2017

SVDNN-NUGET SERVER

Andre Masters

Automated Design Solutions

2/22/2017

Nuget Server Setup Instructions

Contents

Perquisites

Completion of Tutorial #1

Technical Resource Requirements:

- Laptop Computer - Windows 10
- Visual Studio Community Edition 2015 or 2017

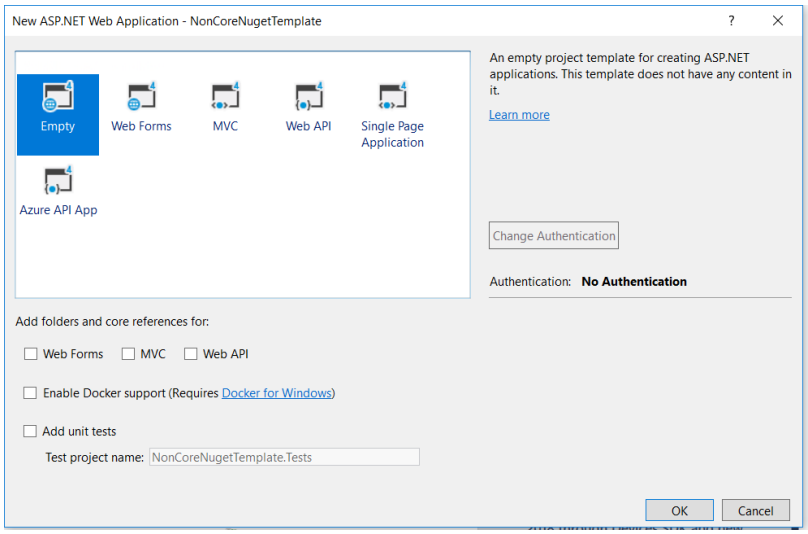
Introduction to Nuget Server

The purpose of having a Nuget server is to serve as an aid in the software development process. A Nuget server will allow for an organized and automated process for updating and storing library code to be reused in a later project.

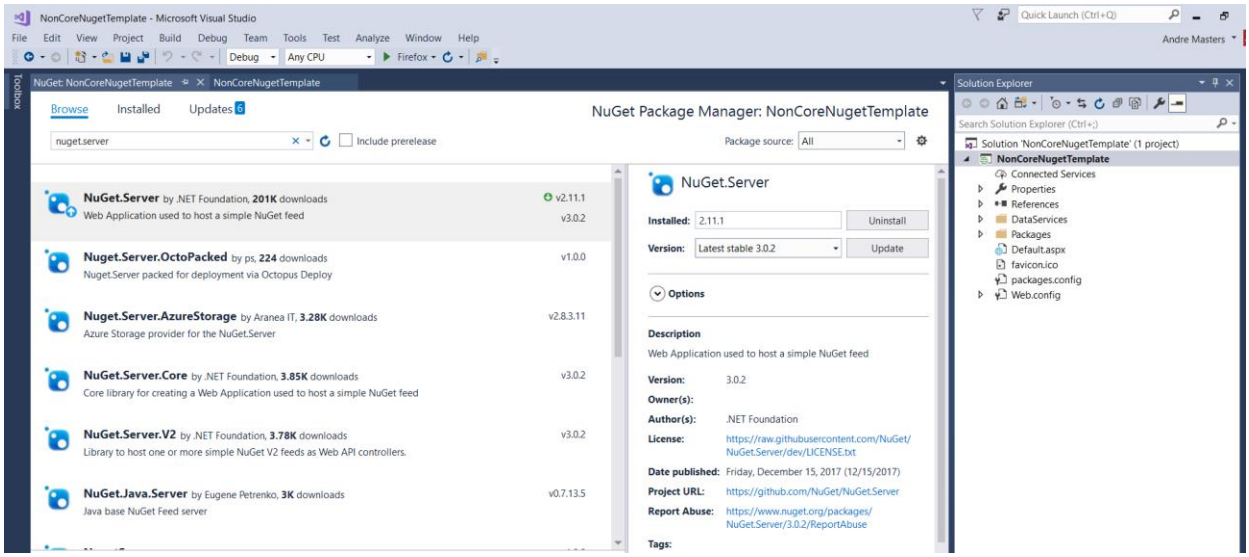
Part I Create a Simple Web Server

Create a web server by completing the following steps as illustrated in the sequence of steps below.

