【OPCAutomation】使用OPCAutomation实现对OPC数据的访问

折腾了一段时间研究OPC，理清了下位机、OPCServer 和OPCClient的关系和通信模型，终于能够来写一篇相关的博客了。

我们使用西门子的 S7 200 SMART作为下位机端，通过3G路由器从vpn与公司服务器通信，服务器运行配置好的PC Access SMART 作为OPC Server， 完成对下位机内存地址的定义后，使用自动化接口开发中间件负责将OPC Server得到的PLC数据存放至SQL Server

中间件和数据库的设计思路是：

数据库按真实设备类型分别建表用作存储，数据库建有数据字典表用作配置功能，包括配置内存地址和opc服务等。

中间件作用是调用OPCAutomation 类访问OPCServer端，并进行可控制读取间隔的OPC数据读取、存储工作。

由于第一次接触这类开发，在设计功能时花费了很多精力，从完全不懂到基本理解这种通信模式和原理，也绕了不少远路。

本文主要介绍OPCAutomation类的使用。

简单说下流程就是：

1. 创建OPC Server的连接

2. 创建OPC组对象并初始化设置

3. 获取组的OPCItems对象，为读取数据作准备

4. opcItem的操作。

5. 退出程序的资源释放

创建连接很简单，需要指定OPCServer所在的服务器（内网可以指定ip或者计算机名），指定OPC服务的名称（同一服务器可能运行多个OPC服务以适配同的下位机）

```
private bool ConnectRemoteServer(string remoteServerIP, string remoteServerName)
{
    try
    {
        this.opcServer.Connect(remoteServerName, remoteServerIP);
        string status;
        if (opcServer.ServerState == (int)OPCServerState.OPCRunning)
        {
            status = "已连接到-" + opcServer.ServerName + "    ";
        }
        else
        {
            //这里你可以根据返回的状态来自定义显示信息，请查看自动化接口API文档
            status = "状态：" + opcServer.ServerState.ToString() + "    ";
        }
        Console.WriteLine(status);
    }
    catch (Exception err)
    {
        Console.WriteLine("连接远程服务器出现错误：" + err.Message, "提示信息");
        return false;
    }
    return true;
}
```

View Code
其中 this.opcServer.Connect(remoteServerName, remoteServerIP);即OPCAutomation类提供的连接方法。

需要注意的是，在实际配置时，需要完全对OPC 服务端所在服务器上配置防火墙出入站规则后，OPC服务才能够被其他服务器上的中间件访问到。故我们选择最简单的方式，在本机运行中间件。

创建组相当于读取到OPC上特定的项目，而具体的数据值是在每个项目下根据开发人员的定义而确定。

```
opcGroups = opcServer.OPCGroups;
opcGroup = opcGroups.Add("OPCDOTNETGROUP");
SetGroupProperty();
opcGroup.DataChange += new DIOPCGroupEvent_DataChangeEventHandler(opcGroup_DataChange);

//opcGroup.AsyncWriteComplete += new DIOPCGroupEvent_AsyncWriteCompleteEventHandler(KepGroup_AsyncWriteComplete);

opcItems = opcGroup.OPCItems;
```

注释的代码是绑定写操作的事件。

这段代码是为OPCGroup对象进行初始化。 比较关键的是两句绑定事件的，第一个绑定的是每当OPC数据有变化时触发的事件。

数据变化的时间是由以下代码中UpdateRate控制。

```
private void SetGroupProperty()
{
    opcServer.OPCGroups.DefaultGroupIsActive = true;
    opcServer.OPCGroups.DefaultGroupDeadband = 0;
    opcGroup.UpdateRate = this.updateRate;
    opcGroup.IsActive = true;
    opcGroup.IsSubscribed = true;
}
```

