using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Interop;

namespace WpfApp

{

class HwndHostEx : HwndHost

{

[DllImport("user32.dll")]

private static extern IntPtr SetParent(IntPtr hWndChild, IntPtr hWndNewParent);

[DllImport("user32.dll")]

private static extern int SetWindowLong(IntPtr hWnd, int nIndex, int dwNewLong);

[DllImport("user32.dll")]

private static extern int GetWindowLong(IntPtr hWnd, int nIndex);

private IntPtr ChildHandle = IntPtr.Zero;

public HwndHostEx(IntPtr handle)

{

this.ChildHandle = handle;

}

protected override HandleRef BuildWindowCore(HandleRef hwndParent)

{

if (ChildHandle != IntPtr.Zero)

{

const int GWL\_STYLE = -16; // Index to change window styles

const int WS\_CHILD = 0x40000000; // Child window style flag

const int WS\_CAPTION = 0x00C00000; // Title bar

const int WS\_THICKFRAME = 0x00040000; // Resizable frame

// Get the current window style

int style = GetWindowLong(ChildHandle, GWL\_STYLE);

// Remove the caption and thick frame styles to disable moving and resizing

style &= ~WS\_CAPTION;

style &= ~WS\_THICKFRAME;

// Add the child window style

style |= WS\_CHILD;

// Apply the updated style to the window

SetWindowLong(ChildHandle, GWL\_STYLE, style);

// Set the parent to the WPF container

SetParent(this.ChildHandle, hwndParent.Handle);

// Return the handle to the hosted window

return new HandleRef(this, this.ChildHandle);

}

return new HandleRef();

}

protected override void DestroyWindowCore(HandleRef hwnd)

{

// Cleanup logic if required

}

}

}

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace WpfApp

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private Process \_process;

public MainWindow()

{

InitializeComponent();

Loaded += MainWindow\_Loaded;

}

private void MainWindow\_Loaded(object sender, RoutedEventArgs e)

{

string processName = "notepad";

var process = System.Diagnostics.Process.Start(processName);

// to create an instance:

var getprocess = System.Diagnostics.Process.GetProcesses()

.FirstOrDefault(item => item.ProcessName.ToLowerInvariant() == processName && item.MainWindowHandle != IntPtr.Zero);

var handle = getprocess?.MainWindowHandle;

if (handle != null)

{

var host = new HwndHostEx(handle.Value);

MainGrid.Children.Add(host);

}

}

protected override void OnClosed(EventArgs e)

{

base.OnClosed(e);

// Ensure the process is cleaned up

if (\_process != null && !\_process.HasExited)

{

\_process.Kill();

\_process.Dispose();

}

}

}

}

<Window x:Class="WpfApp.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApp"

mc:Ignorable="d"

Title="MainWindow" Height="650" Width="1050">

<Grid Name="MainGrid">

<!-- Other UI elements -->

</Grid>

</Window>