

  
 Dr. Holger Schwichtenberg





## Was ein PowerShell-Nutzer über .NET wissen MUSS!

17.04.2018 16:45 bis 17:45 Uhr

**Dr. Holger Schwichtenberg**  
 Softwarearchitekt, Berater, Trainer, Fachjournalist  
[www.dotnet-doktor.de](http://www.dotnet-doktor.de)  
[www.twitter.com/dotnetdoktor](https://www.twitter.com/dotnetdoktor)

Version 5.7  
 17.04.2018 #1

  
 Dr. Holger Schwichtenberg

## Kurze Vorstellung

- MVPs & Spezialisten für .NET, Visual Studio, TFS, SQL Server, SharePoint, BizTalk, Windows Server, Azure, System Center, Xamarin, JavaScript, PowerShell, Java, Oracle, Agile, Scrum u.a.
- **www.IT-Visions.de, Essen**
  - Strategische und technische Beratung
  - Schulungen (individuell/In-House und standardisiert/öffentlich)
- **5Minds IT-Solutions GmbH & Co KG, Gelsenkirchen**
  - Softwareentwicklung
- **Dr. Holger Schwichtenberg**
  - Wirtschaftsinformatiker, MVP, MCSD
  - Entwicklungsleiter, Softwarearchitekt, Berater, Trainer
  - Autor für heise.de, iX, Windows Developer, dotnetpro, O'Reilly, Carl Hanser, Addison-Wesley, Microsoft Press u.a.
  - Blog: [www.DOTNET-DOKTOR.de](http://www.DOTNET-DOKTOR.de)
  - Twitter: [www.twitter.com/DOTNETDOKTOR](https://www.twitter.com/DOTNETDOKTOR)
  - Kontakt: [buero@IT-Visions.de](mailto:buero@IT-Visions.de), 0201 649590-0





### Kundenbeispiele

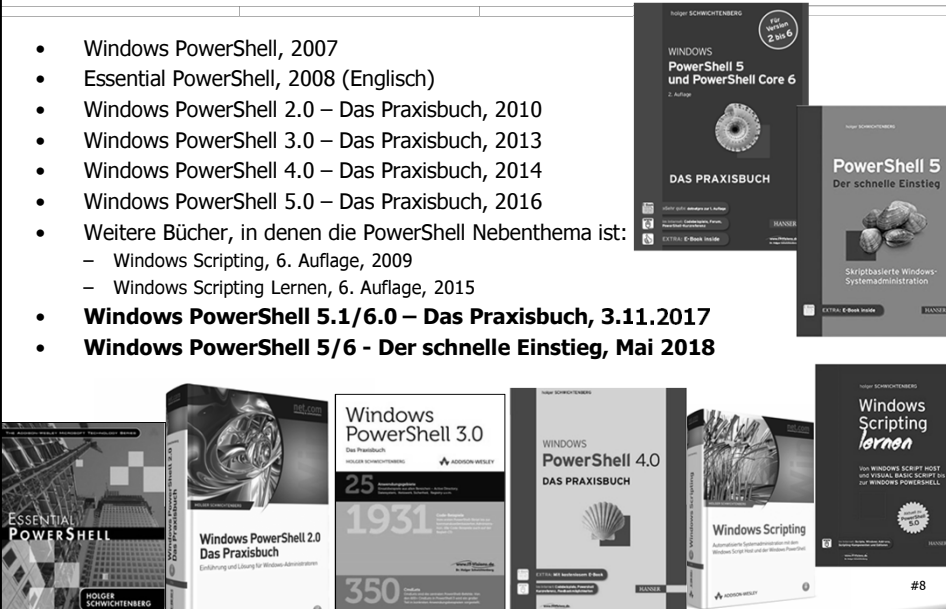
  
  
  


[www.IT-Visions.de](http://www.IT-Visions.de)  
Dr. Holger Schwichtenberg


## Bücher des Vortragenden über die PowerShell

- Windows PowerShell, 2007
- Essential PowerShell, 2008 (Englisch)
- Windows PowerShell 2.0 – Das Praxisbuch, 2010
- Windows PowerShell 3.0 – Das Praxisbuch, 2013
- Windows PowerShell 4.0 – Das Praxisbuch, 2014
- Windows PowerShell 5.0 – Das Praxisbuch, 2016
- Weitere Bücher, in denen die PowerShell Nebenthema ist:
  - Windows Scripting, 6. Auflage, 2009
  - Windows Scripting Lernen, 6. Auflage, 2015
- **Windows PowerShell 5.1/6.0 – Das Praxisbuch, 3.11.2017**
- **Windows PowerShell 5/6 - Der schnelle Einstieg, Mai 2018**



[www.IT-Visions.de](http://www.IT-Visions.de)  
Dr. Holger Schwichtenberg

## Agenda dieses Kurzvortrags



- .NET Framework vs .NET Core
- Namensräume und Klassen
- Objekte instanziiieren
- Klassenmitglieder nutzen
- Die .NET-Klassen-Dokumentation nutzen
- Generische Klassen
- Statische Klassen und statische Mitglieder
- Enumerationen
- Assemblies laden
- Nuget-Pakete nutzen
- Beispiele von C# oder Visual Basic .NET in die PowerShell-Skriptsprache übersetzen
- C# oder Visual Basic .NET in PowerShell-Skripte einbetten

