

鸡啄米

聚焦互联网、数码、软件开发和编程入门的IT休闲吧

[首页](#) [IT互联网](#) [数码生活](#) [软件开发](#) [职场人生](#) [娱乐休闲](#) [编程课堂](#) [安卓开发](#) [留言簿](#)

[首页](#) » [软件开发](#) » [MFC六大核心机制之五、六：消息映射和命令传递](#)

MFC六大核心机制之五、六：消息映射和命令传递

分类标签: [MFC](#) [VC++](#)

作为C++程序员，我们总是希望自己程序的所有代码都是自己写出来的，如果使用了其他的一些库，也总是千方百计想弄清楚其中的类和函数的原理，否则就会感觉不踏实。所以，我们对于在进行MFC视窗程序设计时经常要用到的消息机制也不满足于会使用，而是希望能理解个中道理。本文就为大家剖析MFC消息映射和命令传递的原理。

理解MFC消息机制的必要性

说到消息，在MFC中，“最熟悉的神秘”可以说是消息映射了，那是我们刚开始接触MFC时就要面对的东西。有过SDK编程经验的朋友转到MFC编程的时候，一下子觉得什么都变了样。特别是窗口消息及对消息的处理跟以前相比，更是风马牛不相及的。如文档不是窗口，是怎样响应命令消息的呢？

初次用MFC编程，我们只会用MFC ClassWizard为我们做大量的东西，最主要的是添加消息响应。记忆中，如果是自己添加消息响应，我们应何等的小心翼翼，对BEGIN_MESSAGE_MAP().....END_MESSAGE_MAP()更要奉若神灵。它就是一个魔盒子，把我们的咒语放入恰当的地方，就会发生神奇的力量，放错了，自己的程序就连“命”都没有。

据说，知道得太多未必是好事。我也曾经打算不去理解这神秘的区域，觉得编程的时候知道自己想做什么就行了。MFC外表上给我们提供了东西，直观地说，不但给了我一个程序的外壳，更给我们许多方便。微软的出发点可能是希望达到“傻瓜编程”的结果，试想，谁不会用ClassWizard？大家知道，Windows是基于消息的，有了ClassWizard，你又会添加类，又会添加消息，那么你学的东西似乎学到头了。于是许多程序员认为“我们没有必要走SDK的老路，直接用MFC编程，新的东西通常是简单、直观、易学.....”。

到你真正想用MFC编程的时候，你会发觉光会ClassWizard的你是多么的愚蠢。MFC不是一个普通的类库，普通的类库我们完全可以不理解里面的细节，只要知道这些类库能干什么，接口参数如何就万事大吉。如string类，操作顺序是定义一个string对象，然后修改属性，调用方法。但对于MFC，并不是在你的程序中写上一句“#include MFC.h”，然后就使用MFC类库的。

MFC是一块包着糖衣的牛骨头。你很轻松地写出一个单文档窗口，在窗口中间打印一句“I love MFC!”，然后，恶梦开始了.....想逃避，打算永远不去理解MFC内幕？门都没有！在MFC这个黑暗神秘的洞中，即使你打算摸着石头前行，也注定找不到出口。对着MFC这块牛骨头，微软温和、民主地告诉你“你当然可以选择不啃掉它，咳咳.....但你必然会因此而饿死！”

MFC消息机制与SDK的不同

消息映射与命令传递体现了MFC与SDK的不同。在SDK编程中，没有消息映射的概念，它有明确的回调函数，通过一个switch语句去判断收到了何种消息，然后对这个消息进行处理。所以，在SDK编程中，会发送消息和在回调函数中处理消息就差不多可以写SDK程序了。

在MFC中，看上去发送消息和处理消息比SDK更简单、直接，但可惜不直观。举个简单的例子，如果我们想自定义一个消息，SDK是非常简单直观的，用一条语句：SendMessage(hwnd,message/*一个大于或等于WM_USER的数字*/,wparam,lparam)，之后就可以在回调函数中处理了。但MFC就不同了，因为你通常不直接去改写窗口的回调函数，所以只能亦步亦趋对照原来的MFC代码，把消息放到恰当的地方。这确实是一样很痛苦的劳动。

要了解MFC消息映射原理并不是一件轻松的事情。我们可以逆向思维，想象一下消息映射为我们做了什么工作。MFC在自动化给我们提供了很大的方便，比如，所有的MFC窗口都使用同一窗口过程，即所有的MFC窗口都有一个默认的窗口过程。不像在SDK编程中，要为每个窗口类写一个窗口过程。

订阅鸡啄米

RSS

订阅到

分享到

分享到

一键分享

QQ空间

新浪微博

百度云收藏

人人网

腾讯微博

百度相册

开心网

腾讯朋友

百度贴吧

豆瓣网

搜狐微博

百度新首页

QQ好友

和讯微博

更多...

