基于WPF+C#实现的MIB-2的管理系统

目录

[1.配置环境： 1](#_Toc153035952)

[2.运行展示： 1](#_Toc153035953)

[3.代码详解 3](#_Toc153035954)

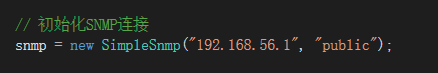
[3.1界面布局详解： 3](#_Toc153035955)

[3.2代码Get,Set等操作详解 5](#_Toc153035956)

[4.不足 8](#_Toc153035957)

1.配置环境：  
 1）Visual Studio中C#模块和WPF模块的安装。

2）本机正确的配置了SNMP协议。可通过修改MainWindow.xaml.cs中，进行对应IP和社区名称的配置。



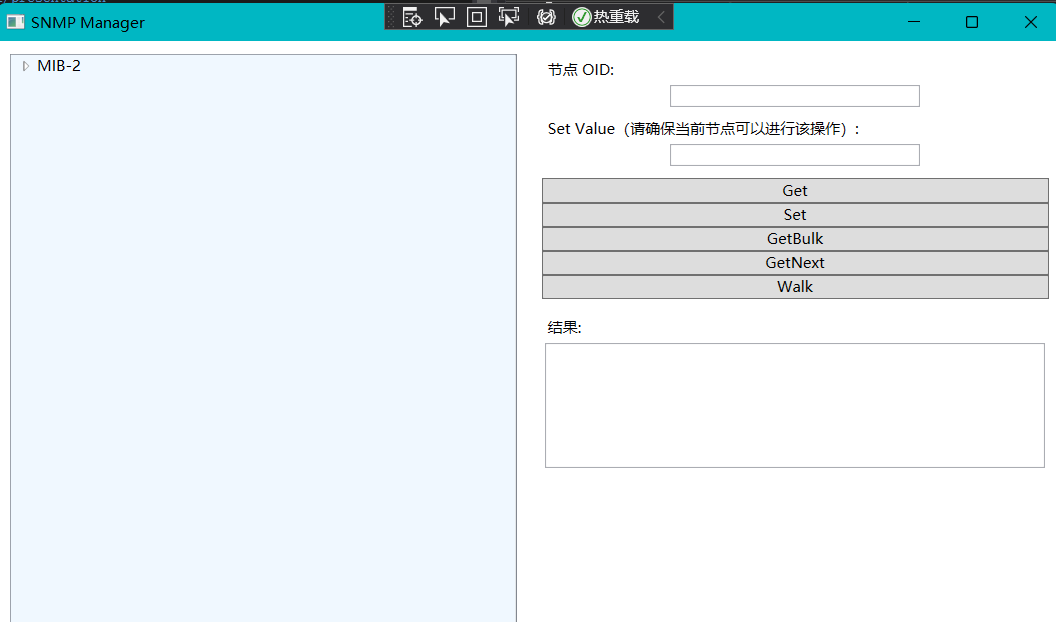
3）正确引入SnmpSharpNet模块，可以通过在vs中点击上方工具🡪Nuget包管理器—>程序包控制器管理台。输入下方命令