#10

[www.IT-Visions.de](http://www.IT-Visions.de)<sup>®</sup>  
Dr. Holger Schwichtenberg

Zielsetzung dieses Vortrags

- Verständnis für .NET wecken
  - .NET ist Unterbau von PowerShell
  - Man kann mit .NET weit über die Möglichkeiten der PowerShell-Commandlets hinausgehen
  - Objekte aus .NET-Commandlets direkt weiterverarbeiten
  - .NET-Klassen direkt nutzen
  - Sie müssen dazu keine .NET-Sprache wie C# erlernen, Sie können weiter in der PowerShell-Sprache arbeiten
  - Aber Sie müssen .NET verstehen
- Verständnis für objektorientierte Programmierung wecken
  - Das ist zu einem gewissen Grad notwendig, um .NET zu verstehen
- Es geht nicht darum, möglichst viele oder gar alle .NET-Klassen zu erläutern: ~ 13.500 Klassen im .NET Framework, viele 10-tausend Third-Party-Klassen

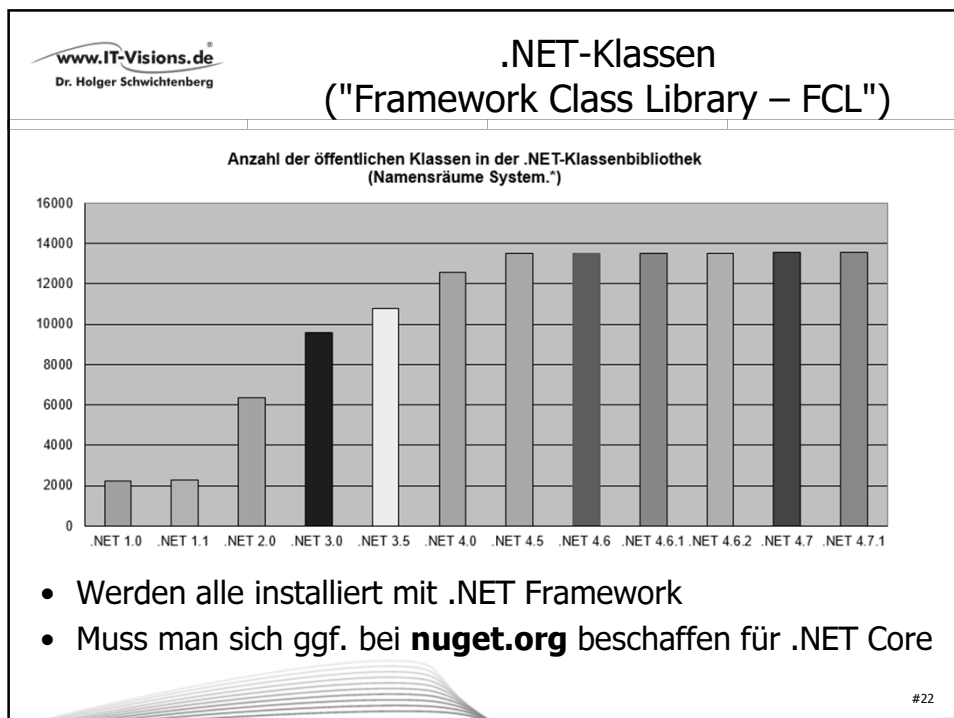
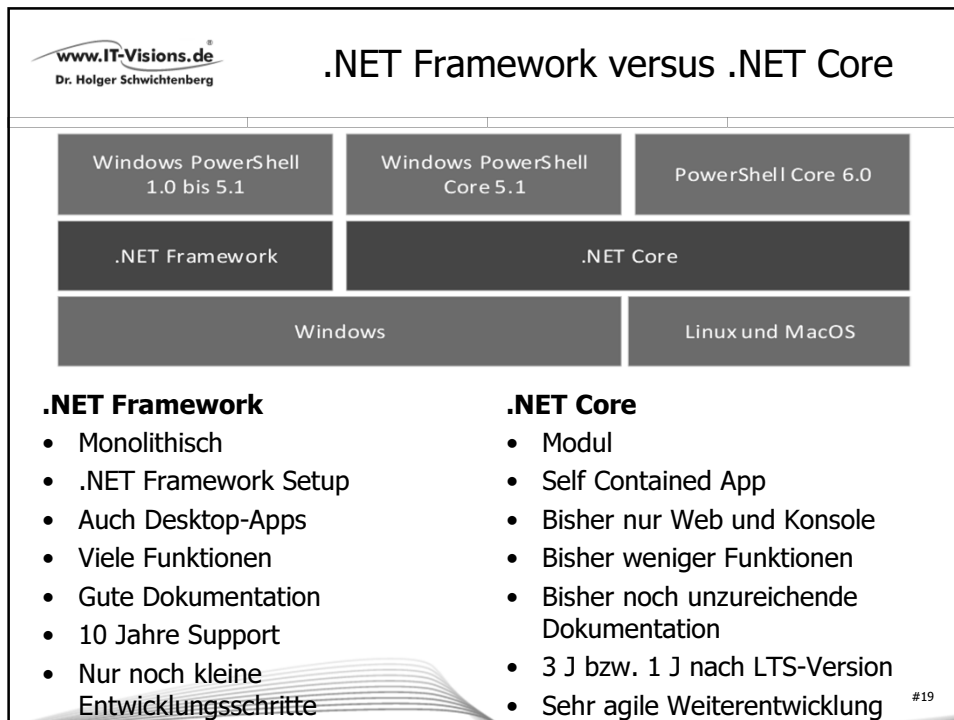
#17

