How to Import Excel in PowerShell

lazyadmin.nl/powershell/import-excel-in-powershell

June 10, 2024

When working with PowerShell we can use Excel files to import data into systems or to use it as a reference list, of users, for example, to update or get settings. To do this we are going to use the ImportExcel module.

Even though we can import Excel data with Import-CSV, the advantage of the ImportExcel module is that you don't need to convert the Excel file and you have more options when it comes to select the data that you need.

In this article, we are going to take a look at how to import an Excel file with PowerShell. We will look at the different features of the modules and I will give you some examples.

Import Excel in PowerShell

To import an excel file into PowerShell, we will first need to install the ImportExcel module. The module is available in the PowerShell Gallery, so we can simply install it with the `Install-Module` cmdlet.

Use the command below to install the module, keep in mind that you will need to run PowerShell in Elevated mode (admin mode) to install modules, or you can use the `scope` parameter to install it for the current user only.

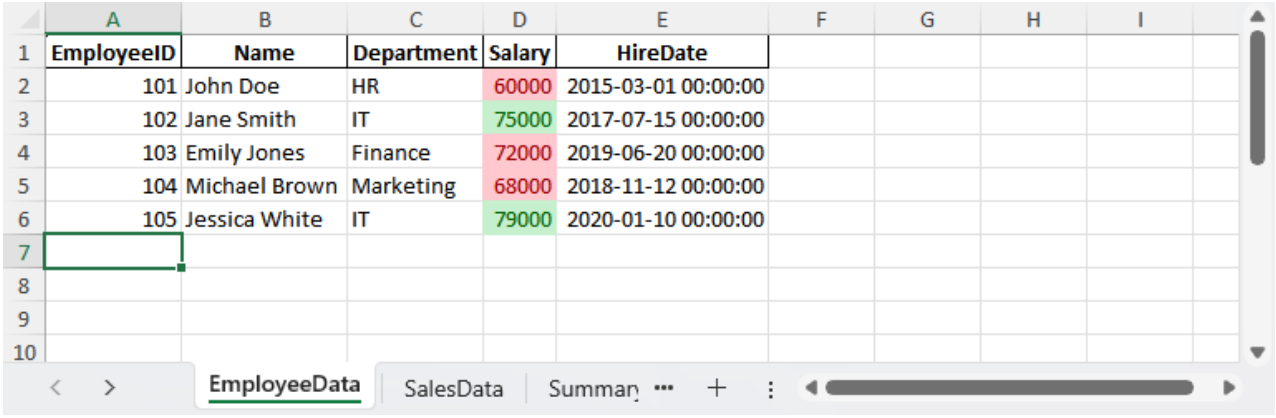
```
# Install the module for all users
Install-Module -Name ImportExcel
# Install the module for the current user only
Install-Module -Name ImportExcel -Scope CurrentUser
```

After you have installed the module, you can load the module using the `Import-Module` cmdlet. If you tend to use the module quite often, then you can also add the module to your PowerShell Profile, so it will always load when you open PowerShell.

```
Import-Module ImportExcel
```

Importing an Excel File

The ImportExcel module comes with a couple of features to help you with importing your Excel file into PowerShell. For the examples below, I am going to use the following Excel file, which has multiple sheets and a date field:



EmployeeID	Name	Department	Salary	HireDate
101	John Doe	HR	60000	2015-03-01 00:00:00
102	Jane Smith	IT	75000	2017-07-15 00:00:00
103	Emily Jones	Finance	72000	2019-06-20 00:00:00
104	Michael Brown	Marketing	68000	2018-11-12 00:00:00
105	Jessica White	IT	79000	2020-01-10 00:00:00

Before we are going to take a look at how to use the ImportExcel module, let's first take a look at the parameters that we can use to import an Excel file:

- **Path** – Required, location of the Excel file
- **ExcelPackage** – Allows you to provide an Excel Package Object instead of a path
- **WorksheetName** – The name of the Excel worksheet that you want to import.
- **HeaderName** – Specify custom header names instead of the ones in the Excel file
- **NoHeader** – Imports the Excel file without using the first row as header.
- **ImportColumns** – Specify which columns to import
- **StartRow** – Defines the row where to start importing from, all rows above will be ignored
- **EndRow** – Import stops at the specified row
- **StartColumn** – Defines from which column to start importing from
- **EndColumn** – Import start at the specified column
- **DataOnly** – Import only rows and columns that contain data. Empty ones will be ignored
- **AsText** – Imports specified columns as text
- **AsDate** – Converts specified columns to a data object
- **Password** – Password to open protected Excel file

When you import an Excel file into PowerShell, then the first sheet is imported only by default. All the headers, rows, and columns are imported, but we can change this if needed. So to simply important an Excel file, you only need to specify the path to the Excel file:

```
import-Excel -Path "C:\temp\employeeData.xlsx" | ft
# Results
EmployeeID Name Department Salary HireDate
---------- ---- ---------- ------ --------
101,00 John Doe HR 60000,00 42064,00
102,00 Jane Smith IT 75000,00 42931,00
103,00 Emily Jones Finance 72000,00 43636,00
104,00 Michael Brown Marketing 68000,00 43416,00
105,00 Jessica White IT 79000,00 43840,00
```

