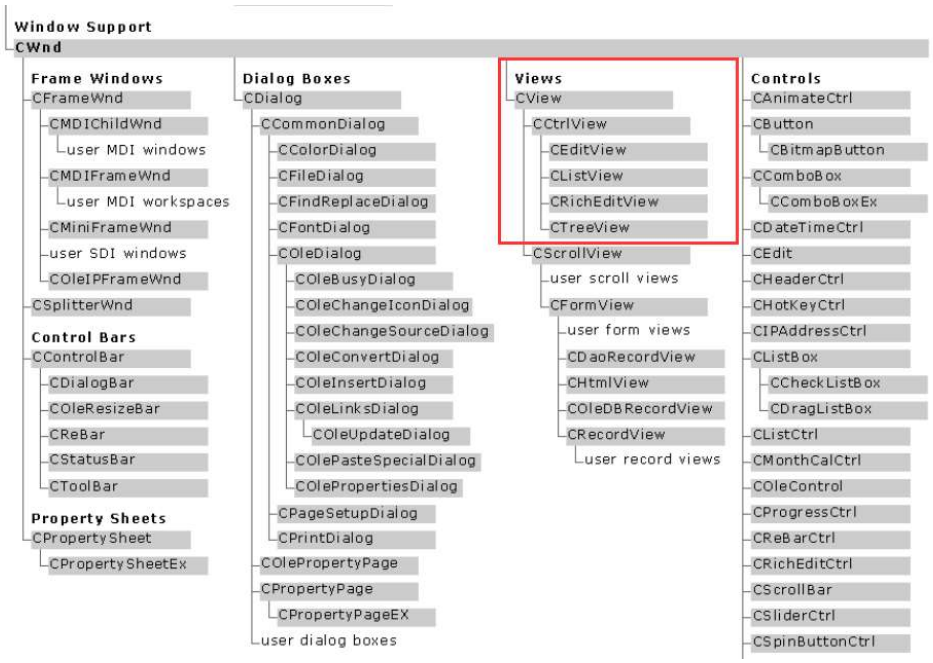
2020/07/14_MFC_第12课_CAD07_试图分割_工具栏_状态栏

笔记本: MFC
创建时间: 2020/7/14 星期二 10:35
作者: ileemi
标签: MFC单文档试图分割, MFC状态栏的使用, 工具栏, 消息反射的几种方法

