1. 禁止/生效
2. 在Main类中增加公有静态方法changeRuleEnableByIndex

//变更Rule的Enable数据

public void changeRuleEnableByIndex(int index)

{

//获取数据

HostModel rule = mainData[index] as HostModel;

//变更状态

rule.Enable = !rule.Enable;

//重新写入文件

DataTool.writeConfigToFile();

}

1. 新建GlobalEvent类，添加partial前缀，并且增加私有方法changeRuleEnable

//是否生效事件

private void changeRuleEnable(object sender, MouseButtonEventArgs e)

{

int index = Convert.ToInt32((sender as Rectangle).Tag.ToString());

//变更状态

Main.changeRuleEnableByIndex(index);

}

1. 修改GlobalStyle.xaml中事件绑定类为GlobalEvent
2. 给选中框添加MouseLeftButtonDown事件

<Rectangle Grid.Column="2" Tag="{Binding Path=Index}" MouseLeftButtonDown="changeRuleEnable" Visibility="{Binding Path=CheckHide}" Style="{StaticResource content\_style\_rect}"></Rectangle>

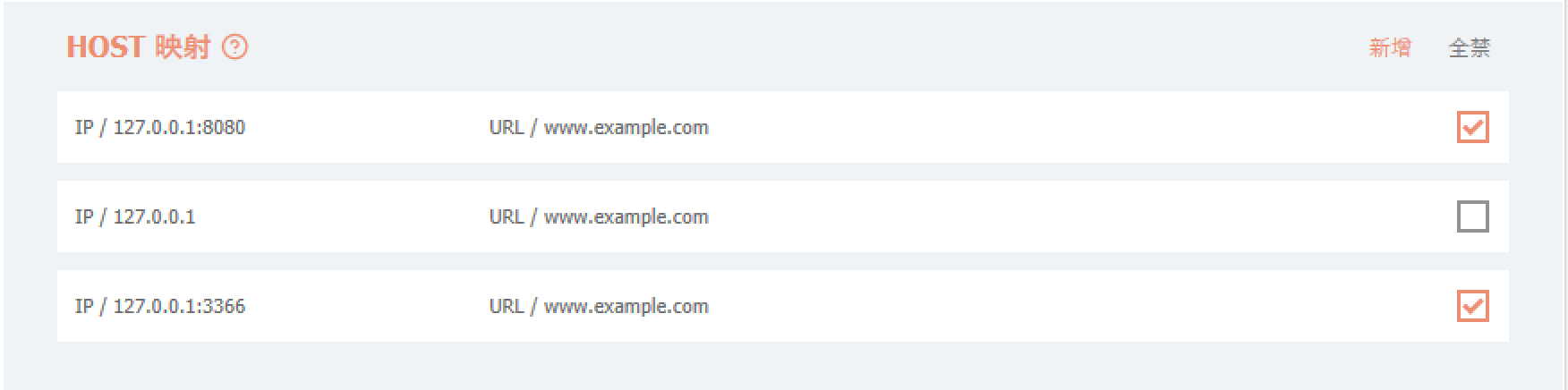
<Canvas Grid.Column="2" Width="12" Height="12" Visibility="{Binding Path=CheckShow}">

<Path Fill="#FFEC8E72" Data="M10.125 1.5l-5.625 5.625-2.625-2.625-1.875 1.875 4.5 4.5 7.5-7.5z" />

</Canvas>

<Rectangle Grid.Column="2" Tag="{Binding Path=Index}" MouseLeftButtonDown="changeRuleEnable" Visibility="{Binding Path=CheckShow}" Style="{StaticResource content\_style\_rect\_check}"></Rectangle>

1. 打包预览



1. 双击修改
2. HostAlertUI类的构造函数增加index参数，并设置默认值为-1
3. HostAlertUI类添加\_index属性，并在构造函数中初始化

private int \_index;

public HostAlertUI(int index = -1)

