目　　录

官方网站：http://www.bmpj.net

15．数据持久化接口的使用

15.1 概述

ServerSuperIO 3.2 版本以前，设备数据仅支持 Xml 序列化的方式，如果以其他方式存储数据，那么只能把持久化操作写在设备驱动中，本质上失去了模块化的灵活性。3.2 版本以后增加了数据持久化接口，方便支持多种形式存储设备的参数数据和实时数据，3.2 版本里现在仅支持 Xml 序列化的方式，后期会支持 MongoDB、influxdb 和实时数据库（类似 PI）等

15.2 定义 IDataPersistence 数据持久化接口

1.接口定义

凡是涉及到不同数据存储方式的都可以继承 IDataPersistence 接口，具体代码定义发下：

```
public interface IDataPersistence
    {
        /// <summary>
        /// 连接数据源
        /// </summary>
        void Connect();

        /// <summary>
        /// 选择数据对象
        /// </summary>
        /// <param name="devid"></param>
        /// <param name="objType"></param>
        /// <returns></returns>
        object Select(string devid, Type objType);

        /// <summary>
        /// 插入数据对象
        /// </summary>
        /// <param name="devid"></param>
        /// <param name="obj"></param>
```

```
        void Insert(string devid, object obj);

        /// <summary>
        /// 更新数据对象
        /// </summary>
        /// <param name="devid"></param>
        /// <param name="obj"></param>
        void Update(string devid, object obj);

        /// <summary>
        /// 删除数据对象
        /// </summary>
        /// <param name="devid"></param>
        /// <param name="obj"></param>
        void Delete(string devid,object obj);

        /// <summary>
        /// 判断是否存在
        /// </summary>
        /// <param name="devid"></param>
        /// <param name="obj"></param>
        /// <returns></returns>
        bool Exist(string devid,object obj);
    }
```

2.接口实现形式

如果以 MongoDB、influxdb 和实时数据库，那么都可以继承 IDataPersistence 接口，下面以 Xml 存储数据为例，代码如下：

```
public class XmlPersistence:IXmlPersistence
    {
        public void Connect()
        {

        }

        public object Select(string devid, Type objType)
        {
            string path = GetSavePath(devid,objType);
            return SerializeUtil.XmlDeserailize(path, objType);
        }

        public void Insert(string devid, object obj)
```

```
{
    string path = GetSavePath(devid,obj.GetType());
    SerializeUtil.XmlSerialize(path, obj);
}

public void Update(string devid, object obj)
{
    this.Insert(devid, obj);
}

public void Delete(string devid,object obj)
{
    string path = GetSavePath(devid,obj.GetType());
    if (System.IO.File.Exists(path))
    {
        System.IO.File.Delete(path);
    }
}

public bool Exist(string devid, object obj)
{
    string path = GetSavePath(devid, obj.GetType());
    return System.IO.File.Exists(path);
}


public string GetSavePath(string devid,Type type)
{
    string path = AppDomain.CurrentDomain.BaseDirectory;
    if (typeof(IDeviceDynamic).IsAssignableFrom(type))
    {
        path = System.IO.Path.Combine(path, "ServerSuperIO/Dynamic");
    }
    else if(typeof(IDeviceParameter).IsAssignableFrom(type))
    {
        path = System.IO.Path.Combine(path, "ServerSuperIO/Parameter");
    }

    if (!System.IO.Directory.Exists(path))
    {
        System.IO.Directory.CreateDirectory(path);
    }
    return System.IO.Path.Combine(path, devid + ".xml");
```

```
        }
    }
```

15.3 使用 IDataPersistence 数据持久化接口

ServerSuperIO 框架二次开发的设备驱动涉及到两类数据：设备参数和实时数据。这两类数据都对应着基类：DeviceParameter 和 DeviceDynamic。每个设备驱动可能对应着不同的设备参数（系数等）和实时数据（温度、湿度、流量等），因为涉及到不同的硬件设备和传感器。可以分别继承 DeviceParameter 和 DeviceDynamic 两个基类，这两个基类都具有 InitDataPersistence(IDataPersistence dataPersistence)接口，实现不同的数据持久化存储接口，并进行 Save、Load 和 Delete 操作。

具体定义方法，如下代码：

```
public class DeviceDyn:DeviceDynamic
{
        public DeviceDyn() : base()
        {
             this.InitDataPersistence(new XmlPersistence());
        }

        public override string GetAlertState()
        {
            throw new NotImplementedException("无报警信息");
        }

        public float Flow{set;get;}
}
```

具体使用方法，如下代码：

```
        DeviceDyn dyn=new DeviceDyn();
        dyn.Save();  //保存
        dyn.Load();  //加载
        dyn.Delete();//删除
```

注：this.InitDataPersistence()不进行此操作，默认使用 XmlPersistence 进行数据持久化。