1.0 Create an empty template as shown in the following image.

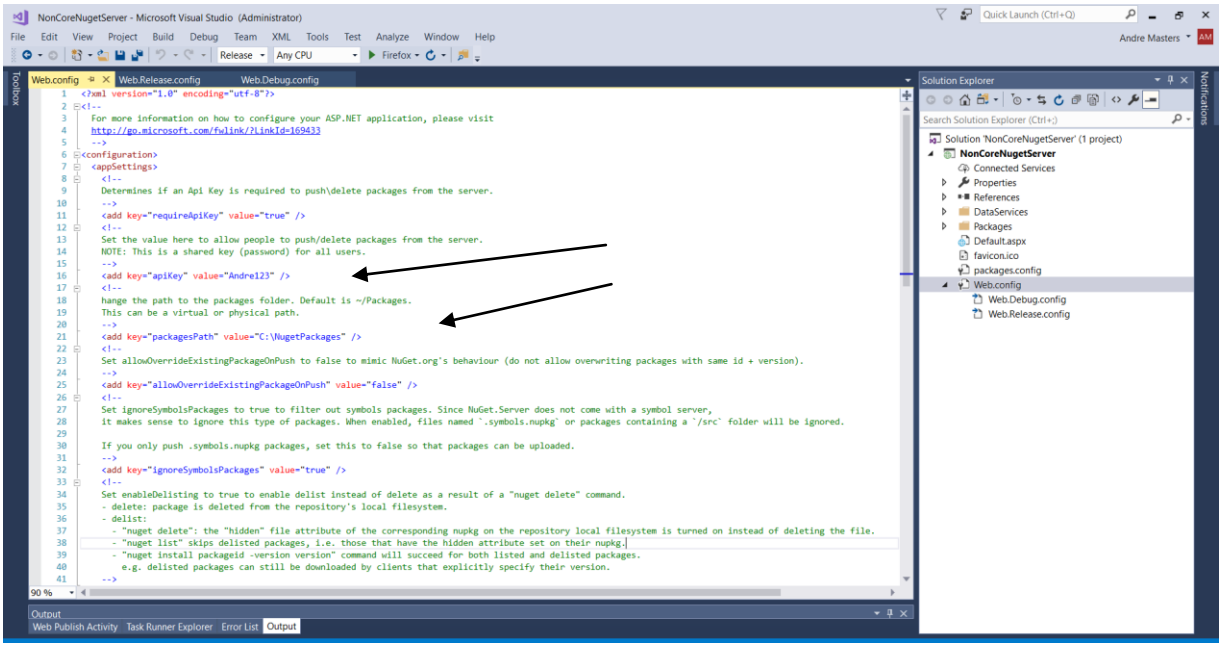


2.0 Create an empty server template as shown in the following image.

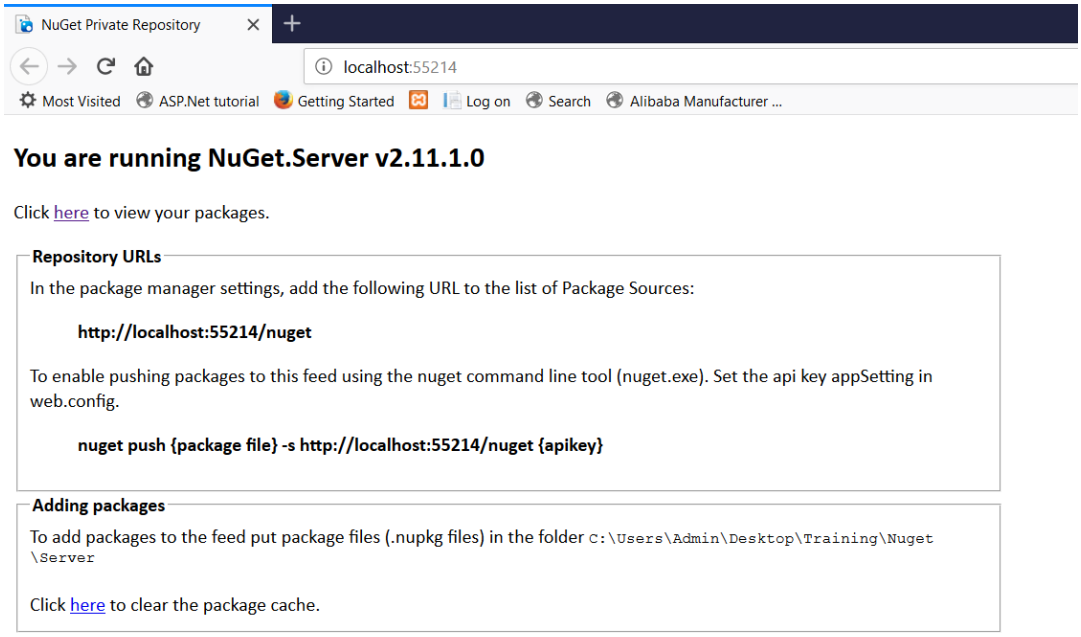


3.0 Edit Web Config file for the desired package path as shown below.

3.0 Enter the following value for apiKey: Andre123 as shown below.

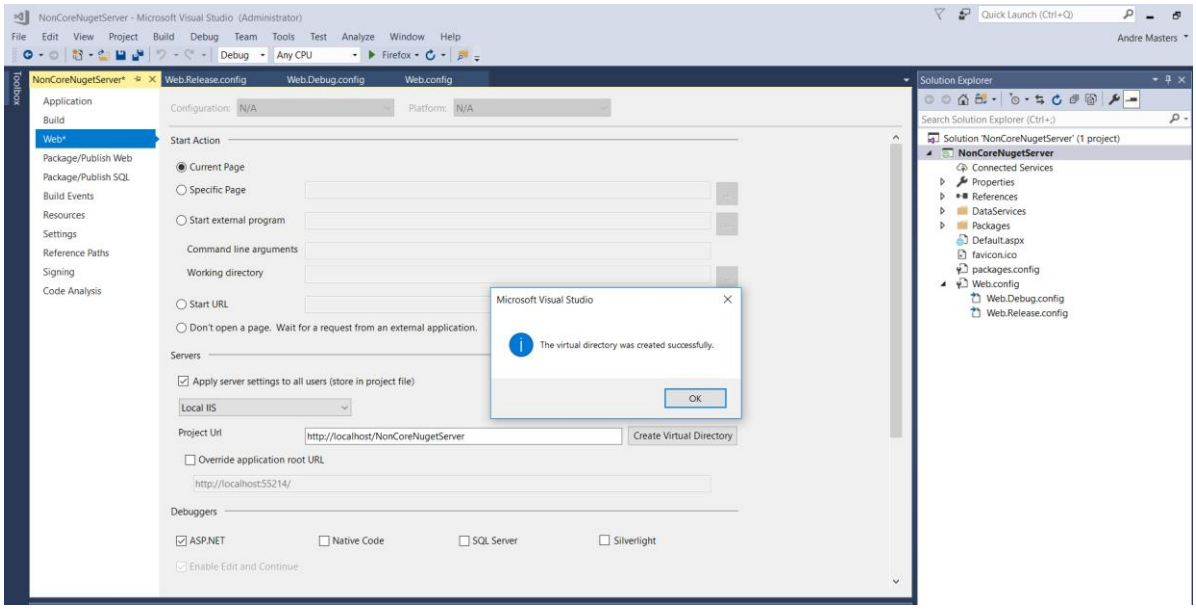


