Advanced Scripting   
Module Manifests

Last Updated: 6/23/2020 11:19 AM Version 1  
Document Prepared for: CIT361 Student

# Name Daniel Harris ID 235868292

# Instructions

Save a copy of this document. Answer all questions directly in this document. You will save and upload this completed document as your homework submission.

# Overview

Now you will create a module manifest to control the loading of your module

# Requirements

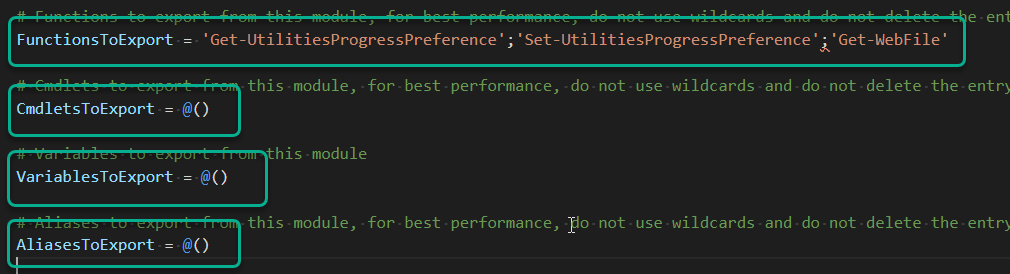
PowerShell  
MyUtilities.psm1 from previous exercise

# Setup

# Task 1—Create a Module Manifest

The easiest way to create a Manifest is to use the New-ModuleManifest cmdlet. You can pass a lot of information to the cmdlet to set those properties of the manifest, or you can modify the manifest yourself. You will use a combination of both.

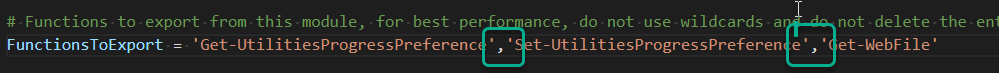
## Steps

1. Create a Manifest for your module. The manifest should be named the same as your module with a psd1 extension. Also you need to tell the manifest what the root module is. It will be your psm1 file.
2. Change directories to where your MyUtilities.psm1 file is.
3. Create the manifest  
   New-ModuleManifest -Path MyUtilities.psd1 -RootModule MyUtilities.psm1
4. Open the Manifest file in your editor and view the results. Notice several default values have been added.
   1. What is the ModuleVersion? 1.0
   2. What is the GUID? 98e6d47f-020a-44b2-9ec4-46e255772132
   3. What is the Author? User Where do you suppose that came from? Current logged on user
5. Notice most of the items are commented out. If you would like to set one you can uncomment it.
6. Partway down you will find a section that exports items. You will modify this section to only export your three functions. Modify the file as follows  
   
7. Save the manifest file.

# Task 2—Test Your Manifest

You can perform some automatic validation on the manifest file with the Test-ModuleManifest cmdlet

## Steps

1. Test the manifest  
   Test-ModuleManifest MyUtilities.psd1
2. Oh oh, something is wrong. The problem is that we should have used commas rather than semicolons when we created the array list for FunctionsToExport.
3. Fix the manifest  
   
4. Save the manifest and test it again. It should pass, if not fix until it does.

# Task 3—Manually Load your Module via Manifest

## Steps

1. If your module is currently loaded unload it.
2. Load the module via manifest.   
   Import-Module MyUtilities.psd1
3. See what commands are available.
4. Test it out.

# Task 4—Install Module

## Steps

1. Create a folder named MyUtilities in your personal module folder. Don’t remember where that is. Look at your $env:PSModulePath.
2. Copy your MyUtilites.psm1 and MyUtilities.psd1 to that folder.
3. Start a new PowerShell instance.
4. Test, Enter:  
   Get-UtilitiesProgressPreference
5. It should work. If not troubleshoot.

