周公（周金桥）的专栏

光荣在于平淡，艰巨在于漫长。请文明用语，勿人身攻击。技术更新快，看一年以上旧文时注意新技术动向。

个人资料

周公

.net中的WMI编程（一）：WMI介绍及简单应用

分类： C#基础 2008-01-14 02:48 11389人阅读 评论(18) 收藏 举报

前言： 由于.net的某些特点使它脱离了操作系统底层，所以如果我们想获得一些底层的操作系统信息的时候比较困难，经常需要DllImport技术，这对一些对C/C++不熟悉的程序员来说简直是一个梦魇（对于我来说也是如此，虽然大学里学习过C/C++，可是我从来没有用过它们做过任何实际的应用程序开发），所以能通过一些简单的办法处理的，我都绕过去了，很多时候我喜欢借助命令行来执行一些命令，然后截取输出流，处理之后返回我需要的结果，这个也是比较痛苦的。经常访问我的博客的朋友应该对我的一篇文章很熟悉，或者从别的地方看到过，因为它被众多网站转载了，这篇文章就是《用C#获取CPU编号、硬盘编号等系统有关环境、属性》，网址是：http://blog.csdn.net/zhoufoxcn/archive/2007/03/20/1534949.aspx。

积分： 21254分
排名： 第63名

原创： 242篇
转载： 76篇
译文： 5篇
评论： 6042条

其实，还有一个为大家所陌生、但功能却又十分强大的工具，它就是WMI。WMI（Windows管理规范：Windows Management Instrumentation）是Microsoft基于Web的企业管理（WBEM）的实现，同时也是一种基于标准的系统管理接口。WMI最早出现在Microsoft Windows 2000系统上，但它同样可以安装在Windows NT 4和Windows 9x计算机上。WMI是一种轻松获取系统信息的强大工具。利用它但是由于缺少WMI的介绍资料和学习资料（特别是中文的，因为国内大部分程序员的E文水平都比较凑合，呵呵，本人也是相当凑合），所以知道WMI的不多，能运用的就少之又少了。

在WMI中有一种查询语言，类似于SQL语言，这种语言叫做WQL(WMI Query Language)，实际上是标准SQL的一个子集加上了WMI的扩展。在接下来的教程中我会给大家演示一些WQL的例子。

用js或者vbs都可以通过WMI来获取系统信息。下面是一个用vbs编写的获取系统序列号的小程序，你可以把它复制到一个文本文件里，然后保存为文件为.vbs的文件，然后运行：

```
Set SNSet = GetObject("winmgmts:").InstancesOf ("Win32_OperatingSystem")

for each SN in SNSet

 MsgBox "当前操作系统的序列号是: " & SN.SerialNumber

next
```

这是在我的机子上运行的结果：

如果通过常规手段在.net里让你写这个方法不知道你需要多少行代码？需要利用几次搜索引擎？一会我将展示如何在.net里利用WMI获取操作系统序列号。

其实在.net里是支持WMI编程的，在.net类库里专门有一个System.Management 命名空间，MSDN介绍这个命名空间如下：

“提供对大量管理信息和管理事件集合的访问，这些信息和事件是与根据 Windows 管理规范 (WMI) 结构对系统、设备和应用程序设置检测点有关的。应用程序和服务可以使用从 ManagementObjectSearcher 和 ManagementQuery 派生的类，查询感兴趣的管理信息（例如在磁盘上还剩多少可用空间、当前 CPU 利用率是多少、某一应用程序正连接到哪一数据库等等）；或者应用程序和服务可以使用 ManagementEventWatcher 类预订各种管理事件。这些可访问的数据可以来自分布式环境中托管的和非托管的组件。”。