Install-Package SnmpSharpNet

4）点击运行即可。

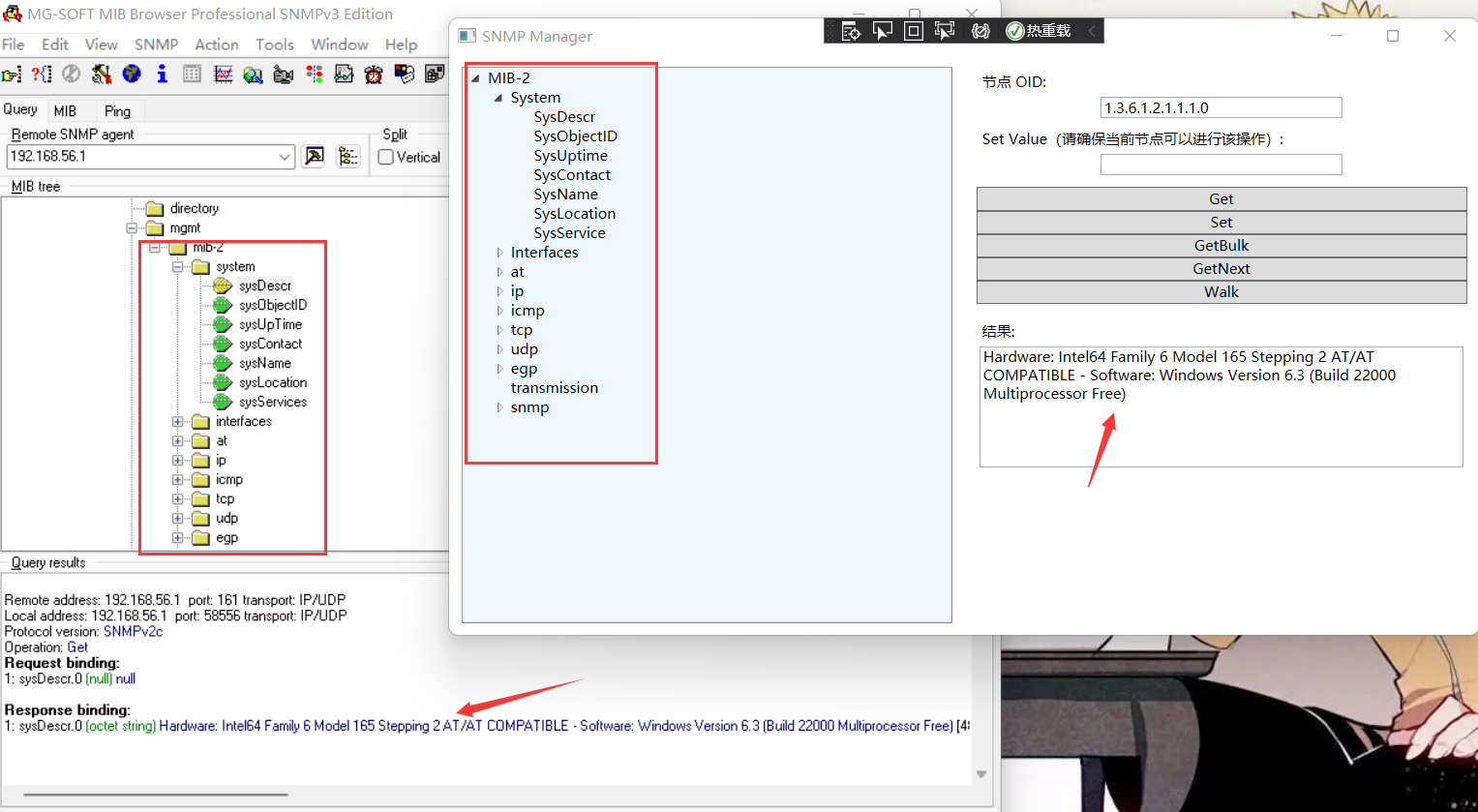
## 2.运行展示：

其中左边为树形结构，参考MIB-Broswer中的结构进行搭建。可以通过点击展开，找到对应的项，会在左边展示对应的Oid，进行右边的五项操作（确保节点可用）。会在结果栏中给出运行结果。

