Managing Errors and Exceptions in PowerShell

Liam Cleary

CEO / MICROSOFT MVP / MICROSOFT CERTIFIED TRAINER

@shareplicity www.shareplicity.com | @helloitsliam www.helloitsliam.com

Overview

Understanding Errors and Exception

- How Errors and Exceptions work
- Differences between Errors and Exceptions

Using TRY/CATCH/FINALLY Statements

Understanding Errors and Exception

Error and Exception Terminology

Exception

Throw and Catch

The Call Stack

Terminating and Non-terminating Errors

Swallowing an Exception

Difference Between Errors and Exceptions

Errors

Errors are returned as PowerShell Objects. Provide terminating and non-terminating errors

Exceptions

Exceptions are created when normal error handling cannot handle the issue. Exceptions are typically non-terminating

Terminating and Non-terminating

Terminating

The error generated by the script, functions, or commands, stop or halts the execution

Non-terminating

Generated by internal commands. Normally automatically handled so the error doesn’t terminate the execution of the pipeline

Error Handling Approaches

Throw

Write-Error

-ErrorAction

Try/Catch

Try/Finally

Try/Catch/Finally

Generating and Handling Errors

```
# Generate an Error using "Throw"
Function New-Error
{
    Throw "This is an Error"
}

# Use Write-Error with "-ErrorAction" to Generate Error
Write-Error -Message "This in an Error" -ErrorAction Stop
```

Generating and Handling Errors

```
# Generate an Error using "Throw"
Function New-Error
{
    $number = 0;
    for ($i = 1; $i -le 10; $i++) {
        Write-Host "The current number is: $i"
        Throw "This in an Error";
        $number += $i
    }
}

# Generate Error using "-ErrorAction"
New-Error -ErrorAction Stop
```

The –ErrorAction Parameter Values

Continue
Log's error, then displays error to console, and continues processing

Stop
Log's error, then displays error to console, and then terminates

SilentlyContinue
Log's error, does not display error, and continues processing

Ignore
Does not log error. Does not display error, and continues processing

Demo

Generate Error and Exceptions

Handling Errors and Exceptions

Using TRY/CATCH/FINALLY Statements

TRY/CATCH/FINALLY Statements

TRY/CATCH
Try a section of code and if it throws an error, catch it

TRY/FNALLY
Don't handle the error, simply execute code if an exception occurs

TRY/CATCH/FINALLY
Combination of both throwing errors, and executing code

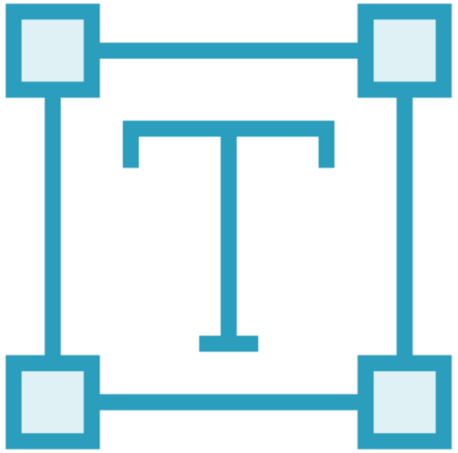
Generating Exceptions

```
# Generate an Error using "Throw"
Function New-Error
{
    Throw "This is an Error"
}

# Use Try/Catch to Capture the Error and raise an Exception
try {
     New-Error
} catch {
    Write-Output "An Exception was Generated"
}
```

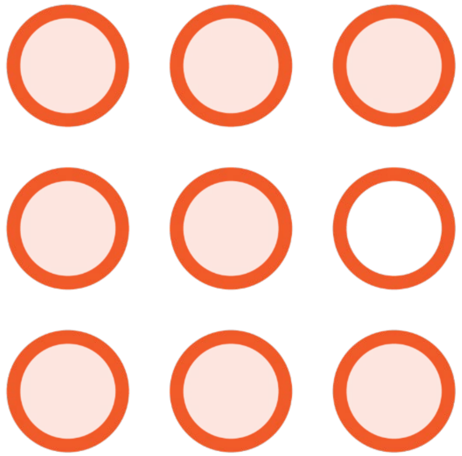
Using Try / Catch / Finally

```
# Use Try/Finally to Capture the Error and Continue Code Execution
try {
     New-Message
} finally {
    Write-Output "Continue Execution"
}

# Use Try/Catch to Capture the Error and raise an Exception
try {
     New-Message
} catch {
    Write-Output "An Exception was Generated"
} finally {
    Write-Output "Continue Execution"
}
```

Typed Exception Handling



Exceptions have a type. You can specify the type of exception you need to catch



Catch multiple exception types with a single try/catch statement

Handling Typed Exceptions

```
$path = "C:\Documents\Code"


try {
    New-Error -Path $path -ErrorAction Stop
}


catch [System.IO.DirectoryNotFoundException],[System.IO.FileNotFoundException]
{
    Write-Output "The Path or File was not found: [$path]"
}


catch [System.IO.IOException]
{
    Write-Output "Error within the selected File: [$path]"
}
```

Demo

Use TRY/CATCH Syntax to Capture Errors and Exceptions

Use TRY/CATCH/FINALLY Syntax to Capture Errors and Exceptions

Summary

Looked at how Errors and Exceptions work

Reviewed the differences between Errors and Exceptions

Used TRY, CATCH, and FINALLY Statements within Functions

Up Next:
Writing Reusable PowerShell Scripts