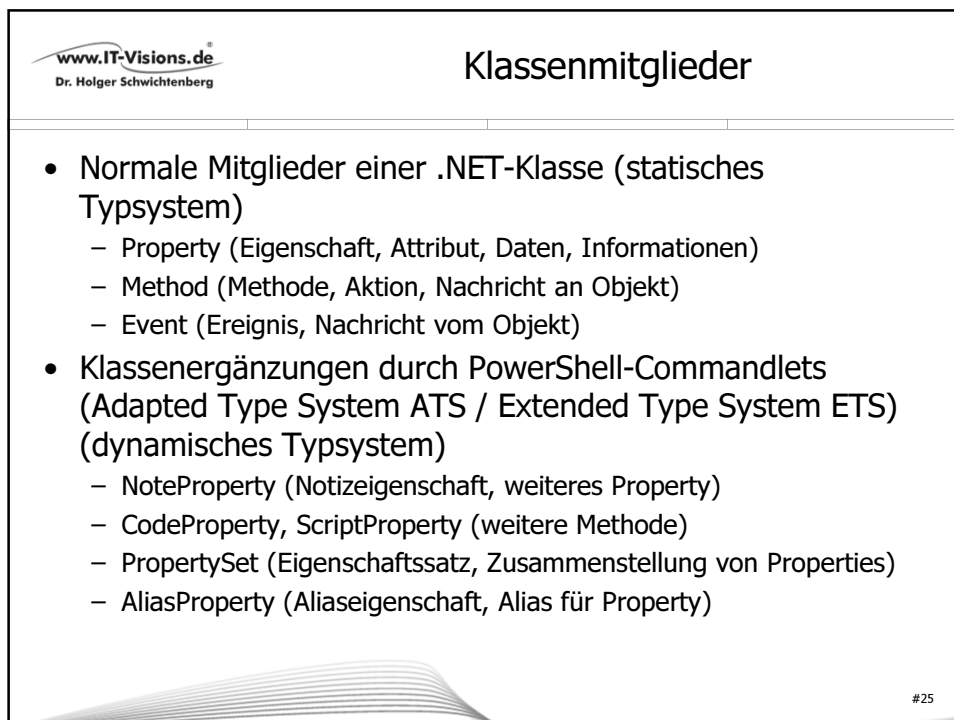
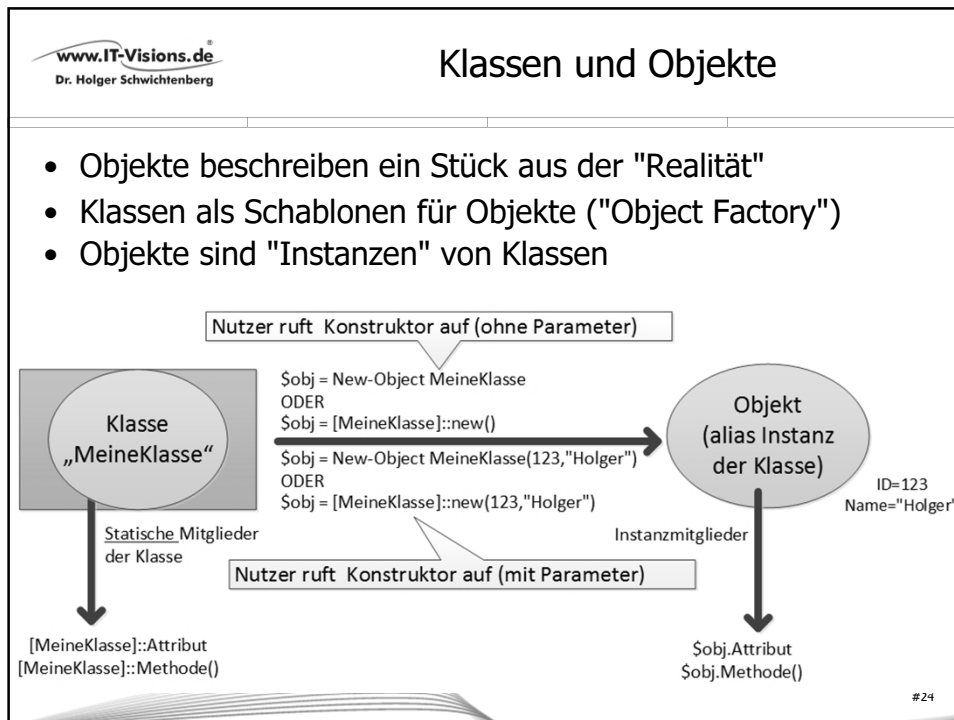
[www.IT-Visions.de](http://www.IT-Visions.de)<sup>®</sup>  
Dr. Holger Schwichtenberg

**Motivationsbeispiel: Datenbankzugriffe**

ADO.NET (System.Data)

