Article
Andrew Krammen · Nov 4, 2016 4m read

Extracting pButtons data to a CSV file on Windows

This is a follow-up to Murray's article for scripting pButtons data extraction on Unix/MacOS:
Extracting pButtons data to CSV for UNIX/MacOS

Extracting pButtons data to a CSV file on Windows

PowerShell scripting was used in this article because it is available by default on Windows platforms.

The script, ***ExtractpButtons.ps1*** , will search the given pButtons file until it finds the marker for the beginning of the section. The section will be printed line-by-line, from the header to the line before the section end marker.

Additional parameters are passed in to the script to include the header line and convert the output to CSV format (spaces will be replaced with commas and any leading spaces on each line are removed). The mgstat and win-perfmon sections are already in comma-delimited format whereas vmstat is not.

```
<# Extract_pButtons.ps1
#
# Usage:    ./Extract_pButtons.ps1 <input pButtons> <input searchName> <header text>
<convert to CSV>
#
# pButtons has the following markers in the html source:
# Metrics                          Parameters to pass
# -------                          ------------------
# mgstat                             mgstat Date 0
# Windows performance monitor        win_perfmon Time 0
# vmstat                             vmstat fre 1
#
# Usage examples:
#   Search for mgstat and redirect to .csv file:
#     ./Extract_pButtons.ps1 CACHE_20161018_0001_24hours.html mgstat Date 0 > mgstatO
uptut.csv
#
#   Search for vmstat, convert to CSV, include header, and redirect to .csv file:
#     ./Extract_pButtons.ps1 CACHE_20161018_0001_24hours.html vmstat fre 1 > vmstatOu
tput.csv
#
#   Search for win_perfmon and redirect to .csv file:
#     ./Extract_pButtons.ps1 CACHE_20161018_0001_24hours.html win_perfmon Time 0 > wi
nperfmonOutput.csv
#>

Param(
    [string]$pButtonFilePath,
    [string]$searchName,
    [string]$headerText,
    [bool]$convertToCSV
)
```

```
# Resolve the full path of the pButtons file and open for reading.
$pButtonFilePath = Resolve-Path $pButtonFilePath
$fileReader = [System.IO.File]::OpenText($pButtonFilePath)

try {
    $printHeader = 0
    $foundString = 0
    $line = $fileReader.ReadLine()
    while (($line -ne $null) -and (-Not $line.Contains("end_"+$searchName))) {
        # When desired section is found, read each line until the end of section is f
ound
        if ($line.Contains("beg_"+$searchName)) {
            $foundString = 1
        }

        # Start extracting at the header line
        if (($foundString -eq 1) -and ($line.Contains($headerText))) {
            $printHeader = 1
        }

        # Print out the current line and read the next line
        if ($printHeader -eq 1) {
            # If specified, convert spaces to commas (remove leading space)
            if ($convertToCSV -eq 1) {
                $line = $line.Trim() -Replace '\s+',','
            }
            # Print out the current line
            $line
        }
        $line = $fileReader.ReadLine()
     }
}
finally {
    $fileReader.Close()
}
```

Note: Windows PowerShell default execution policy does not permit any scripts to run. At PowerShell prompt type get-help about_signing to understand PowerShell policy and instructions for enabling script execution.

Extracting multiple pButtons files in a directory

PowerShell does not seem to have a simple approach to looping through all files in a directory so I created a secondary script to provide this functionality: ***Extract_Directory.ps1***

Given a directory path, the script filters for files ending in .html and calls ***./Extract_pButtons.ps1*** for each file. This script also accepts the same parameters for printing the header, converting to CSV, and also allows for an output file name to append for each pButtons file.

```
<# Extract_Directory.ps1
#
# Usage:  ./Extract_Directory.ps1 <input_Directory> <input_searchName> <header text>
<convert to CSV><output filename>
#
# Process a list of html files for mgstat data:
#   ./Extract_Directory.ps1 C:\Users\akrammen\Documents mgstat Date 0 mgstatOutput.cs
v
```

```
#
# Process a list of html files for vmstat data:
#   ./Extract_Directory.ps1 C:\Users\akrammen\Documents vmstat fre 1 vmstatOutput.csv
#>
Param(
    [string]$directoryPath,
    [string]$searchName,
    [string]$headerText,
    [bool]$convertCSV,
    [string]$outputFileName
)

# Get all files from the given directory, filter *.html extensions
$filelist = Get-ChildItem $directoryPath -Filter *.html

# For each pButtons file in the directory, extract the section data
for ($i=0; $i -lt $filelist.Count; $i++) {
    $filename = $filelist[$i].FullName

    # If an output file name is defined, write to the new file
    if ($outputFileName.Length -gt 0) {
        $outfile = $filename.SubString(0,$filename.IndexOf(".html")) + "_" + $outputF
ileName
        # Call Extract_pButtons.ps1 for each file
        ./Extract_pButtons.ps1 $filename $searchName $headerText $convertCSV > $outfi
le
    } else {    # Print results to the screen
        # Call Extract_pButtons.ps1 for each file
        ./Extract_pButtons.ps1 $filename $searchName $headerText $convertCSV
    }
}
```

Summary

As Murray suggested, review the pButtons .html files to gain a better understanding of what information is contained within as well as how the file is structured. This knowledge will not only make scripting the data manipulation easier but may help discover data that is useful for performance investigations.

PowerShell is not the only option available for Windows but exists by default. I created a similar script using VBScript with a similar level of complexity which can be included if anyone is interested.

#CSV #Performance #System Administration #Tips & Tricks #Other