NAME
    Import-Excel

SYNOPSIS
    Create custom objects from the rows in an Excel worksheet.

SYNTAX
    Import-Excel [-Path] <String> [[-WorksheetName] <String>] [-StartRow <Int32>] [
-EndRow <Int32>] [-StartColumn <Int32>] [-EndColumn <Int32>]
    [-DataOnly] [-Password <String>] [<CommonParameters>]

    Import-Excel [-Path] <String> [[-WorksheetName] <String>] -HeaderName <String[]
> [-StartRow <Int32>] [-EndRow <Int32>] [-StartColumn <Int32>]
    [-EndColumn <Int32>] [-DataOnly] [-Password <String>] [<CommonParameters>]

    Import-Excel [-Path] <String> [[-WorksheetName] <String>] -NoHeader [-StartRow
<Int32>] [-EndRow <Int32>] [-StartColumn <Int32>] [-EndColumn
    <Int32>] [-DataOnly] [-Password <String>] [<CommonParameters>]

DESCRIPTION
    The Import-Excel cmdlet creates custom objects from the rows in an Excel worksh
eet. Each row represents one object. All of this is possible
    without installing Microsoft Excel and by using the .NET library 'EPPLus.dll'.

    By default, the property names of the objects are retrieved from the column hea
ders. Because an object cannot have a blank property name, only
    columns with column headers will be imported.

    If the default behavior is not desired and you want to import the complete work
sheet 'as is', the parameter '-NoHeader' can be used. In case you
    want to provide your own property names, you can use the parameter '-HeaderName
'.

PARAMETERS
    -Path <String>
        Specifies the path to the Excel file.

    -WorksheetName <String>
        Specifies the name of the worksheet in the Excel workbook to import. By def
ault, if no name is provided, the first worksheet will be imported.

    -HeaderName <String[]>
        Specifies custom property names to use, instead of the values defined in th
e column headers of the TopRow.

        In case you provide less header names than there is data in the worksheet,
then only the data with a corresponding header name will be
        imported and the data without header name will be disregarded.

        In case you provide more header names than there is data in the worksheet,
then all data will be imported and all objects will have all the
        property names you defined in the header names. As such, the last propertie
s will be blanc as there is no data for them.

    -NoHeader [<SwitchParameter>]
        Automatically generate property names (P1, P2, P3, ..) instead of the ones
defined in the column headers of the TopRow.

        This switch is best used when you want to import the complete worksheet 'as
 is' and are not concerned with the property names.

    -StartRow <Int32>
        The row from where we start to import data, all rows above the StartRow are
 disregarded. By default this is the first row.

        When the parameters '-NoHeader' and '-HeaderName' are not provided, this ro
w will contain the column headers that will be used as property
        names. When one of both parameters are provided, the property names are aut
omatically created and this row will be treated as a regular row
        containing data.

    -EndRow <Int32>
        By default all rows up to the last cell in the sheet will be imported. If s

pecified, import stops at this row.

    -StartColumn <Int32>
        The number of the first column to read data from (1 by default).

    -EndColumn <Int32>
        By default the import reads up to the last populated column, -EndColumn tel
ls the import to stop at an earlier number.

    -DataOnly [<SwitchParameter>]
        Import only rows and columns that contain data, empty rows and empty column
s are not imported.

    -Password <String>
        Accepts a string that will be used to open a password protected Excel file.

    <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 --------------------------

    PS C:\>Import data from an Excel worksheet. One object is created for each row.
 The property names of the objects consist of the column names
    defined in the first row. In case a column doesn't have a column header (usuall
y in row 1 when '-StartRow' is not used), then the unnamed columns
    will be skipped and the data in those columns will not be imported.

    | File: Movies.xlsx   -   Sheet: Actors |

    | A   B   C |
    |1   First Name   Address |
    |2   Chuck   Norris   California |
    |3   Jean-Claude   Vandamme   Brussels |

    PS C:\> Import-Excel -Path 'C:\Movies.xlsx' -WorkSheetname Actors

    First Name: Chuck
    Address   : California

    First Name: Jean-Claude
    Address   : Brussels

    Notice that column 'B' is not imported because there's no value in cell 'B1' th
at can be used as property name for the objects.

    -------------------------- EXAMPLE 2 --------------------------

    PS C:\>Import the complete Excel worksheet 'as is' by using the '-NoHeader' swi
tch. One object is created for each row. The property names of the
    objects will be automatically generated (P1, P2, P3, ..).

    | File: Movies.xlsx   -   Sheet: Actors |

    | A   B   C |
    |1   First Name   Address |
    |2   Chuck   Norris   California |
    |3   Jean-Claude   Vandamme   Brussels |