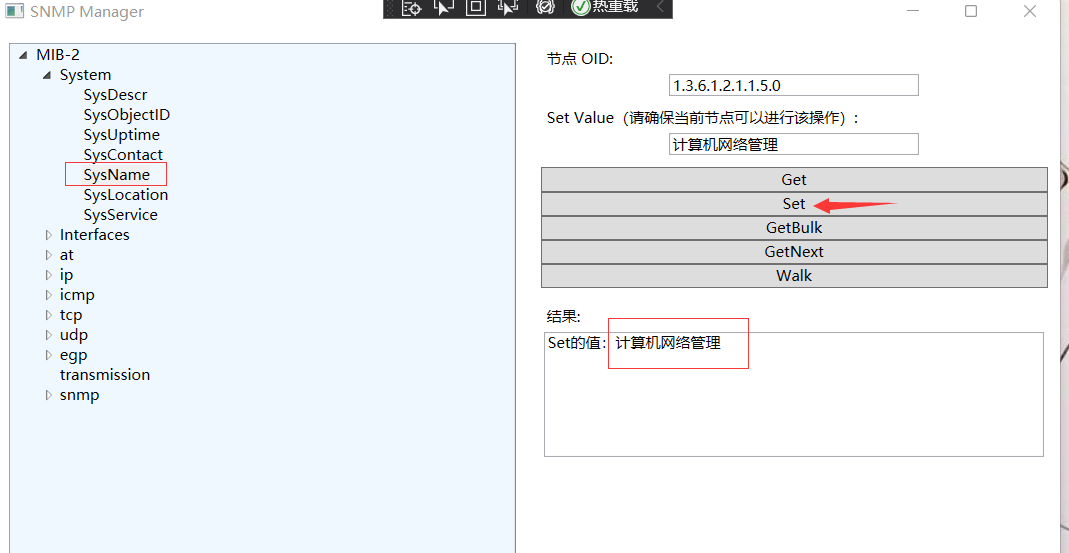


接下来举一些例子：

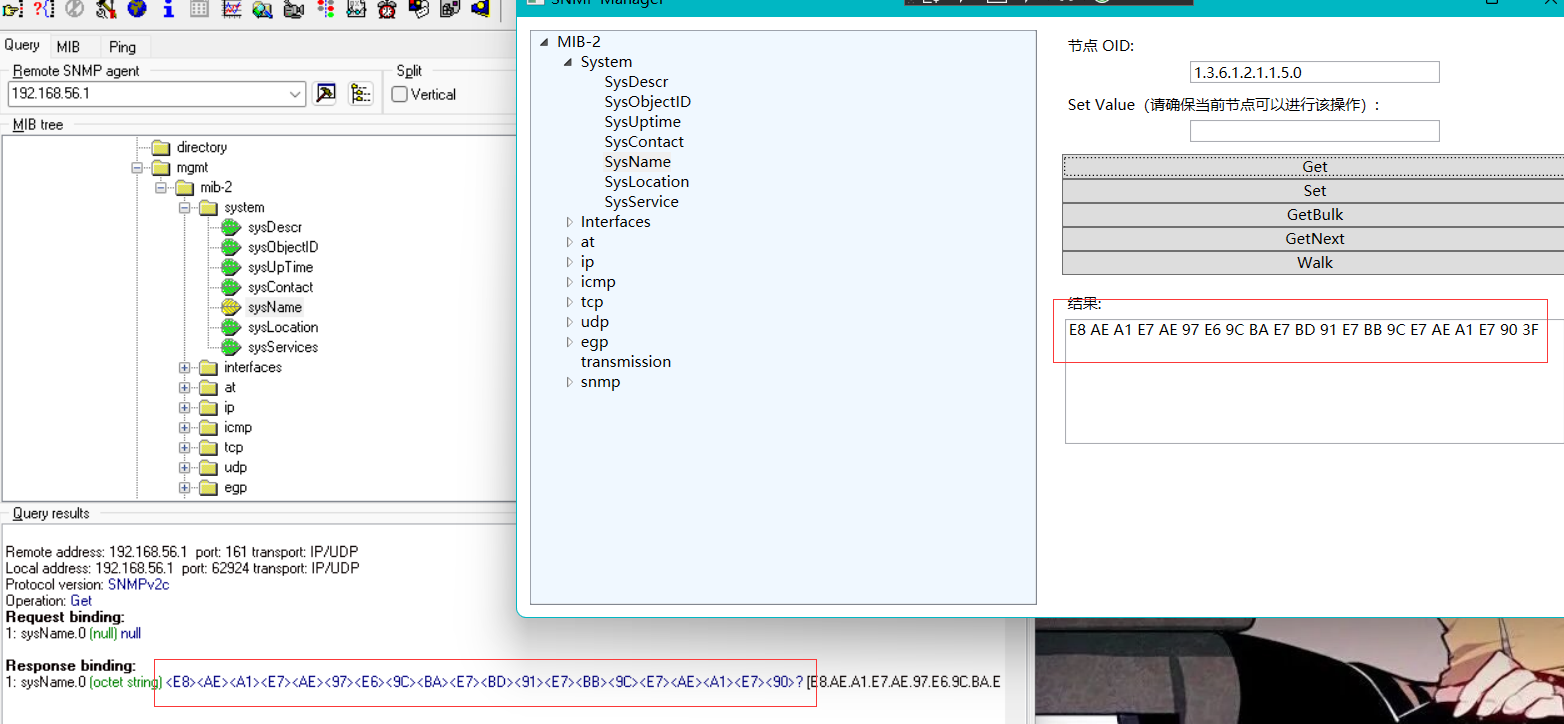
1.Get操作：通过与MG-soft进行比较，结果无误



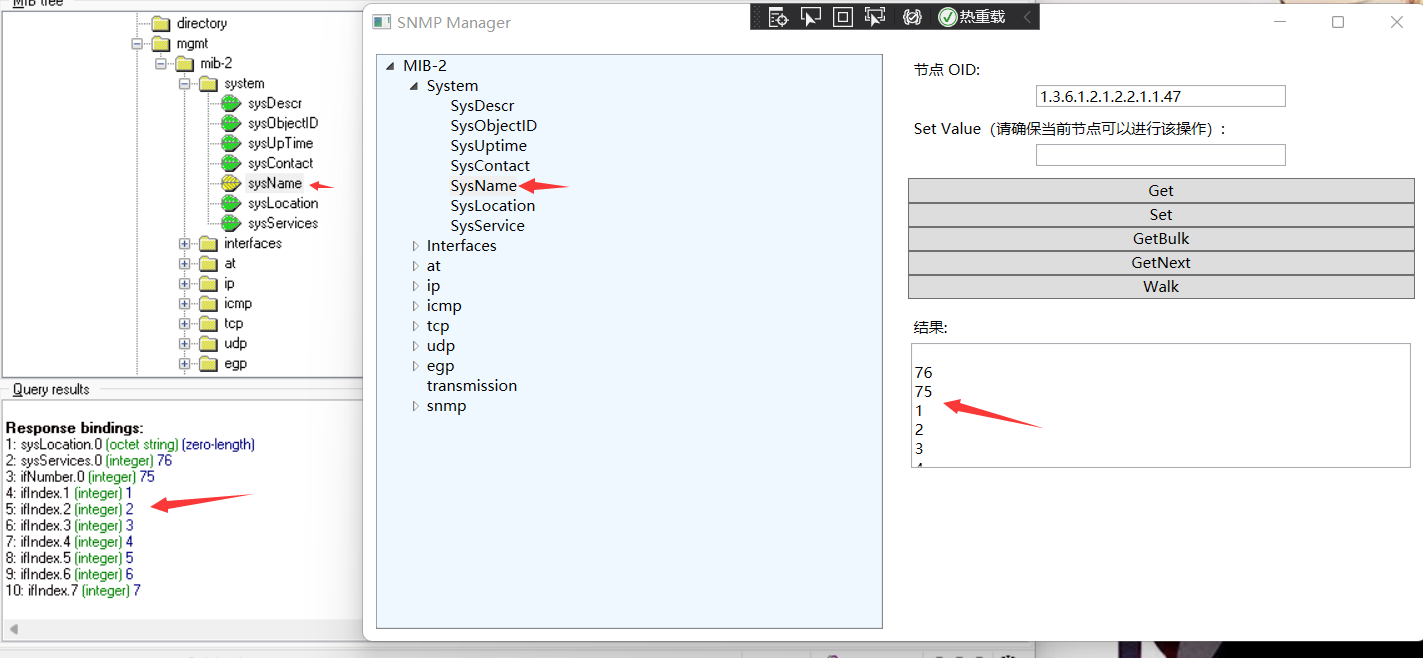
2.Set操作：