System.Management 命名空间包含了下面一些常用类：

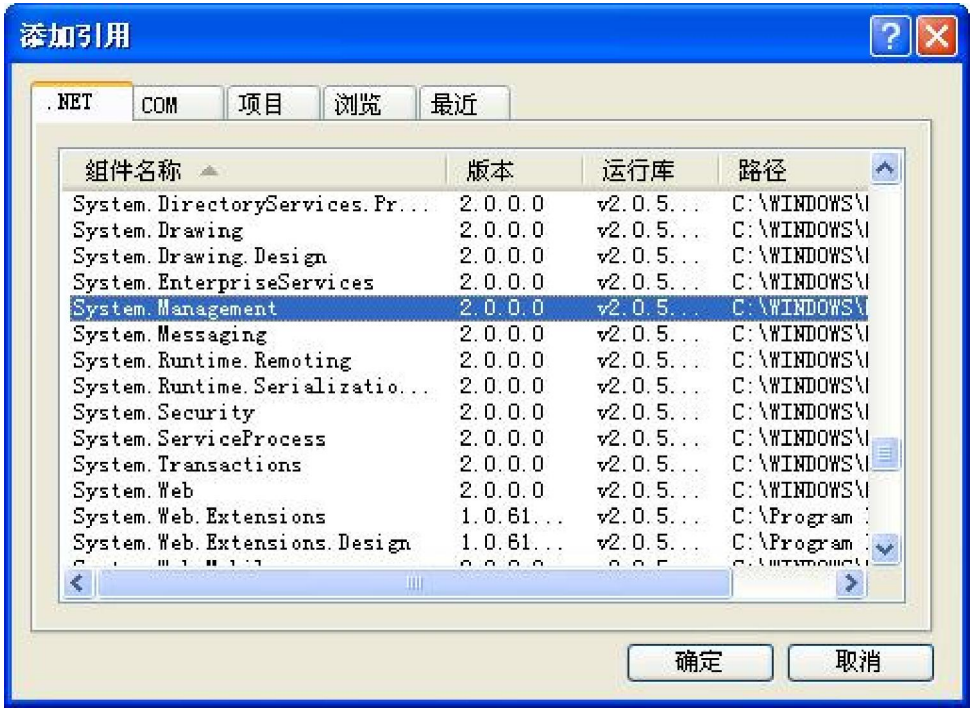
◆ManagementObject 或 ManagementClass：分别为单个管理对象或类。

◆ManagementObjectSearcher：用于根据指定的查询或枚举检索 ManagementObject 或 ManagementClass 对象的集合。

◆ManagementEventWatcher：用于预订来自 WMI 的事件通知。

◆ManagementQuery：用作所有查询类的基础。

在实际编程中需要注意，System.Management 命名空间中的类都存在于System.Management .dll这个文件中，所以在编程的时候一定要添加对这个dll文件的引用，如下图：



然后还需要在我们的程序开始处添加如下代码：System.Management，这样我们采用使用这个命名空间下的类来进行WMI编程。

下面我将用一个方法展示我刚才提到的如何在.net里获取操作系统序列号：

```
/// <summary>
/// 获取操作系统序列号
/// </summary>
/// <returns></returns>
public string GetSerialNumber()
{
    string result = "";
```

```
        ManagementClass mClass = new ManagementClass("Win32_OperatingSystem");
        ManagementObjectCollection moCollection = mClass.GetInstances();
        foreach (ManagementObject mObject in moCollection)
        {
            result += mObject["SerialNumber"].ToString();
        }
        return result;
    }
```

我想这个方法够简单的了。

下面贴出一些常用的方法来获取系统相关的信息，代码如下：

```
using System;
using System.Collections.Generic;
using System.Text;

namespace WMIDemo
{
    /// <summary>
    /// 说明：这个类很简单，主要包含一个Main方法
    /// 作者：周公
    /// 日期：2008-1-14
    /// 首发地址：http://blog.csdn.net/zhoufoxcn
    /// </summary>
    class Program
```

```
    {
        static void Main(string[] args)
        {
            GetSystemInfo getInfo = new GetSystemInfo();
            Console.WriteLine("序列号="+getInfo.GetSerialNumber());
            Console.WriteLine("CPU编号=" + getInfo.GetCpuID());
            Console.WriteLine("硬盘编号=" + getInfo.GetMainHardDiskId());
            Console.WriteLine("主板编号=" + getInfo.GetMainBoardId());
            Console.WriteLine("网卡编号=" + getInfo.GetNetworkAdapterId());
            Console.WriteLine("用户组=" + getInfo.GetGroupName());
            Console.WriteLine("驱动器情况=" + getInfo.GetDriverInfo());
            Console.ReadLine();
        }
    }
}
```

由于是一个命令行程序，所以有上面那个类，下面这个类才包含我要展示的代码：

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Management;

namespace WMIDemo
{
```

```
/// <summary>
/// 说明：这个类主要用来展示如何利用WMI来获取一些操作系统的信息
/// 作者：周公
/// 时间：2008-1-14
/// 首发地址：http://blog.csdn.net/zhoufoxcn
/// </summary>
public class GetSystemInfo
{
    /// <summary>
    /// 获取操作系统序列号
    /// </summary>
    /// <returns></returns>
    public string GetSerialNumber()
    {
        string result = "";
        ManagementClass mClass = new ManagementClass("Win32_OperatingSystem");
        ManagementObjectCollection moCollection = mClass.GetInstances();
        foreach (ManagementObject mObject in moCollection)
        {
            result += mObject["SerialNumber"].ToString() + " ";
        }
        return result;
    }
    /// <summary>
```