#18





www.IT-Visions.de®  
Dr. Holger Schwichtenberg

## Get-Member zeigt die Mitglieder eines Objekts an

```
PS C:\Users\hs.ITU> Get-Process | Get-Member
```

Name	MemberType	Definition
Handles	AliasProperty	Handles = Handlecount
Name	AliasProperty	Name = ProcessName
API	AliasProperty	API = NonpagedSystemMemorySize
PM	AliasProperty	PM = PagedMemorySize
VM	AliasProperty	VM = VirtualMemorySize
WS	AliasProperty	WS = WorkingSet
Disposed	Event	System.EventHandler Disposed(System.Object, System.Eve...
ErrorDataReceived	Event	System.EventHandler ErrorDataReceived(System.Object, System.Eve...
Exited	Event	System.EventHandler Exited(System.Object, System.Eve...
OutputDataReceived	Event	System.EventHandler OutputDataReceived(System.Object, System.Eve...
BeginErrorReadLine	Method	void BeginErrorReadLine()
BeginOutputReadLine	Method	void BeginOutputReadLine()
CancelErrorRead	Method	void CancelErrorRead()
CancelOutputRead	Method	void CancelOutputRead()
Close	Method	void Close()
CloseMainWindow	Method	bool CloseMainWindow()
CreateObjRef	Method	System.Runtime.Remoting.ObjRef CreateObjRef(type requ...
Dispose	Method	void Dispose(). void IDisposable.Dispose()
Equals	Method	bool Equals(System.Object obj)
GetHashCode	Method	int GetHashCode()
GetLifetimeService	Method	System.Object GetLifetimeService()
GetType	Method	type GetType()
InitializeLifetimeService	Method	System.Object InitializeLifetimeService()
Kill	Method	void Kill()
Refresh	Method	void Refresh()
Start	Method	bool Start()
ToString	Method	string ToString()
WaitForExit	Method	bool WaitForExit(int milliseconds, void WaitForExit()
WaitForInputIdle	Method	bool WaitForInputIdle(int milliseconds, void WaitForI...
__NounName	NoteProperty	System.String __NounName=Process
BasePriority	Property	int BasePriority {get;}
Container	Property	System.ComponentModel.IContainer Container {get;}
EnableRaisingEvents	Property	bool EnableRaisingEvents {get;set;}
ExitCode	Property	int ExitCode {get;}
ExitTime	Property	datetime ExitTime {get;}
Handle	Property	System.IntPtr Handle {get;}
HandleCount	Property	int HandleCount {get;}
HasExited	Property	bool HasExited {get;}
WorkingSet	Property	int WorkingSet {get;}
WorkingSet64	Property	long WorkingSet64 {get;}
PSConfiguration	PropertySet	PSConfiguration (Name, Id, PriorityClass, FileVersion)
PSResources	PropertySet	PSResources (Name, Id, Handlecount, WorkingSet, NonPag...
Company	ScriptProperty	System.Object Company {get=\$this.Mainmodule.FileVersio...
CPU	ScriptProperty	System.Object CPU {get=\$this.TotalProcessorTime.TotalS...
Description	ScriptProperty	System.Object Description {get=\$this.Mainmodule.FileVe...
FileVersion	ScriptProperty	System.Object FileVersion {get=\$this.Mainmodule.FileVe...
Path	ScriptProperty	System.Object Path {get=\$this.Mainmodule.FileName;}
Product	ScriptProperty	System.Object Product {get=\$this.Mainmodule.FileVersio...
ProductVersion	ScriptProperty	System.Object ProductVersion {get=\$this.Mainmodule.Fil...

#28

www.IT-Visions.de®  
Dr. Holger Schwichtenberg

## .NET-Klassen-Dokumentation

- Alt: Microsoft Developer Network (MSDN)  
msdn.microsoft.com
  - Beispiele in C#, Visual Basic .NET, C#, C++ (nicht PowerShell)
- Neu: docs.Microsoft.com
  - Beispiele nur noch in C#

Microsoft Technologies Documentation Resources

.NET APIs .NET Core .NET Framework ASP.NET Xamarin

Docs / .NET / .NET API Browser / System.Diagnostics / Process

### Process Class

Namespace: System.Diagnostics  
Assemblies: System.Diagnostics.Process.dll, System.dll, netstandard.dll

Provides access to local and remote processes and enables you to start and stop local system processes.

```
public class Process : System.ComponentModel.Component
```

Inheritance: Object → MarshalByRefObject → Component → Process

Examples

- Klassenergänzungen durch die PowerShell nicht dokumentiert ☹
- Weitere Dokumentation verstreut über diverse Sites auf \*.microsoft.com, nuget.org, github.com u.a.

#29

[www.IT-Visions.de](http://www.IT-Visions.de)  
Dr. Holger Schwichtenberg

Übersetzungsfehler in der deutschen Dokumentation ☹

- [https://msdn.microsoft.com/en-us/library/system.io.driveinfo.driveinfo\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.io.driveinfo.driveinfo(v=vs.110).aspx)

### Remarks

Use this class to obtain information on drives. The drive name must be either an uppercase or lowercase letter from 'a' to 'z'. You **cannot** use this method to obtain information on drive names that are **null** or use UNC (\\server\share) paths.

- [https://msdn.microsoft.com/de-de/library/system.io.driveinfo.driveinfo\(v=vs.110\).aspx](https://msdn.microsoft.com/de-de/library/system.io.driveinfo.driveinfo(v=vs.110).aspx)

### Hinweise

Verwenden Sie diese Klasse, um Informationen zu Laufwerken zu erhalten. Den Namen des Laufwerks muss einen Buchstaben in Groß- oder Kleinschreibung von "a" bis "Z" sein. Sie **können** mithilfe dieser Methode zum Abrufen von Informationen zu den Laufwerknamen, sind **null** oder UNC (\\server\share) Pfade verwenden.