View Code
关于逐项配置的具体说明，建议参考OPCAutomation的api，如果有看到翻译比较合适的后续会补充上来。

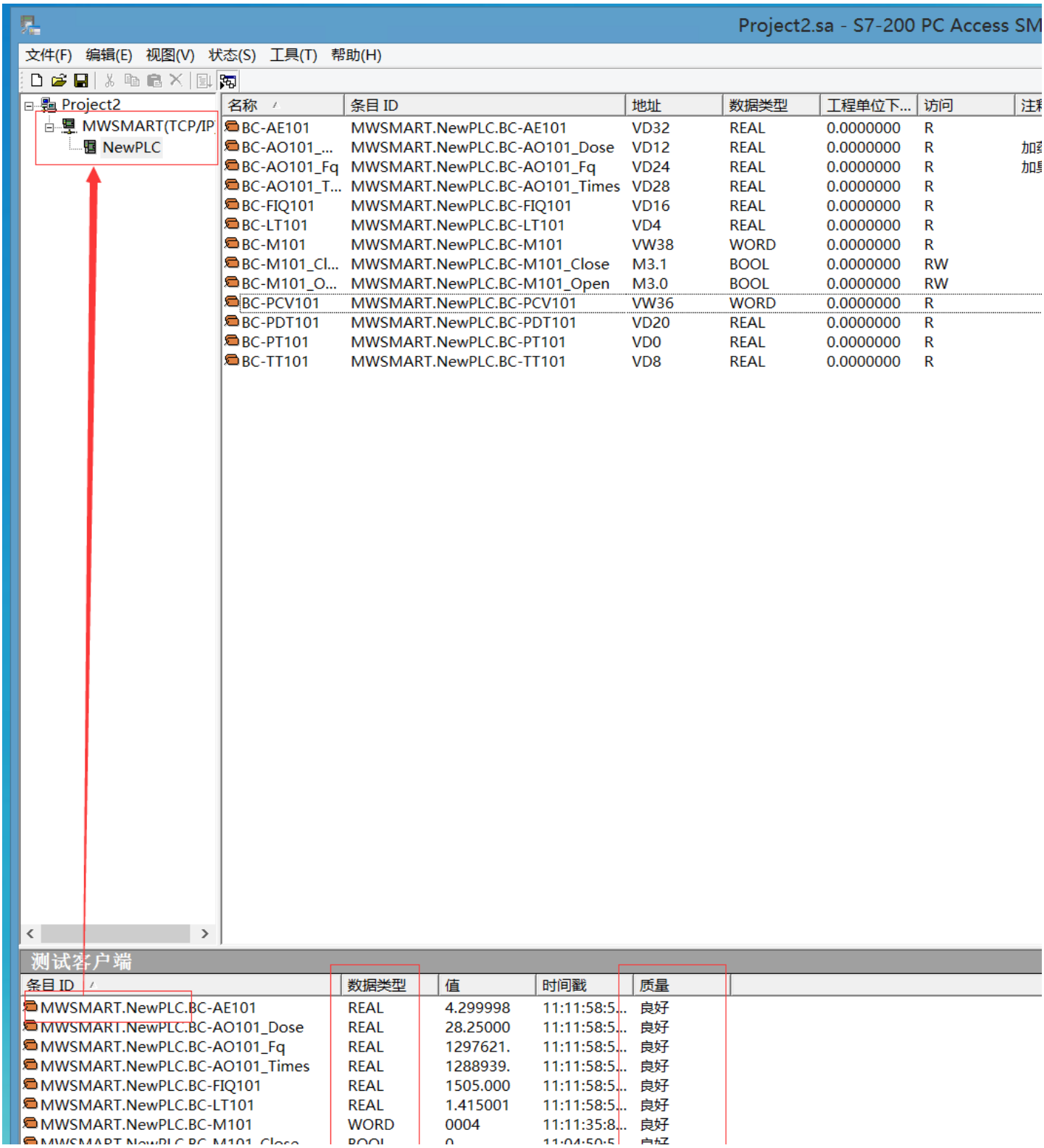
之后是为全局变量里的OPCGroup对象添加Items，在OPC中，每个opcItem会被分配一个客户端句柄值，同时会被配置上服务端句柄。几乎所有有关获取指定OpcItem对象的方法均要求指定客户端句柄值。

```
void AddAllOpcItem()
{
    opcBrowser.ShowBranches();
    opcBrowser.ShowLeafs(true);
    int count = this.opcBrowser.Count;
    if (this.opcItemsArray == null)
    {
        opcItemsArray = new List<string>();
    }
    foreach (var item in opcBrowser)
    {
        opcItemsArray.Add(item.ToString());
    }
    AddOpcItems();
}
//逐项绑定句柄值
void AddOpcItems()
{
    foreach (var item in this.opcItemsArray)
    {
        itmHandleClient = 1234;
        opcItem = opcItems.AddItem(item, itmHandleClient);
        itmHandleServer = opcItem.ServerHandle;
    }
}
```

这段代码是为了把获取到的全部OPCItem添加到所创建的OPCGroup对象的opcItem集合中，以便我们所绑定的DataChange事件能够被触发并且正确对应到每个地址值上。

基本上到此，只要在所绑定的DataChange的实现代码中完成具体的数据读取业务，中间件的核心部分已经完成。

稍微提一下，OPC的数据项主要包含：名称、条目id、地址、数据类型、数据值、工程单位上下限、时间戳和质量几项，在实际业务中我们关注条目ID、数据类型、值、时间戳和质量，对于读取的opc数据值而言，需要注意的是得到值是Dynamic类型的，需要正确进行判断和转换。质量返回值为0时基本可以认为下位机到服务端的数据链断了，或者下位机对应项目没有收到数据，据此可进行日志和提示的业务操作。



在OPC的数据类型中，REAL对应float， WORD对应16位整型，BOOL可以用bit或者Bool进行转换，用bit和sql server的数据项可以比较方便对接。

第一次做这类开发，谬误难免，希望多多指点。

感谢阅读。