Automating tasks with loops and script execution



Andrew Mallett

Author and software developer

Concentrate on cleaning data files, showing the versatility of PowerShell

Script execution policies

Looping using foreach and ForEach-Object

String methods and supplementing them with the culture object

Correct name casing

Output as CSV



Script execution policy

Another advantage of VS Code is the ease of script execution. Scripts can run inside of VS Code but not by default, in a PowerShell terminal

Execution policy

```
Get-ExecutionPolicy #Effective
Get-ExecutionPolicy -List #View hierarchy
Set-ExecutionPolicy -Scope CurrentUser RemoteSigned
```

Within an IDE, scripts typically run with the necessary execution rights already in place. However, when running a script from a PowerShell terminal, you may need to explicitly grant permission.

Setting the execution policy to **"RemoteSigned"** means that scripts created locally can run without a signature, but scripts downloaded from external sources must be signed by a trusted publisher.



Demo

Script execution

Use foreach when you already have data. Use ForEach-Object when working in the pipeline.

foreach

Language keyword

```
$files = ls

foreach ($file in $files) {
  $file.Name
}
```

ForEach-Object

Cmdlet

```
ls | ForEach-Object {
  $_.Name
}
```



Demo

Using iteration or loops



Cleaning data

You may have data in a CSV file

If you are lucky it is clean

Mainly, it will be messy

PowerShell can transform your data

PowerShell gives us ToUpper() and ToLower() directly on strings.

For Title Case, we step into .NET and use the TextInfo class from the current culture.

Casing tools

```
$text = "hello WORLD"
$text.ToUpper()
$text.ToLower()
$culture = [System.Globalization.CultureInfo]::CurrentCulture
$culture.TextInfo.ToTitleCase($text.ToLower())
```

For ToTitleCase() to work reliably, convert all text to lowercase first.



Demo

Working with case tools



Cleaning data

There are 1000 lines that are messy in the names.txt file. Using PowerShell and the knowledge that you have gained, you can create a clean CSV file.

Putting it all together (again)

```
$culture = [System.Globalization.CultureInfo]::CurrentCulture
Get-Content names.txt | ForEach-Object {
    $parts = $_ -split ","
    $lastname  = $parts[0].Trim().ToUpper()
    $firstname = $culture.TextInfo.ToTitleCase($parts[1].Trim().ToLower())
    $dob = $parts[2].Trim()
    [PSCustomObject]@{
        Lastname  = $lastname
        Firstname = $firstname
        DateOfBirth = $dob
    }
} | Export-Csv employees.csv -NoTypeInformation
```

Read in the messy file and output as a clean CSV file with headers



Demo

Cleaning data

Pluralsight

Thank you for your time!