百度分享

站内搜索

请输入搜索内容..

Q

分类标签

编程入门 (135)
C++ (96)
VC++ (77)
MFC (67)
VS2010 (63)
程序员 (55)
Android (51)
Java (51)
苹果 (49)
智能手机 (47)
腾讯 (41)
百度 (37)
阿里巴巴 (33)
谷歌 (32)
平板电脑 (31)
TCP/IP (29)
iPhone (26)
PHP (26)
Javascript (25)
奇虎360 (24)
Mysql (24)
Windows (22)
软件架构 (20)
小米 (20)
设计模式 (19)
iPad (18)
Web (18)
职场攻略 (18)
三星 (16)
创业 (16)
微软 (13)
iOS (13)
微信 (13)
HTML (13)
应用程序 (12)

MFC消息映射原理

对于消息映射，最直截了当地猜想是：消息映射就是用一个数据结构把“消息”与“响应消息函数名”串联起来。这样，当窗口感知消息发生时，就对结构查找，找到相应的消息响应函数执行。其实这个想法也不能简单地实现：我们每个不同的MFC窗口类，对同一种消息，有不同的响应方式。即是说，对同一种消息，不同的MFC窗口会有不同的消息响应函数。

这时，大家又想了一个可行的方法。我们设计窗口基类（CWnd）时，我们让它对每种不同的消息都来一个消息响应，并把这个消息响应函数定义为**虚函数**。这样，从CWnd派生的窗口类对所有消息都有了一个空响应，我们要响应一个特定的消息就重载这个消息响应函数就可以了。但这样做的结果，一个几乎什么也不做的CWnd类要有几百个“多余”的函数，哪怕这些消息响应函数都为**纯虚函数**，每个CWnd对象也要背负着一个巨大的虚拟表，这也是得不偿失的。

许多朋友在学习消息映射时苦无突破，其原因是一开始就认为MFC的消息映射的目的是为了替代SDK窗口过程的编写——这本来没有理解错。但他们还有多一层的理解，认为既然是替代“旧”的东西，那么MFC消息映射身应该是更高层次的抽象、更简单、更容易认识。但结果是，如果我们不通过ClassWizard工具，手动添加消息是相当迷茫的一件事。

所以，我们在学习MFC消息映射时，首先要弄清楚：消息映射的目的，不是为更加快捷地向窗口过程添加代码，而是一种机制的改变。如果不想改变窗口过程函数，那么应该在哪里进行消息响应呢？许多朋友一知半解地认为：我们可以用HOOK技术，抢在消息队列前把消息抓取，把消息响应提到窗口过程的外面。再者，不同的窗口，会有不同的感兴趣的消息，所以每个MFC窗口都应该有一个表把感兴趣的消息和相应消息响应函数连系起来。然后得出——消息映射机制执行步骤是：当消息发生，我们用HOOK技术把本来要发送到窗口过程的消息捕获，然后对照一下MFC窗口的消息映射表，如果是表里面有的消息，就执行其对应的函数。

当然，用HOOK技术，我们理论上可以在不改变窗口过程函数的情况下，可以完成消息响应。MFC确实是这样做的，但实际操作起来可能跟你的想象差别很大。

现在我们来编写消息映射表，我们先定义一个结构，这个结构至少有两个项：一是消息ID，二是响应该消息的函数。如下：

c++代码	
1.	<code>struct AFX_MSGMAP_ENTRY</code>
2.	<code>{</code>
3.	<code>UINT nMessage; // 感兴趣的消息</code>
4.	<code>AFX_PMSG pfn; // 响应以上消息的函数指针</code>
5.	<code>}</code>

当然，只有两个成员的结构连接起来的消息映射表是不成熟的。Windows消息分为标准消息、控件消息和命令消息，每类型的消息都是包含数百不同ID、不同意义、不同参数的消息。我们要准确地判别发生了何种消息，必须再增加几个成员。还有，对于AFX_PMSG pfn，实际上等于作以下声明：

```
void (CCmdTarget::*pfn)(); // 提示：AFX_PMSG为类型标识，具体声明是：typedef void (AFX_MSG_CALL CCmdTarget::*AFX_PMSG)(void);
```

pfn是一个不带参数和返回值的CCmdTarget类型函数指针，只能指向CCmdTarget类中不带参数和返回值的成员函数，这样pfn更为通用，但我们响应消息的函数许多需要传入参数的。为了解决这个矛盾，我们还要增加一个表示参数类型的成员。当然，还有其它.....

