**Introduction to System.Management Namespace**

By [Karthikeyan](http://www.c-sharpcorner.com/authors/karthik_mascon/karthikeyan.aspx) on July 20, 2005

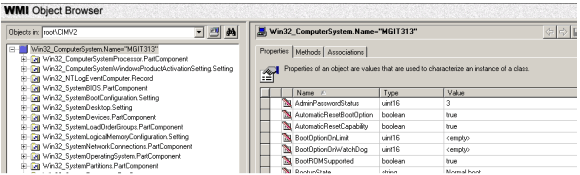
The article provides an insight on the use of System.Management namespace and the advantages of WMI.

* inShare0

The System.Management namespace provides rich set of management information and management events about the system, devices, and applications instrumented to the Windows Management Instrumentation. (WMI) infrastructure. Using the various classes under this namespace we have the ability to query the low level hardware and software related data and information (such as available hard drive space, information on BIOS settings, etc.,).

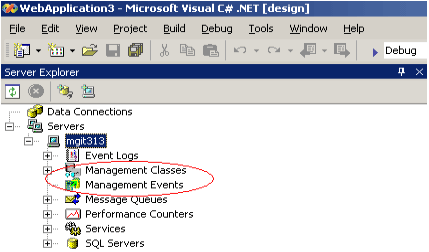
Unfortunately, all classes available under this namespace is written so generically, that Microsoft expects the developers to be aware of the class names (and methods, properties and so on), before they start using them. If you're very serious on learning about WMI and its associated objects, I would recommend you to download the "WMI Tools" available in Microsoft site.   
  
(<http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=6430F853-1120-48DB-8CC5-F2ABDC3ED314>)

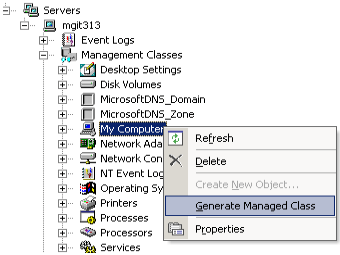
As usual, The WMI objects and libraries provided by Microsoft do not operate with all versions of Windows. Windows 2000 users, have to manually download and register the WMI dll. WMI objects would be installed automatically for Windows 2003 and XP (no need to install or configure them separately).

One of the useful WMI tools (I hope you have downloaded and installed the WMI Tools) is the WMI Object Browser. It provides valuable information about the list of WMI objects, methods and properties that are available. When you try to open the WMI Object Browser, it would prompt for the namespace (and would also display the default namespace "root\CIMV2"). Just click "Ok" and say "Ok" again for the "WMI Object Browser Login dialog box to proceed.   
  


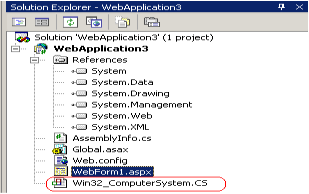
On the left hand side is the available objects and the corresponding Properties, Methods and Associations are listed on the right hand side. I will leave it to the individual developers to explore more about the Object Browser.

Microsoft also provides "Management (WMI) Extensions for Visual Studio .NET 2003 Server Explorer". Just click the following URL and install the package.    
  
<http://www.microsoft.com/downloads/info.aspx?na=47&p=2&SrcDisplayLang=en&SrcCategoryId=&SrcFamilyId=6430F853-1120-48DB-8CC5-2ABDC3ED314&u=details.aspx%3ffamilyid%3d62D91A63-1253-4EA6-8599-68FB3EF77DE1%26displaylang%3den>   
   
After installing the setup, go to the Server Explorer (in view menu), and you'll the see that it now contains "Management Classes" and "Management Events", object(s) added to the list.

   
   
Again, the "Management Classes" (in server explorer) lists the available Classes in WMI, as in WMI Object Browser. But the main advantage is its ability to generate managed wrapper classes on the fly, for the selected object. (The following picture explains it clearly.)



After selecting the "Generate Managed Class", go to the "Solution Explorer". A new .cs file would be automatically added to the solution (Win32\_ComputerSystem.cs). (No need to get alarmed on seeing this new file.).



This automatically generated managed class (Win32\_computersystem.cs), contains the implementation and definition for all the methods and properties available for the selected object (the object in this case is the "My Computer" that we selected in the "Server Explorer's - Management Classes". It's nothing but a managed wrapper for accessing the information about the system.

The contents of this file may be a bit confusing in the beginning, but once you become familiar to the ways and means of using the System.Management namespace, you would find it easier to make modifications in the generated class file.

The following piece of code retrieves the User Name and Manufacturer name.  
  
using WindowsApplication3.ROOT.CIMV2;  
private void button2\_Click(object sender, System.EventArgs e)  
{  
ComputerSystem cs = new ComputerSystem("MGIT313");  
MessageBox.Show(cs.UserName.ToString());  
MessageBox.Show(cs.Manufacturer.ToString());  
}

The WindowsApplication3.ROOT.CIMV2 is nothing but the namespace provided    
for the the managed class which is automatically generated (which is   
Win32\_computersystem.cs - in our case).  Just open the file and you can see  
the namespace name specified.  Use the namespace in your code to avoid fully  
qualifying the same.  
  
Again the ComputerSystem class is the name of the type for the wrapper class   
provided by the tool.  
  
I think now we would have understood the real power and robusness of WMI. Since this article is just an introduction about WMI, I leave it to the individual developers to further explore and experiment with it.

**Conclusion**

WMI objects can also be used in WSH Administrative Scripts, to control, create and modify the parameters and settings of the operating system. These scripts can be written using VBScript or Jscript. Microsoft has provided a utility - "Scriptomatic", to ease the job of creating such scripts.