using OpenQA.Selenium.IE;

IntPtr handle = IntPtr.Zero;

var service = InternetExplorerDriverService.CreateDefaultService();

service.HideCommandPromptWindow = true;

var options = new InternetExplorerOptions();

ptions.IntroduceInstabilityByIgnoringProtectedModeSettings = true;

options.RequireWindowFocus = false;

var before = Process.GetProcessesByName("iexplore");

InternetExplorerDriver driver = new InternetExplorerDriver(service, options);

var after = Process.GetProcessesByName("iexplore");

var processes = new ProcessDiff().GetDiff(before, after);

if (processes.Count() == 1)

{

handle = processes.First().MainWindowHandle;

ShowWindow(handle, WindowShowStyle.Minimize);

}

else if (processes.Count() >= 1)

{

// In case IE opens an extra process, e.g. for Automatic Crash Recovery

foreach (var process in processes)

{

if (process.MainWindowTitle == driver.Title + " - Internet Explorer")

{

handle = process.MainWindowHandle;

ShowWindow(handle, WindowShowStyle.Minimize);

break;

}

}

}

If (DriverInUse) return;

DriverInUse = true;

var task = Task.Run(() =>

{

if (Which1 == ToolType.OPUS) Connect(driverOPUS);

if (Which1 == ToolType.LMT) LMT(driverLMT);

}

);

// cuando termine el task ejecute lo siguiente

task.ContinueWith((cont) =>

{

if (Which1 == ToolType.OPUS)

{

if (OPUSHandle != IntPtr.Zero) BrowserFlashRestore(OPUSHandle);

DriverInUse = false; // TERMINO DE CARGAR

}

}

public void Connect(InternetExplorerDriver driver)

{

string url = ConfigurationManager.AppSettings["phone\_OrderIntegrity"].ToString();

driver.Navigate().GoToUrl(url);

System.Threading.Thread.Sleep(500);

IWebElement element;

element = driver.FindElementById("Username");

element.SendKeys(GlobalData.ATTUserID);

element = driver.FindElementByClassName("attButton");

element.SendKeys(OpenQA.Selenium.Keys.Enter);

}

<!-- Settings whether to restore/flash browsers when done-->

<!-- Values: Flash, Restore-->

<add key="BrowserFlashRestore" value="Flash"/>

private void BrowserFlashRestore(IntPtr handle)

{

string FlashRestore = "";

try

{

FlashRestore = ConfigurationManager.AppSettings["BrowserFlashRestore"];

}

catch (Exception) { }

switch (FlashRestore.ToUpper())

{

case "FLASH":

FlashWindowEx(handle);

break;

case "RESTORE":

ShowWindow(handle, WindowShowStyle.Restore);

break;

default:

break;

}

}

API

[DllImport("user32.dll")]

static extern bool ShowWindow(IntPtr hWnd, WindowShowStyle nCmdShow);

// To support flashing.

[DllImport("user32.dll")]

[return: MarshalAs(UnmanagedType.Bool)]

static extern bool FlashWindowEx(ref FLASHWINFO pwfi);

//Flash both the window caption and taskbar button.

//This is equivalent to setting the FLASHW\_CAPTION | FLASHW\_TRAY flags.

public const UInt32 FLASHW\_ALL = 3;

// Flash continuously until the window comes to the foreground.

public const UInt32 FLASHW\_TIMERNOFG = 12;

[StructLayout(LayoutKind.Sequential)]

public struct FLASHWINFO

{

public UInt32 cbSize;

public IntPtr hwnd;

public UInt32 dwFlags;

public UInt32 uCount;

public UInt32 dwTimeout;

}

// Do the flashing - this does not involve a raincoat.

public static bool FlashWindowEx(IntPtr hWnd)

{

FLASHWINFO fInfo = new FLASHWINFO();

fInfo.cbSize = Convert.ToUInt32(Marshal.SizeOf(fInfo));

fInfo.hwnd = hWnd;

fInfo.dwFlags = FLASHW\_ALL | FLASHW\_TIMERNOFG;

fInfo.uCount = UInt32.MaxValue;

fInfo.dwTimeout = 0;

return FlashWindowEx(ref fInfo);

}