2021/04/12_MFC原理_第1课_框架的设计、MFC框架的模拟实现

笔记本:	MFC原理
创建时间:	2021/4/12 星期一 9:56
作者:	ileemi
标签:	MFC框架

框架的逆向

在Windows平台上不适合对QT框架进行逆向，原因是QT跨平台，内部考虑的平台性等问题较多，QT中的一些界面是操作显卡进行自绘的，比较复杂，且代码量较大。

为了解决框架逆向效率所以在Windows上就适合对MFC框架进行逆向。

提高框架逆向效率：就需要去了解框架（最好达到 "我就是框架设计者" 的水平）。框架逆向的速度就和框架的了解程度有关系。

课程目标

> 从零开始手写一个框架，以及从两个方向了解已知框架结构：

- 正向角度理解框架（懂设计）
- 逆向角度理解框架
- 实现逆向工具

框架的设计

MFC框架就是为了提高开发效率。不使用框架的情况下正常开发就需要使用SDK进行开发，找到SDK效率低的地方，并对这些地方进行改进，进而提高效率。

SDK开发中，效率较低的地方：

- 窗口操作
- 空间操作
- 消息循环
- **消息处理**
- API的调用

解决以上效率低问题的最直接的方法就是对控件或者消息等封装：

- 面向过程（主流框架多使用面向对象）
- 面向对象：设计类，类的核心不是语法，而是抽象（分类）。

软件的设计来源与现实（抽象为代码）。分类封装时就需要注意其重用性。框架的欢迎度取决于框架的设计。

网络发包抽象设计的伪代码：

```
class SocketAddr;
class Socket
  socket();
  bind(SocketAddr);
  DataInstream getDataInstream();
  DataOutstream getDataOutstream();

class TCPSocket :public Socket
  connect();

class UDPSocket :public Socket

// 数据流，这里是对数据的抽象
class Datastream;
class DataInstream :public Datastream // 输入流
   int read();
   float read();

class DataOutstream :public Datastream // 输出流
   int write(int) {
    throw 1;
    return bytes;
   }
   write(char)
class SocketError
   getErrorMsg();

send(socketaddr(127.0.0., 7788));
send(socketaddr(www.baidu.com));

// 不在需要检查返回值
try {
 //File = new File("d:\1.txt");
 Socket s = new UDPSocket("127.0.0.1", 7788);
 s.getOutputStream();
 int bytes = s.write(5);
}
catch() {
```

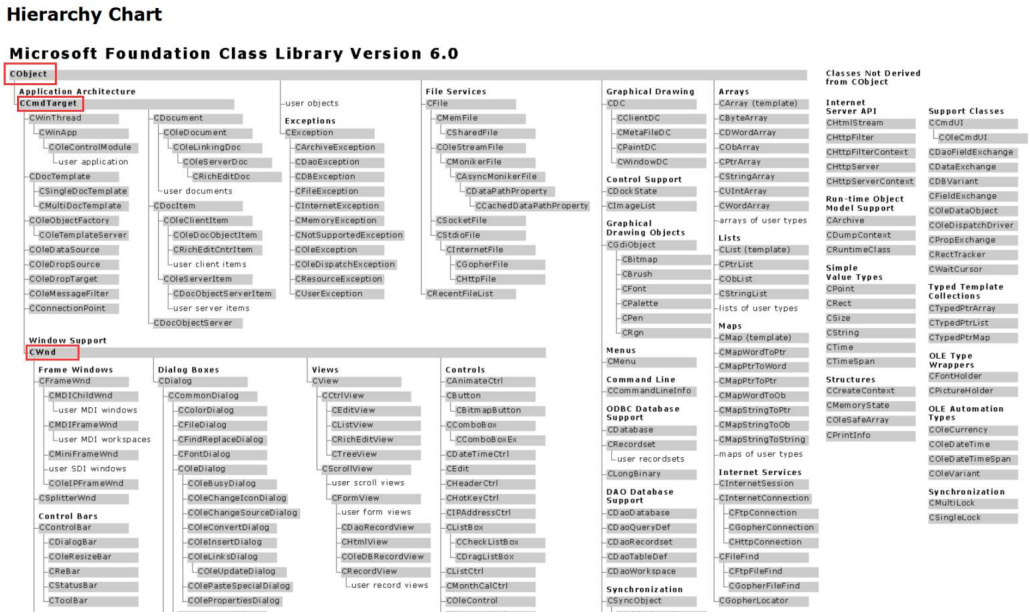
```
 ...
}

try {
 ...
}
catch() {
 ...
```