#30

[www.IT-Visions.de](http://www.IT-Visions.de)  
Dr. Holger Schwichtenberg

Namensräume


*Absoluter Klassenname*

*Wurzelnamensraum* *Relativer Klassenname*

System.DirectoryServices.ActiveDirectory.**Domain**

*Unternamenraum*

#34

www.IT-Visions.de®  
Dr. Holger Schwichtenberg

## Konstruktion und Nutzung von .NET-Objekten

Konstruktor ohne Parameter  
\$sound = new-Object System.Media.SoundPlayer  
Setzen einer Eigenschaft  
\$sound.SoundLocation="c:\WINDOWS\Media\notify.wav"  
Aufruf einer Methode  
\$sound.Play()


Klammern nicht vergessen!!!

Konstruktor mit Parameter  
\$obj = new-Object System.DateTime(2017,4,27)  
oder \$obj = [System.DateTime] "2017,4,27"  
oder \$obj = Get-Date "2017,4,27"  
Abruf einer Eigenschaft  
\$obj.Day  
Aufruf einer Methode, die einen Wert liefert  
\$obj.ToLongDateString()

```
PowerShell-6.0.1
PS T:\> $d = new-object System.DateTime
PS T:\> $d.year = 2017
'year' is a ReadOnly property.
At line:1 char:1
+ $d.year = 2017
~
+ CategoryInfo          : InvalidOperation: (:) [], RuntimeException
+ FullyQualifiedErrorId : PropertyAssignmentException
```

Konstruktion durch Commandlet  
\$notepad = Start-Process notepad -PassThru  
\$notepad.WaitForExit()

#35

www.IT-Visions.de®  
Dr. Holger Schwichtenberg

## Generische Klassen (ab WPS 2.0)

- Eine **generische Klasse** hat einen oder mehrere **Typparameter**, der den Inhalt oder Zwecke näher beschreibt.
- Seit PowerShell 2.0 wird auch die Instanziierung generischer Klassen auf einfache Weise unterstützt.
- Ein oder mehrere Typparameter werden in eckigen Klammern nach dem Klassennamen genannt.

```
$l = New-Object System.Collections.Generic.List[int]
$l.Count
$l.Add(1)
$l.Add(10)
$l.Add("Holger")
$l.Count
$l
```

Vorteil gegenüber PowerShell-Arrays:  
Man kann den Elementtyp in der Liste einschränken!

Achtung: Bei Verwendung von **[string]** als Typparameter kann ich bei Add() alles anfügen, weil PowerShell immer **toString()** aufruft!

#36



**www.IT-Visions.de**  
Dr. Holger Schwichtenberg

## Generische Klassen (ab WPS 2.0)

**WICHTIG:** Wenn es mehr als einen Typparameter gibt, muss der muss der ganze Klassenname inklusive der Typparameterangaben in Anführungszeichen stehen!

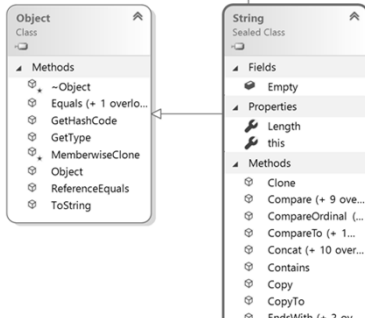
```
$dic = New-Object "System.Collections.Generic.SortedDictionary[int,string]"  
$dic.Add(45257,"Essen")  
$dic.Add(80538,"Schwabing")  
$dic.Add(10789,"Berlin")  
$dic.Count  
"Nur die Schlüssel:"  
$dic.Keys  
"Nur die Werte:"  
$dic.Values  
"Schlüssel und Wert"  
$dic
```

#37


**www.IT-Visions.de**  
Dr. Holger Schwichtenberg

## Vererbung

- Vererbung bedeutet: Klassen übernehmen Attribute, Methoden und Ereignisse von einer anderen Klasse und ergänzen weitere Klassenmitglieder
- Alle .NET-Klassen bis auf eine (System.Object) erben von genau einer anderen Klasse
- Alle Klassen erben direkt oder indirekt von System.Object