```
        /// 查询CPU编号
        /// </summary>
        /// <returns></returns>
        public string GetCpuID()
        {
            string result = "";
            ManagementClass mClass = new ManagementClass("Win32_Processor");
            ManagementObjectCollection moCollection = mClass.GetInstances();
            foreach (ManagementObject mObject in moCollection)
            {
                result += mObject["ProcessorId"].ToString() + " ";
            }
            return result;
        }
        /// <summary>
        /// 查询硬盘编号
        /// </summary>
        /// <returns></returns>
        public string GetMainHardDiskId()
        {
            string result = "";
            ManagementObjectSearcher searcher = new ManagementObjectSearcher("SELECT * FROM Wi
n32_PhysicalMedia");
            ManagementObjectCollection moCollection = searcher.Get();
```

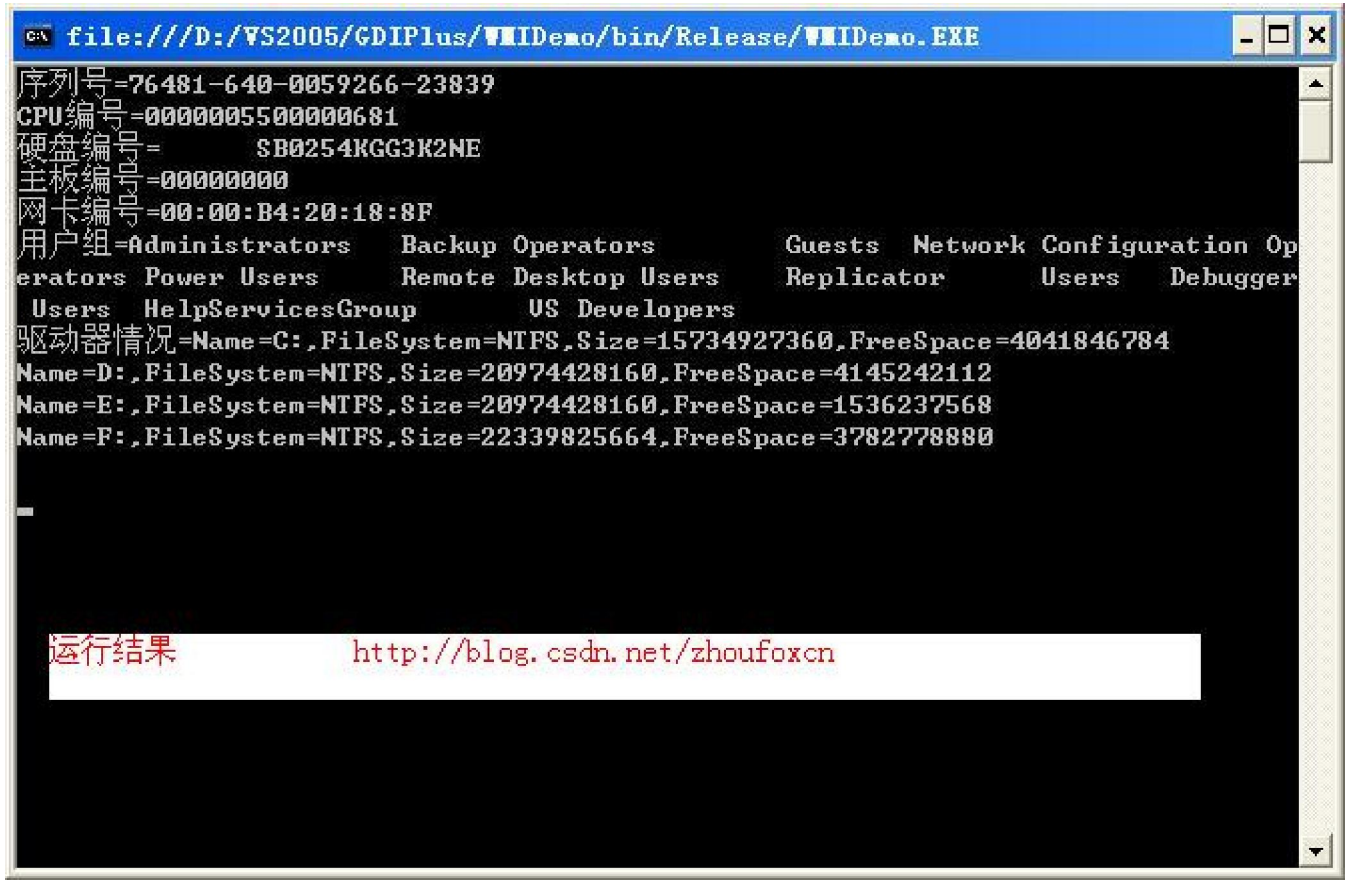
```
        foreach (ManagementObject mObject in moCollection)
        {
            result += mObject["SerialNumber"].ToString() + " ";
        }
        return result;
    }