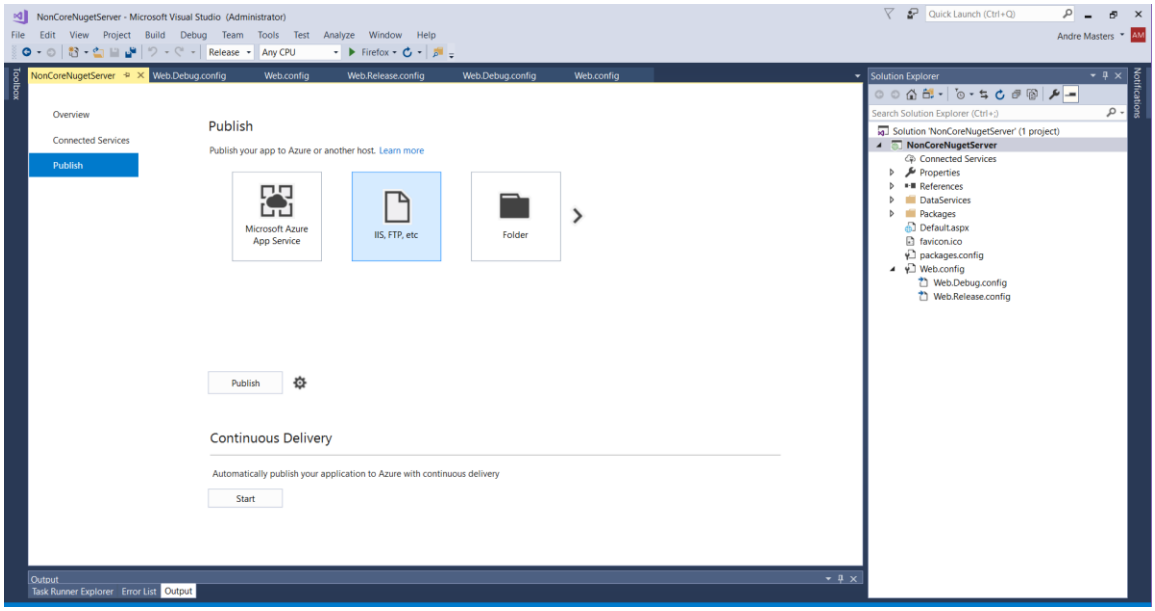
3.0 Press F5 to verify the nuget server is running. The following should appear in the browser as shown below.

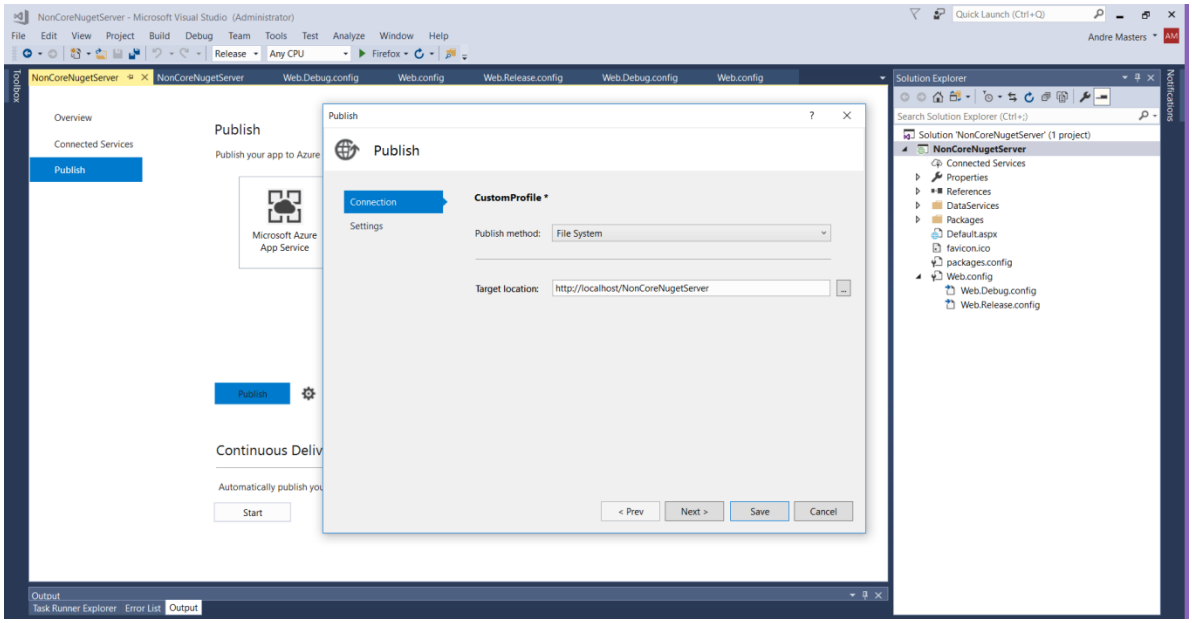


4.0 Set the project into release mode.

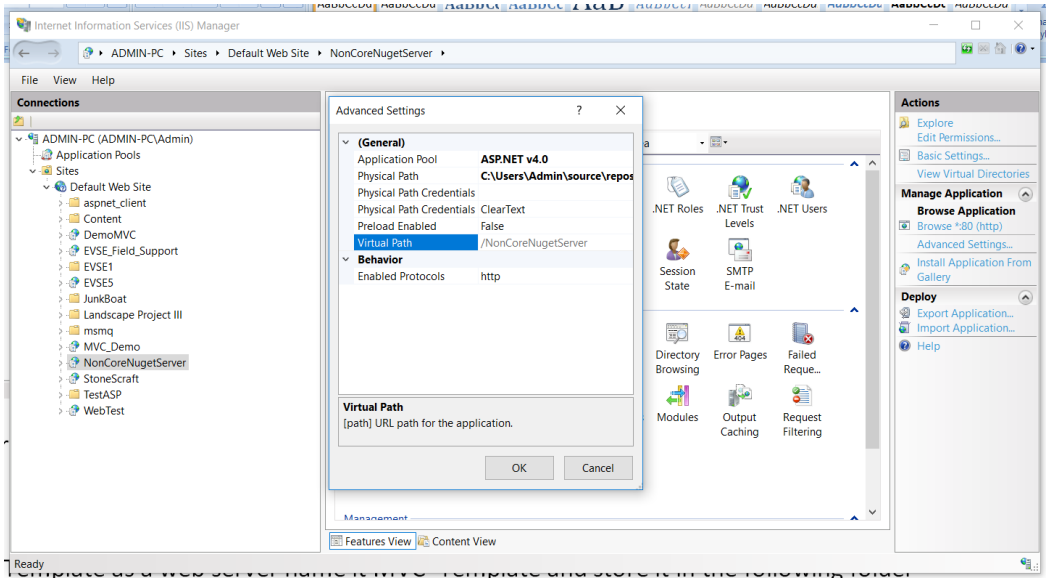
5.0 Configure the project properties for a local IIS web application as shown below.