#39



Dr. Holger Schwichtenberg

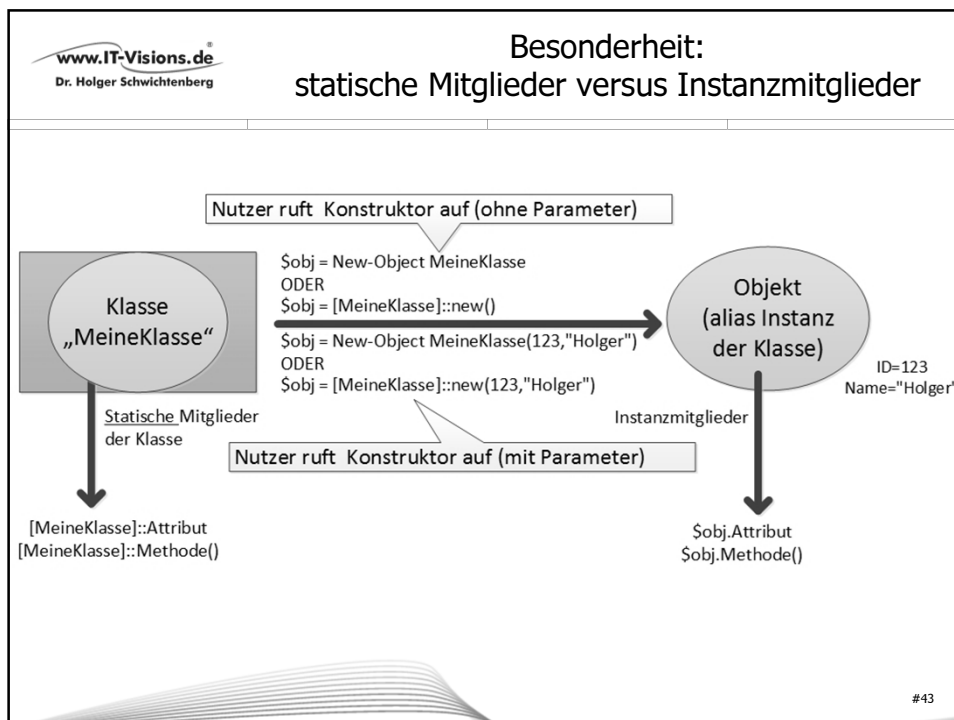
## ToString()


- ToString() wird in jeder Klasse geerbt von System.Object
- Aber ist verschieden realisiert!!!
- (Get-Service)[0].ToString()
- (Get-Process)[0].ToString()
- (Get-Host)[0].ToString()

```
AdobeARMService  
System.Diagnostics.Process (adb)  
System.Management.Automation.Internal.Host.InternalHost
```

- Standard ist die Ausgabe des Klassennamens
- Das ist in den allermeisten .NET-Klassen nur so realisiert!

#40






www.IT-Visions.de<sup>®</sup>  
Dr. Holger Schwichtenberg

## Normale versus statische Mitglieder

---

- **Normale, nicht-statische Mitglieder**
  - Nach New-Object kann man die Mitglieder über die Punktnotation  
\$obj.NameAttribut oder  
\$obj.NameMethode() aufrufen
  - Normale Mitglieder gibt es nur in normalen Klassen
- **Statische Mitglieder / Instanzmitglieder**
  - Können ausschließlich über die Syntax  
[Klassenname]::NameAttribut oder  
[Klassenname]::NameMethode()  
verwendet werden
  - Statische Mitglieder gibt es in statischen Klassen und normalen Klassen

#44




www.IT-Visions.de<sup>®</sup>  
Dr. Holger Schwichtenberg

## Normale versus statische Klassen

---

- **Normale, nicht-statische Klassen**
  - Haben mindestens einen Konstruktor und können daher mit New-Object instanziiert werden
  - Der Konstruktor kann Parameter erwarten
  - Es kann mehrere verschiedene Konstruktoren mit verschiedenen Parametern geben
  - Nutzung mit Punkt: \$obj.Mitglied
- **Statische Klassen**
  - Haben keinen Konstruktor
  - Nutzung mit Doppelpunkt:  
[Klassenname]::Mitglied verwendet werden

#45




Dr. Holger Schwichtenberg

Beispiele

<ul style="list-style-type: none"><li>• Normale, nicht-statische Klasse, verschiedene Konstruktoren \$obj = new-Object System.DateTime \$obj = new-Object System.DateTime(2018,1,26) \$obj = new-Object System.DateTime(2018,1,26,10,12,4)</li></ul>
<ul style="list-style-type: none"><li>• Normale, nicht-statische Klassen, normale Mitglieder (Instanzmitglieder) \$obj.Day \$obj.ToLongDateString()</li></ul>
<ul style="list-style-type: none"><li>• Normale, nicht-statische Klasse, statisches Mitglied [System.DateTime]::Now</li></ul>
<ul style="list-style-type: none"><li>• Statische Klasse, statisches Mitglied [System.Environment]::UserName</li></ul>

#46



Dr. Holger Schwichtenberg

Statisches Klassen und Mitglied in der neuen Dokumentation

NET Core 2.0

Search

- > Uri
- > Decimal
- > Delegate
- > DivideByZeroException
- > DirectoryInfo
- > Double
- > DuplicateWaitObjectException
- > EntryPointNotFoundException
- > Enum
- > Environment
- > Properties
- > Methods

Environment Class

Namespace: System

Assemblies: System.Runtime.Extensions.dll,mscorlib.dll,netstandard.dll

Provides information about, and means to manipulate, the current environment and platform. This class cannot be inherited.

C#  
[System.Runtime.InteropServices.ComVisible(true)]  
public static class Environment

.NET Framework 4.7.1

Search

- > Properties
- > Date
- > Day
- > DayOfWeek
- > DayOfYear
- > Hour
- > Kind
- > Millisecond
- > Minute
- > Month
- > Now

DateTime.Now Property

Namespace: System


Assemblies: System.Runtime.dll,mscorlib.dll,netstandard.dll

Gets a **DateTime** object that is set to the current date and time on this computer,

C#  
public static DateTime Now { get; }

Property Value  
DateTime  
An object whose value is the current local date and time.