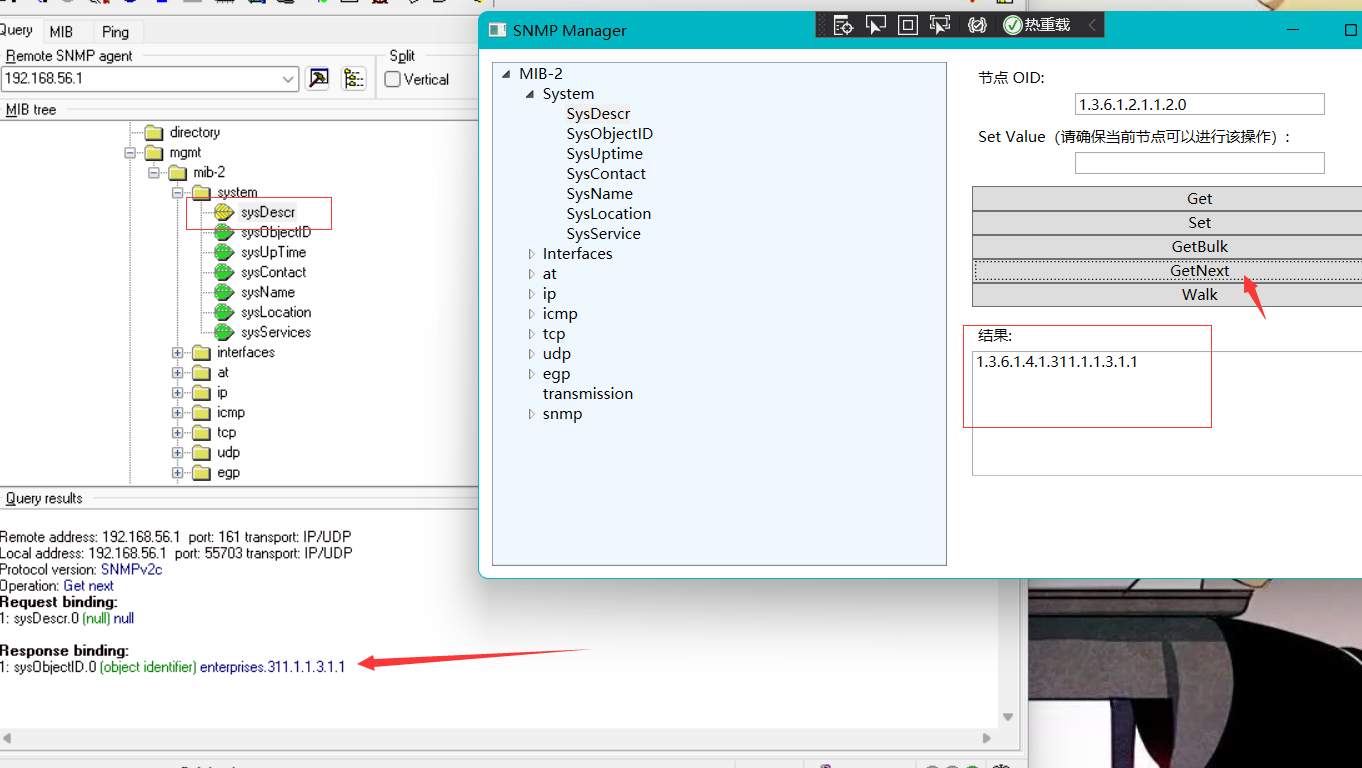
Set后我们再次get,查看是否成功



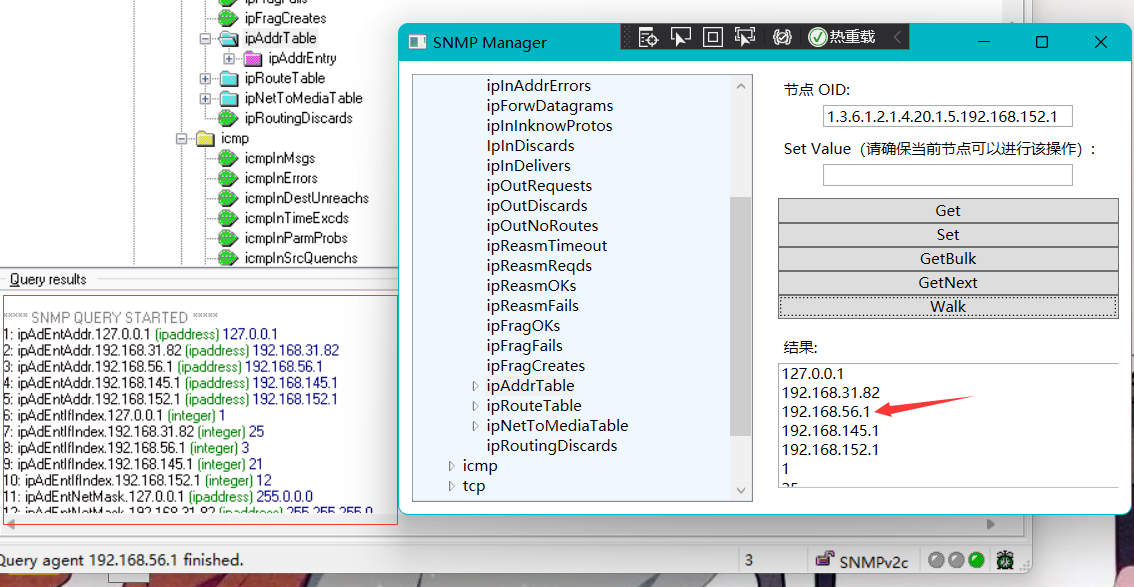
3.GetBulk操作（其中GetBulk操作是针对网络，获取大量数据。会对Oid进行后移，所以Oid会发生改变）



4.GetNext操作：