Fixing date fields

As you can see in the results above, we have an issue with the HireDate column. All the content is imported as text, which renders the date field useless. You can solve this by marking the date column as a date field with the `-AsDate` parameter:

```
Import-Excel -Path "C:\temp\employeeData.xlsx" -AsDate "HireDate" | ft
# Results
EmployeeID Name Department Salary HireDate
---------- ---- ---------- ------ --------
101,00 John Doe HR 60000,00 1-3-2015 00:00:00
102,00 Jane Smith IT 75000,00 15-7-2017 00:00:00
103,00 Emily Jones Finance 72000,00 20-6-2019 00:00:00
104,00 Michael Brown Marketing 68000,00 12-11-2018 00:00:00
105,00 Jessica White IT 79000,00 10-1-2020 00:00:00
```

Import Specific Worksheet

If your Excel workbook has multiple sheets and you want to import the data from another sheet instead of the first one, then we can specify the sheet name with the -WorksheetName parameter.

To import the SalesData from our example worksheet, we can use the command below:

```
Import-Excel -Path "C:\temp\employeeData.xlsx" -WorksheetName "salesdata" | ft
```

In this example, we know the worksheet name that we want to import. But how do you know from the command line if your Excel file has multiple worksheets? And which name the worksheets has? To find this out, we can use the cmdlet `Get-ExcelSheetInfo`, which is also part of the ImportExcel module.

The cmdlet only requires the path to the Excel file and will output all the worksheet, their name and index numbers, and if the worksheet is visible or hidden:

```
Get-ExcelSheetInfo -Path C:\temp\employeeData.xlsx
# Result
Name Index Hidden Path
---- ----- ------ ----
EmployeeData 1 Visible C:\temp\employeeData.xlsx
SalesData 2 Visible C:\temp\employeeData.xlsx
Summary 3 Visible C:\temp\employeeData.xlsx
```

In this example, each worksheet has a different structure, so we can't import them all easily into one object. But if you want to import each worksheet from the Excel file, you can simply use a ForEach loop to import them all:

```
$excelSheets = Get-ExcelSheetInfo -Path "C:\temp\employeeData.xlsx"
$excelSheets | ForEach {
Import-Excel -Path "C:\temp\employeeData.xlsx" -WorksheetName $_.Name | ft
```

}

Import Specific Rows and Columns

Sometimes you only need a part of the data from an Excel file. With the ImportExcel module, you can specify what you want to import and what not.

To do this we can use the cmdlets `StartRow`, `EndRow` and `StartColumn`, `EndColumn`. So if we want to import only the name and department from our Excel file, and only the first three users, we can do the following:

```
Import-Excel -Path "C:\temp\employeeData.xlsx" -StartRow 1 -EndRow 4 -StartColumn 2 -EndColumn 3
# Results
Name Department
---- ----------
John Doe HR
Jane Smith IT
Emily Jones Finance
```

There are a few things to note about this. When programming we normally start counting with 0, but in this case, we start with 1 for the first row or column. The end row or column that you specify is included in the results.

But as you can see, we have specified row 4 as our end row. This is because our Excel file has headers, which we have included as well.

If you don't want to include the headers, then we will need to use the parameter `-NoHeaders` as well and start with row 2 in this example.

When you only need to import specific columns, then you can use the `-ImportColumns` parameter and specify the index numbers (starting at 1). For example, to import only the first two columns, you can do:

```
import-Excel -Path "C:\temp\employeeData.xlsx" -ImportColumns 1,2
```

Advanced Importing Options

When you need to only read specific cells from an Excel file, then you can use the `Open-ExcelPackage` cmdlet, which is also part of the ImportExcel module. This cmdlet does not really import the Excel file but actually opens it in the background.

This not only allows you to read specific cells in your file but also allows you to modify for example. Important to note is that you will need to close your Excel file when you are done.

So to read the value of a specific cell, we can first open the Excel file:

```
$excel = Open-ExcelPackage -Path "C:\temp\employeeData.xlsx"
```

To retrieve the value of a specific cell, we need to use the worksheet and cell name. So for example, to get the value of cell B4 from our example file:

```
$excel.EmployeeData.Cells["B4"].Value
# Returns
Emily Jones
```

When done, make sure that you close the Excel file with the `Close-ExcelPackage` cmdlet.

Wrapping Up

The ImportExcel module is really a powerful module. In this article, we have only looked at its importing capabilities, but the true strength actually lies in exporting to Excel. This is something we will look into in a completely different article, so make sure that you subscribe to the newsletter or follow me on Facebook.

Hope you liked this article, if you have any questions, just drop a comment below!