**C#代码笔记**

# **MouseUtil.cs**

## private ILog \_logger = log4net.LogManager.GetLogger("MouseUtils");利用log4net库引用日志。

## [DllImport("user32.dll")]：以托管的方式引入dll库，而不是引用。

[DllImport("user32.dll")]

extern static bool SetCursorPos(int x, int y);

函数功能：该函数把光标移到屏幕的指定位置。

[DllImport("user32.dll")]

    private static extern bool SetCursorPos(int x, int y);

    [DllImport("user32.dll", EntryPoint = "SetCursorPos")]

    private static extern bool SetCursorPos1(int x, int y);

    [STAThread]

    public static void Main(string[] args)

    {

        int x = 0;

        int y = 0;

        SetCursorPos(x, y);

        SetCursorPos1(x, y);

        Console.ReadKey();

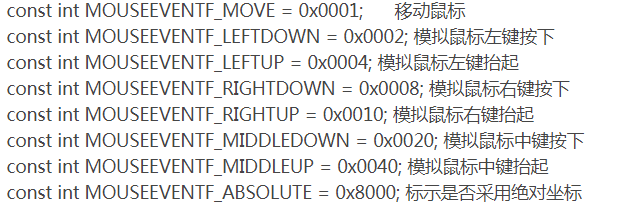
    }

## extern static void mouse\_event(int mouseEventFlag, int incrementX, int incrementY, uint data, UIntPtr extraInfo);

函数功能：模拟鼠标操作；



mouseEventFlag的参数意义：



## 定义枚举类：

Public enum MouseButton{

LeftButton,

MiddleButton,

RightButton

};

/\*

功能描述：

\* 鼠标功能：

\* 1、左键、右键、双击;

\* 2、点击实现的功能：获取element、获取point;

\*/

# KeyboardUtils.cs

## static extern short GetKeyState(int nVirtKey);

函数功能：表示虚拟键码;

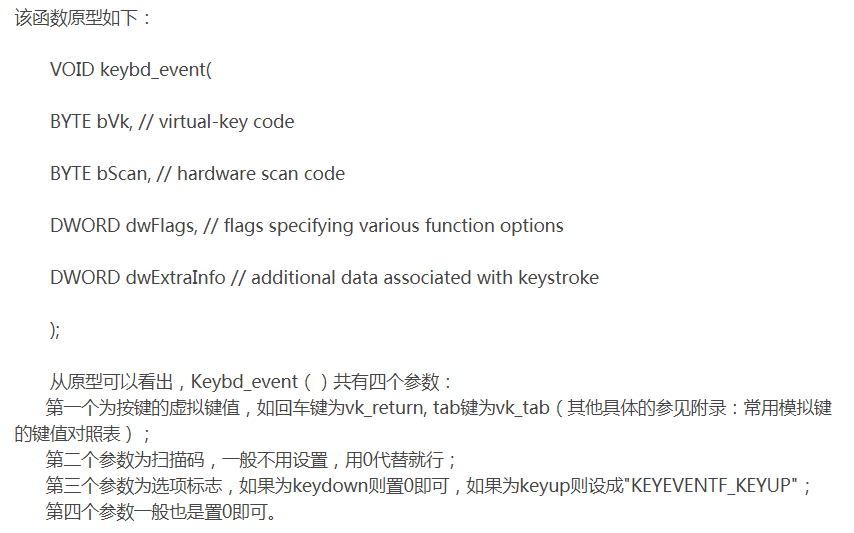
## [DllImport("user32.dll")]

## private static extern short VkKeyScan(char ch);

函数功能：是将一个字符翻译成相应的[虚拟键码](https://baike.baidu.com/item/%E8%99%9A%E6%8B%9F%E9%94%AE%E7%A0%81)和对于当前键盘的转换状态的一个函数。

## static extern void keybd\_event(byte bVk,byte bScan,uint dwFlags,uint dwExtraInfo);

函数功能：触发一个按键事件。





常用模拟键的键值对照表（也可参考<http://msdn.microsoft.com/en-us/library/dd375731(v=vs.85).aspx>

/\*

功能描述：

\* 1、函数：获取键盘状态、获取键码、按压键盘；

\* 2、功能：

\* a. ctrl按下；

\* b. ctrl回起;

\* c. alt+f4;

\* d. 按下、回起;

\* e. 大写按下、回起;

\* f. 输入字符串;(遍历每个字符--获取键码--键盘按回操作)

\* g.模拟键的键值参照。

\*/

# WindowUtils.cs

1、[DllImport("User32.dll", EntryPoint = "SendMessage")]

private static extern int SendMessage(IntPtr hWnd, int Msg, int wParam, int lParam);

函数功能：该函数将**指定的消息**发送到一个或多个**窗口。**

hWnd：接收消息的窗口句柄；

wParam：指定系统命令的类型。

在Windows系统消息处理中，wParam参数区分了类别，lParam参数存放了该类别所存储的信息。

案例1：

SendMessage(hWnd,WM\_KEYDOWN,VK\_TAB,0x000f0001);

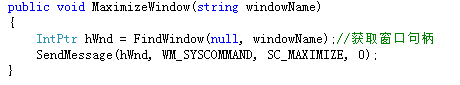
SendMessage(hWnd,WM\_CHAR,VK\_TAB,0x000f0001);

