TECHMENTOR CONFERENCES

ORLANDO 2010

WINDOWS SERVERS SECURITY VIRTUALIZATION MICROSOFT R2 TECHNOLOGY POWERSHELL SHAREPOINT

Scripting, Error Handling and Debugging in Windows PowerShell

Jeffery Hicks

Prof. PowerShell

Pre-requisites for this presentation:

1) Basic PowerShell Experience
2) Some Scripting Experience will help

Level: Intermediate

Agenda

- Why Script?
- Functions or Scripts?
- Scripts as Tools
- Construct Review
- Error Handling and Traps
- A PowerShell v1 Function
- Scripting PowerShell 2.0
- Debugging
- Putting It All Together
- Best Practices

Sorry..

- We won't be covering these topics
  - Modules
  - Windows Form based scripts
  - Script security
  - Comment Based Help
  - Trace-Command
- But, demos will be available on my blog

Why Script?

- Saves typing
- Automate complex tasks
- Share the love

Functions or Scripts?

- Both can accept parameters
- Functions can be written like cmdlets
- Use Functions to modularize
- Use Scripts to organize

Parameters

- Customize a function or script
- Makes your code re-usable
- Positional by default
- Cast parameters
- Provide default values

Parameters

```
Function Foo {
    Param ([string]$name,[int]$x=3)
     #your code
}
Foo jeff 5
Foo 5
Foo –name jeff –x 5
```

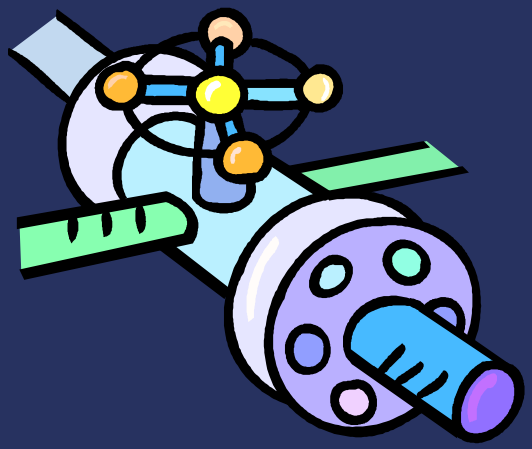
Simple Function

- Let's look at a simple basic function

Functions and the Pipeline

- Begin {}
- Process { $_ }
- End {}
- Scriptblock use is optional

Function Pipeline Demo



Functions or Filters

- A filter is a function with another name
- To pipeline or not, that is the question
- In v2 it's basically all functions

Constructs

- If..(..ElseIf..)..Else
- Switch
- Do While/Until
- For
- ForEach

If

```
If (some condition is true) {
 #do this code
}
```

- No End If
- Bracket positions don't matter

If Else

```
If (some condition is true) {
 #do this code
}
Else {
 #do this code
}
```

If ElseIf

```
If (some condition is true) {
 #do this code }
Elseif (some condition is true) {
 #do this code }
Else {
 #do this code }
```

- No limit to number of ElseIf
- Only first matching code block executes

Switch

```
Switch (some variable) {
 A   {#run this code if variable
matches}
 B   {#run this code if variable
matches}
 C   {#run this code if variable
matches}
 Default  {#run this code if no
matches}
}
```

Switch

- Useful for multiple condition matching
- Supports regular expressions
- Can be used like Select Case
- Default only runs if no match is made
- Help about_switch

Do While

```
Do {
   #run this code
} while (some condition is true)
```

- Script block will always execute at least once
- Make sure your condition will change

Do Until

```
Do {
   #run this code
} Until (some condition is true)
```

- Script block will always execute at least once
- Make sure your condition will change

While

```
while (some condition is true) {
   #run this code
}
```

- Script block may not run
- Make sure your condition will change

For

```
For ($i=0;$i -lt 5;$i++) {
   $i
   #do something
}
```

- Always define starting condition
- Make sure your condition can be met

ForEach

```
$items=get-childitem $env:temp
Foreach ($x in $items) {
  #do something with $x
  write $x.name
}
```

- VBScript Like
- Easier to understand
- Name your own variable

ForEach Pipeline Alternative

```
get-childitem $env:temp | Foreach {
  #do something with each object
  write $_.name
}
```

Error Handling and Trapping

- Errors are Exceptions
- Separate the error message from the exception
- You can only trap exceptions

Error Action

- $errorActionPreference
- -ErrorActionPreference (-ea)
  - Continue
  - SilentlyContinue
  - Stop
  - Inquire