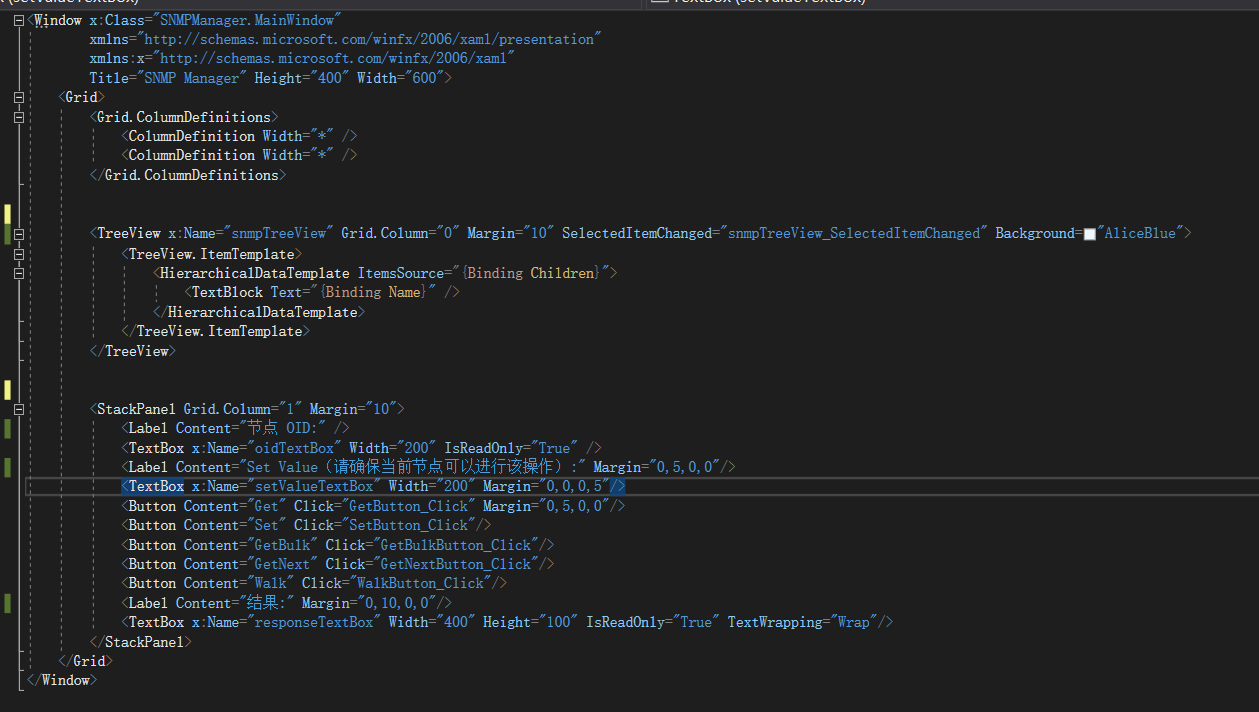
5.Walk操作：（以ip为例）



## 3.代码详解

### 3.1界面布局详解：

1)整体由Grid左右划分，左半部分是一个TreeView的组件，这也是界面布局中的核心。右半部分只有一些简单的button和Text文本展示，增加了一些点击事件，不进行过多的解释。



