PowerShell Scripting & Toolmaking

Classroom Edition

Don Jones and Jeffery Hicks

Classroom Delivery Guide

While instructors are always welcome to change things up to meet their students' needs, we did have some specific ideas in mind for a classroom delivery. This rough outline assumes a starting time of 9am each day, and provides some guidance on how long it takes us to effectively deliver each lecture and lab portion.

This special edition of our book is available in the Microsoft Courseware Marketplace, and is designed specifically for classroom delivery using the information below. Note that the Marketplace edition is not connected to our Leanpub-based Agile publishing pipeline, meaning it represents a "snapshot" of our text. The Marketplace edition also includes additional teaching tools and resources not available to the general public.

Code and Demos

We should note that, for demonstrations, we use the code as it exists in the book - walking through each line of code as needed, running it to show the results, and so on. Everyone has access to these samples as part of the book's downloadable code, so instructors and students can follow along, and students can download the code on their own, even after class.

In cases where a chapter contains numerous snippets of code, we've included them all, sequentially, in a "Snippets.ps1" file, along with brief comments. This can make it easier to review code in the PowerShell ISE or another editor. These snippets files are design *mainly* to make it easier to review, as a group, small chunks of code from the book. In many cases, you can highlight lines or chunks and execute them to follow along with the text. However, please don't assume that the snippets files are entirely standalone. They're a tool, and they're meant to be used in conjunction with the prose in the book. The book will provide the context needed to make the snippets make sense.

Important

When you run `Install-Module PowerShell-Toolmaking` to install the book's downloadable sample code, the resulting module won't "contain" anything. That is, it's empty. You'll need to browse to the install location (by default, it's in Program Files/WindowsPowerShell/Modules/PowerShell-Toolmaking) to find the code samples, neatly organized by chapter.

Slides

We aren't historically big fans of slides, but we do understand their value in keeping things organized and on-track. Rather than using PowerPoint, we've included - in the /Presentation folder of the downloadable sample code - a link to a web browser-presented slide show. Launch it, use the left/right arrow keys to select the chapter you want, and then spacebar as usual through the presentation. The slides serve only to highlight the most critical points and to help you stay on-sequence (mainly by referring to headings in the book text). You'll definitely need to refer to the main book for all the talking points. We also assume you'll be sharing code in the PowerShell ISE (or a similar tool), so we don't often include code on slides. Students who go home and download the sample code from PowerShell Gallery will also have the presentation to use as a reference, if they want.

You can access an updated presentation at https://gitpitch.com/concentrateddon/ToolmakingSlides anytime you wish.

Note that, near the lower portion of the presentation window, you can modify the visual theme to one that you find aesthetically pleasing, switch to full screen mode, print the slides, or download a copy for offline use.

Problems with the slides can be reported using the "Issues" tab at: http://github.com/concentrateddon/ToolmakingSli
This is open-source, so you're free to download it, make your own changes, and whatever. You're also welcome to clone the repo and submit a pull request with your changes, for possible permanent inclusion.

Suggested Schedule

We tend to not build in explicit break times in the following schedules, knowing that labs are a good time for people to stretch, grab a beverage, or whatever they may need to do.

Part 1

Part 1 of the course is a very carefully designed sequence of learning. You shouldn't skip any of this material, you should go through it in the order provided, and students should absolutely perform each hands-on lab offered.

Note that each lab in this Part essentially builds on the one before it. The idea is for students to complete each lab, and then carry their result through to the next one. However, we know sometimes people get stuck, or get called out of class and can't finish in time. That's OK - we provide a sample solution for each lab, which serves as the starting point for the next lab. Feel free to use those whenever needed.