最后，MFC我们消息映射表成员结构如下定义：

c++代码	
1.	<code>struct AFX_MSGMAP_ENTRY</code>
2.	<code>{</code>
3.	<code>UINT nMessage; // Windows 消息ID</code>
4.	<code>UINT nCode; // 控制消息的通知码</code>
5.	<code>UINT nID; // 命令消息ID范围的起始值</code>
6.	<code>UINT nLastID; // 命令消息ID范围的终点</code>
7.	<code>UINT nSig; // 消息的动作标识</code>
8.	<code>AFX_PMSG pfn;</code>

[新浪 \(12\)](#)[微博 \(11\)](#)[软件工程师 \(10\)](#)[诺基亚 \(10\)](#)[京东商城 \(10\)](#)[比特币 \(10\)](#)[Facebook \(9\)](#)[周鸿祎 \(9\)](#)[操作系统 \(8\)](#)[Galaxy \(8\)](#)[社交网络 \(8\)](#)[搜索引擎 \(8\)](#)[移动互联网 \(8\)](#)[C \(8\)](#)[亚马逊 \(7\)](#)[更多标签](#)[分享到](#)[一键分享](#)[QQ空间](#)[新浪微博](#)[百度云收藏](#)[人人网](#)[腾讯微博](#)[百度相册](#)[开心网](#)[腾讯朋友](#)[百度贴吧](#)[豆瓣网](#)[搜狐微博](#)[百度新首页](#)[QQ好友](#)[和讯微博](#)[更多...](#)[百度分享](#)

完全随机文章

[VS2010/MFC](#)[从Facebook和](#)[VS2010/MFC](#)[VS2010/MFC](#)[VS2010/MFC](#)[VS2010/MFC](#)[VS2010/MFC](#)[程序员的選擇：技術vs管理](#)[2015产品校招——阿里腾讯百度360小米...](#)[C、C++、python、Java、php、C#六种流行...](#)[App推广秘籍最全篇](#)[程序员修炼指南——引导你成为真正的...](#)[85后工作5年工资竟然涨了25倍——月薪...](#)[TCP/UDP网络编程入门教程之十五：TC...](#)[memcached使用场景和方法总结](#)[从《奋斗》到《欢乐颂》看青年的价值观...](#)[StackOverflow 创始人推荐程序员看...](#)[魅族的掉队已成事实，生态链不是那么...](#)[详解HTML5 LocalStorage本地存储](#)[如何避免成为下一个雅虎](#)[说说Javascript闭包这点事](#)

最新评论及回复

[CAddSheet\(LPCTSTR ...](#)[有没有XTP的教学？求教](#)[就画个界面，搞这么负责，难怪MFC要被淘汰](#)[讲的太好了，很全，很清楚！楼主你的Q...](#)[楼主，请问如何动态给重写的CList...](#)[普通人只有被剥削的份](#)[已点广，，，，，告支持楼主](#)#include <afxco...[关掉王者荣耀。它就像鸦片，勾引小孩子...](#)[关掉王者荣耀。它就像鸦片，勾引小孩子](#)[楼主，请问CTabCtrl和CLis...](#)[一、初始化函数中在设置好子对话框位置...](#)[创建两组Radio可以在Radio的...](#)[一定要通过【类向导】添加类，【类向导...](#)[写的太棒了。](#)[谢谢楼主](#)[加油](#)[弱的问一声？符号常量的用法是否跟C语...](#)[蛮实用的可是在8年后才看到\[REV...](#)[好的程序员一定是挣钱的](#)

最近发表

[鸡啄米开始承接项目啦](#)

```
9.    };
```

有了以上消息映射表成员结构，我们就可以定义一个AFX_MSGMAP_ENTRY类型的数组，用来容纳消息映射项。定义如下：

```
AFX_MSGMAP_ENTRY _messageEntries[];
```