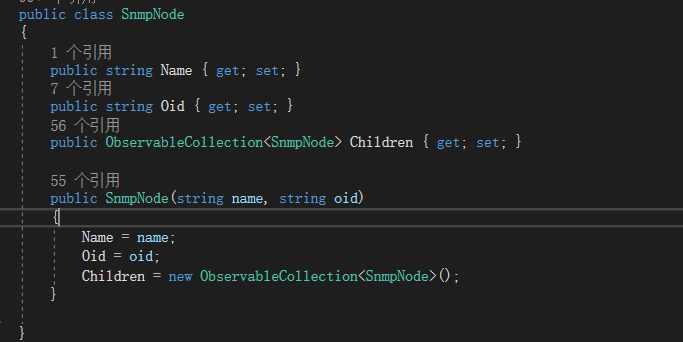
基于MIB中的数据结构，采用树形是最好的选择，自己之前使用了下拉框的组件，使用起来过于繁琐。通过在网上对树形组件的学习，进行顺利的搭建。其中，有静态加入和动态加入两种方法，由于节点数过多，采用动态加入更加适合。

**在本项目的组件中：**

SelectedItemChanged="snmpTreeView\_SelectedItemChanged"：指定当 TreeView 中的选中项发生变化时要触发的事件处理程序。

<HierarchicalDataTemplate ItemsSource="{Binding Children}">：这个模板，允许每一个节点增加子项。模板定义好，之后更重要的是在代码中对节点进行添加管理：

第一步先写一个SnmpNode的类，这个类用于添加节点（自己之前考虑只做SNMP,命名可能有一些问题）



1.这个类中有三个成员变量，其中Children变量是一个ObservableCollection<SnmpNode>类型，这个类型是.Net中的一个集合类发生更改时会触发事件的动态数据集合。

2.构造函数：构造函数接收节点的名称和 OID 作为参数，初始化 Name 和 Oid 属性，并实例化 Children 集合。

整个方法结构主要参考关于Tree组件的使用说明文档，接下来我们来看一下如何进行节点的增加：

SnmpNode root = new SnmpNode("MIB-2", "1.3.6.1");

SnmpNode systemNode = new SnmpNode("System", "1.3.6.1.2.1.1");

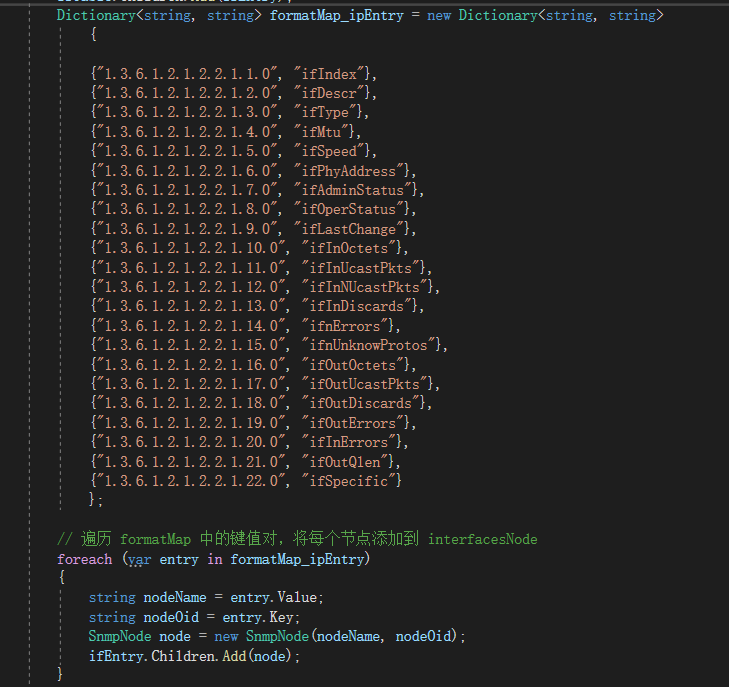
root.Children.Add(systemNode);

// 将树形结构绑定到TreeView

snmpTreeView.ItemsSource = new ObservableCollection<SnmpNode> { root };

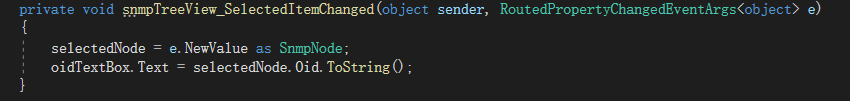
首先先new一个root根节点和System节点。然后通过Children.Add将子节点加入到root节点下面。最后将root挂到Tree上即可。所有的添加流程都是这样的一个方法，只是逻辑上需要注意。

由于部分的节点下，子节点过多，自己写了一个方法，对节点进行循环增加。通过字典，对数据进行循环迭代增加节点即可。（尽管通过这种方法，可以有效减少代码量，但其中字典数据的构建仍然需要大量的精力，去匹配对应的Oid和节点信息。自己想在网上直接搜索对应的Oid规定文档，但是一直没找到能用的合适的，只能自己构建一个）



