## Writing Windows PowerShell Scripts

**Lab Time**: 45 minutes

**Lab Folder**: C:\Student\Labs\06.PowerShell

**Lab Overview**: Windows PowerShell is a command-line scripting tool that provides an administrator full access to the SharePoint object model which offers the capability to interact directly with SharePoint 2010 objects to manipulate Web applications, site collections, sites, lists and much more. In addition, the farm administrator has the ability to write and execute scripts which call cmdlets (pronounced "command-lets") which makes for an improved scripting experience especially when compared to writing scripts using DOS-style batch files.

In the following lab, you will learn to write Windows PowerShell Scripts to manage SharePoint 2010. First you will learn how to work with the Windows PowerShell add-in for SharePoint 2010 and later on in the lab you will learn how to write your own scripts and how to debug these scripts.

### Exercise 1: Working with the Windows PowerShell Console Window

In this exercise you will get some exposure to working with Windows PowerShell and the Windows PowerShell add-in created for SharePoint 2010.

1. Log into the **WingtipServer** VM.
2. Launch the Windows PowerShell console from Windows Start menu. You’ll find the PowerShell shortcut at Start » All Programs » Accessories » Windows PowerShell » Windows PowerShell.

NOTE: Make sure to launch the 64 bit version with the shortcut label of Windows PowerShell and not the 32-bit version with a shortcut label of Windows PowerShell (x86). Also take note that in this step you should be launching the PowerShell console from standard PowerShell menu item and not SharePoint-specific menu item under Microsoft SharePoint 2010 Products.

1. Once you have a command prompt in the Windows PowerShell console windows, type the following command using the Set-Location cmdlet and single parameter with a path on the local C:\ drive and press [Enter]. *Remember the Lab Folder location at the very beginning of this lab for the exact location of [[LAB FILES]].* This is the equivalent of the well-known DOS CD command.

Set-Location “[[LAB FILES]]\Starter Files\”

1. The current folder of the Windows PowerShell console should now be located in a folder that has several Windows PowerShell scripts (\*.ps1 files) that will be used in this lab. Type the following command which uses the Get-ChildItem cmdlet and passes no parameters and press [Enter] to see a listing of all the .ps1 files in this folder.

Get-ChildItem

1. Type in the following command to open Notepad and the script named Hello.ps1. Press [Enter] after typing the command to execute it.

Notepad Hello.ps1

1. After examining the Windows PowerShell code inside Hello.ps1, close Notepad and attempt to run it by typing .\Hello.ps1 and pressing [Enter]. If the Windows PowerShell scripting support on your VM still has the default execution policy of restricted, the script will not run at all. If the execution policy has been changed to unrestricted, the console will prompt you whether to run the script or not. When writing and testing Windows PowerShell scripts, it is easiest to change the execution policy to Bypass so that scripts can freely run without any user prompts. Type the following command and press [Enter] twice to enable scripting support.

Set-ExecutionPolicy ByPass

1. Now, try to run the script named Hello.ps1 a second time by attempt to run it by typing .\Hello.ps1 and pressing [Enter]. You should now see the script is able to run and output a simple message to the Windows PowerShell console.

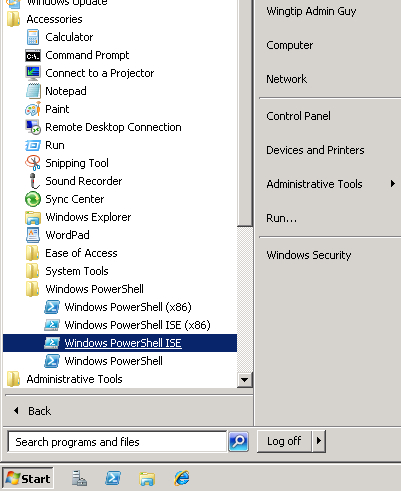
In this exercise you ran your first Windows PowerShell cmdlets and scripts.

### Exercise 2: Writing and Debugging Windows PowerShell Scripts

In this exercise you will write and debug your first Windows PowerShell scripts.

1. Now it's time to move to a better script editor. Open the Windows PowerShell ISE from the Windows Start menu. The shortcut can be found at **Start » All Programs » Accessories » Windows PowerShell »** Windows PowerShell ISE as shown below.

ISE = Integrated Scripting Environment



1. Open Hello.ps1 once again in the Windows PowerShell ISE. Drop down the **Debug** menu and you should see it gives you the ability to run and debug the code inside a Windows PowerShell script. Execute Hello.ps1 by pressing [F5] and examine the output.
2. Make a small change to the text message inside Hello.ps1 that is assigned to the variable named $HelloMessage and save your work.

