```
PowerShell 6 (x64)
                                                class Session {
  [string]$Title = "PowerShell Classes"
  [string]$Speaker = "Jan-Henrik Damaschke"
  [string]$Event = "PowerShell Saturday"
  Session ()
    Write-Host "Welcome!"
```

Jan-Henrik Damaschke

Senior Cloud Architect

MVP Azure, MCT

- @jandamaschke
- itinsights.org

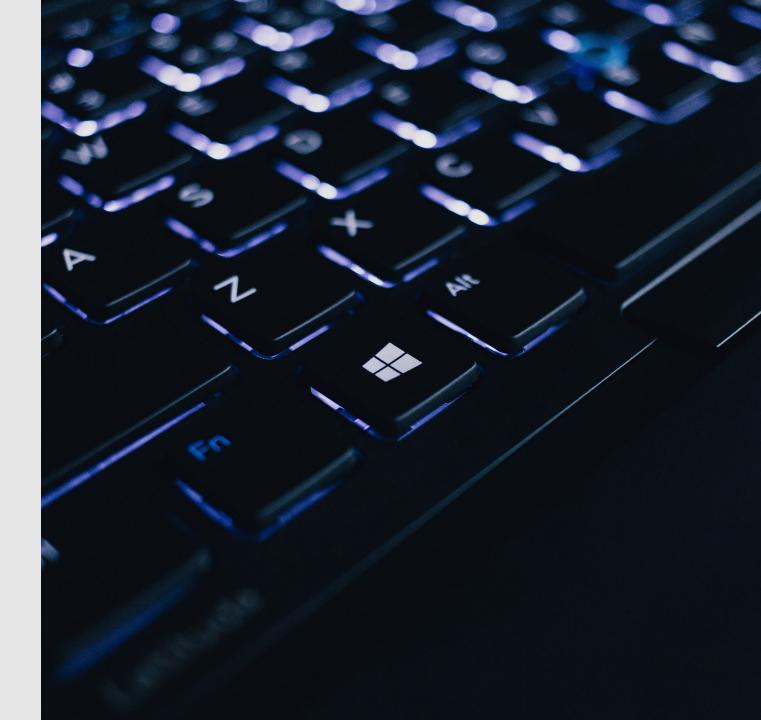


Agenda

- Class basics
- Working with classes
- Cross Platform
- Azure Storage Table wrapper
- Questions

Class basics – Class keyword

```
Class Calculator
  [double]$operand
  Calculator([int]$operand)
    $this.operand = $operand
  [double]Add([int]$operand)
    Write-Host "Increasing $($this.operand) by $($operand)"
    $this.operand = $this.operand + $operand
    return $this.operand
```



Class basics - Overloads

```
Class Calculator
 hidden [double]$operand = 0
 Calculator([int]$operand)
   $tmpValue = 0
    [double]::TryParse($operand, [ref]$tmpValue)
   $this._operand = $tmpValue
 Calculator() {}
```



Class basics – Class Methods

```
Class Calculator
 hidden [double]$operand = 0
 static $history = @()
  [void]Add([int]$operand)
   $message = "Increasing $($this.operand) by $($operand)"
   $this.operand = $this.operand + $operand
   $this. WriteHistory($message)
  hidden [void]_WriteHistory($entry)
   $entry = "$entry : $($this.operand)"
   Write-Host $entry
    [Calculator]::history += $entry
```



Class basics – Inheritance

```
Class VmEntity : Microsoft.Azure.Cosmos.Table.TableEntity {
  [string]$Name
  [string]$Id
  [string]$VmSize
  [string]$0sType
  VmEntity($region, $resourceId, $name, $vmSize, $osType) {
    $this.PartitionKey = $region
    $this.RowKey = $resourceId.replace('/','')
    $this.Name = $name
    $this.Id = $resourceId
    $this.VmSize = $vmSize
    $this.OsType = $osType
```

Class basics – Interfaces

```
class MyComparable : system.IComparable
{
    [int] CompareTo([object] $obj)
    {
       return 0;
    }
}
```

Working with classes

- Inside of scripts
- Inside of modules
- Multiple classes in multiple files
- Load classes only with: using module .\TableStorage.psm1



Cross platform

- Based on .Net (core)
- PowerShell >= 5 compatible
- Cross platform PowerShell 6/7
- COM/Win32 APIs not available
- Debugging with VSCode/WSL

Cross platform - setup

- Activate and install WSL (e.g. Ubuntu 18.04 LTS)
- Install PowerShell in Ubuntu (aka.ms/install-powershell)
- Install "Remote WSL" (ms-vscode-remote.remote-wsl) extension
- Open new WSL Windows and click "Install in WSL" for PS extension



Azure Storage Table Wrapper

- Use Table Storage Client from within PowerShell
- Abstract handling of .Net assemblies
- Use .Net Namespaces imported by AzTable
- Alternative to PowerShell Module AzTable (CosmosDB)

Microsoft Implementation

AzureRmStorageTableCoreHelper.psm1:

```
if ($UpdateExisting)
{
    return ($Table.Execute([Microsoft.Azure.Cosmos.Table.TableOperation]::InsertOrReplace($entity)))
}
else
{
    return ($Table.Execute([Microsoft.Azure.Cosmos.Table.TableOperation]::Insert($entity)))
}
```

Source: https://github.com/paulomarquesc/AzureRmStorageTable



Advanced topics – Create setter/getter

Constructor:

Method:

Links

https://github.com/it-insights/PowerShell-Examples
https://www.itinsights.org/Create-Advanced-Ping-Class-in-PowerShell/
https://docs.microsoft.com/enus/powershell/module/microsoft.powershell.core/about/about classes
https://github.com/PowerShell/PowerShell

https://github.com/paulomarquesc/AzureRmStorageTable

Questions