{

\_index = index;

InitializeComponent();

}

1. HostAlertUI类增加initInputText方法，并在构造函数中调用

#region 初始化输入框内容

private void initInputText()

{

//小于0代表是新增，直接返回

if (\_index < 0)

{

return;

}

//获取数据

HostModel rule = Main.mainData[\_index] as HostModel;

//设置数据

this.ip.Text = rule.IP;

this.port.Text = rule.Port;

this.url.Text = rule.Url;

}

#endregion

1. Main类中增加公有静态方法modifyRuleByIndex

//修改Rule数据

public static void modifyRuleByIndex(int index, string ip, string port, string url)

{

//获取规则

HostModel rule = mainData[index] as HostModel;

//更新数据

rule.IP = ip;

rule.Port = port;

rule.Url = url;

//重新写入文件

DataTool.writeConfigToFile();

}

1. 修改HostAlertUI类的addHostRule方法，增加修改相关逻辑
2. AlertTool类的showHostAlertUI方法增加index参数，设置默认值为-1，并传递给HostAlertUI
3. GlobalEvent类增加私有方法modifyRule

//修改规则

private void modifyRule(object sender, MouseButtonEventArgs e)

{

int index = Convert.ToInt32((sender as Label).Tag.ToString());

//显示弹框

AlertTool.showHostAlertUI(index);

