

9. Manejo de errores

Wednesday, September 9, 2020 8:44 PM

Como regla general, siempre debes de considerar que habrá algo que "romperá" el flujo de tu código. Por ello, lo mejor que podemos hacer es código que se "rompa" responsablemente.

☆ excepciones

Cuando el código encuentra un problema, este cambia su flujo normal, sufre una "disrupción", a este evento que lo genera se le llama *excepción*.

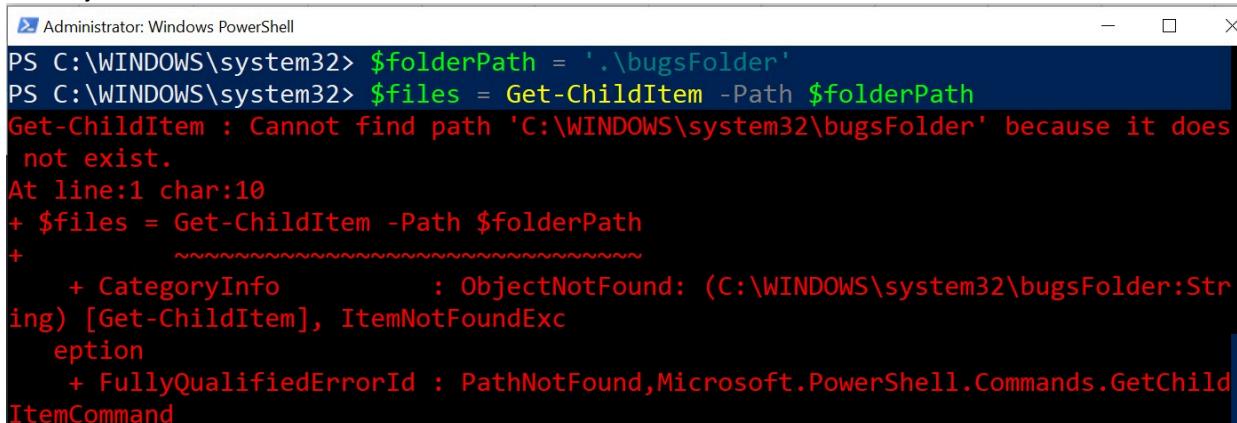
PowerShell tiene 2 tipos de errores:

- **Errores terminantes.** Cualquier error que detiene la ejecución del código.
- **Errores no terminantes.** Ocurren errores en acciones independientes, por lo que no es tan severo como para afectar al resto del código.

Un error no terminante puede convertirse en un error terminante para prevenir la ejecución del resto del código.

