1. 说明按钮
2. 新建AlertUI文件夹，同时新建一个WPF控件，命名为ExplainAlertUI.xaml
3. 在GlobalStyle.xaml中新增相关样式

<!-- 说明 -->

<Style x:Key="alert\_style\_explain" TargetType="TextBlock">

<Setter Property="Padding" Value="10 5"></Setter>

<Setter Property="Foreground" Value="#FF4B4B4B"></Setter>

</Style>

<!-- 说明\_\_Label -->

<Style x:Key="alert\_style\_explain\_label" TargetType="Label">

<Setter Property="Foreground" Value="#FF4B4B4B"></Setter>

<Setter Property="Padding" Value="0,5,0,0"></Setter>

</Style>

1. 在ExplainAlertUI.xaml中引入全局样式文件，同时增加相关元素结构

<ScrollViewer VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto" Background="White">

<StackPanel Margin="10,0">

<StackPanel x:Name="host">

<Label HorizontalContentAlignment="Center" Margin="0,10">HOST映射说明</Label>

<TextBlock Style="{StaticResource alert\_style\_explain}">1、IP字段：填写映射机器的IP地址即可</TextBlock>

<TextBlock Style="{StaticResource alert\_style\_explain}">2、端口字段：非必填，填写则替换为指定端口号，为空则使用请求本身的端口号</TextBlock>

<TextBlock Style="{StaticResource alert\_style\_explain}">

3、URL字段：<LineBreak/>

<Label xml:space="preserve" Style="{StaticResource alert\_style\_explain\_label}"> (1)、不局限于域名，以映射https://www.example.com/test/index.html为例</Label><LineBreak/>

<Label xml:space="preserve" Style="{StaticResource alert\_style\_explain\_label}"> (2)、可以指定具体的Path，如：www.example.com/test</Label><LineBreak/>

<Label xml:space="preserve" Style="{StaticResource alert\_style\_explain\_label}"> (3)、可以是URL的任意部分，如：com/test/i</Label><LineBreak/>

<Label xml:space="preserve" Style="{StaticResource alert\_style\_explain\_label}"> (4)、支持正则表达式，如：\S\*.example.com</Label>

</TextBlock>

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

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

</StackPanel>

<Label Height="20"></Label>

</StackPanel>

</ScrollViewer>

1. 在Tools文件夹中新增AlertTool类，同时添加私有静态方法initWindow

//初始化窗体

private static Window initWindow(string type, int height, int width = 500)

{

Window window = new Window();

//设置宽和高

window.Width = width;

window.Height = height + 30;//状态栏的高度是30

//去掉最小化、最大化按钮

window.ResizeMode = 0;

window.Title = type == "explain" ? "配置说明" : "Host配置";

//设置显示在中间

window.WindowStartupLocation = WindowStartupLocation.CenterScreen;

//返回对应的窗体

return window;

}

1. AlertTool类中添加公有静态方法showExplainAlertUI

//显示说明弹框

public static void showExplainAlertUI(string type)

{

ExplainAlertUI explainAlertUI = new ExplainAlertUI();

//初始化窗体

Window window = initWindow("explain", 450, 700);

//设置window窗体内容

window.Content = explainAlertUI;

//显示窗体

window.ShowDialog();

}

1. 在Container类中增加私有方法showExplainAlertUI

#region Alert--事件

//显示说明弹框

private void showExplainAlertUI(object sender, MouseButtonEventArgs e)

{

AlertTool.showExplainAlertUI();

}

#endregion

1. 在Container控件中给说明按钮绑定MouseLeftButtonDown事件

<Label MouseLeftButtonDown="showExplainAlertUI" Template="{StaticResource top\_guide}" HorizontalAlignment="Left"></Label>

1. 打包预览



1. 新增按钮
2. 在AlertUI文件夹中新建一个WPF控件，命名为HostAlertUI.xaml
3. 在GlobalStyle.xaml中新增相关样式

<!-- 标题 -->

<Style x:Key="alert\_style\_title" TargetType="Label">

<Setter Property="Height" Value="40"></Setter>

<Setter Property="HorizontalAlignment" Value="Center"></Setter>

<Setter Property="VerticalAlignment" Value="Top"></Setter>

<Setter Property="HorizontalContentAlignment" Value="Center"></Setter>

<Setter Property="VerticalContentAlignment" Value="Center"></Setter>

<Setter Property="FontSize" Value="16"></Setter>

<Setter Property="Foreground" Value="#FF666666"></Setter>

<Setter Property="Margin" Value="0,20,0,0"></Setter>

</Style>

<!-- 容器 -->

<Style x:Key="alert\_style\_wrap" TargetType="StackPanel">

<Setter Property="Height" Value="50"></Setter>

<Setter Property="Orientation" Value="Horizontal"></Setter>

</Style>

<!-- 容器\_\_TextArea -->

<Style x:Key="alert\_style\_wrap\_textarea" TargetType="StackPanel" BasedOn="{StaticResource alert\_style\_wrap}">