但这样还不够，每个AFX_MSGMAP_ENTRY数组，只能保存着当前类感兴趣的消息，而这仅仅是我们想处理的消息中的一部分。对于一个MFC程序，一般有多个窗口类，里面都应该有一个AFX_MSGMAP_ENTRY数组。

我们知道，MFC还有一个消息传递机制，可以把自己不处理的消息传送给别的类进行处理。为了能查找各下MFC对象的消息映射表，我们还要增加一个结构，把所有的AFX_MSGMAP_ENTRY数组串联起来。于是，我们定义了一个新结构体：

c++代码

```
1.    struct AFX_MSGMAP
2.    {
3.        const AFX_MSGMAP* pBaseMap;           //指向别的类的AFX_MSGMAP
        对象
4.        const AFX_MSGMAP_ENTRY* lpEntries;     //指向自身的消息表
5.    };
```

之后，在每个打算响应消息的类中声明这样一个变量：AFX_MSGMAP messageMap，让其其中的pBaseMap指向基类或另一个类的messageMap，那么将得到一个AFX_MSGMAP元素的单向链表。这样，所有的消息映射信息形成了一张信息网。

当然，仅有消息映射表还不够，它只能把各个MFC对象的消息、参数与相应的消息响应函数连成一张网。为了方便查找，MFC在上面的类中插入了两个函数（其中theClass代表当前类）：

一个是_GetBaseMessageMap()，用来得到基类消息映射的函数。函数原型如下：

c++代码

```
1.    const AFX_MSGMAP* PASCAL theClass::_GetBaseMessageMap() /
2.    { return &baseClass::messageMap; } /
```

另一个是GetMessageMap()，用来得到自身消息映射的函数。函数原型如下：

c++代码

```
1.    const AFX_MSGMAP* theClass::GetMessageMap() const /
2.    { return &theClass::messageMap; } /
```

有了消息映射表之后，我们得讨论到问题的关键，那就是消息发生以后，其对应的响应函数如何被调用。大家知道，所有的MFC窗口，都有一个同样的窗口过程——AfxWndProc(...)。在这里顺便要提一下的是，看过MFC源代码的朋友都得，从AfxWndProc函数进去，会遇到一大堆曲折与迷团，因为对于这个庞大的消息映射机制，MFC要做的事情很多，如优化消息，增强兼容性等，这一大量的工作，有些甚至用汇编语言来完成，对此，我们很难深究它。所以我们要省略大量代码，理性地分析它。

对已定型的AfxWndProc来说，对所有消息，最多只能提供一种默认的处理方式。这当然不是我们想要的。我们想通过AfxWndProc最终执行消息映射网中对应的函数。那么，这个执行路线是怎样的呢？

从AfxWndProc下去，最终会调用到一个函数OnWndMsg。请看代码：

c++代码

```
1.    LRESULT CALLBACK AfxWndProc(HWND hWnd,UINT nMsg,WPARAM wParam, LPARAM
        M lParam)
2.    {
3.        .....
4.        CWnd* pWnd = CWnd::FromHandlePermanent(hWnd); //把对句柄的操作转换
```

小白照样读懂的VLAN原理讲解

SSH电商项目实战之十：商品类基本模块的搭建

SSH电商项目实战之九：添加和更新商品类别功能的实现

SSH电商项目实战之八：查询和删除商品类别功能的实现

SSH电商项目实战之七：Struts2和Json的整合

长文：内容产业的赢家与输家

SSH电商项目实战之六：基于DataGrid的数据显示

SSH电商项目实战之五：完成数据库的级联查询和分页

分享到

SSH电商项目

SSH电商项目

面框架

SSH电商项目

和Action的

大妈：我们

SSH电商项目

和Spring

面临连续亏损

一键分享

新浪微博

人人网

百度相册

腾讯朋友

豆瓣网

百度新首页

和讯微博

QQ空间

百度云收藏

腾讯微博

开心网

百度贴吧

搜狐微博

QQ好友

更多...

