MyWebUniversity

python PowerShell FPDF Library PDF generator

Full credit is given to the above companies including the OS that this PDF file was generated!

PowerShell Get-Help on command 'ConvertTo-Csv'

PS C:\Users\wahid> Get-Help ConvertTo-Csv

NAME

ConvertTo-Csv

SYNOPSIS

Converts .NET objects into a series of character-separated value (CSV) strings.

SYNTAX

```
    ConvertTo-Csv [-InputObject] <System.Management.Automation.PSObject>
    [[-Delimiter] <System.Char>] [-NoTypeInformation] [<CommonParameters>]

    ConvertTo-Csv [-InputObject] <System.Management.Automation.PSObject>
    [-NoTypeInformation] [-UseCulture] [<CommonParameters>]
```

DESCRIPTION

The `ConvertTo-CSV` cmdlet returns a series of character-separated value (CSV) strings that represent the objects that you submit. You can then use the `ConvertFrom-Csv` cmdlet to recreate objects from the CSV strings. The objects converted from CSV are string values of the original objects that contain property values and no methods.

You can use the `Export-Csv` cmdlet to convert objects to CSV strings. `Export-CSV` is similar to `ConvertTo-CSV`, except that it saves the CSV strings to a file.

The `ConvertTo-CSV` cmdlet has parameters to specify a delimiter other than a comma or use the current culture as the delimiter.

PARAMETERS

-Delimiter <System.Char>

Specifies the delimiter to separate the property values in CSV strings. The default is a comma (`,`). Enter a character, such as a colon (`:`). To specify a semicolon (`;`) enclose it in single quotation marks.

-InputObject <System.Management.Automation.PSObject>

Specifies the objects that are converted to CSV strings. Enter a variable that contains the objects or type a command or expression that gets the objects. You can also pipe objects to `ConvertTo-CSV`.

-NoTypeInformation <System.Management.Automation.SwitchParameter>

Removes the #TYPE information header from the output. This parameter became the default in PowerShell 6.0 and is included for backwards compatibility.

-UseCulture <System.Management.Automation.SwitchParameter>

Uses the list separator for the current culture as the item delimiter. To find the list separator for a culture, use the following command: `(Get-Culture).TextInfo.ListSeparator`.

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable,

OutBuffer, PipelineVariable, and OutVariable. For more information, see
about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

------------- Example 1: Convert an object to CSV -------------

```
Get-Process -Name 'PowerShell' | ConvertTo-Csv -NoTypeInformation
```

```
"Name","SI","Handles","VM","WS","PM","NPM","Path","Company","CPU","FileVersion"
, ...
"powershell","11","691","2204036739072","175943680","132665344","33312", ...
```

The `Get-Process` cmdlet gets the Process object and uses the Name parameter
to specify the PowerShell process. The process object is sent down the
pipeline to the `ConvertTo-CSV` cmdlet. The `ConvertTo-CSV` cmdlet converts
the object to CSV strings. The NoTypeInformation parameter removes the #TYPE
information header from the CSV output.

--------- Example 2: Convert a DateTime object to CSV ---------

```
$Date = Get-Date
ConvertTo-Csv -InputObject $Date -Delimiter ';' -NoTypeInformation
```

```
"DisplayHint";"DateTime";"Date";"Day";"DayOfWeek";"DayOfYear";"Hour";"Kind";"Mi
llisecond";"Minute";"Month";"Second";"Ticks";"TimeOfDay";"Year"
"DateTime";"Friday, January 4, 2019 14:40:51";"1/4/2019 00:00:00";"4";"Friday";
"4";"14";"Local";"711";"40";"1";"51";"636822096517114991";"14:40:51.7114991";"2
019"
```

The `Get-Date` cmdlet gets the DateTime object and saves it in the `$Date`
variable. The `ConvertTo-Csv` cmdlet converts the DateTime object to strings.
The InputObject parameter uses the DateTime object stored in the `$Date`
variable. The Delimiter parameter specifies a semicolon to separate the string
values. The NoTypeInformation parameter removes the #TYPE information header
from the CSV output.

------ Example 3: Convert the PowerShell event log to CSV ------

```
(Get-Culture).TextInfo.ListSeparator
Get-WinEvent -LogName 'Windows PowerShell' | ConvertTo-Csv -UseCulture
-NoTypeInformation
```

```
,
"Message","Id","Version","Qualifiers","Level","Task","Opcode","Keywords","Recor
dId", ...
"Error Message = System
error","403",,"0","4","4",,"36028797018963968","46891","PowerShell", ...
```

The `Get-Culture` cmdlet uses the nested properties TextInfo and ListSeparator and displays the current culture's default list separator. The `Get-WinEvent` cmdlet gets the event log objects and uses the LogName parameter to specify the log file name. The event log objects are sent down the pipeline to the `ConvertTo-Csv` cmdlet. The `ConvertTo-Csv` cmdlet converts the event log objects to a series of CSV strings. The UseCulture parameter uses the current culture's default list separator as the delimiter. The NoTypeInformation parameter removes the #TYPE information header from the CSV output.

REMARKS

To see the examples, type: "get-help ConvertTo-Csv -examples".
For more information, type: "get-help ConvertTo-Csv -detailed".
For technical information, type: "get-help ConvertTo-Csv -full".
For online help, type: "get-help ConvertTo-Csv -online"