单文档和多文档

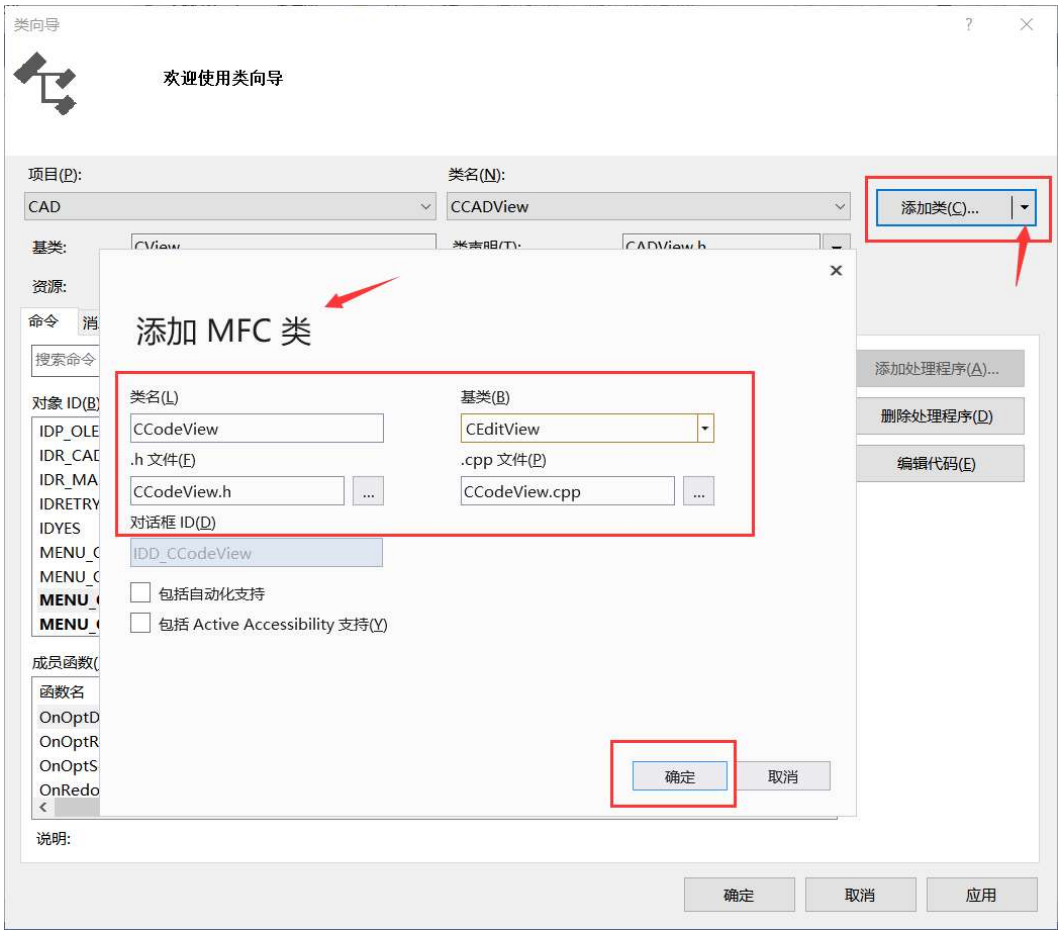
单文档 -- 单试图
有一个 CDocument --> CDoc

多文档 -- 多视图
有多个 CDocument --> CDoc



将MFC默认视图改为Edit

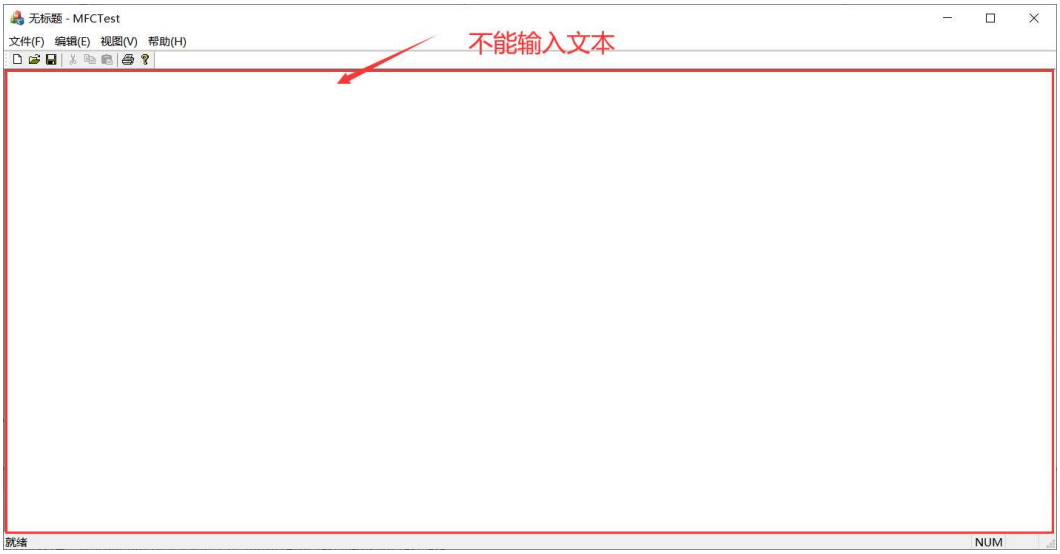
MFC 类的创建在类向导中创建：



MFC 单文档中新建的 MFC类（CCodeView）可以在 "...App" 类的 "InitInstance" 方法中进行切换试图：

```
101     // 注册应用程序的文档模板。  文档模板
102     // 将用作文档、框架窗口和视图之间的连接
103     CSingleDocTemplate* pDocTemplate;
104     pDocTemplate = new CSingleDocTemplate(
105         IDR_MAINFRAME,
106         RUNTIME_CLASS(CMFCTestDoc),
107         RUNTIME_CLASS(CMainFrame),       // 主 SDI 框架窗口
108         RUNTIME_CLASS(CMFCTestView));
109     if (!pDocTemplate)
110         return FALSE;
111     AddDocTemplate(pDocTemplate);
112
113
114     // 分析标准 shell 命令、DDE、打开文件操作的命令行
115     CCommandLineInfo cmdInfo;
116     ParseCommandLine(cmdInfo);
117
```