    /// <summary>
    /// 主板编号
    /// </summary>
    /// <returns></returns>
    public string GetMainBoardId()
    {
        string result = "";
        ManagementObjectSearcher searcher = new ManagementObjectSearcher("root/CIMV2",
                "SELECT * FROM Win32_BaseBoard");
        ManagementObjectCollection moCollection = searcher.Get();
        foreach (ManagementObject mObject in moCollection)
        {
            result += mObject["SerialNumber"].ToString() + " ";
        }
        return result;
    }
```

```
        /// <summary>
        /// 主板编号
        /// </summary>
        /// <returns></returns>
        public string GetNetworkAdapterId()
        {
            string result = "";
            ManagementObjectSearcher searcher = new ManagementObjectSearcher("SELECT MACAddre
ss FROM Win32_NetworkAdapter WHERE ((MACAddress Is Not NULL)AND (Manufacturer <> 'Microsof
t'))");
            ManagementObjectCollection moCollection = searcher.Get();
            foreach (ManagementObject mObject in moCollection)
            {
                result += mObject["MACAddress"].ToString() + " ";
            }
            return result;
        }

        /// <summary>
        /// 主板编号
        /// </summary>
        /// <returns></returns>
        public string GetGroupName()
        {
```

```
            string result = "";
            ManagementObjectSearcher searcher = new ManagementObjectSearcher("root/CIMV2", "SELE
CT * FROM Win32_Group");
            ManagementObjectCollection moCollection = searcher.Get();
            foreach (ManagementObject mObject in moCollection)
            {
                result += mObject["Name"].ToString() + " ";
            }
            return result;
        }

        /// <summary>
        /// 获取本地驱动器信息
        /// </summary>
        /// <returns></returns>
        public string GetDriverInfo()
        {
            string result = "";
            ManagementObjectSearcher searcher = new ManagementObjectSearcher("root/CIMV2", "SELE
CT * FROM Win32_LogicalDisk");
            ManagementObjectCollection moCollection = searcher.Get();
            foreach (ManagementObject mObject in moCollection)
            {
                //mObject["DriveType"]共有6中可能值，分别代表如下意义：
```

```
            //1:No type   2:Floppy disk   3:Hard disk
            //4:Removable drive or network drive   5:CD-ROM   6:RAM disk
            //本处只列出固定驱动器（硬盘分区）的情况
            if (mObject["DriveType"].ToString() == "3")
            {
                result += string.Format("Name={0},FileSystem={1},Size={2},FreeSpace={3} ", mObject["Na
me"].ToString(),
                    mObject["FileSystem"].ToString(), mObject["Size"].ToString(), mObject["FreeSpace"].ToSt
ring());
            }
        }
        return result;
    }
  }
}
```

程序的运行结果如下（不同机器上的结果会不相同）：



最后说明：虽然绝大部分Windows操作系统已经安装了WMI（根据微软官方说法是自WinME以后的Windows系统包括WinME都安装了），但是不能保证上面的代码能在所有的Windows操作系统之上，如Win95之类，如果在不支持WMI的Windows系统上运行WMI，那么需要从MSDN下载WMI。

另外，运行WMI需要当前Windows登录用户有一定的权限，由于本人开发时是以管理员的身份登录并运行程序的，所以没有权限异常提示，但是我不保证代码在你处运行也正常，特别是在WebForm的环境下，因为asp.net程序默认是较低的权限运行的（安全原因）。

请看下一篇《.net中的WMI编程（二）：WMI中的WQL语言和WQL的测试工具》：