当把所有的节点加入完毕后，便可以显示出WPF的正确布局页面。

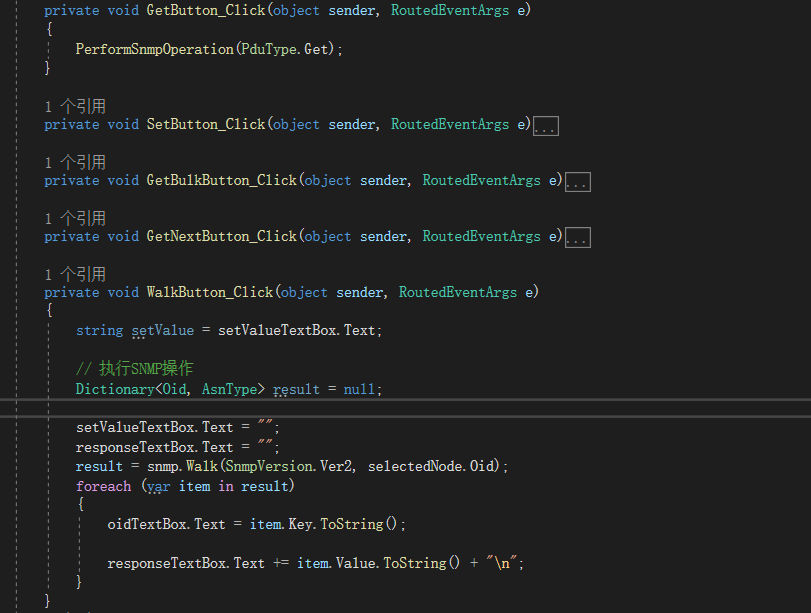
最后的一个snmpTreeView\_SelectedItemChanged，这个方法用于获取当前选择了哪一个节点，并把该节点的Oid展示到前端Text里。



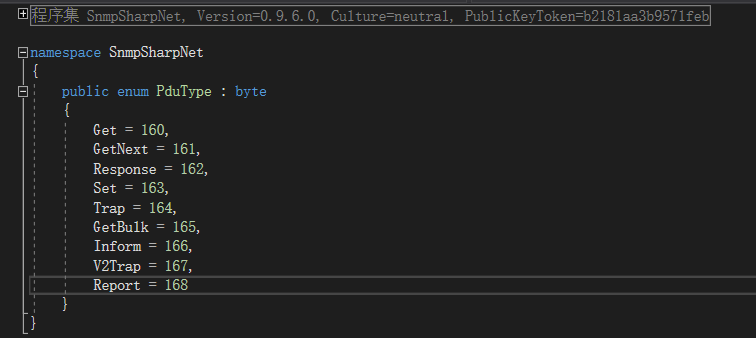
### 3.2代码Get,Set等操作详解

首先，我们要明白整个项目中运行的流程。

1.WPF中用户选择节点，给出对应的Oid。点击对应的Button，触发事件，展示Oid和结果。Button事件对应是下面的五个。他们都会去调用一个PerformSnmpOperation的方法，这个方法接收一个PduType类型的值，通过这个值去判断进行的是什么操作。（PduType，这个类型中，没有Walk类型，下面源码中有展示。自己便直接把Walk操作放到点击事件中了。也可以使用重载实现，操作不多，自己就直接写进去了）