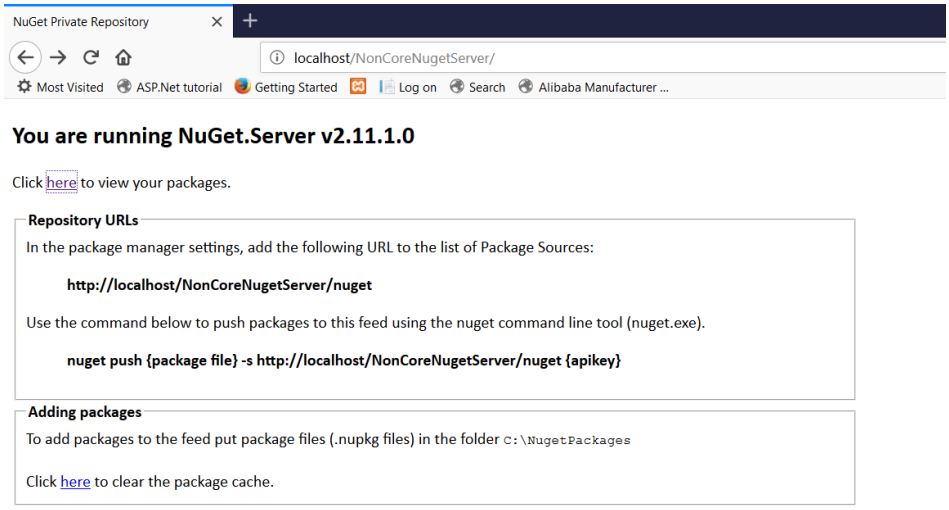

NonCoreNugetServer - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Release
Any CPU
Firefox
Andre Masters
NonCoreNugetServer
Web.Debug.config
Web.config
Web.Release.config
Web.Debug.config
Web.config
Overview
Connected Services
Publish
Publish
Publish your app to Azure or another host. Learn more
Microsoft Azure App Service
IIS, FTP, etc
Folder
Publish
Continuous Delivery
Automatically publish your application to Azure with continuous delivery
Start
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'NonCoreNugetServer' (1 project)
NonCoreNugetServer
Connected Services
Properties
References
DataServices
Packages
Default.aspx
favicon.ico
packages.config
Web.config
Web.Debug.config
Web.Release.config
Output
Task Runner Explorer
Error List
Output


NonCoreNugetServer - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Release
Any CPU
Firefox
Andre Masters
NonCoreNugetServer
NonCoreNugetServer
Web.Debug.config
Web.config
Web.Release.config
Web.Debug.config
Web.config
Overview
Connected Services
Publish
Publish
Publish your app to Azure
Microsoft Azure App Service
Publish
Continuous Deliv
Automatically publish you
Start
Publish
Publish
Connection
Settings
CustomProfile *
Publish method:
File System
Target location:
http://localhost/NonCoreNugetServer
< Prev
Next >
Save
Cancel
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'NonCoreNugetServer' (1 project)
NonCoreNugetServer
Connected Services
Properties
References
DataServices
Packages
Default.aspx
favicon.ico
packages.config
Web.config
Web.Debug.config
Web.Release.config
Output
Task Runner Explorer
Error List
Output

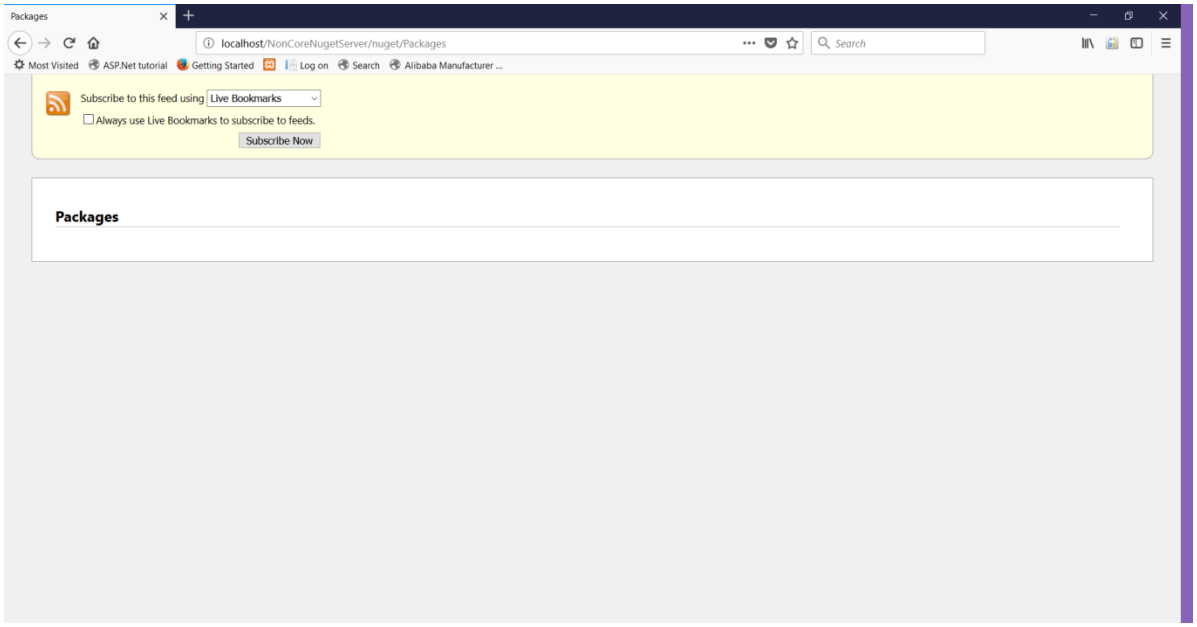
4.0 In the web application verify the following application pool matches matches the figure below.