Chapter	Lecture/Demo	Lab Time
DAY ONE		
(Introductions and class housekeeping)	9:00-9:30	
Tool Design	9:30-10:15	10:15-10:45
Start With a Command	10:45-11:15	11:15-11:45
Lunch	11:45-12:45	
Build a Basic Function and Module	12:45-13:30	13:30-14:30
Adding CmdletBinding and Parameterizing	14:30-15:15	15:15-15:45
Emitting Objects as Output	15:45-16:30	16:30-17:00
DAY TWO		
(Prior day review, Q&A, and finishing up)	9:00-9:30	
Interlude: Changing Your Approach	9:30-10:00	
Using Verbose, Warning, and Informational Output	10:00-11:00	11:00-11:30
Lunch	11:30-12:30	
Comment-Based Help	12:30-13:00	13:00-13:30
Handling Errors	13:30-14:15	14:15-15:00
Basic Debugging	15:00-15:45	15:45-16:45

Part 2

Part 2 starts diving into more advanced techniques, and so we allow a bit of extra time for labs. Students will need to download, from the PowerShell Gallery, a couple of modules; their lab computers must have Internet access for that to work. These labs are a bit less sequential than the previous ones. In many cases, we'll ask students to use their results from the "Handling Errors" lab as a starting point for these. Of course, students can use their own work, or they can use our sample solution from that chapter.

Chapter	Lecture/Demo	Lab Time
DAY THREE		
(Prior day review, Q&A, and finishing up)	9:00-9:30	
Going Deeper with Parameters	9:30-10:15	
Writing Full Help	10:15-11:00	11:00-11:30
Lunch	11:30-12:30	
Unit Testing Your Code	12:30-13:15	13:15-14:00
Extending Output Types	14:00-14:30	
Analyzing Your Script	14:30-15:00	15:00-15:30
Publishing Your Tools	15:30-16:15	

For a 3-day delivery, you can reasonably stop at this point. You'll have covered the "main" narrative of the book, and students will be fully capable of producing their own tools, following core patterns and practices.

Parts 3 and 4

Part 3 introduces some very basic concepts for writing controller scripts, and is fairly short. It relies on students having completed Part 1, or at least being willing to use the sample results from the "Handling Errors" chapter as a starting point.

Part 4 covers three distinct types of data usage, two of which offer labs. Students may sometimes be feeling tired by this point, but being able to work with these data sources is crucial to becoming a real-world toolmaker, so encourage them to stick with it through this home stretch. Note that the SQL Server chapter does not include practical demos or a exercise, in order to simplify the lab environment.

Chapter	Lecture/Demo	Lab Time
DAY FOUR		
(Prior day review, Q&A, and finishing up)	9:00-9:30	
Basic Controller Scripts and Menus	9:30-10:15	10:15-11:00
Proxy Functions	11:00-11:30	11:30-12:00
Lunch	12:00-13:00	
Working with XML	13:00-13:45	13:45-14:30
Working with JSON	14:30-15:15	15:15-16:00
Working with SQL Server	16:00-16:30	

For a 4-day delivery, you can reasonably stop at this point. Students will have enough information to produce effective tools and automation scripts.

"Final Exam" Lab

If you're targeting a 5-day delivery, then the last day is a series of structured labs. The idea is to get students to repeat all of the main points they've learned so far, and put those things to use in a "from scratch" lab. This kind of hands-on reinforcement can make all the difference in the world in terms of solidifying key concepts and techniques. Stopping points and sample solutions are provided in case students get stuck or lost.

Most students will need 3-4 hours to complete everything in the lab, enabling them to wrap up early and begin their weekend.

Delivery Notes

These notes offer some guidance on delivering each chapter.

Final Exam

Lab One should be done as a group, breaking students into small groups when they conduct their actual design. Use a whiteboard to collect student thoughts on the tool design. Then, walk through

"Code the Tool" section by section, analyzing each chunk of code, the accompanying notes, and making sure students understand *why* everything is there.

For Lab Two, consider taking the nine-item list of steps and making each a mini-lab. Give a reasonable amount of time for the class to complete each step, or perform some steps as a group. Let the class come together at the end of each step before proceeding to the next one.

A Note on Code Listings

The code formatting in this book only allows for about 60-odd characters per line. We've tried our best to keep our code within that limit, although sometimes you may see some awkward formatting as a result.