$HelloMessage = "Sample message"

Write-Host "-----------------------------------"

Write-Host "Hello World of Powershell Scripting"

Write-Host "Host name: "$(Get-Item env:\computerName).value

Write-Host $HelloMessage

Write-Host "-----------------------------------"

Run Hello.ps1 one more time in the Windows PowerShell console. You should see your changes and also observe how easy it is to edit and run Windows PowerShell scripts.

1. Now, it's time to practice debugging and single stepping through a Windows PowerShell script. Set a breakpoint by right-clicking a line and selecting Toggle Breakpoint or by pressing [F9] in the first line in Hello.ps1. Now, press [F5] to begin execution. The execution should stop and the breakpoint you set. Now, press [F11] repeatedly to single step through the remaining lines of code inside the script.

### Exercise 3: Using Cmdlets from the Microsoft.SharePoint.PowerShell Snap-in

In this exercise you will learn what the Microsoft.SharePoint.PowerShell snap-in has to offer and how to use cmdlets from this snap-in.

1. Open the script named LoadSharePointSnapin.ps1. Examine the PowerShell script code inside. There is call to the Add-PSSnapin cmdlet that loads the snap-in for SharePoint 2010 named Microsoft.SharePoint.Powershell. This snap-in contains a lot of SharePoint 2010 related cmdlets. A cmdlet is a set of PowerShell statements that are needed to execute a desired functionality.
2. Now, leave the Windows PowerShell ISE and return the PowerShell console window. Run script by typing .\LoadSharePointSnapin.ps1 and pressing [Enter]. At this point, you can call the cmdlets provided by the Microsoft.SharePoint.PowerShell snap-in.
3. Run the Get-Command cmdlet with the following parameter and press return to see a listing of all the cmdlet included with the Microsoft.SharePoint.PowerShell snap-in.

Get-Command -PSSnapin Microsoft.SharePoint.PowerShell

1. In the last step, there are too many cmdlets in Microsoft.SharePoint.PowerShell to be able to see them all at once in the console windows. Try running the Get-Command cmdlet again but this time requesting only the cmdlets based on the cmdlet verb of Get.

Get-Command -PSSnapin Microsoft.SharePoint.PowerShell -Verb Get

1. Run the Get-Command again with the Verb parameter and pipe the output to a new text file named SP2010Cmdlets.txt.

Get-Command -PSSnapin Microsoft.SharePoint.PowerShell –Verb Get | out-file –filepath “C:\Student\SP2010Cmdlets.txt”

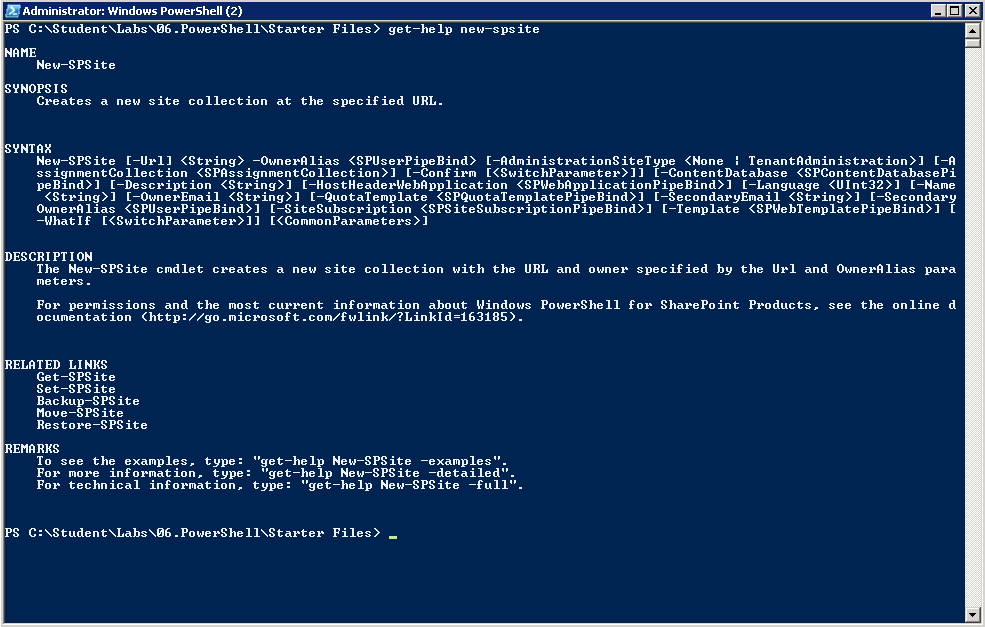
1. Open SP2010Cmdlets.txt with Notepad using the following command to inspect the cmdlets provided by the Microsoft.SharePoint.PowerShell snap-in.