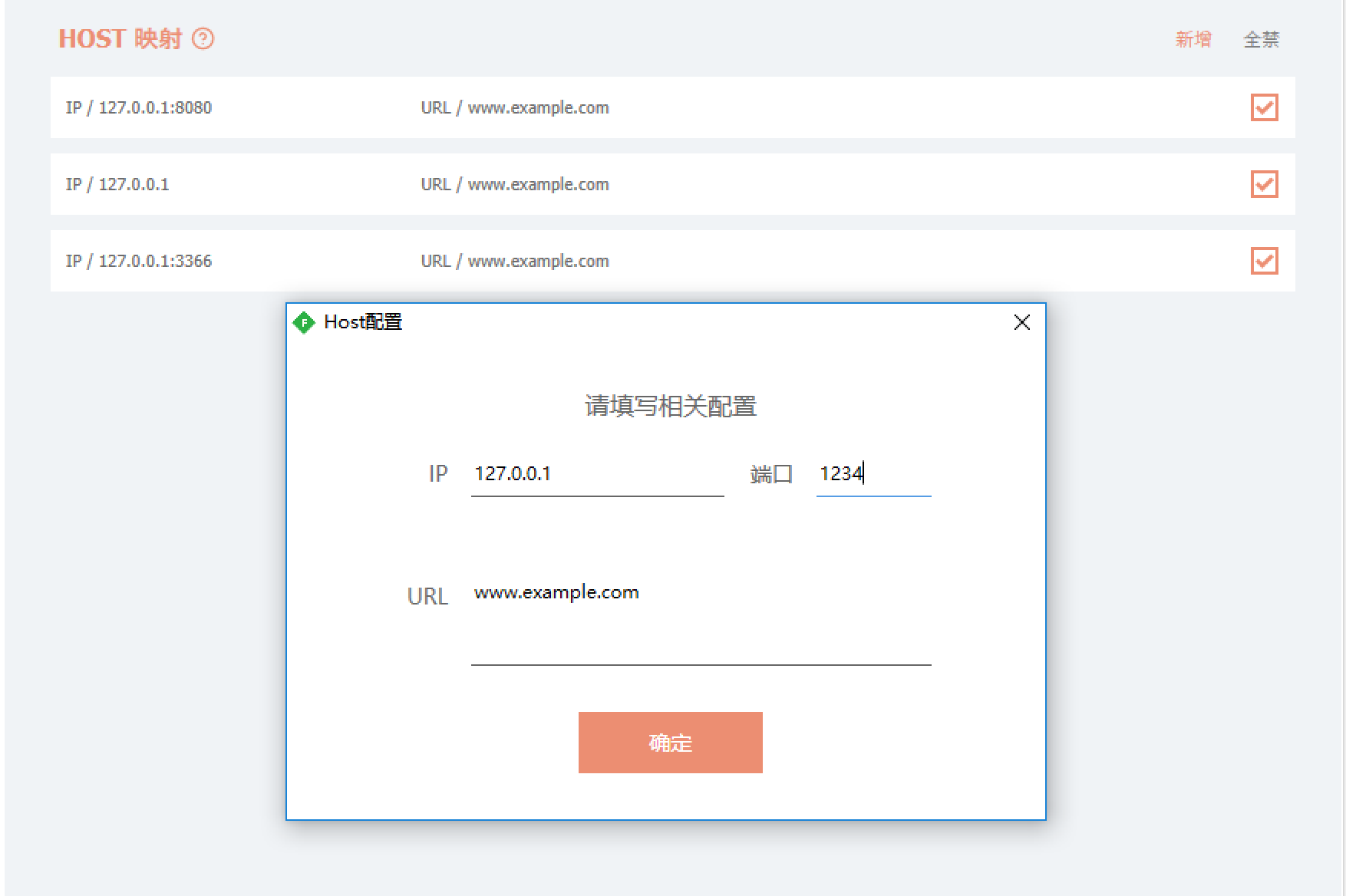
}

1. 在GlobalStyle中给对应控件绑定双击事件

<Label Grid.Column="0" Tag="{Binding Path=Index}" MouseDoubleClick="modifyRule" Content="{Binding Path=IpAndPort}" Template="{StaticResource content\_text}"></Label>

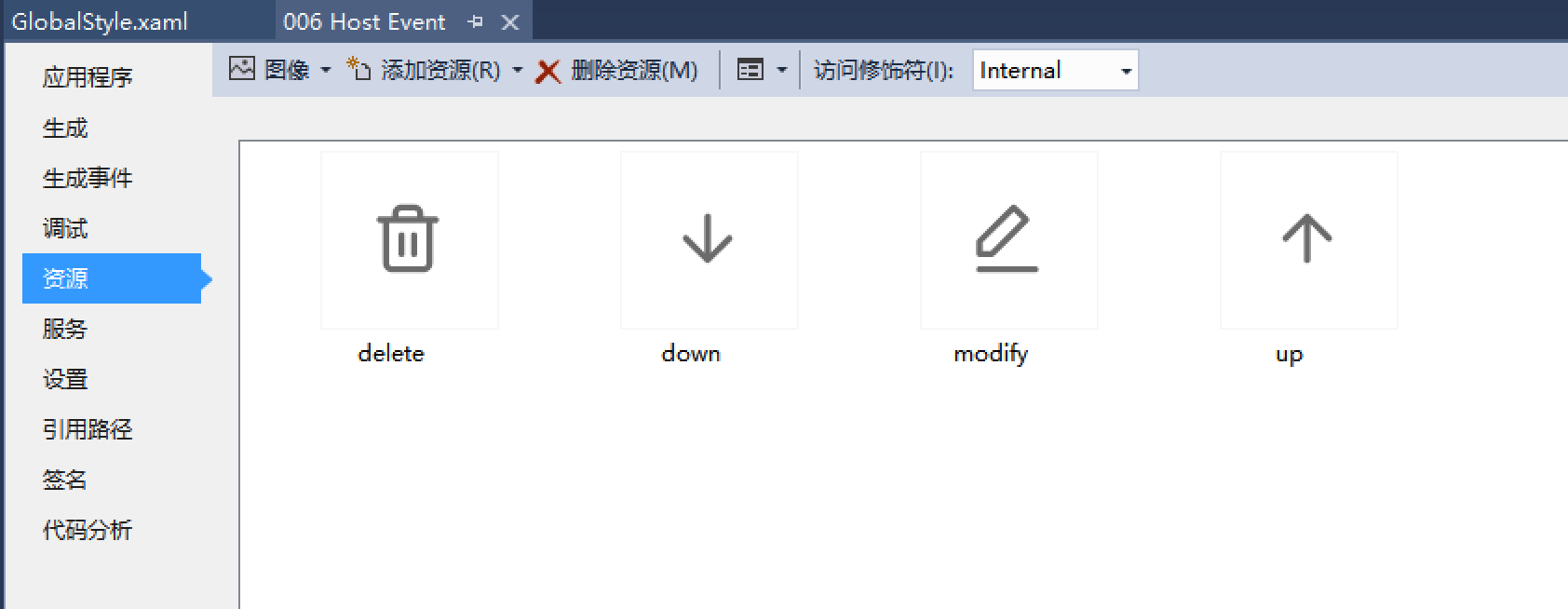
<Label Grid.Column="1" Tag="{Binding Path=Index}" MouseDoubleClick="modifyRule" Content="{Binding Path=TipsAndUrl}" Template="{StaticResource content\_text}"></Label>