MFC 框架的模拟实现

设计框架中，在抽象时，必须抽象一个共同的基类（比如MFC的CObject），再写的类都是该基类的派生类，这样设计主要是为了方便以后**对类扩展功能。**

MFC框架继承关系图：



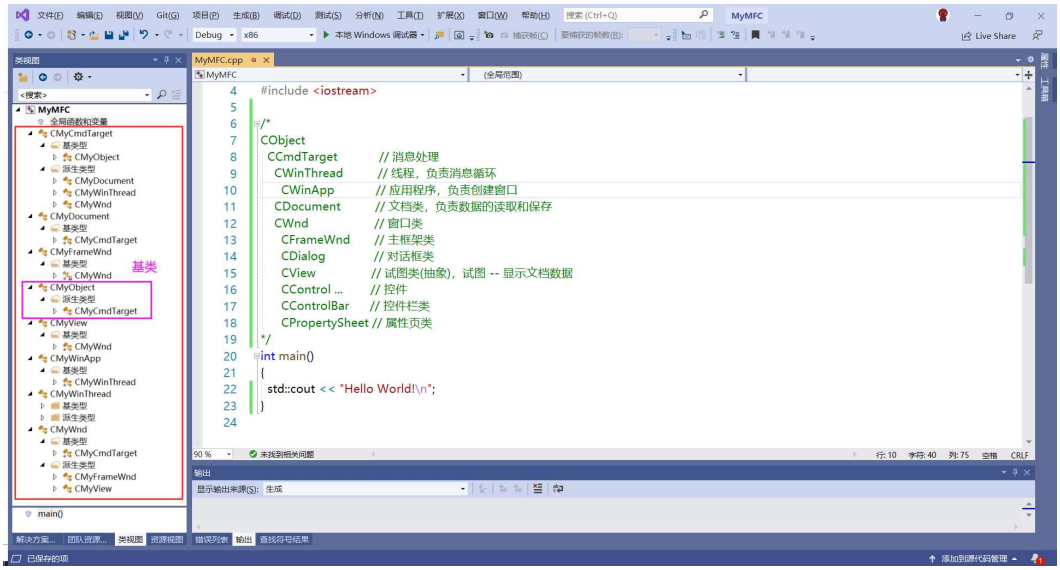
集成关系简述：

```
CObject
  CCmdTarget // 消息处理
    CWinThread  // 线程，负责消息循环
      CWinApp // 应用程序，负责创建窗口
    CDocument // 文档类，负责数据的读取和保存
    CWnd // 窗口类
      CFrameWnd // 主框架类
      CDialog // 对话框类
      CView  // 试图类(抽象)，试图 -- 显示文档数据
      CControl ... // 控件
      CControlBar // 控件栏类
      CPropertySheet // 属性页类
```

