**Using Static Classes and Methods**

* Article
* 10/12/2021
* 4 minutes to read
* 3 contributors

Not all .NET Framework classes can be created by using New-Object. For example, if you try to create a **System.Environment** or a **System.Math** object with New-Object, you will get the following error messages:

PowerShellCopy

New-Object System.Environment

OutputCopy

New-Object : Constructor not found. Cannot find an appropriate constructor for

type System.Environment.

At line:1 char:11

+ New-Object <<<< System.Environment

PowerShellCopy

New-Object System.Math

OutputCopy

New-Object : Constructor not found. Cannot find an appropriate constructor for

type System.Math.

At line:1 char:11

+ New-Object <<<< System.Math

These errors occur because there is no way to create a new object from these classes. These classes are reference libraries of methods and properties that do not change state. You don't need to create them, you simply use them. Classes and methods such as these are called **static classes** because they are not created, destroyed, or changed. To make this clear we will provide examples that use static classes.

**Getting Environment Data with System.Environment**

Usually, the first step in working with an object in Windows PowerShell is to use Get-Member to find out what members it contains. With static classes, the process is a little different because the actual class is not an object.

**Referring to the Static System.Environment Class**

You can refer to a static class by surrounding the class name with square brackets. For example, you can refer to **System.Environment** by typing the name within brackets. Doing so displays some generic type information:

PowerShellCopy

[System.Environment]

OutputCopy

IsPublic IsSerial Name BaseType

-------- -------- ---- --------

True False Environment System.Object

**Note**

As we mentioned previously, Windows PowerShell automatically prepends '**System.**' to type names when you use New-Object. The same thing happens when using a bracketed type name, so you can specify **[System.Environment]** as **[Environment]**.

The **System.Environment** class contains general information about the working environment for the current process, which is powershell.exe when working within Windows PowerShell.

If you try to view details of this class by typing **[System.Environment] | Get-Member**, the object type is reported as being **System.RuntimeType** , not **System.Environment**:

PowerShellCopy

[System.Environment] | Get-Member

OutputCopy

TypeName: System.RuntimeType

To view static members with Get-Member, specify the **Static** parameter:

PowerShellCopy

[System.Environment] | Get-Member -Static

OutputCopy

TypeName: System.Environment

Name MemberType Definition

---- ---------- ----------

Equals Method static System.Boolean Equals(Object ob...

Exit Method static System.Void Exit(Int32 exitCode)

...

CommandLine Property static System.String CommandLine {get;}

CurrentDirectory Property static System.String CurrentDirectory ...

ExitCode Property static System.Int32 ExitCode {get;set;}

HasShutdownStarted Property static System.Boolean HasShutdownStart...

MachineName Property static System.String MachineName {get;}

NewLine Property static System.String NewLine {get;}

OSVersion Property static System.OperatingSystem OSVersio...

ProcessorCount Property static System.Int32 ProcessorCount {get;}

StackTrace Property static System.String StackTrace {get;}

SystemDirectory Property static System.String SystemDirectory {...

TickCount Property static System.Int32 TickCount {get;}

UserDomainName Property static System.String UserDomainName {g...

UserInteractive Property static System.Boolean UserInteractive ...

UserName Property static System.String UserName {get;}

Version Property static System.Version Version {get;}

WorkingSet Property static System.Int64 WorkingSet {get;}

TickCount ExitCode

We can now select properties to view from System.Environment.

**Displaying Static Properties of System.Environment**

The properties of System.Environment are also static, and must be specified in a different way than normal properties. We use :: to indicate to Windows PowerShell that we want to work with a static method or property. To see the command that was used to launch Windows PowerShell, we check the **CommandLine** property by typing:

PowerShellCopy

[System.Environment]::Commandline

OutputCopy

"C:\Program Files\Windows PowerShell\v1.0\powershell.exe"

To check the operating system version, display the OSVersion property by typing:

PowerShellCopy

[System.Environment]::OSVersion

OutputCopy

Platform ServicePack Version VersionString

-------- ----------- ------- -------------

Win32NT Service Pack 2 5.1.2600.131072 Microsoft Windows...

We can check whether the computer is in the process of shutting down by displaying the **HasShutdownStarted** property:

PowerShellCopy

[System.Environment]::HasShutdownStarted

OutputCopy

False

**Doing Math with System.Math**

The System.Math static class is useful for performing some mathematical operations. The important members of **System.Math** are mostly methods, which we can display by using Get-Member.

**Note**

System.Math has several methods with the same name, but they are distinguished by the type of their parameters.

Type the following command to list the methods of the **System.Math** class.

PowerShellCopy

[System.Math] | Get-Member -Static -MemberType Methods

OutputCopy

TypeName: System.Math

Name MemberType Definition

---- ---------- ----------

Abs Method static System.Single Abs(Single value), static Sy...

Acos Method static System.Double Acos(Double d)

Asin Method static System.Double Asin(Double d)

Atan Method static System.Double Atan(Double d)

Atan2 Method static System.Double Atan2(Double y, Double x)

BigMul Method static System.Int64 BigMul(Int32 a, Int32 b)

Ceiling Method static System.Double Ceiling(Double a), static Sy...

Cos Method static System.Double Cos(Double d)

Cosh Method static System.Double Cosh(Double value)

DivRem Method static System.Int32 DivRem(Int32 a, Int32 b, Int3...

Equals Method static System.Boolean Equals(Object objA, Object ...

Exp Method static System.Double Exp(Double d)

Floor Method static System.Double Floor(Double d), static Syst...

IEEERemainder Method static System.Double IEEERemainder(Double x, Doub...

Log Method static System.Double Log(Double d), static System...

Log10 Method static System.Double Log10(Double d)

Max Method static System.SByte Max(SByte val1, SByte val2), ...

Min Method static System.SByte Min(SByte val1, SByte val2), ...

Pow Method static System.Double Pow(Double x, Double y)

ReferenceEquals Method static System.Boolean ReferenceEquals(Object objA...

Round Method static System.Double Round(Double a), static Syst...

Sign Method static System.Int32 Sign(SByte value), static Sys...

Sin Method static System.Double Sin(Double a)

Sinh Method static System.Double Sinh(Double value)

Sqrt Method static System.Double Sqrt(Double d)

Tan Method static System.Double Tan(Double a)

Tanh Method static System.Double Tanh(Double value)

Truncate Method static System.Decimal Truncate(Decimal d), static...

This displays several mathematical methods. Here is a list of commands that demonstrate how some of the common methods work:

PowerShellCopy

[System.Math]::Sqrt(9)

3

[System.Math]::Pow(2,3)

8

[System.Math]::Floor(3.3)

3

[System.Math]::Floor(-3.3)

-4

[System.Math]::Ceiling(3.3)

4

[System.Math]::Ceiling(-3.3)

-3

[System.Math]::Max(2,7)

7

[System.Math]::Min(2,7)

2

[System.Math]::Truncate(9.3)

9

[System.Math]::Truncate(-9.3)

-9