1. 打包预览

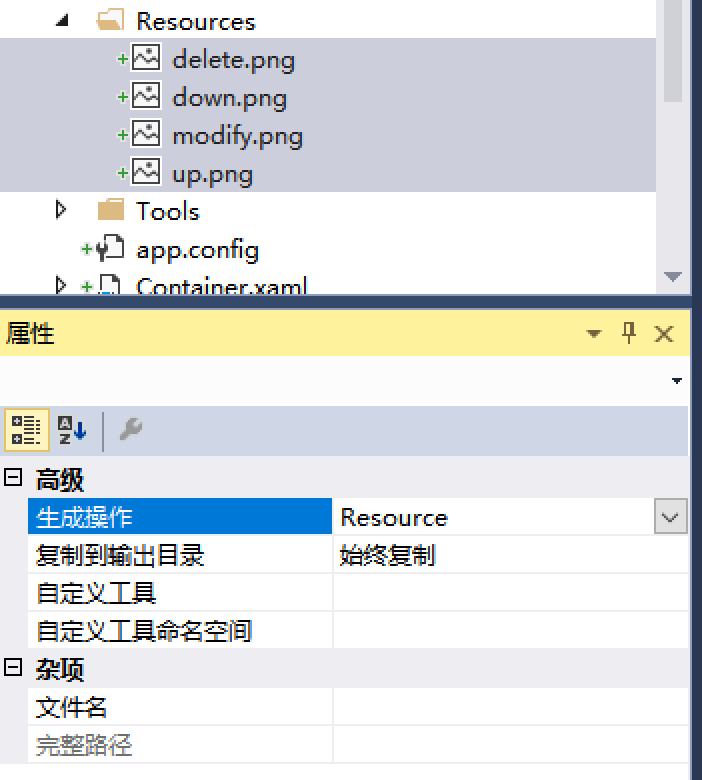




1. 菜单
2. 项目右键选择属性，切换到资源Tab，点击创建资源文件，创建完成以后，切换为图像资源，然后点击添加资源，选择添加现有文件，导入modify、delete、up、down四个Icon