SendMessage(HWND, WM\_SYSCOMMAND, SC\_CLOSE, ByVal 0&)   '关闭

SendMessage(HWND, WM\_SYSCOMMAND, SC\_MINIMIZE, ByVal 0&) '最小化

SendMessage(HWND, WM\_PASTE, 0, 0) '粘贴

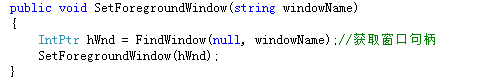
案例2：



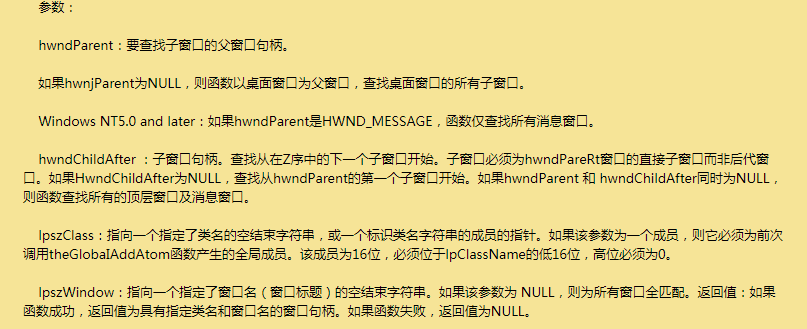
2、private static extern IntPtr FindWindow(string lpClassName, string lpWindowName);

功能描述：获得窗口的句柄。

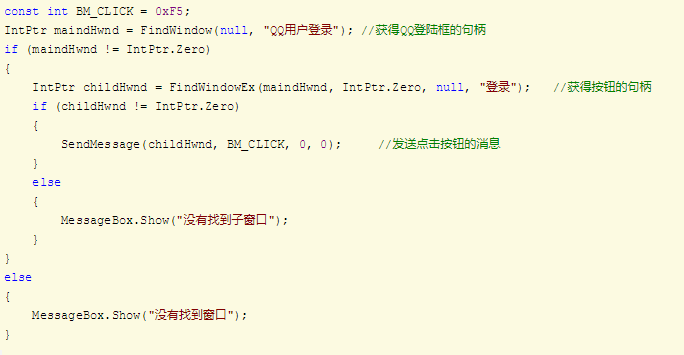
案例：



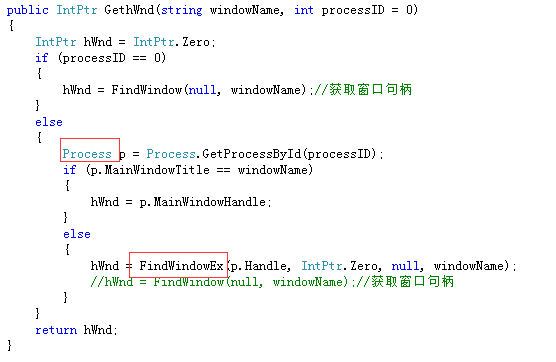
3、static extern IntPtr FindWindowEx(IntPtr hwndParent, IntPtr hwndChildAfter, string lpszClass, string lpszWindow);

功能描述：

举例说明：



案例2：

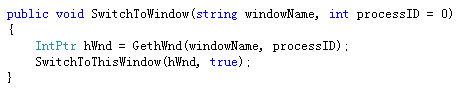


4、extern static bool SetForegroundWindow(IntPtr hWnd);

功能描述：将创建指定窗口的线程设置到前台，并且激活该窗口。

5、private static extern void SwitchToThisWindow(IntPtr hWnd, bool fAltTab);

功能描述：将焦点切换指定的窗口，并将其带到前台。



# InputUtils.cs

1、private static extern UInt32 SendInput(UInt32 nInputs, ref INPUT pInputs, int cbSize);

功能描述：模拟键盘输入；

2、private static extern IntPtr GetMessageExtraInfo();

功能描述：取得当前线程的额外信息；

3、

HmiAutomation.cs代码主要功能：

1. 获取主窗口的标题；
2. 关闭窗口；
3. 得到进程ID;
4. 获取主菜单状态；
5. 点击主菜单；
6. 通过右键菜单操作战场设备；
7. 通过右键菜单操作列车标识；
8. 根据战场图的显示名称获取设备名称、ID、设备对象；
9. 点击设备、使能设备；
10. 单击按钮；
11. 单击对话框按钮；
12. 获?取¨?GridCell文?本À?
13. 获?取¨?DataGrid总Á¨¹行D数ºy
14. 检¨¬查¨¦操¨´作Á¡Â堆?栈?是º?否¤?为a空?.
15. 获取控件的IsEnabled属性；
16. 获取设备菜单状态；
17. 执行使能；
18. 获取visualstatus;
19. 获取设备状态；
20. 获取真实的设备状态；
21. 获取轨道区段设备状态、真实的轨道区段设备状态；
22. 获取显示状态；
23. 验证状态；
24. 检查超时；
25. 检查状态标识是否在状态对象列表中；
26. 获取区段显示状态；
27. 获取对象属性与值的字典；