原来为CAD绘图：

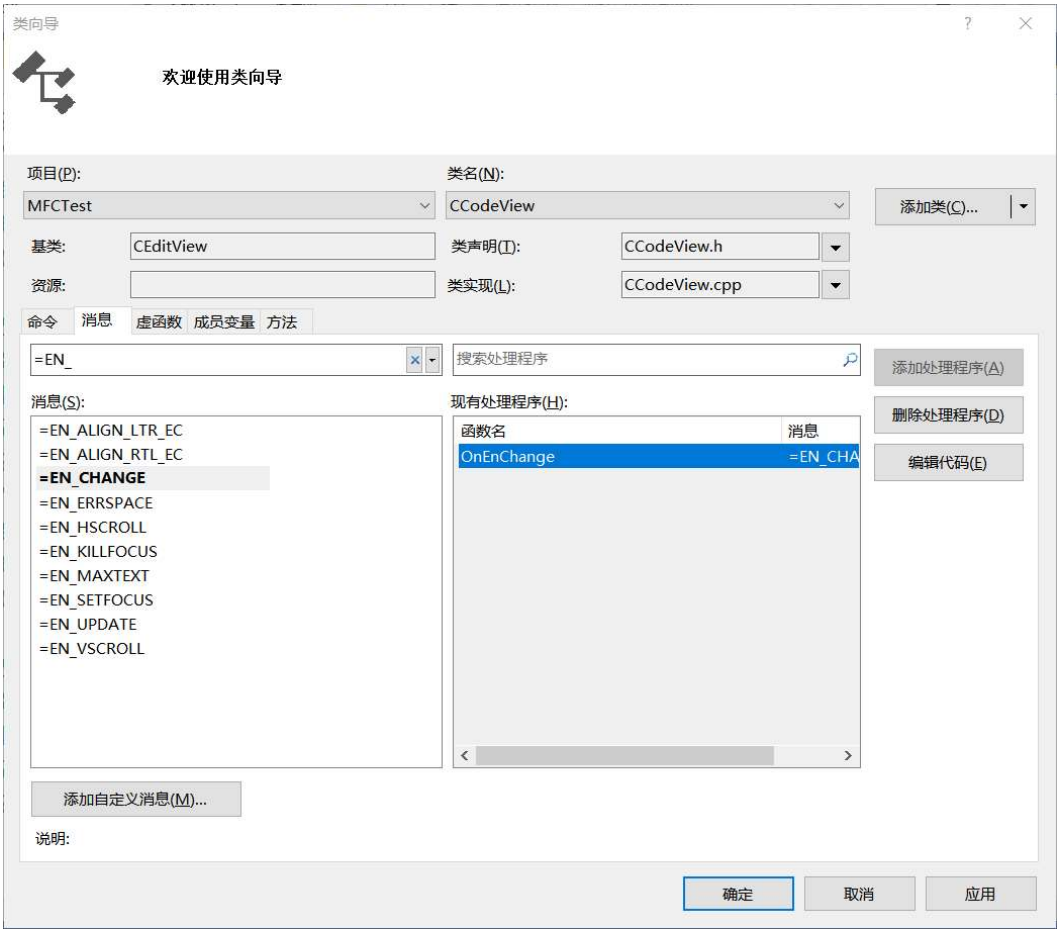


做如下更改后，操作如下：

```
100
101     // 注册应用程序的文档模板。  文档模板
102     // 将用作文档、框架窗口和视图之间的连接
103     CSingleDocTemplate* pDocTemplate;
104     pDocTemplate = new CSingleDocTemplate(
105         IDR_MAINFRAME,
106         RUNTIME_CLASS(CMFCTestDoc),
107         RUNTIME_CLASS(CMainFrame),       // 主 SDI 框架窗口
108         RUNTIME_CLASS(CCodeView));
109     if (!pDocTemplate)
110         return FALSE;
111     AddDocTemplate(pDocTemplate);
112
113
114     // 分析标准 shell 命令、DDE、打开文件操作的命令行
115     CCommandLineInfo cmdInfo;
116     ParseCommandLine(cmdInfo);
117
```