For example:

```
        Invoke-CimMethod -ComputerName $computer `
                         -MethodName Change `
                         -Query "SELECT * FROM Win32_Service WHERE Name = '$Servi\
ceName'" `
```

Here, you can see the default action for a too-long line - it gets word-wrapped, and a backslash inserted at the wrap point to let you know. We try to avoid those situations, but they may sometimes be unavoidable. When we *do* avoid them, it may be with awkward formatting, such as in the above where we used backticks (‘) or:

```
        Invoke-CimMethod -ComputerName $computer `
                         -MethodName Change `
          -Query "SELECT * FROM Win32_Service WHERE Name = '$ServiceName'" `
```

Here, we've given up on neatly aligning everything to prevent a wrap situation. Ugly, but oh well.

You may also see this crop up in `inline code` snippets, especially the backslash.

If you are reading this book on a Kindle, tablet or other e-reader, then we hope you'll understand that all code formatting bets are off the table. There's no telling what the formatting will look like due to how each reader might format the page. We trust you know enough about PowerShell to not get distracted by odd line breaks or whatever.

When *you* write PowerShell code, you should not be limited by these constraints. There is no reason for you to have to use a backtick to "break" a command. Simply type out your command. If you want to break a long line to make it easier to read without a lot of horizontal scrolling, you can hit `Enter` after any of these characters:

- Open parenthesis (
- Open curly brace {

- Pipe |
- Comma ,
- Semicolon ;
- Equal sign =

This is probably not a complete list, but breaking after any of these characters makes the most sense.

Anyway, we apologize for these artifacts. Keep in mind that you can, and should, use `Install-Module PowerShell-Toolmaking` to download and install the code samples from the PowerShell Gallery. They'll end up in `\Program Files\WindowsPowerShell\Modules\PowerShell-Toolmaking`, typically broken down by chapter. We do update that download pretty often, so if you don't have the latest version installed, do that.

Lab Setup

We hope that you plan to follow along with us in this book, and to help you do so we've provided hands-on exercises at the end of most chapters. To complete those, you won't need much of a lab environment - just a Windows 10 (or later) computer to which you have Administrator access, and which has Internet connectivity. We've built the labs so that there's no need for a domain controller, servers, or anything else. In this short chapter, we'll walk you through the lab setup process, just in case you're building a new Windows 10 computer and aren't familiar with that procedure.

What You'll Need

Before you begin, you'll need Windows 10 installation media. This is usually an ISO image; for the following examples, we will assume it's called **windows10.iso**. In reality, it's probably got a different filename, so be sure to ensure that you change the filename in the examples below. Also, make sure that you're installing a 64-bit edition of Windows 10. Other than that, any business edition (Professional, Enterprise, etc.) is supported.

Setting Up a Virtual Machine

If you plan to run your lab environment in a virtual machine (VM), you'll need to set up your VM host first. You can use a variety of hosts, including Amazon Web Services (AWS), Microsoft Azure, VMware vSphere, Hyper-V, and so on.

Setting Up a VM on Windows Server Hyper-V

We'll assume that you're using Microsoft Hyper-V, on Windows Server, and provide setup instructions for that. We assume that your host computer has at least 4096MB (4GB) of memory free for the VM to use. We also assume that the Hyper-V role is already installed and functional on your server, and that the Windows 10 ISO image is available locally on the server.

1. Launch Hyper-V Manager.
2. In the left panel, select your server.
3. In the Actions panel on the right, select "Virtual Switch Manager…".
4. In the Virtual Switches section, select "New Virtual Network Switch".
5. On the right, select "External" and click `Create Virtual Switch`.
6. Name the network "Internet Access" and click `OK`.

7. Click "Yes" to acknowledge the warning. Note that this will disconnect you from the machine if you are performing the action remotely. Reconnect and proceed.
8. In Hyper-V Manager, select your server in the left panel.
9. From the Action menu, select `New > Virtual Machine`.
10. On the Before You Begin page click `Next`.
11. On the Specify Name and Location page, enter "CLIENT".
12. On the Specify Generation page, select 'Generation 2' and click Next.
13. On the Assign Memory page, enter "4096" and click `Next`. You can assign a lower number, if needed, but the performance of the VM may be poor.
14. On the Configure Networking page, select "Internet Access" and click `Next`.
15. On the Connect Virtual Hard Disk page, click `Next`.
16. On the Installation Options page, select "Install an operating system from a boot CD/DVD-ROM".
    1. Select the "Image file (.iso):" option and click `Browse`.
    2. Locate the Windows ISO image (windows10.iso, or whatever filename you used) and click `Open`.
    3. Click `Next`.
17. On the Completing the New Virtual Machine Wizard page, click `Finish`.
18. In Hyper-V Manager, right-click CLIENT and select `Connect` to launch the VM in a console window.
19. In the Action menu of VM console window, select `Start`.