Examples




Dr. Holger Schwichtenberg

Leider nicht immer logisch, was  
Konstruktor bedeutet & was statisch ist

<pre># Neue GUID - so nicht !!! \$x = <b>New-Object</b> System.Guid \$x.Guid → Liefert immer nur "0"-Guid ☹ # Neue GUID - richtig [System.Guid]::NewGuid().Guid → Das ist eine neue GUID!</pre>
<pre># neue Zufallszahl \$r = <b>New-Object</b> System.Random(12) → 12 ist nicht der Bereich, sondern nur die Kalkulationsbasis \$r.Next(100) → Das ist eine Zufallszahl zwischen 0 und 100</pre>

#51



Dr. Holger Schwichtenberg


Enumerationen (Auflistungstypen)

<ul style="list-style-type: none"><li>• Ist eine .NET-Klasse, die eine Menge statischer Werte enthält, die andere Klassen nutzen.</li><li>• Z.B. hat die Klasse System.IO.DriveInfo ein Attribut mit Namen DriveType, das den Datentyp System.IO.DriveType besitzt.</li><li>• Verwendung mit [EnumTyp]::Wert</li></ul> <p>Beispiel:</p> <ul style="list-style-type: none"><li>• [System.IO.DriveInfo]::GetDrives()   where { \$_.DriveType -eq [System.IO.DriveType]::CDRom }   FL Name, DriveType, IsReady</li><li>• Alle möglichen Werte: [System.Enum]::GetNames([System.IO.DriveType])</li></ul>
--

#52

**www.IT-Visions.de**  
**Dr. Holger Schwichtenberg**

- ```
PS C:\Users\hs.ITU> [System.Windows.Forms.MessageBox]::Show("Text", "Ueberschrift", [System.Windows.Forms.MessageBoxButtons]::OK)
Unable to find type [System.Windows.Forms.MessageBox]: make sure that the assembly containing this type is loaded.
line 1 char 1:
[System.Windows.Forms.MessageBox]::Show("Text", "Ueberschrift", [System.Windows.F ...
+ CategoryInfo          : InvalidOperation: (System.Windows.Forms.MessageBox:TypeName) [], RuntimeException
+ FullyQualifiedErrorId : TypeNotFound
```

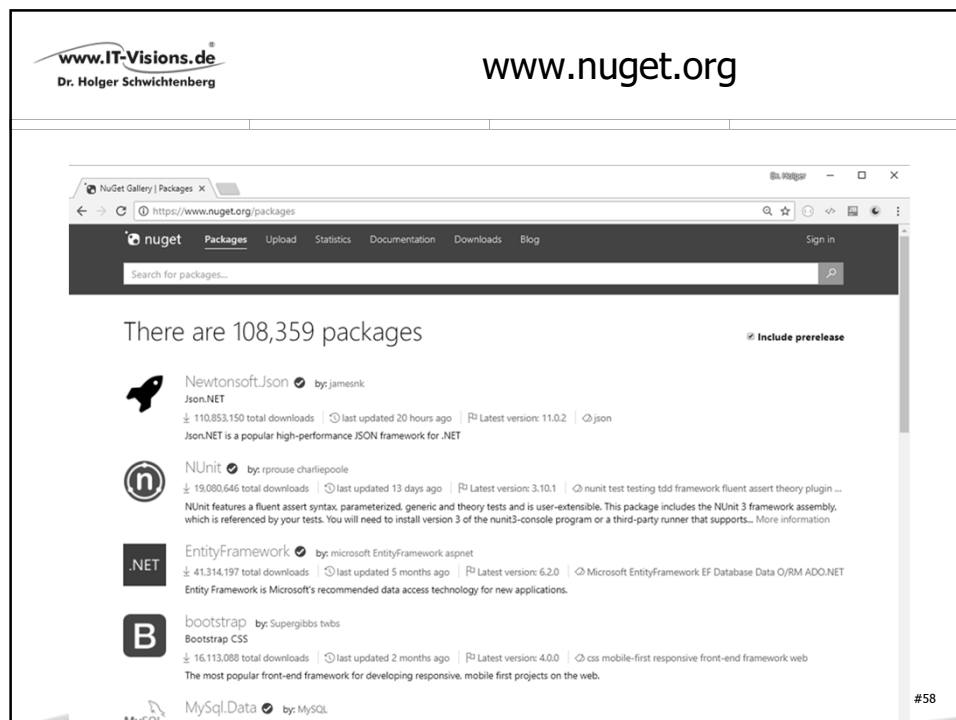
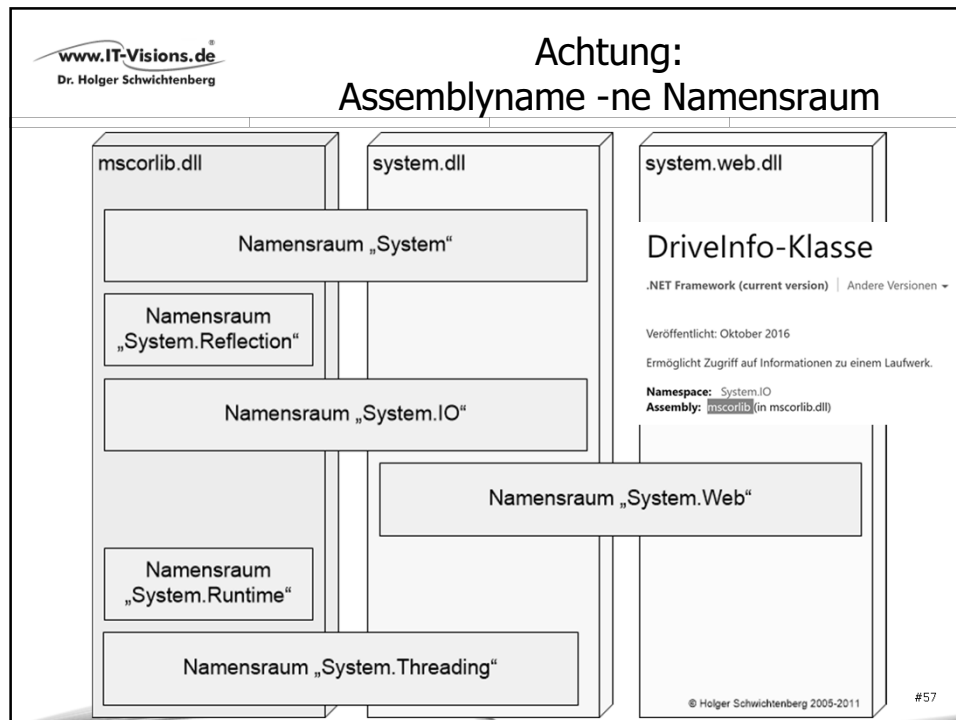