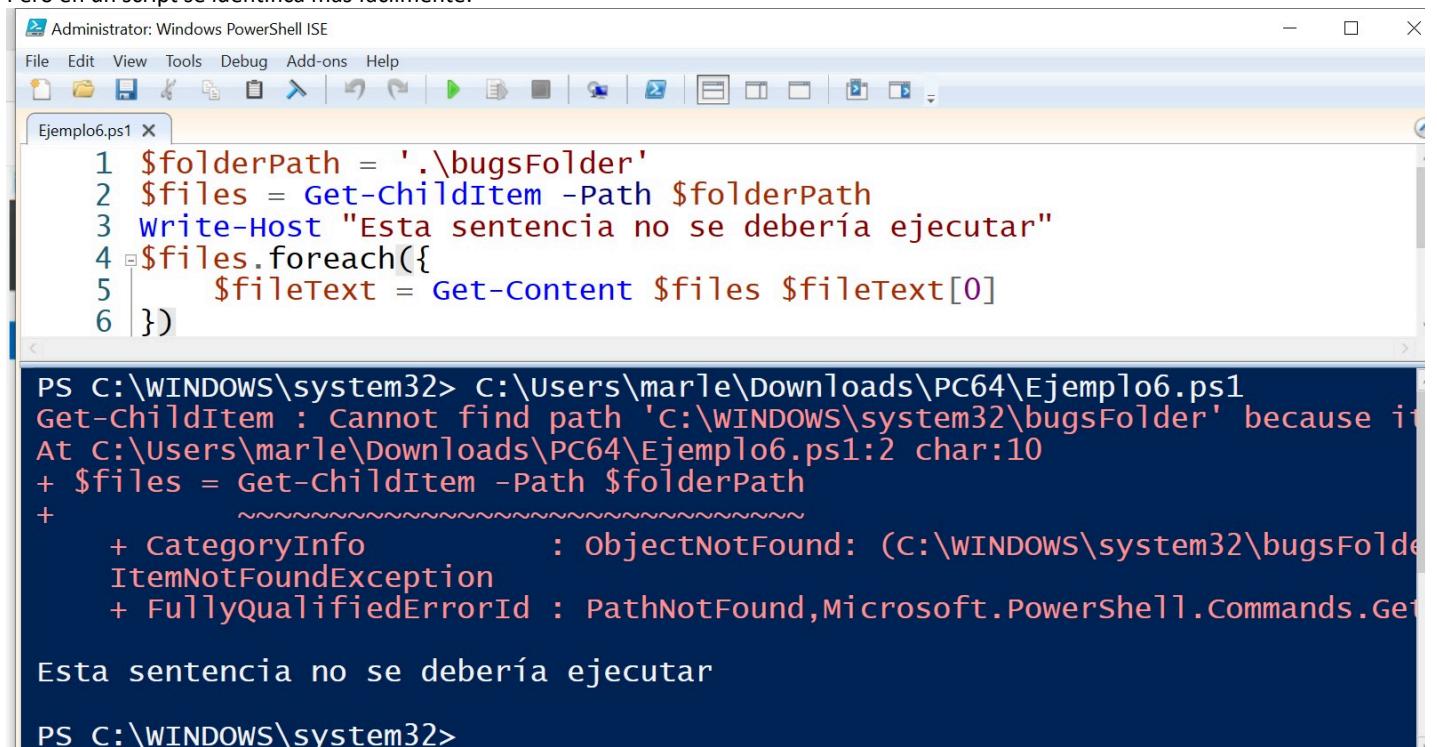
Además, la decisión de convertir una excepción en un error terminante o no terminante es tomada por el desarrollador.

Al trabajar en terminal es difícil saber si es o no un error terminante:



```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> $FolderPath = '.\bugsFolder'
PS C:\WINDOWS\system32> $files = Get-ChildItem -Path $FolderPath
Get-ChildItem : Cannot find path 'C:\WINDOWS\system32\bugsFolder' because it does
not exist.
At line:1 char:10
+ $files = Get-ChildItem -Path $FolderPath
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (C:\WINDOWS\system32\bugsFolder:String) [Get-ChildItem], ItemNotFoundException
+ FullyQualifiedErrorMessage : PathNotFound,Microsoft.PowerShell.Commands.GetChild
ItemCommand
```

Pero en un script se identifica más facilmente:



```
Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Ejemplo6.ps1 x
1 $FolderPath = '.\bugsFolder'
2 $files = Get-ChildItem -Path $FolderPath
3 Write-Host "Esta sentencia no se debería ejecutar"
4 $files.foreach({
5     $fileText = Get-Content $files $fileText[0]
6 })
```

```
PS C:\WINDOWS\system32> c:\users\marle\Downloads\PC64\Ejemplo6.ps1
Get-ChildItem : Cannot find path 'C:\WINDOWS\system32\bugsFolder' because it does
not exist.
At C:\users\marle\Downloads\PC64\Ejemplo6.ps1:2 char:10
+ $files = Get-ChildItem -Path $FolderPath
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (C:\WINDOWS\system32\bugsFolder:String) [Get-ChildItem], ItemNotFoundException
+ FullyQualifiedErrorMessage : PathNotFound,Microsoft.PowerShell.Commands.GetChild
ItemCommand
```

Esta sentencia no se debería ejecutar

```
PS C:\WINDOWS\system32>
```



El sistema nos indica el error, **ItemNotFoundException**. Y con el parámetro **ErrorAction**, que es un parámetro común, es decir, que está integrado en cada cmdlet de PowerShell, es como podemos convertir el error ocurrido en un error terminante.