Notepad C:\Student\SP2010Cmdlets.txt

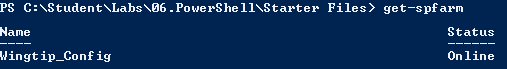
1. You can also get some help on cmdlets. Type in the following:

Get-Help New-SPSite

1. Running this cmdlet will display information about the New-SPSite cmdlet. It lists the different arguments that you can use to create a new SharePoint site. Each argument is listed within brackets ([]) and the data type of the argument is listed within “<”.



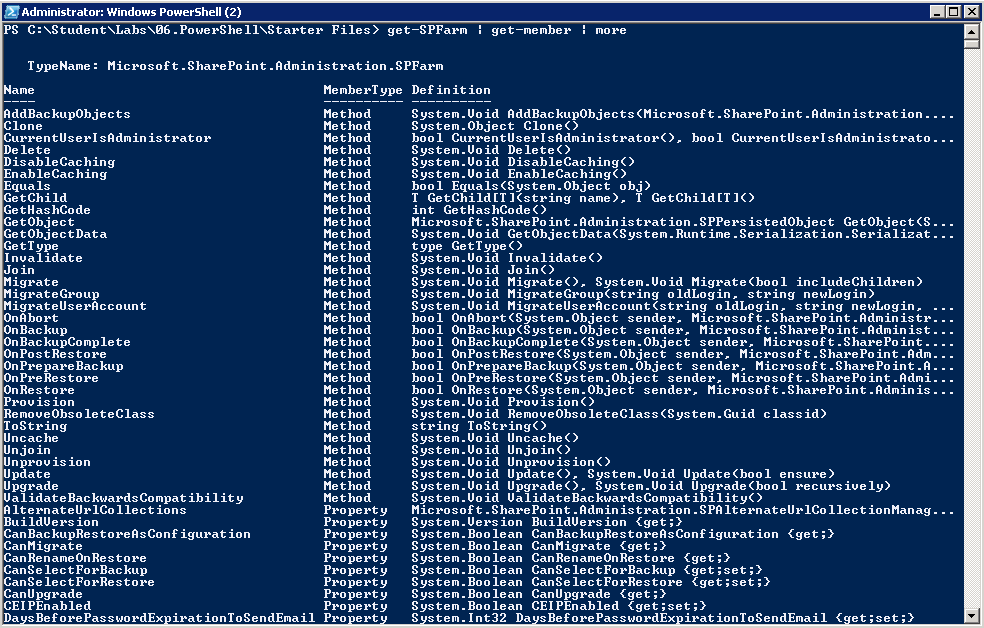
1. Now enter Get-SPFarm. The only thing you get is the name of the configuration database.



1. If you want to see all members available on the SPFarm object you can type the following:

Get-SPFarm | get-member | more

1. This results in the following (the screenshot is not complete as the list of properties is much longer):



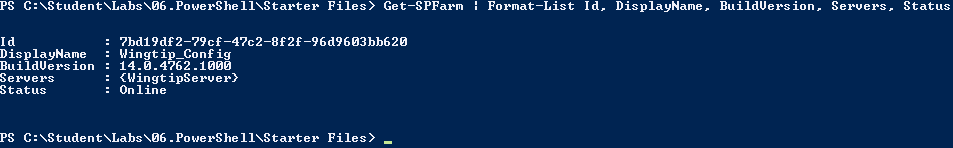
1. In this list you can also see which member is a property and which member is a method. If you want to see all the details, type the following:

Get-SPFarm | Select \*

1. If you are interested in only a few of the properties, you can restrict the number of properties that are displayed by using the following command.

Get-SPFarm | Format-List Id, DisplayName, BuildVersion, Servers, Status

1. This results in the following:



1. As you can see DisplayName returns the name of the configuration database and the servers in the farm is WingtipServer.
2. You can also retrieve information about:
   1. Web applications:

Get-SPWebApplication

* 1. Managed accounts:

Get-SPManagedAccount

* 1. Retrieve information about the service applications in the farm:

Get-SPServiceApplication

1. Besides retrieving data, you can also create and configure every object in the SharePoint farm. As an example you are going to remove the Active Directory account as a managed account. You can use the PowerShell cmdlet **New-SPManagedAccount** which needs an argument of type PSCredential. This object on its turn can only be created specifying a user name and a password of type SecureString.
2. Create a variable with the name **$password**:

$passwordstring = "Password1"

* 1. Create the object of type SecureString(Press [Enter] twice to make the PowerShell cmdlet execute.)**:**