Error Action Demo

Traps

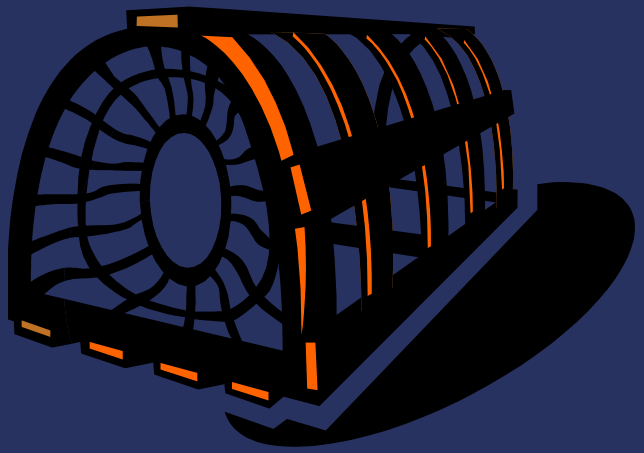
- You can only trap exceptions
- Traps can handle any exception or a specific type
- You can run PowerShell commands when a trap catches an error

```
trap [[<error type>]] {
  #your code here
}
```

Traps – Scope

- PowerShell looks for trap handlers in the current scope first
- Break = terminate current scope and pass exception up the scope hierarchy
- Return = Pass a value (optional) and exit current scope
- Continue = go to next line and no exception
- Help about_trap

Trap Demo



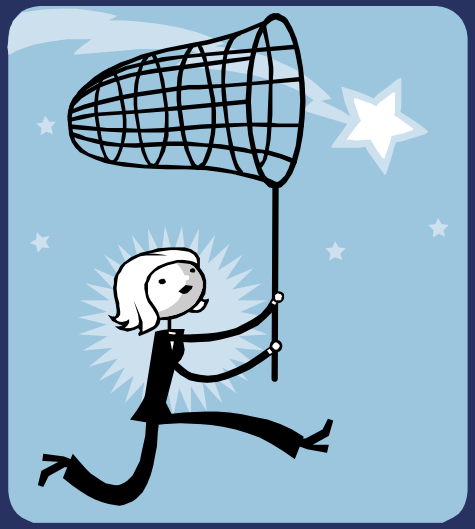
A PowerShell v1 Function

- Let's walk through a sample function

Try to Catch Me

- PowerShell v2
- Try {} Catch {} Finally{}
- You must have at least one Catch or Finally scriptblock
- Help about_try_catch_finally

Try Catch Demo



Scripting PowerShell 2.0

- Advanced Functions
- Use cmdlet binding
- Include comment based help with examples
- Much easier to create cmdlet-like tools
- Help about_functions*

A PowerShell 2.0 Function

- Let's walk through a simple function

Debugging

- Where Reality Does Not Meet Expectation
- Syntax Errors
- Logic Errors
- Squash bugs with a script editor
- Follow a process – Never Guess!

Debugging

- Add Write-* commands to your code
- Debug line by line (trust me)
- Set-PSDebug
- Set-PSBreakpoint

Set-PSDebug

- Step
- Strict
- Trace
  - 0 Turn script tracing off
  - 1 Trace script lines as they are executed
  - 2 Trace everything

PSBreakpoint

- PowerShell 2.0
- *-PSBreakpoint cmdlets
- Works locally only
- PowerShell stops at breakpoints
- Help about_debuggers

Debugging Demo

Scripts as Tools

- Create single purpose and flexible scripts and functions
- Dot Source function script files as needed
  - Profile
  - PowerShell session
  - PowerShell scripts

Putting It All Together

- Let's examine a PowerShell 2.0 function-based tool

Best Practices

- Use full cmdlet and parameter names
- No aliases
- Document
- Include verbose/trace/debugging from the beginning
- Think “object”-ively

Best Practices

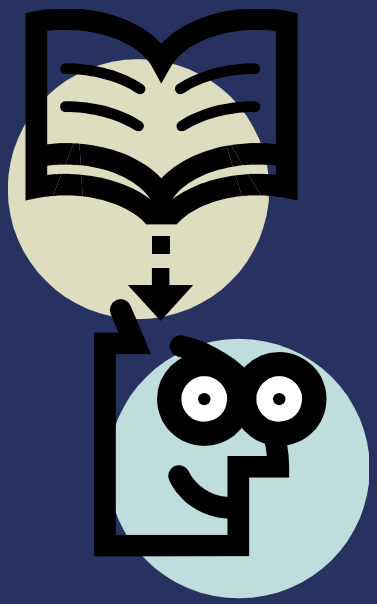
- Think modularly
- Use a standard template and format
- Sign your scripts
- Use source control
- Use a script editor

Resources

- *Windows PowerShell 2.0: TFM* (Don Jones & Jeff Hicks)
- *PowerShell in Action* (Bruce Payette)
- *Windows PowerShell 2.0 Best Practices* (Ed Wilson)
- ScriptingAnswers.com
- PowerShellCommunity.org
- Jdhitsolutions.com/blog
- Blogs.msdn.com/PowerShell

Questions

Now's the time to pick my brain



Thank You

- Enjoy the conference
- Meet people
- Learn a lot
- Have Fun!
- Jdhitsolutions.com/blog