1. 选择四个新增的Icon图片，修改复制到输出目录属性为始终复制，修改生成操作属性为Resource



1. 在GlobalStyle中新增menu\_rule定义

<!-- Rule菜单 -->

<ContextMenu x:Key="menu\_rule">

<MenuItem Header="修改">

<MenuItem.Icon>

<Image Source="Resources/modify.png" Width="16" Height="16"></Image>

</MenuItem.Icon>

</MenuItem>

<MenuItem Header="删除">

<MenuItem.Icon>

<Image Source="Resources/delete.png" Width="16" Height="16"></Image>

</MenuItem.Icon>

</MenuItem>

<Separator></Separator>

<MenuItem Header="上移">

<MenuItem.Icon>

<Image Source="Resources/up.png" Width="16" Height="16"></Image>

</MenuItem.Icon>

</MenuItem>

<MenuItem Header="下移">

<MenuItem.Icon>

<Image Source="Resources/down.png" Width="16" Height="16"></Image>

</MenuItem.Icon>

</MenuItem>

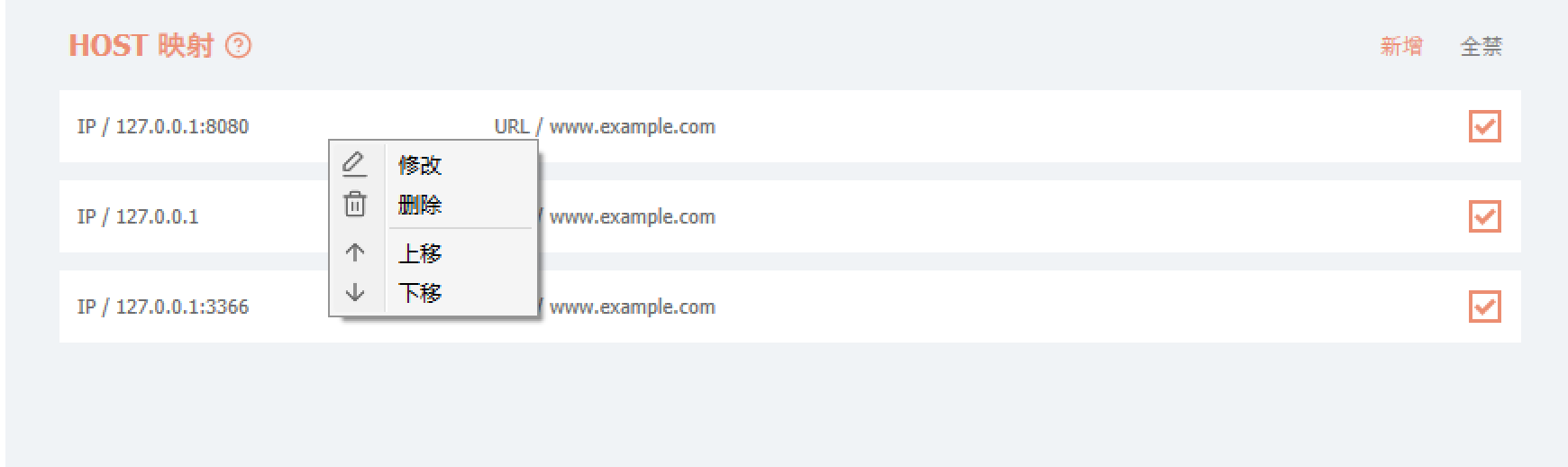
</ContextMenu>

1. 在GlobalStyle中给对应控件绑定ContextMenu属性

<Label Grid.Column="0" Tag="{Binding Path=Index}" MouseDoubleClick="modifyRule" Content="{Binding Path=IpAndPort}" ContextMenu="{StaticResource menu\_rule}" Template="{StaticResource content\_text}"></Label>

<Label Grid.Column="1" Tag="{Binding Path=Index}" MouseDoubleClick="modifyRule" Content="{Binding Path=TipsAndUrl}" ContextMenu="{StaticResource menu\_rule}" Template="{StaticResource content\_text}"></Label>

1. 打包预览



1. 菜单--修改
2. GlobalEvent类增加私有方法handleRuleMenuClick

#region 菜单点击事件

private void handleRuleMenuClick(object sender, RoutedEventArgs e)

{

string type = (sender as MenuItem).Tag.ToString();

object target = ((sender as MenuItem).Parent as ContextMenu).PlacementTarget as object;//获取点击源控件

int index = (int)(target as Label).Tag;

if (type == "modify")

{

modifyRule(target, null);

