String Handling

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clone | CompareTo | Contains | CopyTo | EndsWith |
| Equals | GetEnumerator | GetHashCode | GetType | GetTypeCode |
| IndexOf | IndexOfAny | Insert | IsNormalized | LastIndexOf |
| LastIndexOfAny | Normalize | PadLeft | PadRight | Remove |
| Replace | Split | StartsWith | Substring | ToBoolean |
| ToByte | ToChar | ToCharArray | ToDateTime | ToDecimal |
| ToDouble | ToInt16 | ToInt32 | ToInt64 | ToLower |
| ToLowerInvariant | ToSByte | ToSingle | ToString | ToType |
| ToUInt16 | ToUInt32 | ToUInt64 | ToUpper | ToUpperInvariant |
| Trim | TrimEnd | TrimStart |  |  |

Comparing strings

In general, you can work with the same comparison operators as for numerical values to determine differences between strings. Primarily, this includes -eq and -ne, as well as -like, which supports wildcards.

String objects also offer methods for this purpose. If the first string is “bigger” than the second string (that is, if it comes first in the sort order), the cmdlet returns 1; if the first string is smaller, the result is ‑1.

("Hello world").CompareTo("Hello" + " " + "world")

In the above example, CompareTo returns 0 because the strings are identical. In contrast, the comparable call with Equals returns True:

("Hello world").Equals("Hello" + " " + "world")

## Searching and replacing characters

("Hello World").Replace("Hello","New")

("Hello world").Contains("ll")

("Hello world").IndexOf("ll")

## Extracting substrings

("Hello world").Substring(2,5)

("Hello world").Remove(2,3)

$dbUserName.length

Split

$string = 'FirstPart SecondPart'

$a,$b = $string.split(' ')

$a

$b

Traverse folder and files

Dir

## Nested folders

Get-ChildItem -Path C:\data\ScriptingGuys –recurse

function GetFiles($path = $pwd, [string[]]$exclude)

{

foreach ($item in Get-ChildItem $path)

{

if ($exclude | Where {$item -like $\_}) { continue }

if (Test-Path $item.FullName -PathType Container)

{

$item

GetFiles $item.FullName $exclude

}

else

{

$item

}

}

}

Get-ChildItem "C:\Users\gerhardl\Documents\My Received Files" -Filter \*.log |

Foreach-Object {

$content = Get-Content $\_.FullName

#filter and save content to the original file

$content | Where-Object {$\_ -match 'step[49]'} | Set-Content $\_.FullName

#filter and save content to a new file

$content | Where-Object {$\_ -match 'step[49]'} | Set-Content ($\_.BaseName + '\_out.log')

}

$files = Get-ChildItem "C:\Users\gerhardl\Documents\My Received Files\"

for ($i=0; $i -lt $files.Count; $i++) {

$outfile = $files[$i].FullName + "out"

Get-Content $files[$i].FullName | Where-Object { !($\_ -match 'step4' -or $\_ -match 'step9') } | Set-Content $outfile

}

**$DB = Get-Content $file**

**foreach ($Data in $DB) {**

**$First, $Second, $Third, $Fourth = $Data -split ',' -replace '^\s\*|\s\*$'**

**write-host "First is: "$First**

**write-host "Second is: "$Second**

**write-host "Third is: "$Third**

**write-host "Fourth is: "$Fourth**

**Write-Host ""**

**}**