百度分享

成对CWnd对象。

```
5.         Return AfxCallWndProc (pWnd,hWnd,nMsg,wParam,lParam);
6.     }
```

把对句柄的操作转换成对CWnd对象是很重要的-一件事，因为AfxWndProc只是一个全局函数，当然不知怎么样去处理各种windows窗口消息，所以它聪明地把处理权交给windows窗口所关联的MFC窗口对象。

现在，大家几乎可以想象得到AfxCallWndProc要做的事情，不错，它当中有一句：

```
pWnd->WindowProc(nMsg,wParam,lParam);
```

到此，MFC窗口过程函数变成了自己的一个成员函数。WindowProc是一个虚函数，我们甚至可以通过改写这个函数去响应不同的消息，当然，这是题外话。

WindowProc会调用到CWnd对象的另一个成员函数OnWndMsg，下面看看大概的函数原型是怎么样的：

c++代码

```
1.  BOOL CWnd::OnWndMsg (UINT message, WPARAM wParam, LPARAM lParam, LRESULT
    * pResult)
2.  {
3.      if (message==WM_COMMAND)
4.      {
5.          OnCommand (wParam, lParam);
6.          .....
7.      }
8.      if (message==WM_NOTIFY)
9.      {
10.         OnCommand (wParam, lParam, &lResult);
11.         .....
12.     }
13.     const AFX_MSGMAP* pMessageMap= GetMessageMap ();
14.     const AFX_MSGMAP_ENTRY* lpEntry;
15.     /*以下代码作用为：用AfxFindMessageEntry函数从消息入口pMessageMap处查找
    指定消息，如果找到，返回指定消息映射表成员的指针给lpEntry。然后执行该结构成员的pfn
    所指向的函数*/
16.     if ((lpEntry=AfxFindMessageEntry (pMessageMap->lpEntries,message,
    0,0) !=NULL)
17.     {
18.         lpEntry->pfn(); /*注意：真正MFC代码中没有用这一条语句。上面提到，不
    同的消息参数代表不同的意义和不同的消息响应函数有不同类型的返回值。而pfn是一个不带参
    数的函数指针，所以真正的MFC代码中，要根据对象lpEntry的消息的动作标识nSig给消息处
    理函数传递参数类型。这个过程包含很复杂的宏代换，大家在此知道：找到匹配消息，执行相应
    函数就行！*/
19.     }
20. }
```

MFC命令传递

在上面的代码中，大家看到了OnWndMsg能根据传进来的消息参数，查找到匹配的消息和执行相应的消息响应。但这还不够，我们平常响应菜单命令消息的时候，原本属于框架窗口（CFrameWnd）的WM_COMMAND消息，却可以放到视对象或文档对象中去响应。其原理如下：

我们看上面函数OnWndMsg原型中看到以下代码：

```
if (message==WM_COMMAND)
{
    OnCommand(wParam,lParam);
    .....
```

分享到

一键分享	QQ空间
新浪微博	百度云收藏
人人网	腾讯微博
百度相册	开心网
腾讯朋友	百度贴吧
豆瓣网	搜狐微博
百度新首页	QQ好友
和讯微博	更多...

百度分享

分享

```
}
```

即对于命令消息，实际上是交给OnCommand函数处理。而OnCommand是一个虚函数，即WM_COMMAND消息发生时，最终是发生该消息所对应的MFC对象去执行OnCommand。比如点框架窗口菜单，即向CFrameWnd发送一个WM_COMMAND，将会导致CFrameWnd::OnCommand(wParam, lParam)的执行。且看该函数原型：

c++代码

```
1.  BOOL CFrameWnd::OnCommand(WPARAM wParam, LPARAM lParam)
2.  {
3.      .....
4.      return CWnd::OnCommand(wParam, lParam);
5.  }
```

可以看出，它最后把该消息交给CWnd::OnCommand处理。再看：

c++代码

```
1.  BOOL CWnd::OnCommand(WPARAM wParam, LPARAM lParam)
2.  {
3.      .....
4.      return OnCmdMsg(nID, nCode, NULL, NULL);
5.  }
```

这里包含了一个C++多态性很经典的问题。在这里，虽然是执行CWnd类的函数，但由于这个函数在CFrameWnd::OnCmdMsg里执行，即当前指针是CFrameWnd类指针，再有OnCmdMsg是一个虚函数，所以如果CFrameWnd改写了OnCommand，程序会执行CFrameWnd::OnCmdMsg(...)。