4.0 In the web browser enter the following url and observe the following displays as shown below.



5.0 select click here link and observe the following screen appears with no packages installed.

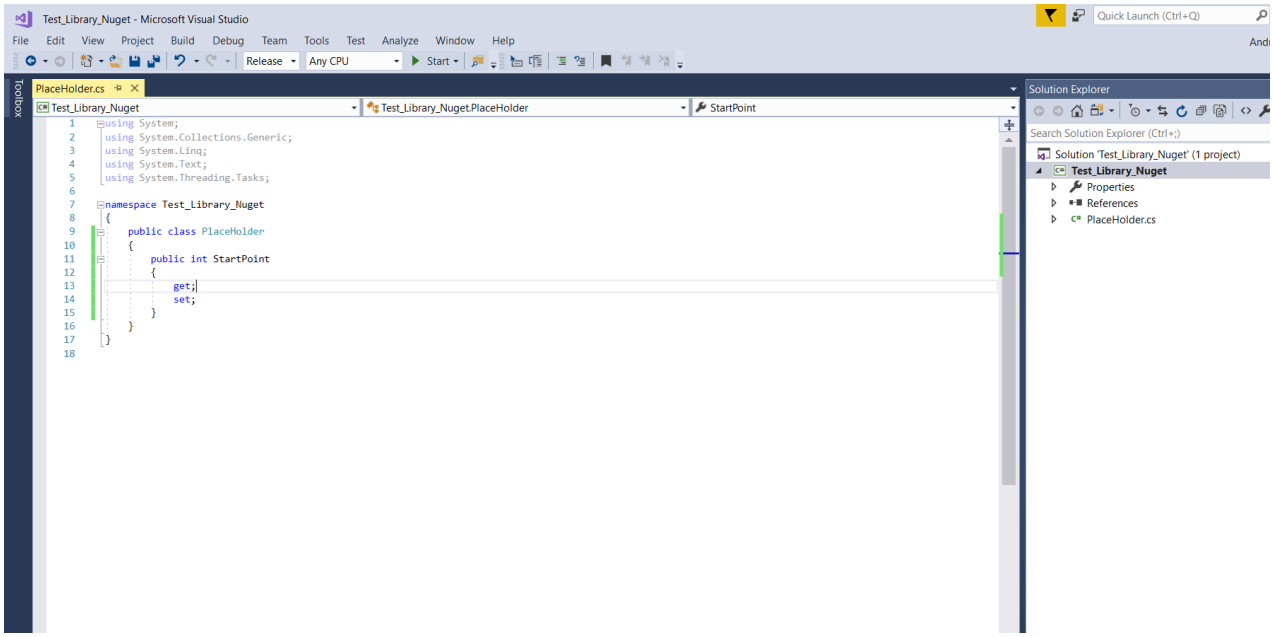


NOTE: THIS SECTION CANNOT BE BROWSED DIRECTLY FROM WITHIN IIS, IT MUST BE ACCESSED IN THE BROWSER URL.

THIS COMPLETES PART I SETTING UP A NUGET SERVER

Part II Create a Nuget Package

1.0 Create a Test Template as a web server name it MVC_Template and store it in the following folder structure: **C:_xCode\Nuget Libraries\Test_Library_Nuget**



2.0 Compile the project in Release Mode, and verify its success.

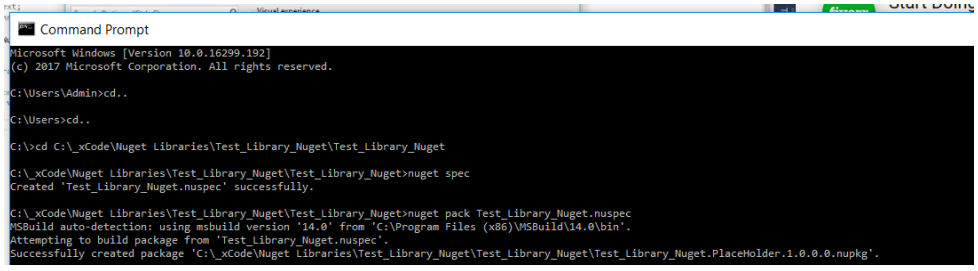
3.0 Open a command window and navigate to the project address.

C:_xCode\Nuget Libraries\Test_Library_Nuget\Test_Library_Nuget

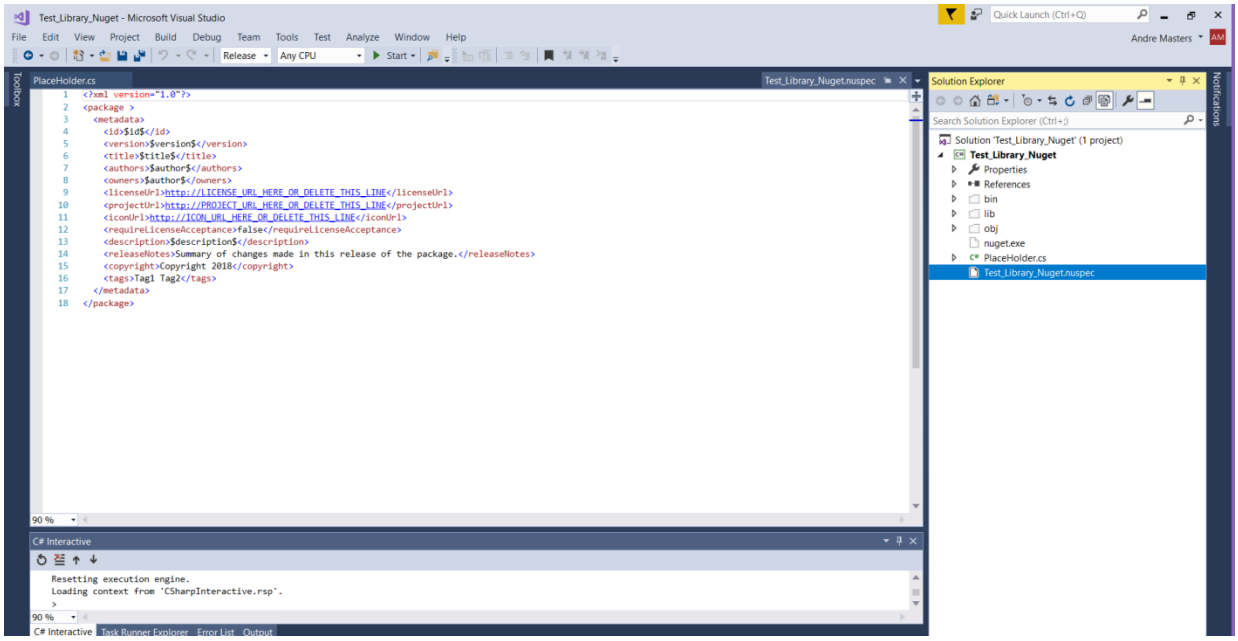
3.0 Observe the following files are present in the Test_Library_Nuget folder.

