Iron Scripter 2018: Prequel 3

A commentary

I've decided to call these notes "A commentary" rather than "A solution" because I'm mainly commenting on how to solve the puzzle. The solution you adopt will be based on your faction. I'll leave the faction-based details to you and just supply some ideas about how your faction may approach the puzzle. Be sure to check the forum and Slack channel to view what other people have done and to join the conversation around your faction.

The Puzzle

Greetings Iron Scripters. The march to Iron Scripter continues. Obtaining, manipulating and presenting data is an important skill to master as you'll learn in this challenge. You'll also need to do some research to solve the puzzle.

Your task is to create a reusable PowerShell artefact that will access the PowerShell.org feed - https://powershell.org/feed. For each entry in the feed the following information must be displayed:

- Title
- Publication date
- Link
- Author

The data should be displayed in a manner that the allows the user to select a post from the feed. The selected post should then be opened in a browser OR the text of the post should be downloaded and displayed to the user. The user should be able to decide on browser or console display.

Use best practice if it doesn't conflict with your faction's aims. The solution must be acceptable to your faction:

- Daybreak Faction - beautiful code
- Flawless Faction - flawless code
- Battle Faction - good enough to get the job done

For a bonus can you make the code work for any feed?

As in previous challenges the standard is PowerShell v5.1. Will the code work on PowerShell v6?

Good luck and good coding.

The Commentary

Not a lot to go on for this week's puzzle but that's what makes it fun.

First thing – how can we access the PowerShell feed?

PowerShell has two cmdlets for working with web data:

- Invoke-WebRequest
- Invoke-RestMethod

Invoke-WebRequest returns the content from a page on the web in the form of collections of forms, links, images, and other significant HTML elements.

Invoke-RestMethod sends HTTP and HTTPS requests to Representational State Transfer (REST) web services that returns richly structured data. PowerShell formats the response based to the data type. For an RSS or ATOM feed, Windows PowerShell returns the Item or Entry XML nodes. For JavaScript Object Notation (JSON) or XML, PowerShell converts (or deserializes) the content into objects.

Try these two commands to see the difference:

Invoke-WebRequest -Uri https://powershell.org/feed

Invoke-RestMethod -Uri https://powershell.org/feed

You need to be using Invoke-RestMethod. You'll see a bunch of data for each entry in the feed that looks like this:

```
title       : Iron Scripter 2018 Prequel: Puzzle 3
link        : https://powershell.org/2018/01/28/iron-scripter-2018-prequel-puzzle-3/
comments    : {https://powershell.org/2018/01/28/iron-scripter-2018-prequel-puzzle-3/#respond,
0}
pubDate     : Sun, 28 Jan 2018 00:01:38 +0000
creator     : creator
category    : {category, category, category}
guid        : guid
description : description
encoded     : encoded
commentRss  : https://powershell.org/2018/01/28/iron-scripter-2018-prequel-puzzle-3/feed/
```

So, we can pull the data but we were asked for a specific set of properties to be displayed. Title, publication date and link are obvious but author is a bit trickier.

```
PS> Invoke-RestMethod -Uri https://powershell.org/feed | select -First 1 | select creator

creator
-------
creator


PS> Invoke-RestMethod -Uri https://powershell.org/feed | select -First 1 | select -expand
creator

#cdata-section
--------------
Don Jones
```

You have to extract the data from the #cdata-section. To avoid errors (#cdata-section starts with a # so is treated as a comment!) use quotes:

```
Invoke-RestMethod -Uri https://powershell.org/feed |
select Title, PubDate, Link,
@{Name="Author";Expression={$_.creator.'#cdata-section'}} | Format-List
```