对CFrameWnd::OnCmdMsg(...)函数的原理扼要分析如下：

c++代码

```
1.  BOOL CFrameWnd::OnCmdMsg(...)
2.  {
3.      CView pView = GetActiveView(); //得到活动视指针。
4.      if(pView->OnCmdMsg(...))
5.          return TRUE; //如果CView类对象或其派生类对象已经处理该消息，则返回。
6.      ..... //否则，同理向下执行，交给文档、框架、及应用程序执行自身的OnCmdMsg。
7.  }
```

到此，CFrameWnd::OnCmdMsg完成了把WM_COMMAND消息传递到视对象、文档对象及应用程序对象实现消息响应。

写了这么多，我们已经清楚了MFC消息映射与命令传递的大致过程。

MFC消息映射宏

现在，我们来看MFC“神秘代码”，会发觉好看多了。

先看DECLARE_MESSAGE_MAP()宏，它在MFC中定义如下：

c++代码

```
1.  #define DECLARE_MESSAGE_MAP() /
2.  private: /
3.      static const AFX_MSGMAP_ENTRY _messageEntries[]; /
4.  protected: /
5.      static AFX_DATA const AFX_MSGMAP messageMap; /
6.      virtual const AFX_MSGMAP* GetMessageMap() const; /
```

可以看出DECLARE_MESSAGE_MAP()定义了我们熟悉的两个结构和一个函数，显而易见，这个宏为每个需要实现消息映射的类提供了相关变量和函数。

分享到

一键分享	QQ空间
新浪微博	百度云收藏
人人网	腾讯微博
百度相册	开心网
腾讯朋友	百度贴吧
豆瓣网	搜狐微博
百度新首页	QQ好友
和讯微博	更多...

百度分享

分享

现在集中精力来看一下BEGIN_MESSAGE_MAP, END_MESSAGE_MAP和ON_COMMAND三个宏，它们在MFC中定义如下（其中ON_COMMAND与另外两个宏并没有定义在同一个文件中，把它放到一起是为了好看）：

c++代码

```
1.  #define BEGIN_MESSAGE_MAP(theClass, baseClass) /
2.  const AFX_MSGMAP* theClass::GetMessageMap() const /
3.  { return &theClass::messageMap; } /
4.  AFX_COMDAT AFX_DATADEF const AFX_MSGMAP theClass::messageMap = /
5.  { &baseClass::messageMap, &theClass::_messageEntries[0] }; /
6.  AFX_COMDAT const AFX_MSGMAP_ENTRY theClass::_messageEntries[] = /
7.  { /
8.
9.  #define ON_COMMAND(id, memberFxn) /
10.     { WM_COMMAND, CN_COMMAND, (WORD)id, (WORD)id, AfxSig_vv, (AFX_P
    MSG)&memberFxn },
11.
12.  #define END_MESSAGE_MAP() /
13.  {0, 0, 0, 0, AfxSig_end, (AFX_PMSG)0 } /
14.  }; /
```

一下子看三个宏觉得有点复杂，但这仅仅是复杂，公式性的文字代换并不是很难。且看下面例子，假设我们框架中有一菜单项为“Test”，即定义了如下宏：

c++代码

```
1.  BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
2.      ON_COMMAND(ID_TEST, OnTest)
3.  END_MESSAGE_MAP()
```

那么宏展开之后得到如下代码：

c++代码

```
1.  const AFX_MSGMAP* CMainFrame::GetMessageMap() const
2.
3.  { return &CMainFrame::messageMap; }
4.
5.  ///以下填入消息表映射信息
6.
7.  const AFX_MSGMAP CMainFrame::messageMap =
8.
9.  { &CFrameWnd::messageMap, &CMainFrame::_messageEntries[0] };
10.
11.  //下面填入保存着当前类感兴趣的消息，可填入多个AFX_MSGMAP_ENTRY对象
12.
13.  const AFX_MSGMAP_ENTRY CMainFrame::_messageEntries[] =
14.
15.  {
16.
17.  { WM_COMMAND, CN_COMMAND, (WORD)ID_TEST, (WORD)ID_TEST, AfxSig_vv, (AF
    X_PMSG)&OnTest },          //      加入的ID_TEST消息参数
18.
19.  {0, 0, 0, 0, AfxSig_end, (AFX_PMSG)0 } //本类的消息映射的结束项
20.
21.  };
```

大家知道，要完成ID_TEST消息映射，还要定义和实现OnTest函数。即在头文件中写afx_msg v

分享到

一键分享	QQ空间
新浪微博	百度云收藏
人人网	腾讯微博
百度相册	开心网
腾讯朋友	百度贴吧
豆瓣网	搜狐微博
百度新首页	QQ好友
和讯微博	更多...

