1. Icon
2. 右键工程->属性->资源->图像，新增icon图标资源，修改新增资源的复制到输出目录属性为始终复制，生成操作属性为Resource
3. Main类增加构造函数

public Main()

{

//添加tab的icon图片进入列表

FiddlerApplication.UI.tabsViews.ImageList.Images.Add("FiddlerExampleIcon", Properties.Resources.icon);

}

1. Main类的OnLoad增加初始化Icon的相关逻辑

//初始化icon

page.ImageIndex = FiddlerApplication.UI.tabsViews.ImageList.Images.IndexOfKey("FiddlerExampleIcon");

1. AlertTool类的initWindow方法增加初始化Icon相关逻辑

IntPtr iconPtr = Properties.Resources.icon.GetHbitmap();

ImageSource icon = System.Windows.Interop.Imaging.CreateBitmapSourceFromHBitmap(iconPtr, IntPtr.Zero, Int32Rect.Empty, BitmapSizeOptions.FromEmptyOptions());

//设置icon

window.Icon = icon;

1. 打包预览



1. 按需加载
2. Main类中将page提取成公有静态属性，并在构造函数中增加初始化相关逻辑

//新建一个Fiddler插件的page

page = new TabPage("PlugExample");

//将page加入Fiddler的tab选项卡中

FiddlerApplication.UI.tabsViews.TabPages.Add(page);

//初始化icon

page.ImageIndex = FiddlerApplication.UI.tabsViews.ImageList.Images.IndexOfKey("FiddlerExampleIcon");

1. Main类中增加私有Init方法，并删除OnLoad方法中初始化page的相关逻辑

private void Init()

{

//将WinForm和WPF联系起来(在WinForm中调用WPF)

ElementHost element = new ElementHost();

element.Child = container;

element.Dock = DockStyle.Fill;

//将WPF挂载对象添加到page中

page.Controls.Add(element);

}

1. Main类中的OnLoad方法增加Fiddler Tab选中委托相关逻辑

//创建委托对象

TabControlEventHandler tabSelectedEvent = null;

tabSelectedEvent = delegate (object obj, TabControlEventArgs e)

{

if (e.TabPage == page)

{

//初始化UI

Init();

//移除委托监听

FiddlerApplication.UI.tabsViews.Selected -= tabSelectedEvent;

FiddlerApplication.Log.LogString("FiddlerExample初始化完成！");

}

};

//添加委托监听

FiddlerApplication.UI.tabsViews.Selected += tabSelectedEvent;

1. 数据备份
2. DataTool类新增私有静态属性backupPath

//备份数据路径

private static string backupPath = path + "\\Backup";

1. 替换DataTool类中initFolder方法的path为backupPath
2. DataTool类新增私有静态方法backupConfigFile

private static void backupConfigFile()

{

try

{

int fileNum = Directory.GetFiles(backupPath, "\*.json").Length;

if (fileNum < 10)

{

FileStream fs = new FileStream(backupPath + "\\backup\_" + (fileNum + 1) + ".json", FileMode.Create);

StreamWriter sw = new StreamWriter(fs);

//开始写入

sw.Write(formatConfigData().ToString());

//清空缓冲区

sw.Flush();

//关闭流

sw.Close();

fs.Close();

}

else

{

//首先删除第一个文件

File.Delete(backupPath + "\\backup\_1.json");

//然后将之前的全部改名

for (int i = 1; i < 10; i++)

{

File.Move(backupPath + "\\backup\_" + (i + 1) + ".json", backupPath + "\\backup\_" + i + ".json");

}

//重新写入最新备份文件

FileStream fs = new FileStream(backupPath + "\\backup\_10.json", FileMode.Create);

StreamWriter sw = new StreamWriter(fs);

//开始写入

sw.Write(formatConfigData().ToString());

//清空缓冲区

sw.Flush();

//关闭流

sw.Close();

fs.Close();

}

}

catch (Exception e)

{

FiddlerApplication.Log.LogString("FiddlerExample出现错误(backupConfigFile函数)：" + e.ToString());

}

}

1. DataTool类的writeConfigToFile方法增加数据备份逻辑

//数据备份

backupConfigFile();

1. 补全说明
2. 在Resources文件夹中新增ExplainImages文件夹
3. 截取相关配置图片，并在ExplainImages文件夹上右键选择添加现有项，将截取的图片添加到ExplainImages文件夹
4. 修改新增图片的生成操作属性为Resource，复制到输出目录属性为始终复制
5. ExplainAlertUI控件添加相关图片代码

<TextBlock Style="{StaticResource alert\_style\_explain}">4、配置示例：</TextBlock>

<Image Source="../Resources/ExplainImages/001.png" Width="625"></Image>

<TextBlock Style="{StaticResource alert\_style\_explain}">5、映射结果示例：</TextBlock>

<Image Source="../Resources/ExplainImages/002.png" Width="625"></Image>

1. 打包预览



1. Release版

切换打包模式为Release，然后重新生成