    PS C:\> Import-Excel -Path 'C:\Movies.xlsx' -WorkSheetname Actors -NoHeader

    P1: First Name
    P2:
    P3: Address

    P1: Chuck
    P2: Norris
    P3: California

```
    P1: Jean-Claude
    P2: Vandamme
    P3: Brussels

    Notice that the column header (row 1) is imported as an object too.



    -------------------------- EXAMPLE 3 --------------------------

    PS C:\>Import data from an Excel worksheet. One object is created for each row.
 The property names of the objects consist of the names defined in
    the parameter '-HeaderName'. The properties are named starting from the most le
ft column (A) to the right. In case no value is present in one of
    the columns, that property will have an empty value.

    ------------------------------------------------------------
    | File: Movies.xlsx            -           Sheet: Movies |
    ------------------------------------------------------------
    |           A            B            C          D       |
    |1     The Bodyguard   1992           9                   |
    |2     The Matrix      1999           8                   |
    |3                                                        |
    |4     Skyfall         2012           9                   |
    ------------------------------------------------------------

    PS C:\> Import-Excel -Path 'C:\Movies.xlsx' -WorkSheetname Movies -HeaderName '
Movie name', 'Year', 'Rating', 'Genre'

    Movie name: The Bodyguard
    Year      : 1992
    Rating    : 9
    Genre     :

    Movie name: The Matrix
    Year      : 1999
    Rating    : 8
    Genre     :

    Movie name:
    Year      :
    Rating    :
    Genre     :

    Movie name: Skyfall
    Year      : 2012
    Rating    : 9
    Genre     :

    Notice that empty rows are imported and that data for the property 'Genre' is n
ot present in the worksheet. As such, the 'Genre' property will be
    blanc for all objects.



    -------------------------- EXAMPLE 4 --------------------------

    PS C:\>Import data from an Excel worksheet. One object is created for each row.
 The property names of the objects are automatically generated by
    using the switch '-NoHeader' (P1, P@, P#, ..). The switch '-DataOnly' will spee
d up the import because empty rows and empty columns are not
    imported.

    ------------------------------------------------------------
    | File: Movies.xlsx            -           Sheet: Movies |
    ------------------------------------------------------------
    |           A            B            C          D       |
    |1     The Bodyguard   1992           9                   |
    |2     The Matrix      1999           8                   |
    |3                                                        |
    |4     Skyfall         2012           9                   |
    ------------------------------------------------------------

    PS C:\> Import-Excel -Path 'C:\Movies.xlsx' -WorkSheetname Movies -NoHeader -Da
```

taOnly

    P1: The Bodyguard
    P2: 1992
    P3: 9

    P1: The Matrix
    P2: 1999
    P3: 8

    P1: Skyfall
    P2: 2012
    P3: 9

    Notice that empty rows and empty columns are not imported.

    -------------------------- EXAMPLE 5 --------------------------

    PS C:\>Import data from an Excel worksheet. One object is created for each row.
 The property names are provided with the '-HeaderName' parameter.
    The import will start from row 2 and empty columns and rows are not imported.

    | File: Movies.xlsx   -   Sheet: Actors |

    | A   B   C   D |
    |1   Chuck   Norris   California |
    |2 |
    |3   Jean-Claude   Vandamme   Brussels |

    PS C:\> Import-Excel -Path 'C:\Movies.xlsx' -WorkSheetname Actors -DataOnly -He
aderName 'FirstName', 'SecondName', 'City' -StartRow 2

    FirstName : Jean-Claude
    SecondName: Vandamme
    City   : Brussels

    Notice that only 1 object is imported with only 3 properties. Column B and row
2 are empty and have been disregarded by using the switch
    '-DataOnly'. The property names have been named with the values provided with t
he parameter '-HeaderName'. Row number 1 with 'Chuck Norris' has
    not been imported, because we started the import from row 2 with the parameter
'-StartRow 2'.

    -------------------------- EXAMPLE 6 --------------------------

    >

    PS> ,(Import-Excel -Path .\SysTables_AdventureWorks2014.xlsx) |
        Write-SqlTableData -ServerInstance localhost\DEFAULT -Database BlankDB -Sch
emaName dbo -TableName MyNewTable_fromExcel -Force

        Imports data from an Excel file and pipe the data to the Write-SqlTableData
 to be INSERTed into a table in a SQL Server database.
        The ",(...)" around the Import-Excel command allows all rows to be import
ed from the Excel file, prior to pipelining to the
    Write-SqlTableData cmdlet. This helps prevent a RBAR scenario and is important
 when trying to import thousands of rows.
        The -Force parameter will be ignored if the table already exists. However,
 if a table is not found that matches the values provided by
    -SchemaName and -TableName parameters, it will create a new table in SQL Server
 database. The Write-SqlTableData cmdlet will inherit the column
    names & datatypes for the new table from the object being piped in.
        NOTE: You need to install the SqlServer module from the PowerShell Gallery
in oder to get the Write-SqlTableData cmdlet.

REMARKS

```
To see the examples, type: "get-help Import-Excel -examples".
For more information, type: "get-help Import-Excel -detailed".
For technical information, type: "get-help Import-Excel -full".
For online help, type: "get-help Import-Excel -online"
```