更改为自定义的MFC类，该类继承CEditView

消息反射



子窗口 --> 消息 --> 父窗口
子窗口 <-- 消息 <-- 父窗口

反射由框架自动转发给子窗口，父窗口接收到消息发送给子窗口

反射消息的各种方法：
自定义消息

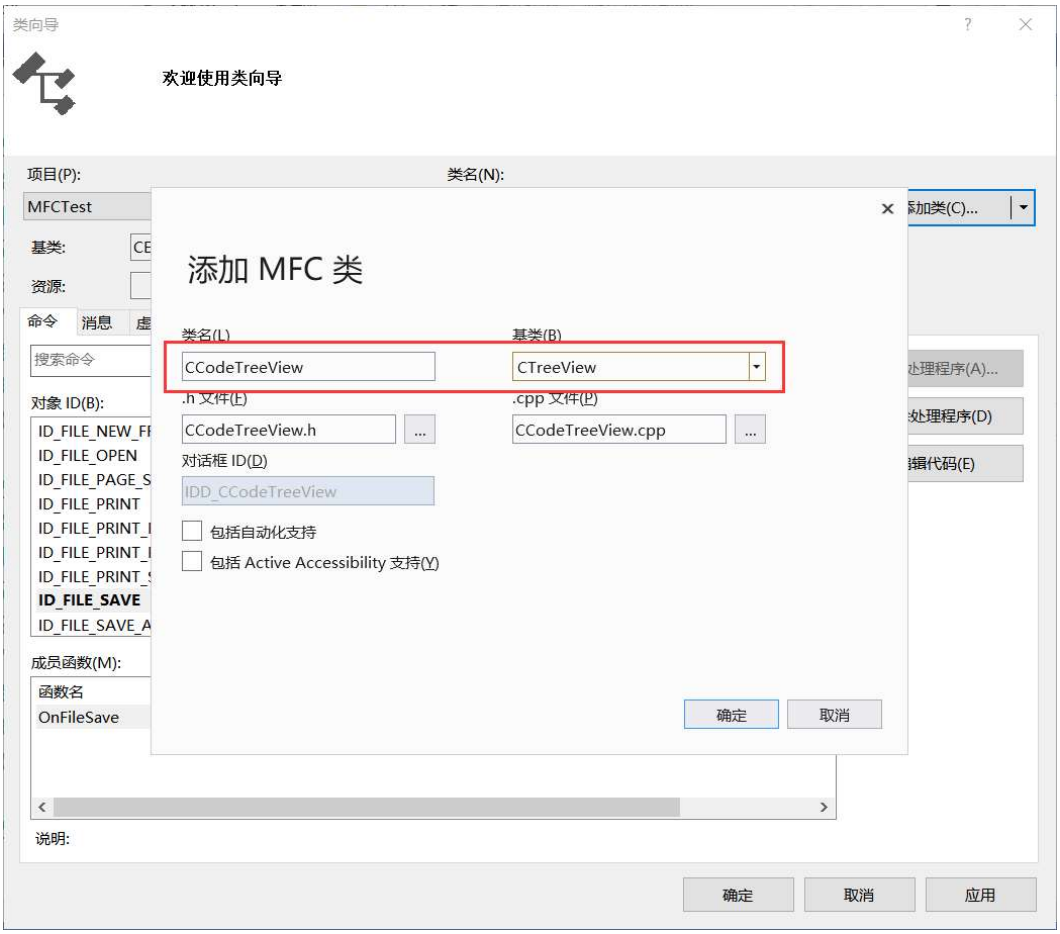
EN_ 开头的是获取类消息：
EN_CHANGE 等

EM_开头的是设置类消息：
EM_SETRECT 等

WM_COMMAND 消息存在的问题：
每一个窗口都有一个窗口过程函数，**WM_COMMAND** 消息不是发送给控件本身的，是先发送给父窗口。

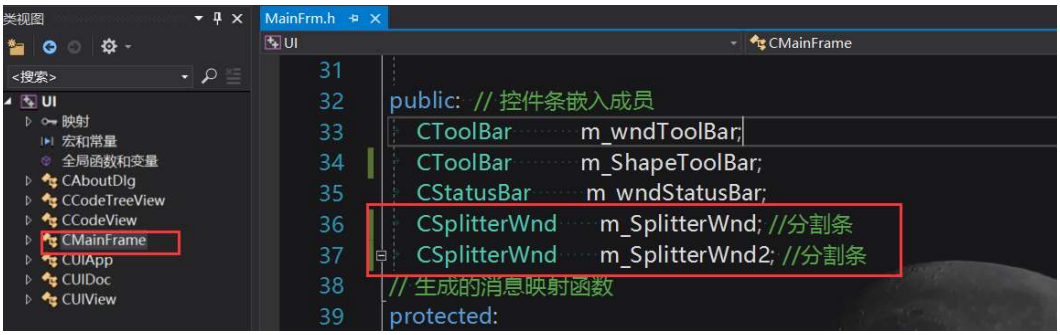
试图分割

创建一个继承 **CTreeView** 的类 "**CCodeTreeView**"：



视图分割也需要一个**分割的控件**或者**窗口**。MFC 已经封装了一个分割条的类 "**CSplitterWnd**"，可以将一个窗口拆分成多个视图。

定义对象：



再窗口创建之前进行分割，在 **CMainFrame** 的 **OnCreateClient**() 方法内进行（原因是需要使用 **OnCreateClient** 的 *CCreateContext pContext** 参数）。不在 **OnCreate** 中进行，是因为 **OnCreate** 缺少信息，不够早。

使用 **CSplitterWnd** 类的 **CreateStatic** 方法进行分割，分割后的每列应该说明其视图种类（使用 **CreateView** 来说明），使用过程代码如下（**注意返回值，返回值不能**

是基类，返回基类会造成视图分割失败）：

```
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT lpcs, CCreateContext* pContext)
{
// TODO: 在此添加专用代码和/或调用基类
//pContext->m_pCurrentDoc;

// 调用创建一个静态拆分器窗口并将其附加到CSplitterWnd对象
BOOL bCreateSpltr = m_SplitterWnd.CreateStatic(this, 1, 2);
//SIZE sz;

// 调用在拆分器窗口中创建窗格
m_SplitterWnd.CreateView(0, 0, RUNTIME_CLASS(CCodeTreeView), CSize(200, 50), pContext);
m_SplitterWnd.CreateView(0, 1, RUNTIME_CLASS(CCodeView), CSize(200, 50), pContext);

return bCreateSpltr;
```

MFC 中可以使用 **CSize** 直接给定长、宽。分割的视图各个视图响应自己的数据即可。文档数据是同一个文档数据。

分割窗口窗口间的间距：

- 窗口间上下的高度由 **CreateView** 的 **参数4** 决定（**CSize**）
- 窗口间左右的宽度由 **SetColumnInfo** 决定