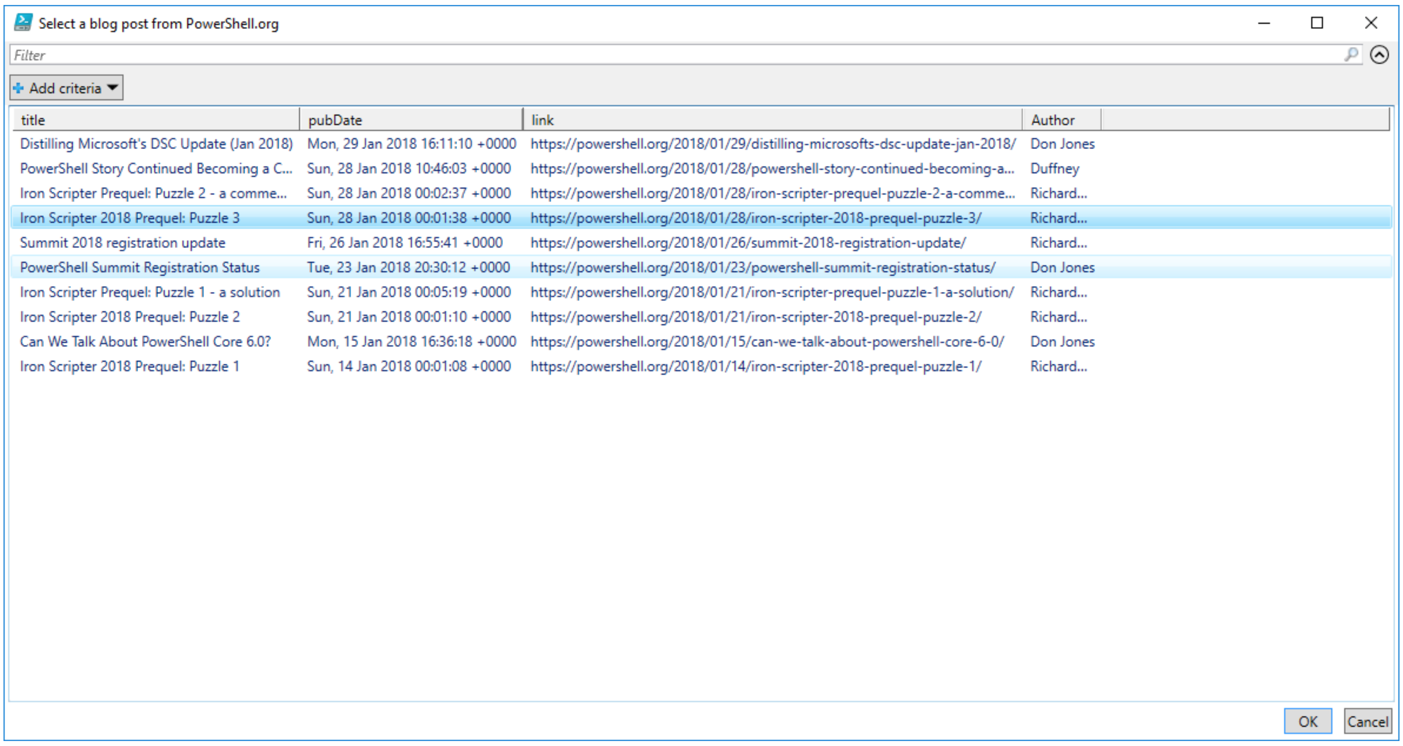
```
title   : Iron Scripter 2018 Prequel: Puzzle 3
pubDate : Sun, 28 Jan 2018 00:01:38 +0000
link    : https://powershell.org/2018/01/28/iron-scripter-2018-prequel-puzzle-3/
Author  : Richard Siddaway
```

Now the next bit can get tricky – how are you going to allow the user to select a particular entry. You could assign a number to each feed entry and then use Read-Host to get the desired one but that's very messy. You could create a GUI application and create a selectable list but that's too much work.

As Goldilocks might say - using Out-GridView is just right:

```
Invoke-RestMethod -Uri https://powershell.org/feed |
select Title, PubDate, Link,
@{Name="Author";Expression={$_.creator.'#cdata-section'}} |
Out-GridView -Title "Select a blog post from PowerShell.org" -OutputMode Single |
foreach {Start-Process $_.link }
```

You can give the view a title. The important parameter is -OutpuMode Single. Out-GridView by default doesn't send anything down the pipeline – it's a pipeline terminator. Using -OutpuMode Single sends the selected item down the pipeline when you click on OK.