百度分享

oid OnTest()并在源文件中实现它。根据以上所学的东西，我们知道了当ID为ID_TEST的命令消息发生，最终会执行到我们写的OnTest函数。

至此，MFC六大关键技术写完了。其中写得最难的是消息映射与命令传递，除了技术复杂之外，最难的是有许多避不开的代码。为了大家看得轻松一点，我把那繁杂的宏放在文章最后，希望能给你带来阅读带来方便。

来自：http://blog.csdn.net/liyi268/article/details/623391

除非特别注明，鸡啄米文章均为原创
转载请标明本文地址：<http://www.jizhuomi.com/software/275.html>

2012年12月11日

作者:鸡啄米 分类:软件开发 浏览:240240 评论:8

相关文章:

- MFC六大核心机制之四：永久保存（串行化）（2012-12-4 21:50:31）
- MFC六大核心机制之三：动态创建（2012-11-30 21:43:21）
- MFC六大核心机制之二：运行时类型识别（RTTI）（2012-11-26 21:5:33）
- MFC六大核心机制之一：MFC程序的初始化（2012-11-22 22:20:43）
- VS2010中如何实现自定义MFC控件（2012-11-18 22:56:16）
- VS2010/MFC编程入门教程之目录和总结（2012-10-31 22:4:12）
- VS2010/MFC编程入门之五十四（Ribbon界面开发：使用更多控件并为控件添加消息处理函数）（2012-10-27 21:56:47）
- VS2010/MFC编程入门之五十三（Ribbon界面开发：为Ribbon Bar添加控件）（2012-10-21 21:53:16）
- VS2010/MFC编程入门之五十二（Ribbon界面开发：创建Ribbon样式的应用程序框架）（2012-10-17 22:0:34）
- VS2010/MFC编程入门之五十一（图形图像：GDI对象之画刷CBrush）（2012-10-12 22:19:2）

分享到

分享到

一键分享

QQ空间

新浪微博

百度云收藏

人人网

腾讯微博

百度相册

开心网

腾讯朋友

百度贴吧

豆瓣网

搜狐微博

百度新首页

QQ好友

和讯微博

更多...

百度分享

分享

1楼. 足球比分

不懂代码，还在学习中

2012/12/12 11:55:13 回复该留言

2楼. 贺紫菲

小白飘过，留字表示支持

2012/12/13 10:38:54 回复该留言

3楼. 润初颜

技术牛人呀！

2012/12/13 11:22:12 回复该留言

4楼. 旅途者

额滴神，那么长的，我真没看完.....

鸡啄米 于 2012-12-13 23:21:16 回复
呵呵，是有点小长

2012/12/13 15:42:31 回复该留言

5楼. 分享美丽

这篇写得最有技术含量了，言简意赅，真是不可多得的好文！

2012/12/24 7:15:40 回复该留言

6楼. 谢小白

看了一半后感觉不是非常懂，看来看后要自己想

2015/3/24 20:23:29 回复该留言

7楼. [wegen](#)

看了两，有点明白，还是迷迷糊糊。

2015/8/21 11:44:14 [回复该留言](#)

8楼. [星夜](#)

还是没怎么看懂额...
指出一点小问题，关于宏那，应该是反斜线吧！

2017/2/23 10:02:17 [回复该留言](#)

上一篇: [苹果惧怕三星的理由](#)

下一篇: [低调华为的开放之路](#)


发表评论:

名称(*)

邮箱(选填)

网站链接(选填)

验证(*)



正文*)(留言最长字数:1000)

☐ 记住我,下次回复时不用重新输入个人信息

[\[URL\]](#) [\[URL2\]](#) [\[EMAIL\]](#) [\[EMAIL2\]](#) [\[B\]](#) [\[I\]](#) [\[U\]](#) [\[S\]](#) [\[QUOTE\]](#) [显示UBB表情>>](#)

◎欢迎参与讨论，请在这里发表您的看法、交流您的观点。

分享到

分享到

一键分享	QQ空间
新浪微博	百度云收藏
人人网	腾讯微博
百度相册	开心网
腾讯朋友	百度贴吧
豆瓣网	搜狐微博
百度新首页	QQ好友
和讯微博	更多...

百度分享

[无觅相关文章插件](#)