为对话框分割的窗口设置初始化数值，分别在对应的类中的初始化消息函数（**OnInitialUpdate**）内设置。

初始化文本信息：

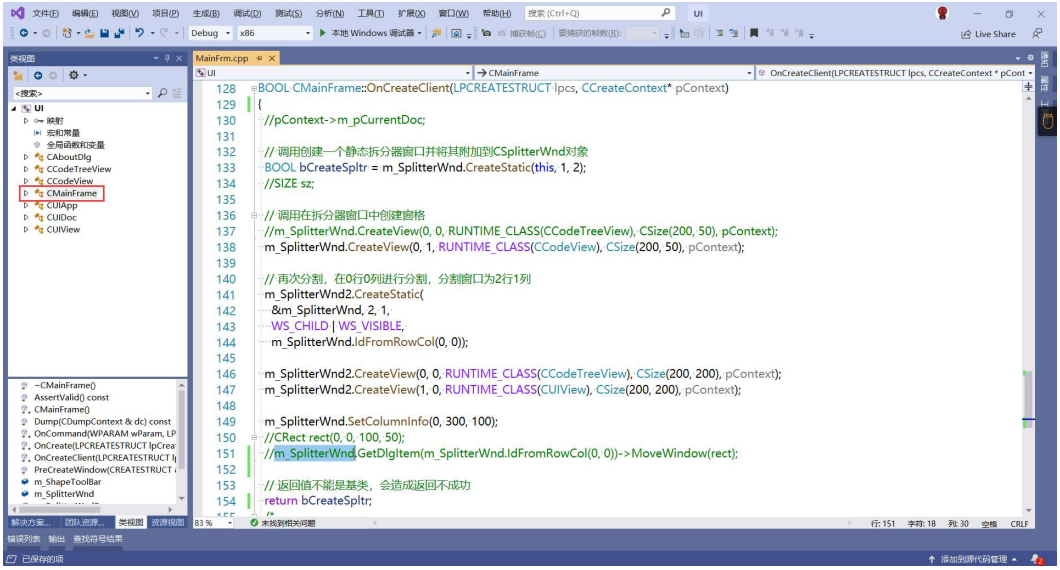
```
}

void CCodeView::OnInitialUpdate()
{
CEditView::OnInitialUpdate();

// 获取文档
CUIDoc* pDoc = (CUIDoc*)GetDocument();
// 这只初始文本数据到窗口
SetWindowText(pDoc->m_csCode);

//CEdit& edit = GetEditCtrl();
//SetWindowLong(edit.m_hWnd, GWL_STYLE, GetWindowLong(edit.m_hWnd, GWL_STYLE) | ES_READONLY);
//GetEditCtrl()
}
```

添加树控件：

在对用类中的OnInitialUpdate内：

```
void CCodeTreeView::OnInitialUpdate()
{
CTreeView::OnInitialUpdate();

// 获取文档
CUIDoc* pDoc = (CUIDoc*)GetDocument();

CTreeCtrl& tree = GetTreeCtrl();    // 检索与视图关联的树控件的引用

// 设置样式
SetWindowLong(tree.m_hWnd, GWL_STYLE, WS_VISIBLE | WS_TABSTOP | WS_CHILD | WS_BORDER
    | TVS_HASBUTTONS | TVS_LINESATROOT | TVS_HASLINES
    | TVS_DISABLEDRAGDROP);

// 插入结点
HTREEITEM hRoot = tree.InsertItem("CTest");
tree.InsertItem("CTest()", hRoot);
tree.InsertItem("~CTest()", hRoot);
tree.InsertItem("Fun1(int n1, int n2)", hRoot);
}
```