Opciones del parámetro **ErrorAction**:

- **Continue**. Despliega el mensaje de error y continua ejecutando el cmdlet. Es el valor por default.
- **Ignore**. Continua ejecutando el cmdlet sin desplegar el error o guardar lo en la variable \$Error.
- **Inquire**. Despliega el mensaje de error y le pide al usuario una entrada antes de continuar.
- **SilentlyContinue**. Continua ejecutando el cmdlet sin desplegar mensaje de error, pero lo guarda en la variable \$Error.
- **Stop**. Despliega un mensaje de error y detiene la ejecución del cmdlet.

Así podemos cambiar la acción en caso de error en un cmdlet:

```
1 $FolderPath = '.\bugsFolder'  
2 $files = Get-ChildItem -Path $FolderPath -ErrorAction "Stop"  
3 Write-Host "Esta sentencia no se debería ejecutar"  
4 $files.foreach({  
5     $fileText = Get-Content $files $fileText[0]  
6 })
```

```
PS C:\WINDOWS\system32> c:\users\marle\Downloads\PC64\Ejemplo7.ps1  
Get-childitem : Cannot find path 'C:\WINDOWS\system32\bugsFolder' because it does not exist.  
At C:\Users\marle\Downloads\PC64\Ejemplo7.ps1:2 char:10  
+ $files = Get-ChildItem -Path $FolderPath -ErrorAction "stop"  
+ ~~~~~~  
+ CategoryInfo          : ObjectNotFound: (C:\WINDOWS\system32\bugsFolder:String) [Get-ChildItem],  
ItemNotFoundException  
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetChildItemCommand
```

E, inclusive, podemos manejarlo con las instrucciones **try** y **catch**:

```
1 $FolderPath = '.\bugsFolder'  
2 try{  
3     $files = Get-ChildItem -Path $FolderPath -ErrorAction "Stop"  
4     Write-Host "Esta sentencia no se debería ejecutar"  
5     $files.foreach({ $fileText = Get-Content $files $fileText[0] })  
6 } catch{  
7     $_.Exception.Message  
8 }
```

```
PS C:\WINDOWS\system32> c:\users\marle\Downloads\PC64\Ejemplo8.ps1  
Cannot find path 'C:\WINDOWS\system32\bugsFolder' because it does not exist.
```

NOTA: **\$_** es una variable automática del sistema.

Try/catch solo pueden encontrar errores terminantes, por lo que no podremos desplegar nuestro mensaje personalizado en el siguiente ejemplo:

```
Ejemplo6.ps1 Ejemplo7.ps1 Ejemplo8.ps1 Ejemplo9.ps1 X  
1 $filePath = '.\bugsFile.txt'  
2 try{  
3     Get-Content $filePath  
4 } catch{  
5     Write-Host "Ocurrió un error"  
6 }  
  
PS C:\WINDOWS\system32> c:\users\marle\Downloads\PC64\Ejemplo9.ps1  
Get-Content : Cannot find path 'C:\WINDOWS\system32\bugsFile.txt' because it does not exist.  
At C:\Users\marle\Downloads\PC64\Ejemplo9.ps1:3 char:5  
+     Get-Content $filePath  
+ ~~~~~~  
+ CategoryInfo          : ObjectNotFound: (C:\WINDOWS\system32\bugsFile:String) [Get-Content],  
ItemNotFoundException  
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetContentCommand
```

```
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.Get
```

Revisa el contenido de la variable automática **\$Error** para que veas el registro de los errores de los ejemplos. Si quieres ver el más reciente, usa **\$Error[0]**.

Referencias:

- Bertram, Adam (2020) . *PowerShell for SysAdmins*. "no starch press"
<https://nostarch.com/powershellsysadmins>
- <https://leanpub.com/thebigbookofpowershellerrorhandling>