- #59

**www.IT-Visions.de**  
Dr. Holger Schwichtenberg

- GAC = Global Assembly Cache (gibt es nicht in .NET Core)





**www.IT-Visions.de**  
Dr. Holger Schwichtenberg

## Nuget-Pakete nutzen

- Nuget.exe install PaketName oder:
- Manueller Download: .nupkg-Datei → .zip → auspacken

**www.IT-Visions.de**  
Dr. Holger Schwichtenberg

## Von C# zu PS-Skript

- Manche C#-Beispiele sind einfach zu übertragen

```
using System;
using System.Globalization;
using System.Threading;

namespace Demo_Convert_CS_to_WPS
{
    class Program
    {
        static void Main(string[] args)
        {
            // aus MSDN-Doku: http://msdn.microsoft.com/de-de/library/
            // system.datetime.tolongdatestring.aspx?cs-save-lang=1&cs-lang=csharp#code-snippet-2
            DateTime myDateTime = new System.DateTime(2001, 5, 16, 3, 2, 15);    $myDateTime = New-Object System.DateTime(2001, 5, 16, 3, 2, 15)

            // Display the name of the current culture.
            CultureInfo ci = Thread.CurrentThread.CurrentCulture;              # Display the name of the current culture.
            Console.WriteLine("Current culture: \"{0}\"\\n", ci.Name);           $ci = [System.Threading.Thread]::CurrentThread.CurrentCulture
                                     "Current culture: $($ci.Name)"

            // Display the long date pattern and string.
            Console.WriteLine("Long date pattern: \"{0}\"\\n", ci.DateTimeFormat.LcDatePattern);
            Console.WriteLine("Long date string: \"{0}\"\\n", myDateTime.ToLongDateString());
            # Display the long date pattern and string.
            "Long date pattern: $($ci.DateTimeFormat.LongDatePattern)"
            "Long date string: $($myDateTime.ToLongDateString())"
        }
    }
}
```

- Andere hingegen gar nicht einfach, weil zu umfassend:  
<http://msdn.microsoft.com/de-de/library/system.media.soundplayer.aspx>

#60



**Was geht noch?**

- C# in WPS-Skripte einbetten
- C#-Dateien in WPS laden und ausführen
- Commandlets in C# schreiben und als binäre DLL-Module ausliefern

```

1 # C#-Code eingebettet in PowerShell-Skripte
2 $CSKlasseCode = @"
3 public class HelloWorld
4 {
5     // Statische Methode mit Parameter
6     public static string Hallo1(string name)
7     {
8         return "Hallo " + name + "!";
9     }
10
11     // Instanz-Methode mit Parameter
12     public string Hallo2(string name)
13     {
14         return "Hallo " + name + "!";
15     }
16
17     // Attribut (Field)
18     public string Name = "";
19
20     // Instanz-Methode ohne Parameter
21     public string Hallo3()
22     {
23         return "Hallo " + this.Name + "!";
24     }
25 }
26 "@
27
28 # Dynamisches Kompilieren
29 Add-Type -typedefinition $CSKlasseCode -language csharp
30
31 # Testen
32 [HelloWorld]::Hallo1("Holger")
33 $o = New-Object HelloWorld
34 $o.Hallo2("Holger")
35 $o.Name = "Holger"
36 $o.Hallo3()

```

**Am Ziel**



Vielen Dank für Ihre Aufmerksamkeit!

Fragen ???

Download der Präsentation:  
<http://IT-Visions.de/vortraege>

**Brauchen Sie Unterstützung?**  
**Schulung, Beratung, Support, Implementierung**  
Architektur, .NET, HTML5, JavaScript, Apps, PowerShell, SQL Server, SharePoint, Windows, BizTalk, CRM u.v.a. Microsoft-Produkte sowie Java, Python, Angular, Oracle, node.js, MySQL ...  
[www.IT-Visions.de](http://www.IT-Visions.de) • [anfragen@IT-Visions.de](mailto:anfragen@IT-Visions.de)