$password = New-Object System.Security.SecureString

for ($i=0; $i -lt $passwordstring.Length; $i++) {

$password.AppendChar($passwordstring[$i])

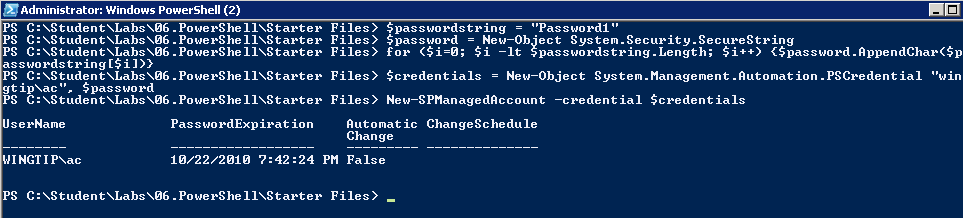
}

* 1. Create a credentials object:

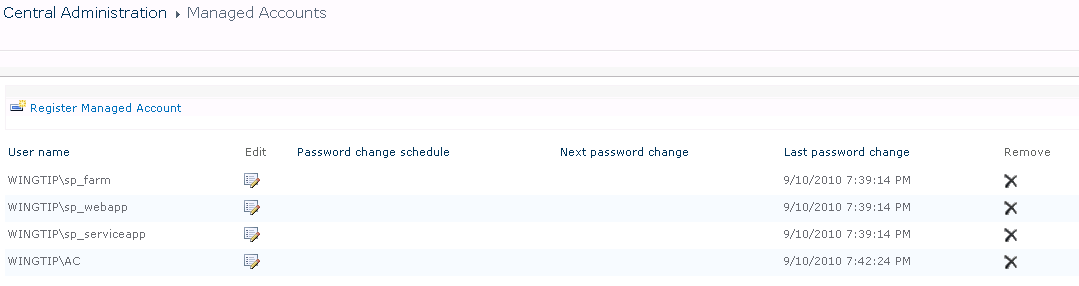
$credentials = New-Object System.Management.Automation.PSCredential "wingtip\ac", $password

* 1. Create the managed account:

New-SPManagedAccount -credential $credentials



1. Open SharePoint 2010 Central Administration and check the managed accounts by browsing to **Central Administration** » **Security** » **Configure Managed accounts**. Your account should be added to the list.



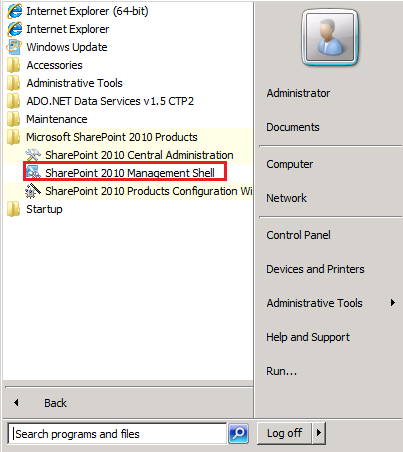
In this exercise you got some hands-on experience in working with some of the provided SharePoint 2010 Windows PowerShell cmdlets as well as working with custom Windows PowerShell scripts.

### Exercise 4: Working with the SharePoint 2010 Management Shell

In this exercise you will explore the power of the SharePoint 2010 Management Shell. With the management shell, you do not have to register the snap-in that contains the cmdlets.

This tool has already loaded the Microsoft.SharePoint.PowerShell snap-in; as a result of the line Add-PSSnapin Microsoft.SharePoint.PowerShell in the SharePoint.ps1 file located in %CommonProgramFiles%\Microsoft Shared\Web Server Extensions\14\Config\PowerShell\Registration.

1. SharePoint 2010 comes with its own PowerShell tool, named SharePoint 2010 Management Shell. You can open it from **Start** » **All Programs** » **Microsoft SharePoint 2010 Products** » **SharePoint 2010 Management Shell**.



1. Execute again the following PowerShell cmdlets:
2. Retrieve farm details:

Get-SPFarm | Select \*

* 1. Retrieve information about the SharePoint assemblies located in the SharePoint directories. (Note: Press [Enter] twice to make the PowerShell cmdlet execute.)

$path = "C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14"

Get-ChildItem $path -recurse |

Where-Object {$\_.name -like "\*SharePoint\*.dll"} |

Sort-Object -Property Name -unique |

Format-Table Name

* 1. Retrieve the managed accounts:

Get-SPManagedAccount

1. Removing a managed account is far more simple:

Remove-SPManagedAccount –identity wingtip\ac

1. Verify in Central Administration that your account has been added successfully as a managed account.