> QT 框架封装的所有类：官方文档

在MFC程序中，CWinApp类是必须存在的，文档类、视图类、主框架类都不是必须的。

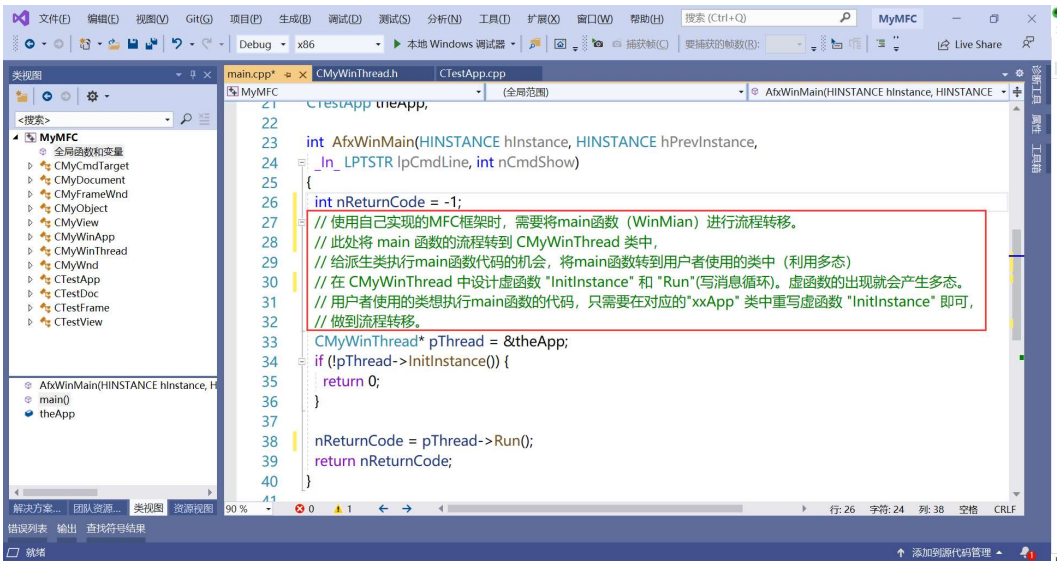
> 模拟MFC框架，主要类的设计如下图所示：

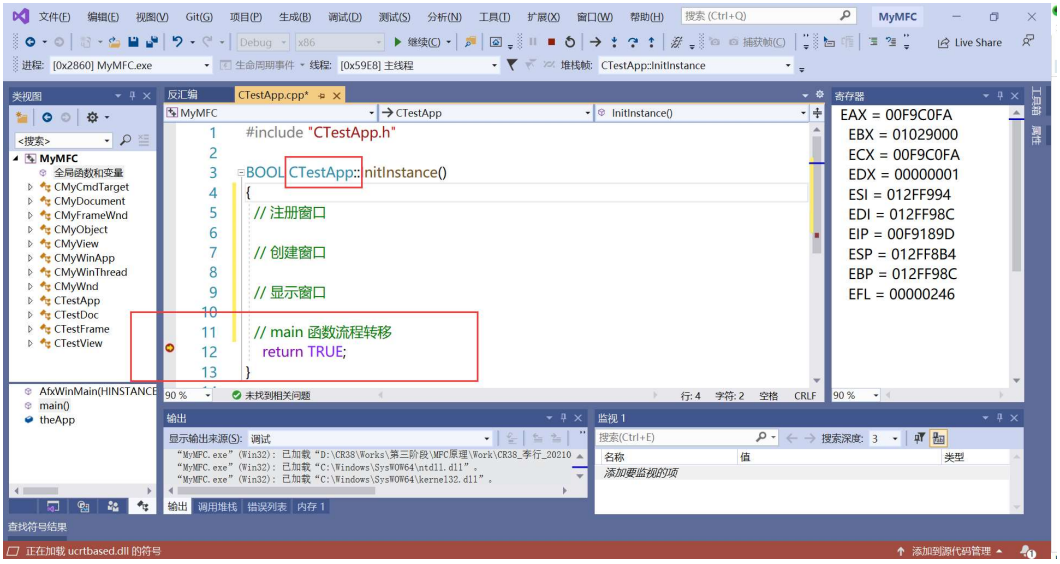


使用自己实现的MFC框架，需要对应的类去继承框架中对应的类，比如创建一个MFC单文档程序就有以下的继承关系：

- "CTestApp" 继承 "CMyWinApp"
- "CTestDoc" 继承 "CMyDocument"
- "CTestFrame" 继承 "CMyFrameWnd"
- "CTestView" 继承 "CMyView"

main 函数流程转移，如下图所示：





> 利用多态特性，接管流程，执行初始化操作（注册窗口类并创建、显示、更新窗口），之后进入消息循环处理消息，代码示例如下：

```
#include "CTestApp.h"

CTestApp theApp;
int main()
{
  //AfxWinMain(NULL, NULL, NULL, NULL);

  int nReturnCode = -1;
  CMyWinThread* pThread = &theApp;
  // 利用多态，初始化示例，创建窗口
  if (!pThread->InitInstance()) {
    // 初始化失败
    return -2;
  }

  // 利用多态，进入消息循环
  nReturnCode = pThread->Run();
  return nReturnCode;
}
```