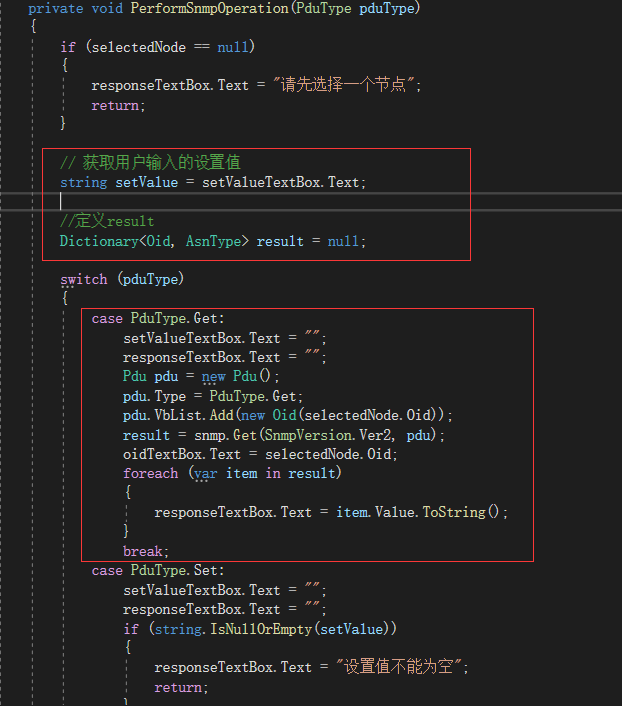


其中，PduType是一个枚举类型的值，这是通过按住Ctrl+点击，去查看官方文档的定义



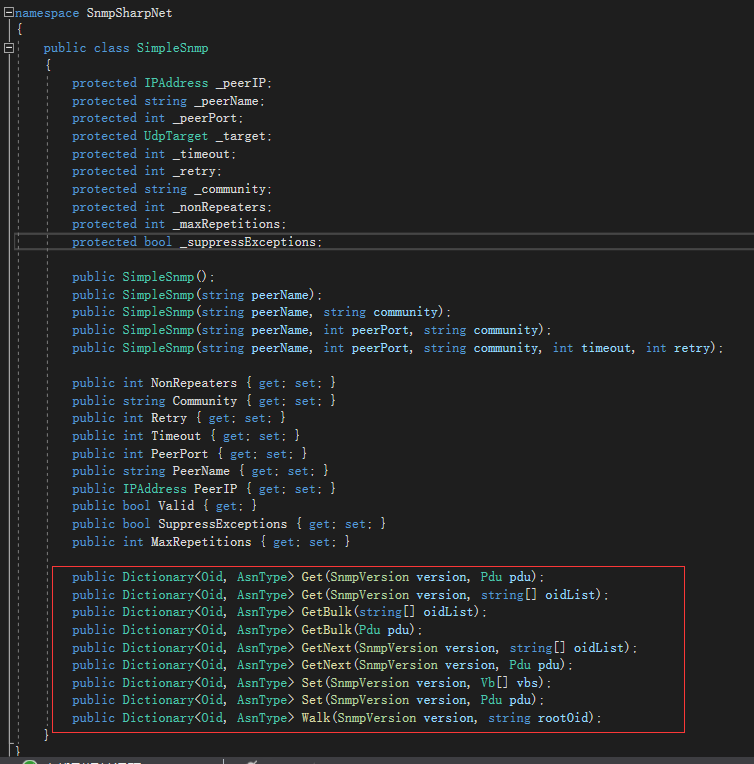
**接下来我们详细介绍PerformSnmpOperation：**

**先以Get为例：**

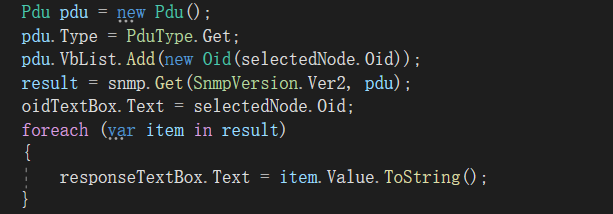


**Snmp的初始化是**private SimpleSnmp snmp;通过这句代码，调用库，创建的对象。

setValue是从WPF里获取到的数据，**其中，result为什么要定义成这个类型，还是一样的方法，查看官方源码（crtl+点击。Snmp.Get），这个源码会根据版本不同有一些区别。我们不是要只会呆板的写，重要的是查看为什么要这么接收参数，传递参数，这个对象都有什么方法，这才是最重要的。**



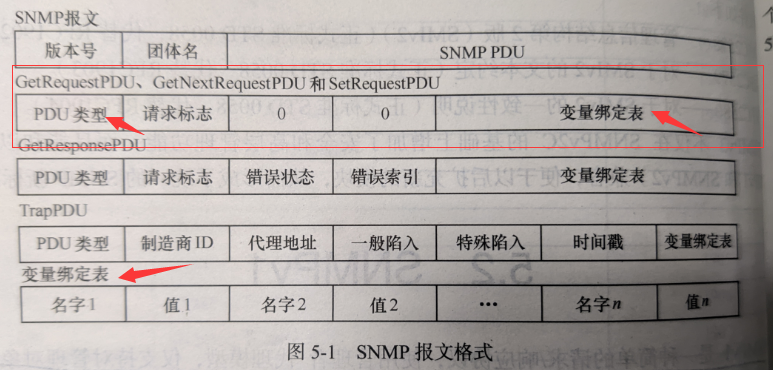
源码中清晰给出，Simplesnmp中有这些方法，其中Get,有两个重载，要么接收version,也就是版本，另外是一个pdu类型的数据。或者，接收一个OidList的string数组。其中会返回一个Dictionary类型的数据，这个数据包含了Oid和AsnType。这也是为什么我们要设置result为呢样的类型。



因此，我们需要先new一个Pdu对象。设置其类型（Type）为Get。然后Add一个VbList,也就是变量绑定表，之后便可以进行Get操作，返回给result。再通过foreach进行遍历展示即可。

**【那么为什么new出Pdu后要进行这些步骤呢？，我们可以结合课本内容查看】**

报文中指出了报文的结构，其中我们需要对类型和变量绑定表（VbList）进行添加Oid，但是此时我们只给出了名字1。等数据传递回来后，我们便能获取到值1.。然后就可以使用遍历，在对应的地方进行内容的展示。



**后面的Set,GetBulk,GetNext也是一样的方法。注意查看源码，确定对应需要传递的参数。其中针对Walk操作，由于我们之前查看，pdu枚举类型中，没有walk,所以需要单独拿出来。**

**至此，整个代码的介绍也就结束了**

## 4.不足

**当然代码还有一些bug,针对一些Oid的非法操作处理不够，信息提示不足，通过Text展示布局比较混乱。自己会继续不断优化改善**