# Wrap-up

Copy your Manifest file here

#  
# Module manifest for module 'MyUtilities'  
#  
# Generated by: User  
#  
# Generated on: 6/4/2023  
#  
  
@{  
  
# Script module or binary module file associated with this manifest.  
RootModule = 'MyUtilities.psm1'  
  
# Version number of this module.  
ModuleVersion = '1.0'  
  
# Supported PSEditions  
# CompatiblePSEditions = @()  
  
# ID used to uniquely identify this module  
GUID = '98e6d47f-020a-44b2-9ec4-46e255772132'  
  
# Author of this module  
Author = 'User'  
  
# Company or vendor of this module  
CompanyName = 'Unknown'  
  
# Copyright statement for this module  
Copyright = '(c) 2023 User. All rights reserved.'  
  
# Description of the functionality provided by this module  
# Description = ''  
  
# Minimum version of the Windows PowerShell engine required by this module  
# PowerShellVersion = ''  
  
# Name of the Windows PowerShell host required by this module  
# PowerShellHostName = ''  
  
# Minimum version of the Windows PowerShell host required by this module  
# PowerShellHostVersion = ''  
  
# Minimum version of Microsoft .NET Framework required by this module. This prerequisite is valid for the PowerShell Desktop edition only.  
# DotNetFrameworkVersion = ''  
  
# Minimum version of the common language runtime (CLR) required by this module. This prerequisite is valid for the PowerShell Desktop edition only.  
# CLRVersion = ''  
  
# Processor architecture (None, X86, Amd64) required by this module  
# ProcessorArchitecture = ''  
  
# Modules that must be imported into the global environment prior to importing this module  
# RequiredModules = @()  
  
# Assemblies that must be loaded prior to importing this module  
# RequiredAssemblies = @()  
  
# Script files (.ps1) that are run in the caller's environment prior to importing this module.  
# ScriptsToProcess = @()  
  
# Type files (.ps1xml) to be loaded when importing this module  
# TypesToProcess = @()  
  
# Format files (.ps1xml) to be loaded when importing this module  
# FormatsToProcess = @()  
  
# Modules to import as nested modules of the module specified in RootModule/ModuleToProcess  
# NestedModules = @()  
  
# Functions to export from this module, for best performance, do not use wildcards and do not delete the entry, use an empty array if there are no functions to export.  
FunctionsToExport = 'get-utilitiesprogressprefrence','set-utilitiesprogressprefrence','Get-Webfile'  
  
# Cmdlets to export from this module, for best performance, do not use wildcards and do not delete the entry, use an empty array if there are no cmdlets to export.  
CmdletsToExport = '\*'  
  
# Variables to export from this module  
VariablesToExport = '\*'  
  
# Aliases to export from this module, for best performance, do not use wildcards and do not delete the entry, use an empty array if there are no aliases to export.  
AliasesToExport = '\*'  
  
# DSC resources to export from this module  
# DscResourcesToExport = @()  
  
# List of all modules packaged with this module  
# ModuleList = @()  
  
# List of all files packaged with this module  
# FileList = @()  
  
# Private data to pass to the module specified in RootModule/ModuleToProcess. This may also contain a PSData hashtable with additional module metadata used by PowerShell.  
PrivateData = @{  
  
 PSData = @{  
  
 # Tags applied to this module. These help with module discovery in online galleries.  
 # Tags = @()  
  
 # A URL to the license for this module.  
 # LicenseUri = ''  
  
 # A URL to the main website for this project.  
 # ProjectUri = ''  
  
 # A URL to an icon representing this module.  
 # IconUri = ''  
  
 # ReleaseNotes of this module  
 # ReleaseNotes = ''  
  
 } # End of PSData hashtable  
  
} # End of PrivateData hashtable  
  
# HelpInfo URI of this module  
# HelpInfoURI = ''  
  
# Default prefix for commands exported from this module. Override the default prefix using Import-Module -Prefix.  
# DefaultCommandPrefix = ''  
  
}

# Deliverable

Upload this document with completed answers to i-learn.