分割视图代码示例：



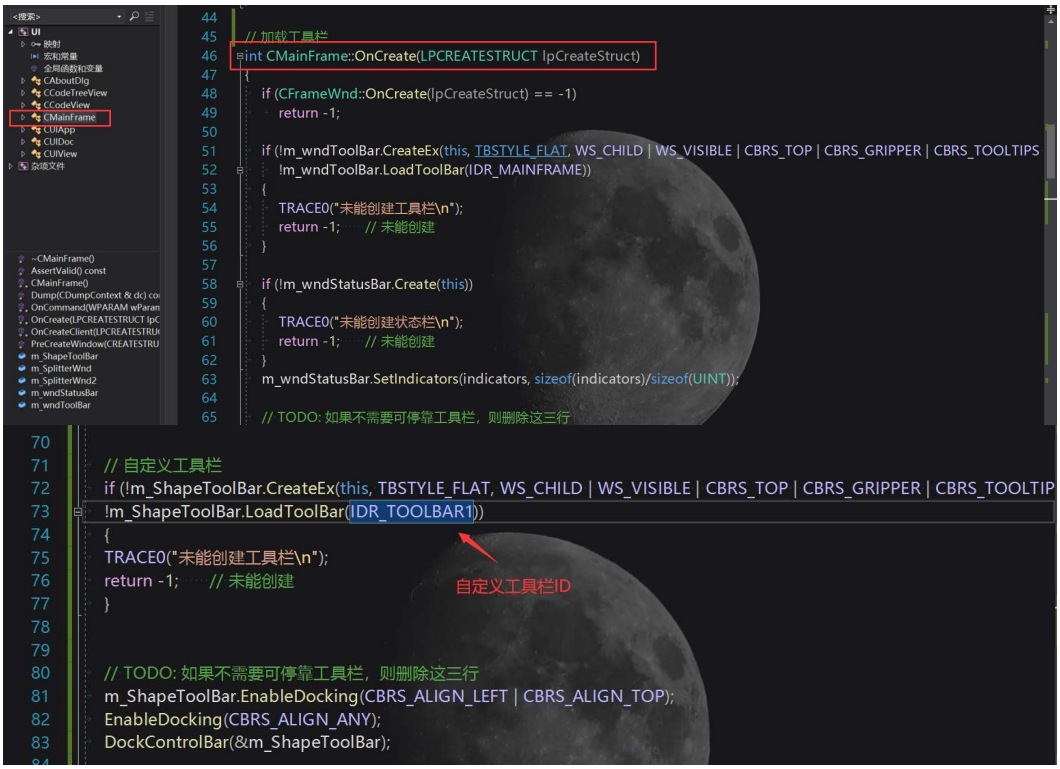
```
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT lpcs, CCreateContext* pContext)
{
	//pContext->m_pCurrentDoc;

	// 调用创建一个静态拆分器窗口并将其附加到CSplitterWnd对象
	BOOL bCreateSpltr = m_SplitterWnd.CreateStatic(this, 1, 2);
	//SIZE sz;

	// 调用在拆分器窗口中创建窗格
	//m_SplitterWnd.CreateView(0, 0, RUNTIME_CLASS(CCodeTreeView), CSize(200, 50), pContext);
	m_SplitterWnd.CreateView(0, 1, RUNTIME_CLASS(CCodeView), CSize(200, 50), pContext);

	// 再次分割，在0行0列进行分割，分割窗口为2行1列
	m_SplitterWnd2.CreateStatic(
		&m_SplitterWnd, 2, 1,
		WS_CHILD | WS_VISIBLE,
		m_SplitterWnd.IdFromRowCol(0, 0));

	m_SplitterWnd2.CreateView(0, 0, RUNTIME_CLASS(CCodeTreeView), CSize(200, 200), pContext);
	m_SplitterWnd2.CreateView(1, 0, RUNTIME_CLASS(CUIView), CSize(200, 200), pContext);

	m_SplitterWnd.SetColumnInfo(0, 300, 100);
	//CRect rect(0, 0, 100, 50);
	//m_SplitterWnd.GetDlgItem(m_SplitterWnd.IdFromRowCol(0, 0))->MoveWindow(rect);

	// 返回值不能是基类，会造成返回不成功
	return bCreateSpltr;
	/*
```

工具栏的创建和使用

对应的MFC类：

- **CToolBar**

在项目的资源中添加对应的 "**ToolBar**" 工具条的样式以及各种相关属性。工具栏的创建也是在主窗口进行的，"**CMainFrame**"，在 CMainFrame 类中的 "**OnCreate**" 方法中进行。同时还需要在 "**CMainFrame**"类中添加一个工具栏对应的成员。



```
// 加载工具栏
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
		return -1;

	if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE | CBRS_TOP | CBRS_GRIPPER | CBRS_TOOLTIPS
		!m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
	{
		TRACE0("未能创建工具栏\n");
		return -1;      // 未能创建
	}

	if (!m_wndStatusBar.Create(this))
	{
		TRACE0("未能创建状态栏\n");
		return -1;      // 未能创建
	}
	m_wndStatusBar.SetIndicators(indicators, sizeof(indicators)/sizeof(UINT));

	// TODO: 如果不需要可停靠工具栏，则删除这三行
```

```
	// 自定义工具栏
	if (!m_ShapeToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE | CBRS_TOP | CBRS_GRIPPER | CBRS_TOOLTIP
	!m_ShapeToolBar.LoadToolBar(IDR_TOOLBAR1))
	{
	TRACE0("未能创建工具栏\n");
	return -1;      // 未能创建
	}

	// TODO: 如果不需要可停靠工具栏，则删除这三行
	m_ShapeToolBar.EnableDocking(CBRS_ALIGN_LEFT | CBRS_ALIGN_TOP);
	EnableDocking(CBRS_ALIGN_ANY);
	DockControlBar(&m_ShapeToolBar);
```