title	pubDate	link	Author
Distilling Microsoft's DSC Update (Jan 2018)	Mon, 29 Jan 2018 16:11:10 +0000	https://powershell.org/2018/01/29/distilling-microsofts-dsc-update-jan-2018/	Don Jones
PowerShell Story Continued Becoming a C...	Sun, 28 Jan 2018 10:46:03 +0000	https://powershell.org/2018/01/28/powershell-story-continued-becoming-a...	Duffney
Iron Scripter Prequel: Puzzle 2 - a comme...	Sun, 28 Jan 2018 00:02:37 +0000	https://powershell.org/2018/01/28/iron-scripter-prequel-puzzle-2-a-comme...	Richard...
Iron Scripter 2018 Prequel: Puzzle 3	Sun, 28 Jan 2018 00:01:38 +0000	https://powershell.org/2018/01/28/iron-scripter-2018-prequel-puzzle-3/	Richard...
Summit 2018 registration update	Fri, 26 Jan 2018 16:55:41 +0000	https://powershell.org/2018/01/26/summit-2018-registration-update/	Richard...
PowerShell Summit Registration Status	Tue, 23 Jan 2018 20:30:12 +0000	https://powershell.org/2018/01/23/powershell-summit-registration-status/	Don Jones
Iron Scripter Prequel: Puzzle 1 - a solution	Sun, 21 Jan 2018 00:05:19 +0000	https://powershell.org/2018/01/21/iron-scripter-prequel-puzzle-1-a-solution/	Richard...
Iron Scripter 2018 Prequel: Puzzle 2	Sun, 21 Jan 2018 00:01:10 +0000	https://powershell.org/2018/01/21/iron-scripter-2018-prequel-puzzle-2/	Richard...
Can We Talk About PowerShell Core 6.0?	Mon, 15 Jan 2018 16:36:18 +0000	https://powershell.org/2018/01/15/can-we-talk-about-powershell-core-6-0/	Don Jones
Iron Scripter 2018 Prequel: Puzzle 1	Sun, 14 Jan 2018 00:01:08 +0000	https://powershell.org/2018/01/14/iron-scripter-2018-prequel-puzzle-1/	Richard...

You've solved most of the puzzle at this point. The only other thing to consider is displaying the text of the post on screen rather than in the browser.

```
function get-feed {
param (
  [string]$feed = 'https://powershell.org/feed',
  [switch]$onscreen
)

$posts = Invoke-RestMethod -Uri $feed

$posts |
select Title, PubDate, Link,
@{Name="Author";Expression={$_.creator.'#cdata-section'}} |
Out-GridView -Title "Select a blog post from PowerShell.org" -OutputMode Single |
foreach {
  $link = $_.link
  if ($onscreen) {
    $text = $posts | where link -EQ $link |
    select -ExpandProperty encoded |
    select -ExpandProperty '#cdata-section'

    ((((($text -replace '<\w*>','') -replace '</\w*>','') -replace '<span id=".*">', '') -
replace '<\w\s\w*="', '') -replace '">', ' ') -replace '&nbsp;', ''
  }
  else {
    Start-Process -FilePath $link
  }
}
}
```

I've turned the code into a function with a feed parameter defaulting to PowerShell.org and a switch parameter onscreen. If you don't use the switch you get browser display. Using the switch selects the text of the post, removes the HTML tags (there should be a better way) and displays the data.

You can tighten up the code by using | in your regular expression to signify alternative patterns on which to match:

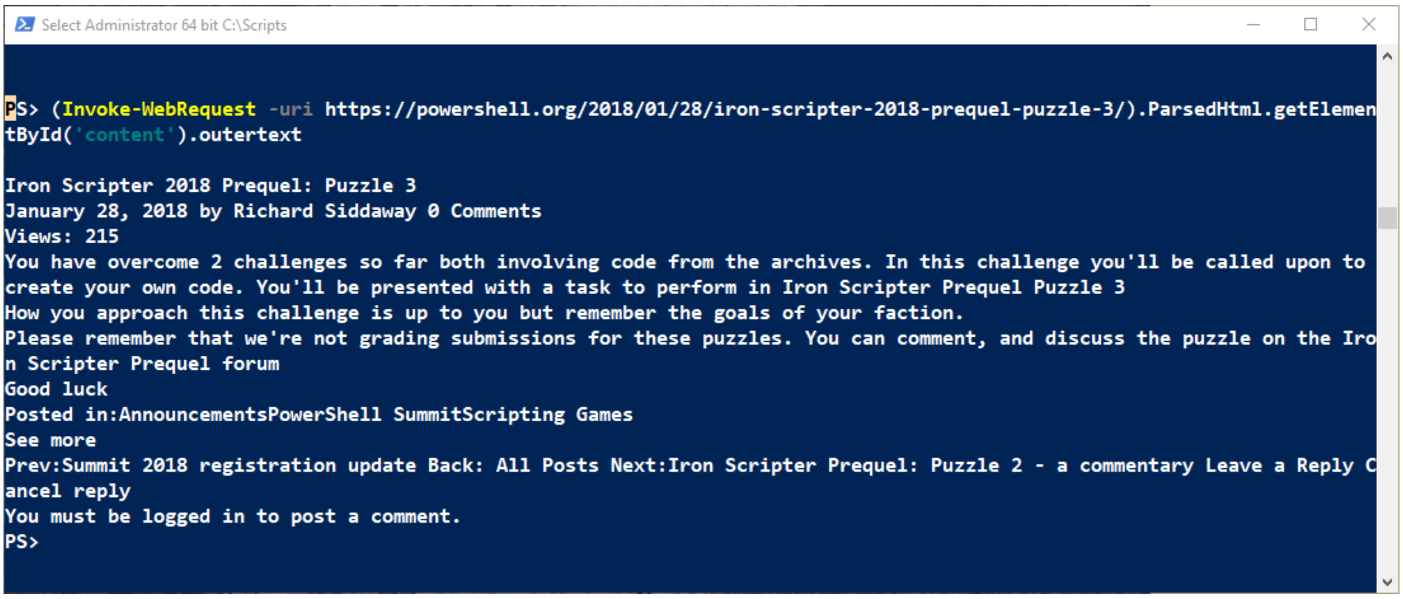
```
function get-feed {
param (
  [string]$feed = 'https://powershell.org/feed',
  [switch]$onscreen
)

$posts = Invoke-RestMethod -Uri $feed

$posts |
select Title, PubDate, Link,
@{Name="Author";Expression={$_.creator.'#cdata-section'}} |
Out-GridView -Title "Select a post from this feed" -OutputMode Single |
foreach {
  $link = $_.link
  if ($onscreen) {
    $text = $posts | where link -EQ $link |
    select -ExpandProperty encoded |
    select -ExpandProperty '#cdata-section'

    ($text -replace '(<\w*>|</\w*>|<span id=".*">|<\w\s\w*="|">|&nbsp;|<\w*\s/>)', ' ') -replace
'&#8217;', "'"
  }
  else {
    Start-Process -FilePath $link
  }
}

}
```

An alternative feed can be called:

```
get-feed -feed 'http://blogs.msdn.com/b/powershell/rss.aspx'
```

I probably don't have all of the possible HTML tags but it works and gives usable results – guess that aligns me with battle faction at heart 😊

If you don't want to go down the regular expression route – and I can't blame you for that decision – you could try using Invoke-WebRequest to get the text of the post:



Select Administrator 64 bit C:\Scripts

```
PS> (Invoke-WebRequest -uri https://powershell.org/2018/01/28/iron-scripter-2018-prequel-puzzle-3/).ParsedHtml.getElemen
tById('content').outertext

Iron Scripter 2018 Prequel: Puzzle 3
January 28, 2018 by Richard Siddaway 0 Comments
Views: 215
You have overcome 2 challenges so far both involving code from the archives. In this challenge you'll be called upon to
create your own code. You'll be presented with a task to perform in Iron Scripter Prequel Puzzle 3
How you approach this challenge is up to you but remember the goals of your faction.
Please remember that we're not grading submissions for these puzzles. You can comment, and discuss the puzzle on the Iro
n Scripter Prequel forum
Good luck
Posted in:AnnouncementsPowerShell SummitScripting Games
See more
Prev:Summit 2018 registration update Back: All Posts Next:Iron Scripter Prequel: Puzzle 2 - a commentary Leave a Reply C
ancel reply
You must be logged in to post a comment.
PS>
```

You get a few more details and some extra text but it’s a very viable alternative.

The code won’t run on PowerShell v6 because Out-GridView isn’t available. You’d have to use the menu type approach outlined above if you wanted to run on v5.1 and v6. This is one time where I think it’s better to stick with v5.1.

Battle faction will probably stick with what we have. It meets the requirements and gets the job done. If things change the code can be easily modified. Time to move on to another problem.

Daybreak faction will want to format the code so that it looks good. Maybe move the removal of HTML characters to a separate function or use Invoke-Webrequest and strip off the extra text. A well designed GUI would be a good possibility for Daybreak coders.

Flawless faction will want to add all the bells and whistles – help file, parameter validation, Write-Debug and Write-Verbose commands as appropriate, try-catch blocks as necessary and anything else that ensures the code executes flawlessly.

Enjoy!

Puzzle 4 will be available around the time you’re reading this.