4.0 Open a command prompt and navigate to the project directory.

5.0 Enter the following Nuget command to create a nuspec file as shown below.

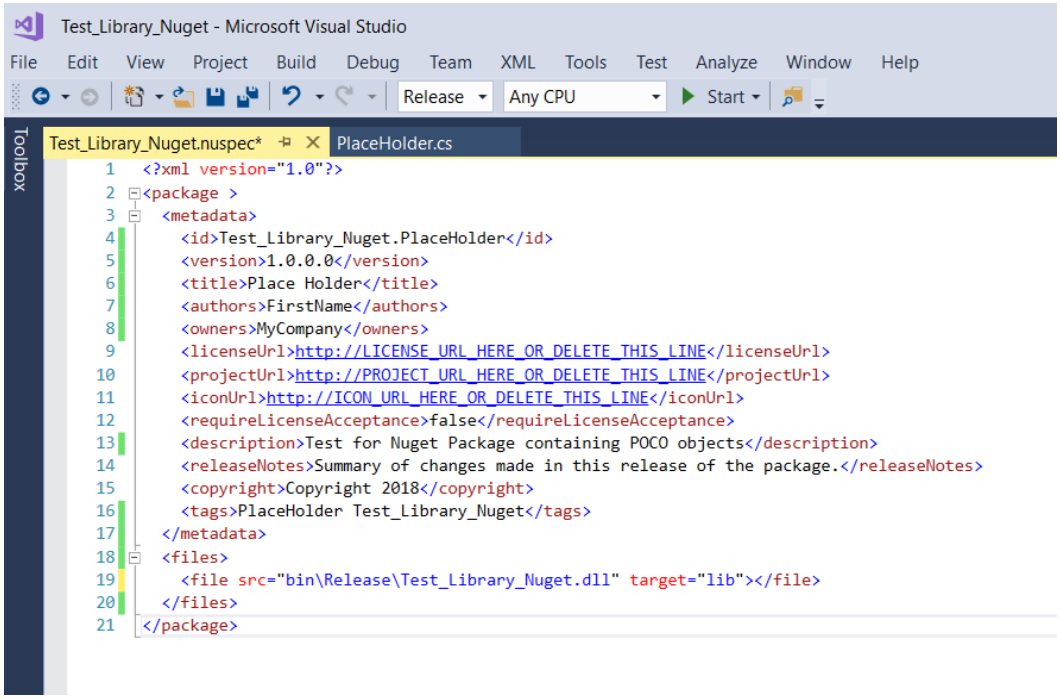


6.0 Include the Nuspec file into the project and open it for editing. Observe the following appears as shown below.



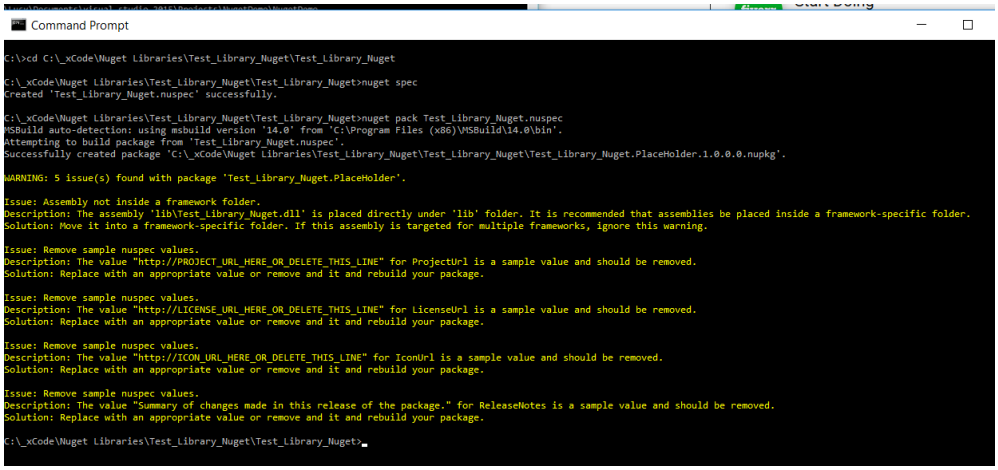
NOTE: To Include the Nuspec file select it RMB -> Select Include in Project option.