自定义工具栏ID

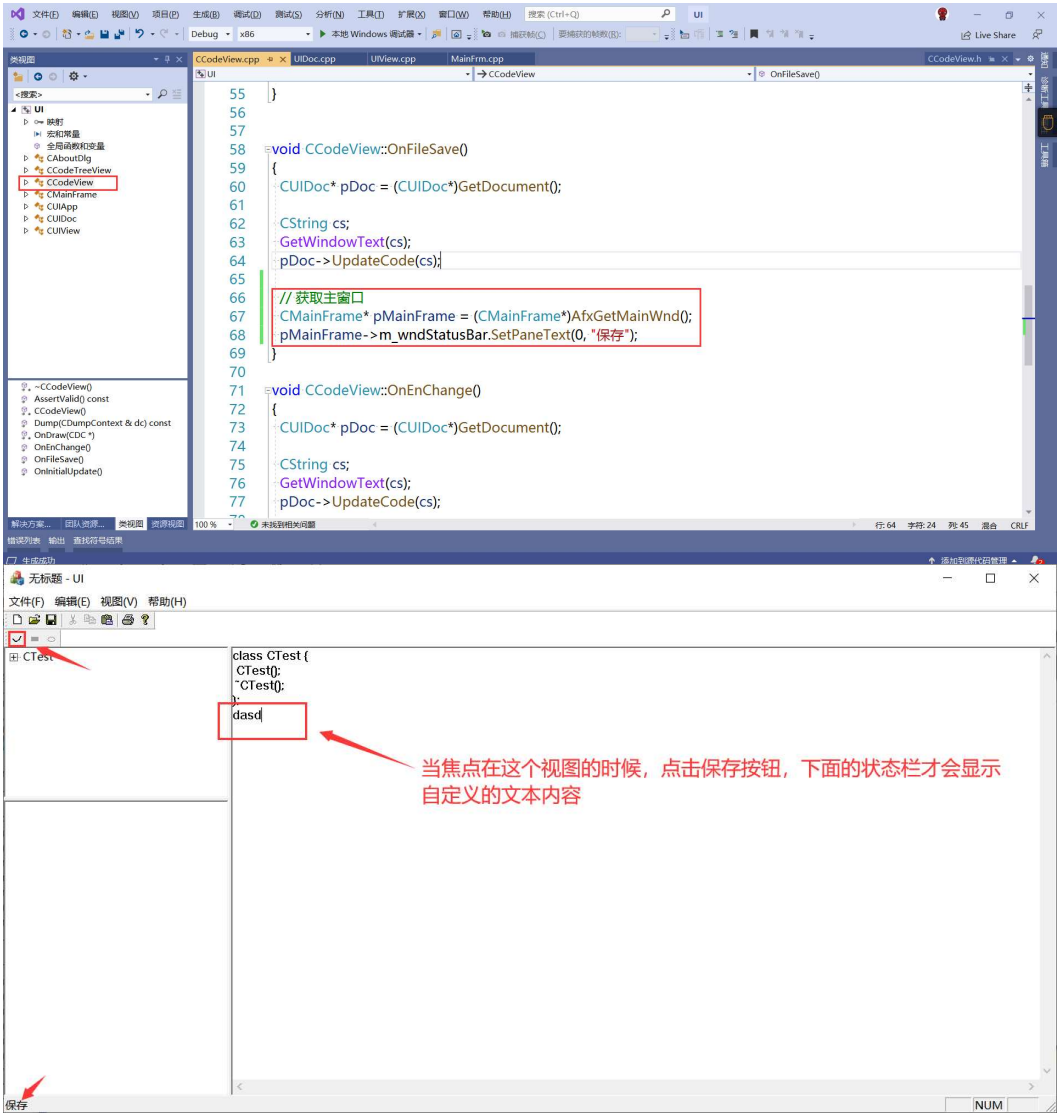
状态栏

对应的MFC类：

- **CStatusBar**

在 **CMainFrame** 类中的 "**OnCreate**" 方法中已经创建了，只需要使用对应的API进行对应的更改即可。
SetPaneText -- 设置给定索引的指示符文本。

注意窗口是有焦点的，鼠标点击对应的窗口，再点击状态栏才会响应对应的消息：



获取主窗口：

```
CMainFrame* pMainFrame = (CMainFrame*)AfxGetMainWnd();
pMainFrame.xxx
......
```