}

}

#endregion

1. 修改菜单Item绑定Click，并设置Tag为modify

<MenuItem Header="修改" Click="handleRuleMenuClick" Tag="modify">

<MenuItem.Icon>

<Image Source="Resources/modify.png" Width="16" Height="16"></Image>

</MenuItem.Icon>

</MenuItem>

1. 菜单--删除
2. Main类中添加公有静态方法deleteRuleByIndex

//删除Rule

public static void deleteRuleByIndex(int index)

{

//删除对应的数据

mainData.RemoveAt(index);

//遍历修改下标值

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

{

HostModel item = mainData[i] as HostModel;

item.Index = i;

}

//重新写入文件

DataTool.writeConfigToFile();

}

1. Container类中添加公有方法deleteRuleFromUI

//删除Rule控件

public void deleteRuleFromUI(int index)

{

this.host.Children.RemoveAt(index);

}

1. GlobalEvent类的handleRuleMenuClick增加delete逻辑

else if (type == "delete")

{

//删除数据

Main.deleteRuleByIndex(index);

//删除对应UI

Main.container.deleteRuleFromUI(index);

}

1. GlobalStyle中的删除菜单Item绑定Click属性，并设置Tag为delete

<MenuItem Header="删除" Click="handleRuleMenuClick" Tag="delete">

<MenuItem.Icon>

<Image Source="Resources/delete.png" Width="16" Height="16"></Image>

</MenuItem.Icon>

</MenuItem>

1. 菜单--上移/下移
2. Main类中添加公有静态方法moveRuleByType

//移动Rule

public static void moveRuleByType(int index, string moveType)

{

//第一个数据

if (index == 0 && moveType == "up")

{

return;

}

//最后一个数据

if (index == mainData.Count - 1 && moveType == "down")

{

return;

}

//移动数据

if (moveType == "up")

{

mainData.Insert(index - 1, mainData[index]);

mainData.RemoveAt(index + 1);

}

else

{

mainData.Insert(index, mainData[index + 1]);

mainData.RemoveAt(index + 2);

}

//遍历修改下标值

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

{

HostModel item = mainData[i] as HostModel;

item.Index = i;

}

//重新写入文件

DataTool.writeConfigToFile();

}

1. Container类中添加私有方法initRuleToUI

#region 私有方法(内部工具方法)

//初始化Rule面板

private void initRuleToUI()

{

ArrayList items = Main.mainData;

//遍历添加Rule到UI

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

{

addHostRule(items[i] as HostModel);

}

}

#endregion

1. Container类中添加公有方法moveRuleFromUI

//移动Rule控件

public void moveRuleFromUI(int index, string moveType)

{

if (index <= 0 && moveType == "up")

{

Fiddler.FiddlerApplication.DoNotifyUser("已在最顶部", "无法上移");

return;

}

StackPanel panel = this.host;

if (index == panel.Children.Count - 1 && moveType == "down")

{

Fiddler.FiddlerApplication.DoNotifyUser("已在最底部", "无法下移");

return;

}

//移除所有的Item

panel.Children.Clear();

//重新渲染所有的Item

initRuleToUI();

}

1. GlobalEvent类的handleRuleMenuClick增加up/down逻辑

else if (type == "up" || type == "down")

{

//移动对应的数据

Main.moveRuleByType(index, type);

//移动对应的UI

Main.container.moveRuleFromUI(index, type);

}

1. GlobalStyle中的上移/下移菜单Item绑定Click属性，并设置Tag为up/down

<MenuItem Header="上移" Click="handleRuleMenuClick" Tag="up">

<MenuItem.Icon>

<Image Source="Resources/up.png" Width="16" Height="16"></Image>

</MenuItem.Icon>

</MenuItem>

<MenuItem Header="下移" Click="handleRuleMenuClick" Tag="down">

<MenuItem.Icon>

<Image Source="Resources/down.png" Width="16" Height="16"></Image>

</MenuItem.Icon>

</MenuItem>