7.0 Make the following changes to the Nuspec file so that is matches what is shown in the image below.



8.0 Compile the project in release mode and verify it compiles successfully.

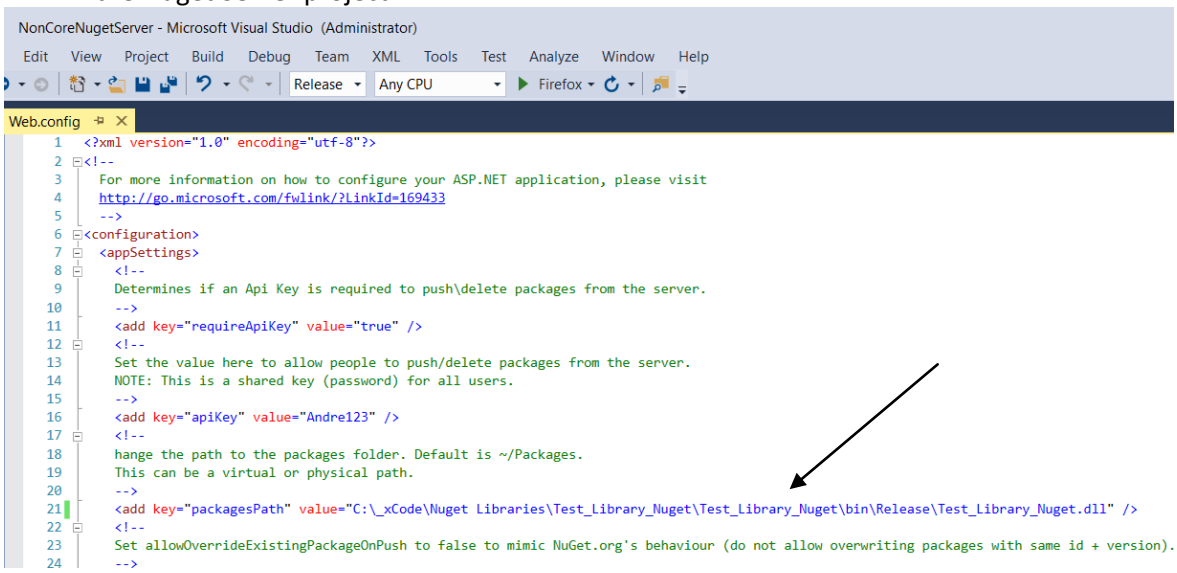
9.0 in the command prompt enter the pack command using the nuspec file requirements as shown in the image below.



THIS COMPLETES PART II CREATING A NUGET PACKAGE

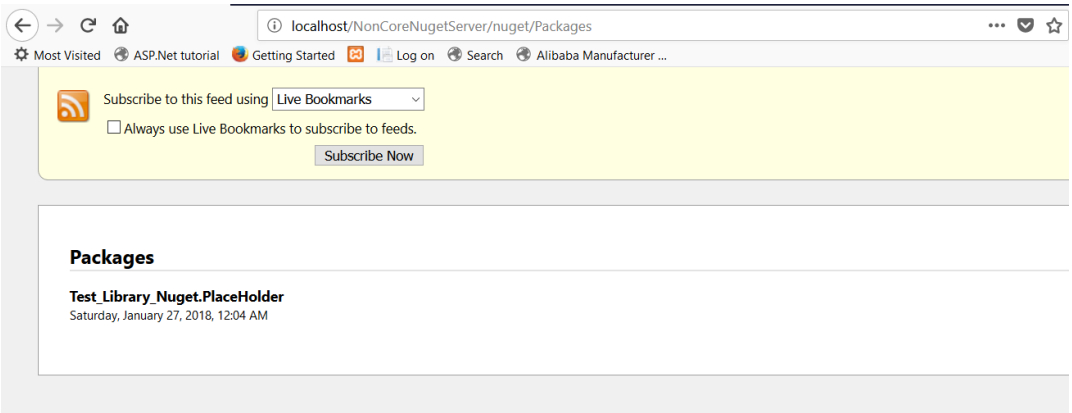
Part III Push a Nuget Package to the Local Nuget Server

10.0 To push a package to the Local Nuget Server, the first step it to update the Web Config file in the in the Nuget Server project.



11.0 In the Command prompt push the package to the allowed location on the Nuget Server. The Nuget Server home page details the correct location for package migrations.

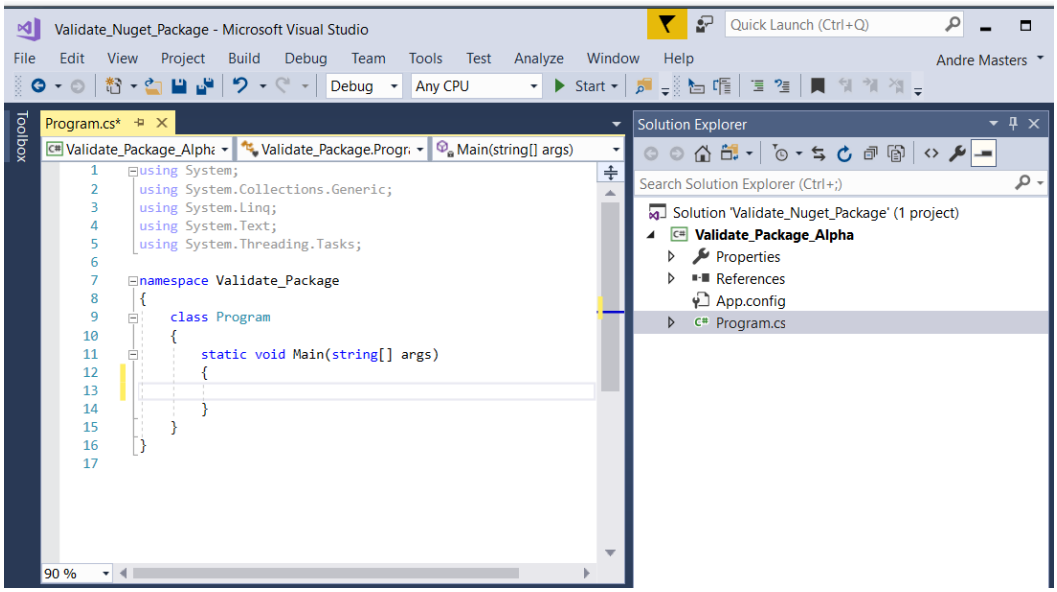
12.0 Verify the package exists by clicking on view packages on the main Nuget server page.



