The Pipeline Part 2

Parameter binding byproperty

As you already know pipeline parameter binding is the method of stringing together the right cmdlets to perform a task.

Byvalue or Byproperty is the glue that PowerShell uses to allow the output of one command to pass to the input of the second command

In the last lecture we **demonstrated** how to **get commands** through the pipeline **Byvalue.**

In this lecture, we'll demonstrate how PowerShell uses parameter binding byproperty.

We'll introduce two **new commands:**

- **import-csv** and the **new-alias** command.
- The **Import-Csv** cmdlet creates table-like custom objects from the **items** in **CSV files**. Each **column** in the CSV file becomes a **property** of the custom object and the **items in rows** become the **property values**
- The **New-Alias** cmdlet creates a new alias in the current PowerShell session.

let's **Open windows explorer.** From your c: drive **create a folder** named **test.** Close explorer

Now we'll use notepad to **create a csv** (Comma separated value) file and copy that to our test folder.

- From the search bar **type notepad**.
- Now type the following:

  Name, Value
  L, eventlog
  List, get-childitem
  P, ping
  W, winver
  Now from file, click **save-as**. Click the **C: drive** and **click the test folder**.
  For a **file name** type **aliases.csv**
  Change **save as type**, press the down arrow and **change .txt** to **All Files.**
  Now click save., then click close.

Now, the parameter binding process that PowerShell uses to determine what commands pass through the pipeline is basically the same for byproperty and for byvalue, with a few differences.

For command #1 you still use get-member and for command #2 you still use get-help. The difference is that now you are looking for things that are byproperty instead of byvalue.

- **Command 1** which in this **case is import-csv.** We'll use **get-member** to get the property and methods of the object import-csv.

- **Command 2** will **be New-Alias.** We'll us **get-help –full** to get the parameters that take byproperty.

- Now we'll open two PowerShell windows.

- **Type import-csv** -path c:\test\aliases.csv | gm now press return

If you recall from our object's lecture. Objects can have properties **or methods.** But what we're interested in is properties and in **this case** Powershell displays them as noteProperty

- (NoteProperties are just generic properties that are created by Powershell)
- Notice that **Name and Value** match the column headings from **our csv file.**

Now we need to use get-help to determine if **Command 2** or in this case **New-alias** has a parameter that will **bind** with **command 1 ByProperty**

- **Type get-help new-alias -full press return**

- Scroll up until you see a parameter that says ByPropertyName

- So, we see that the **-Value and -Name** parameters **both accept pipeline input BypropertyName**

These two commands should work because:

Command one which **is import-csv** passed **two properties** called **name and value** across the **pipeline to the second command which is new-alias**.

From the second command **new-alias** Get-help told us that the parameters **-name and -value** can take pipeline input by **propertyname**

So, these command should work. We can test that by typing (press return)

- **import-csv -path c:\test\aliases.csv | new-alias -verbose**

We see that the command ran.

now let's **check out our new aliases**

- **Type L security** and our security log is displayed
- Type **list** and the files on the root of my C: drive are displayed
- Type **P 127.0.0.1** and we can test our loopback.
- Press **w** and my current version of windows is displayed.

For the next byproperty example we'll need to create another csv file.

So let's go ahead and **open notepad** again. We'll use the **same process** as we did before.

- **Type svcname, svcstatus Press return**
- **Bits, running**
- **Now for a file name** type **bits.csv** and save this to the **test directory as well**.

This time we'll just run the command.

- Type **import-csv -path c:\test\bits.csv | stop-service**
- **And we see that the command failed, but why?**
- **The error log says** that stop-service Cannot find any service with service name '@{svcName=bits; SvcStatus=running }'.

Now we'll do some troubleshooting

We'll open two powershell windows again

First we'll check command 1, using get-member.

- Type **import-csv -path c:\test\bits | gm**
- **Notice svcName noteproperty** and **string (which in this case is a text string)**

Now lets check the full help on the second command

- **Type get-help stop-service -full**

- **Scroll up** until you see a **parameter** that **supports a string**.

  And **we see** that **-Name supports a string**

  it also accepts pipeline input ByPropertyName and ByValue

- Ok now open our **bits.csv** file.

- **Open file explorer**, click **C:\test**, **right click and choose edit on the bits.csv file**.

  all we have to do is to rename our **column header** from **scvname to name**.

  The column header determines the names of the properties of the object that Import-csv creates and in this case are passed across the pipeline to our second command which is stop-service

- Go ahead del **svc** and then **click save and close notepad.**

- Now run the command again, in this case go ahead and add the parameter -passthru, press return.
- And you see that the status of bits is now stopped, so you can see that the command ran successfully.

Let's review

- The only parameter that **stop-service** (our second cmd) would take bypropertyname was -**name.**

- **Import-csv** (1st Cmd) tried to push SVCname across the pipeline to **stop-service**. But **stop-service** will only match up with -name not svcname.

- **After we modified** the column header inn notepad from SvcName to the **property, Name, Stop-service** was able to use the **parameter -name as a match** and the **command ran.**

Thanks for watching as we'll see you in the next lecture.