<Setter Property="Height" Value="110"></Setter>

</Style>

<!-- 名称 -->

<Style x:Key="alert\_style\_name" TargetType="Label">

<Setter Property="Width" Value="80"></Setter>

<Setter Property="Height" Value="50"></Setter>

<Setter Property="HorizontalContentAlignment" Value="Right"></Setter>

<Setter Property="VerticalContentAlignment" Value="Center"></Setter>

<Setter Property="FontSize" Value="14"></Setter>

<Setter Property="Margin" Value="30,0,0,0"></Setter>

<Setter Property="Foreground" Value="#FF666666"></Setter>

</Style>

<!-- 输入框 -->

<Style x:Key="alert\_style\_input" TargetType="TextBox">

<Setter Property="Width" Value="300"></Setter>

<Setter Property="Height" Value="30"></Setter>

<Setter Property="Margin" Value="10,10,0,0"></Setter>

<Setter Property="Background" Value="{x:Null}"></Setter>

<Setter Property="Foreground" Value="Black"></Setter>

<Setter Property="BorderBrush" Value="#FF666666"></Setter>

<Setter Property="BorderThickness" Value="0,0,0,1"></Setter>

<Setter Property="VerticalAlignment" Value="Top"></Setter>

<Setter Property="VerticalContentAlignment" Value="Center"></Setter>

</Style>

<!-- 输入框\_\_TextArea -->

<Style x:Key="alert\_style\_textarea" TargetType="TextBox" BasedOn="{StaticResource alert\_style\_input}">

<Setter Property="Height" Value="90"></Setter>

<Setter Property="TextWrapping" Value="Wrap"></Setter>

<Setter Property="Padding" Value="0,0,0,5"></Setter>

</Style>

<!-- 确定按钮 -->

<Style x:Key="alert\_style\_sure\_btn" TargetType="Label">

<Setter Property="Width" Value="120"></Setter>

<Setter Property="Height" Value="40"></Setter>

<Setter Property="Padding" Value="0"></Setter>

<Setter Property="Margin" Value="0,20,0,0"></Setter>

<Setter Property="FontSize" Value="14"></Setter>

<Setter Property="Cursor" Value="Hand"></Setter>

<Setter Property="Background" Value="#FFEC8E72"></Setter>

<Setter Property="Foreground" Value="White"></Setter>

<Setter Property="HorizontalContentAlignment" Value="Center"></Setter>

<Setter Property="VerticalContentAlignment" Value="Center"></Setter>

</Style>

1. 在HostAlertUI.xaml中引入全局样式文件，同时增加相关元素结构，最后分别设置输入控件的唯一Name为：ip、port、url

<!--#region 样式、模板资源 -->

<UserControl.Resources>

<ResourceDictionary Source="../GlobalStyle.xaml"></ResourceDictionary>

</UserControl.Resources>

<!--#endregion-->

<StackPanel Background="White">

<Label Content="请填写相关配置" Style="{StaticResource alert\_style\_title}"></Label>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="295"/>

<ColumnDefinition Width="100\*"/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0" Style="{StaticResource alert\_style\_wrap}">

<Label Style="{StaticResource alert\_style\_name}">IP</Label>

<TextBox x:Name="ip" Style="{StaticResource alert\_style\_input}" Width="165"/>

</StackPanel>

<StackPanel Grid.Column="1" Style="{StaticResource alert\_style\_wrap}">

<Label Style="{StaticResource alert\_style\_name}" Width="40" Margin="0">端口</Label>

<TextBox x:Name="port" Style="{StaticResource alert\_style\_input}" Width="75" Margin="10,10,0,0"/>

</StackPanel>

</Grid>

<StackPanel Style="{StaticResource alert\_style\_wrap\_textarea}">

<Label Style="{StaticResource alert\_style\_name}">URL</Label>

<TextBox x:Name="url" Style="{StaticResource alert\_style\_textarea}"/>

</StackPanel>

<Label Style="{StaticResource alert\_style\_sure\_btn}">确定</Label>

</StackPanel>

1. 在HostAlertUI类中增加私有方法addHostRule和inputKeyDown

#region 鼠标点击事件

private void addHostRule(object sender, MouseButtonEventArgs e)

{

string ip = this.ip.Text;

string port = this.port.Text;

string url = this.url.Text;

if (ip.Length == 0)

{

Fiddler.FiddlerApplication.DoNotifyUser("请填写IP", "输入提示");

return;

}

if (url.Length == 0)

{

Fiddler.FiddlerApplication.DoNotifyUser("请填写URL", "输入提示");

return;

}

//添加UI

Main.addHostRule(ip, port, url);

//关闭弹框

(this.Parent as Window).Close();

}

#endregion

#region 输入框按键监听事件

private void inputKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

addHostRule(null, null);

}

}

#endregion

1. 给对应输入控件绑定KeyDown事件，确定按钮绑定MouseLeftButtonDown事件
2. AlertTool类增加公有静态方法showHostAlertUI

//显示Host弹框

public static void showHostAlertUI()

{

HostAlertUI hostAlertUI = new HostAlertUI();

//初始化窗体

Window window = initWindow("host", 310);

//设置window窗体内容

window.Content = hostAlertUI;

//自动聚焦

hostAlertUI.ip.Focus();

//显示窗体

window.ShowDialog();

}

1. Container类增加私有方法addHost

//增加Host

private void addHost(object sender, MouseButtonEventArgs e)

{

AlertTool.showHostAlertUI();

}

1. 给Container的新增按钮绑定MouseLeftButtonDown事件

<Label MouseLeftButtonDown="addHost" Style="{StaticResource top\_style\_operation}">新增</Label>

1. 打包预览





1. 全禁按钮
2. Main类增加公有静态方法disabledAllHostFromData

//禁止所有Host规则

public static void disabledAllHostFromData()

{

//遍历更新数据

for (int i = 0, len = mainData.Count; i < len; i++)

{

HostModel item = mainData[i] as HostModel;

//更新数据

item.Enable = false;

}

//重新写入文件

DataTool.writeConfigToFile();

}

1. Container类增加私有方法disabledAllItem

//禁止所有Item

private void disabledAllItem(object sender, MouseButtonEventArgs e)

{

Main.disabledAllHostFromData();

}

1. 给Container的新增按钮绑定MouseLeftButtonDown事件

<Label MouseLeftButtonDown="disabledAllItem" Style="{StaticResource top\_style\_operation\_disable}">全禁</Label>

1. 打包预览