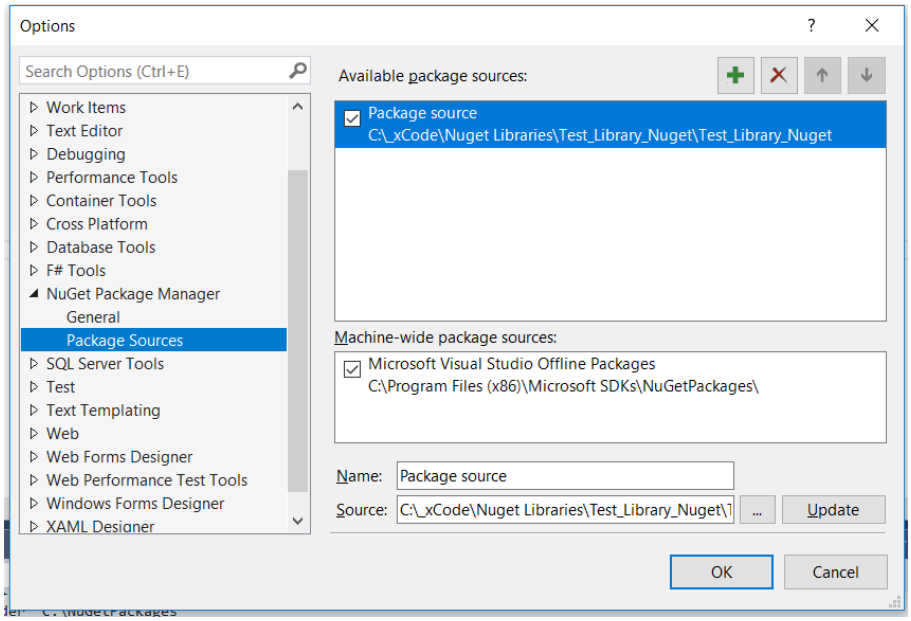
THIS COMPLETES PART III PUSHING A NUGET PACKAGE

Part IV Consume a Nuget Package in a Visual Studio Project

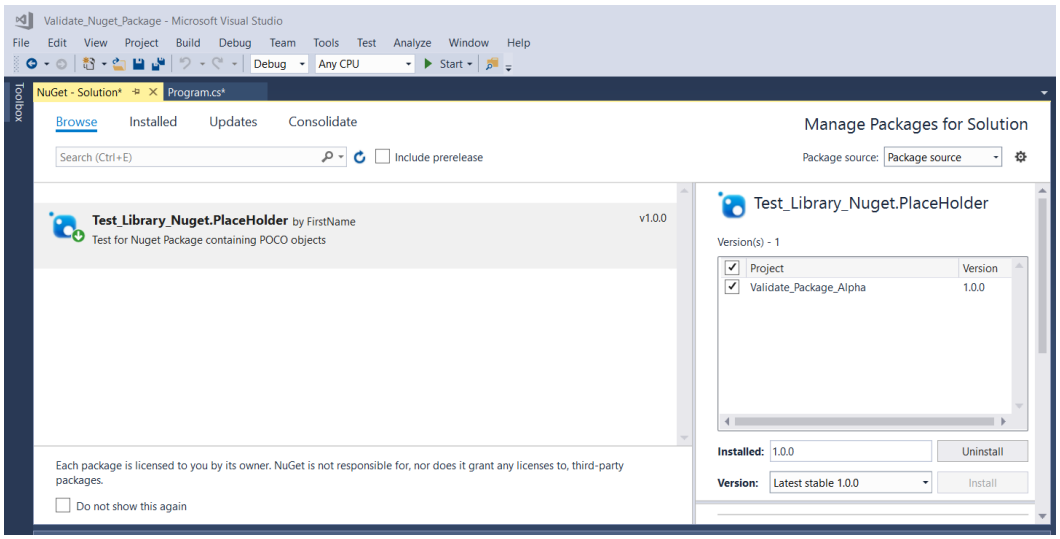
13.0 Create a new console application in Visual Studio. Call it Validate_Nuget_Package and store it in _xCode directory.



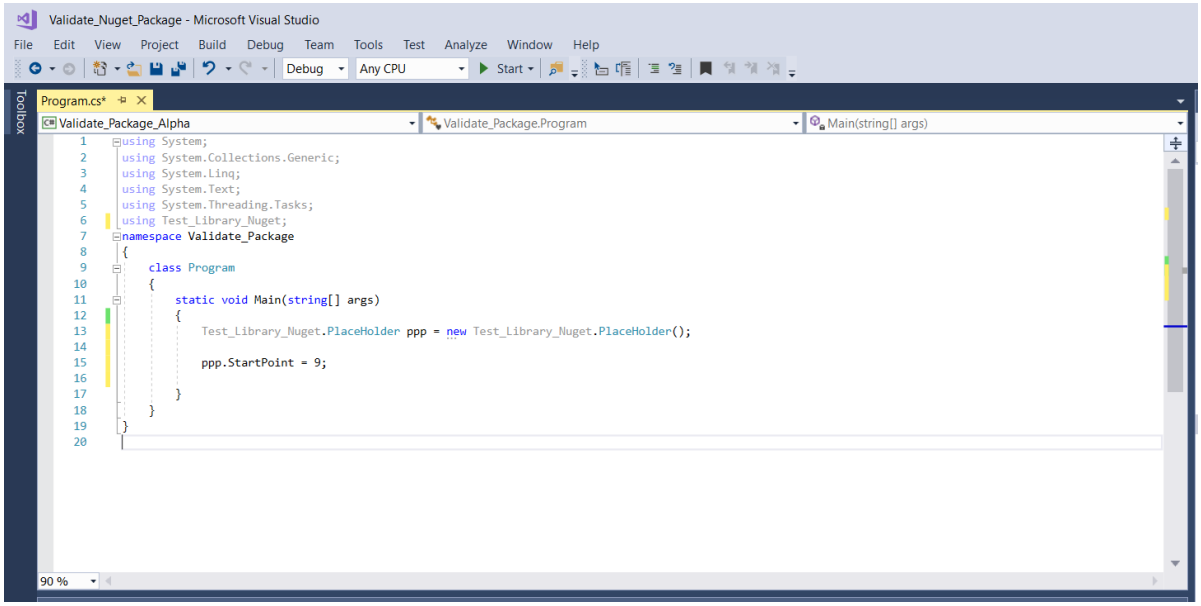
14.0 Update the Nuget Settings to include a Nuget Package, by Selecting Nuget Settings under the Tools Menu as shown below.



15.0 Navigate to the package source for the local Nuget packages as shown below.



16.0 Verify the reference was accepted by the project by entering the following code construction as shown below.



THIS COMPLETES PART IV PULLING DOWN A NUGET PACKAGE