Setting up a VM on Windows 10 Hyper-V

Windows 10 supports Hyper-V on computers with a compatible processor. If you have already installed and enabled **both** "Hyper-V Management Tools" and "Hyper-V Platform," then you should be able to open Hyper-V Manager and set up a VM, using the instructions above for Windows Server Hyper-V. Note that we recommend your computer have at least 8GB of RAM, so that 4GB can be reserved for your host, and 3-4GB allocated to the VM.

Installing Windows 10

You can follow these steps whether you're installing Windows in a VM or on a regular host computer. This assumes you are installing from a DVD (or equivalent ISO image). **Note that these instructions were written for Windows 10 build 1607; future builds may introduce a somewhat different Setup process.**

1. On the Windows Setup dialog box, select your language, time and currency format, and keyboard. Then click `Next` (or press `Alt+N`).
2. Press `Enter` or click `Install now`.

3. You must provide a valid activation key. The activation key determines the edition of Windows that is installed; we recommend installing either a Professional or Enterprise edition. You can also click "I don't have a Product key" to continue installing. If you do so, you may be asked to choose an edition to install. Again, we recommend Professional ("Pro") or Enterprise. Be sure the Architecture states "x64." **We do not recommend using a Home edition**.
4. Click `Next` (or press `Alt+N`).
5. Select the checkbox to accept the licensing terms, and click `Next`.
6. Click "Custom: Install Windows only (advanced)".
7. Select the drive to install Windows on. This will usually be "Drive 0" on a new system or VM.
8. Click `Next`.
9. Wait while Setup completes.
10. After Setup completes, you may see a "Get going fast" screen. You can click Use Express settings. Or, if you need to customize the settings described, select `Customize`. We will assume you are choosing to use the Express settings.
11. You may see a "Who owns this PC?" screen. Select "I own it" and click `Next`.
12. Select "Skip this step" (located near the bottom-left of the screen).
13. For "Create an account for this PC," enter the user name "User". For the password, enter "P@ssw0rd" twice, enter "password" for the hint, and then click `Next`.
14. For "Meet Cortana," click "Not now".
15. Wait while Windows sets up your PC.

Adding Lab Files and Configuring PowerShell

We have published all of the lab files for this book on the PowerShell Gallery, to make it easier to install them.

1. On your Windows computer, press `Windows+R`, type "powershell", and press `Enter`.
2. Right-click the PowerShell icon on the Task Bar, and select Run as Administrator.
3. Click `Yes`.
4. In the new PowerShell window (which should say "Administrator: Windows PowerShell" in the title bar), type `Install-Module PowerShell-Toolmaking` and press `Enter`.
5. You may be notified that "NuGet provider is required to continue." Type `Y` and press `Enter`.
6. You may be notified of an "Untrusted repository." Type `Y` and press `Enter`.
7. Type `Set-ExecutionPolicy Bypass` and press `Enter`.
8. Type `Y` and press `Enter`.

If the installation fails, or if you see an error or warning when setting the execution policy, ensure that PowerShell is running as Administrator and that the computer has unrestricted Internet access. On a company-owned computer, restrictions may be in place that prevent the installation of files or the changing of the execution policy. You will need to consult your company's IT administrators to remedy that.

Assumptions Going Forward

Because scripting and toolmaking are not entry-level tasks, we assume that readers are already aware of the need to run PowerShell "as Administrator" when developing scripts and tools. We assume a basic level of familiarity with the PowerShell Integrated Scripting Environment (ISE), and we assume an intermediate or higher level of familiarity with PowerShell itself. If you don't feel you meet these expectations, we suggest first completing *Learn Windows PowerShell in a Month of Lunches*[3] available from most booksellers or from Manning.com.

We also assume that your lab computer (or virtual machine) will have Internet access.

[3] https://www.manning.com/books/learn-windows-powershell-in-a-month-of-lunches-third-edition