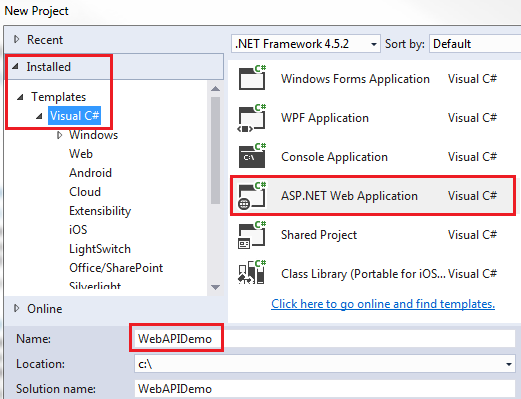
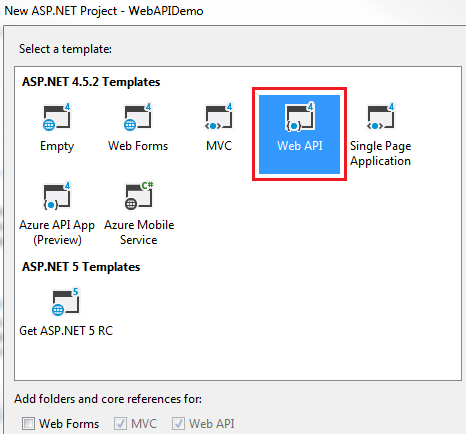
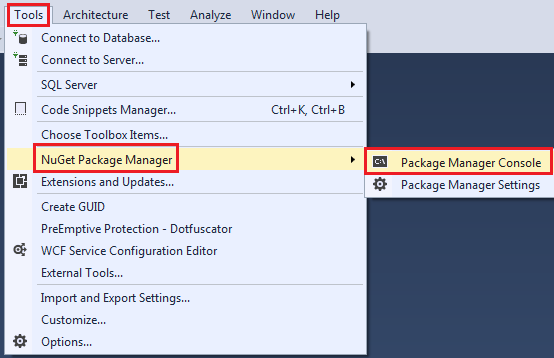
**In this video we will discuss**  
1. Creating a new ASP.NET Web API Project  
2. Explore and understand the Web API code auto-generated by Visual Studio   
  
  
  
For this course, we will be using Visual Studio 2015.  
  
**Creating a new ASP.NET Web API Project**  
1. Open Visual Studio and select File - New - Project  
  
2. In the "New Project" window  
    Select "Visual C#" under "Installed - Templates"  
    From the middle pane select, ASP.NET Web Application  
    Name the project "WebAPIDemo" and click "OK"   
  
   
  
3. On the next window, select "Web API" and click "OK"   
  
   
  
While creating the Web API project, you may get the following errors  
Package Installation Error - Could not add all required packages to the project. The following packages failed to install from 'C:\Program Files (x86)\Microsoft ASP.NET\ASP.NET Web Stack 5\Packages'  
  
Failed to initialize the PowerShell host.  
  
**If you do then follow the below steps which may help resolve the issue**  
1. Close all instances of Visual Studio  
  
2. Open Windows Powershell as an Administrator and execute the following command  
Set ExecutionPolicy AllSigned  
  
3. Run Visual Studio 2015 as an Administrator  
  
4. Open Package Manager Console window in Visual Studio. To do this click on Tools - NuGet Package Manager - Package Manager Console   
  
   
  
5 In the Package Manager Console, type [R] for Run once and press the Enter key  
  
At this point, you will be able to create a new Web API project.  
  
**Now, let us explore and understand the Web API code auto-generated by Visual Studio**  
  
1. If you have worked with ASP.NET MVC, then project folder structure should be familiar to you. Notice with in the **Controllers** folder we have **ValuesController** which inherits from ApiController class that is present in System.Web.Http namespace. This is different from the MVC controller. The MVC Controller class inherits from the Controllerclass that is present in System.Web.Mvc namespace. The HomeController class which is an MVC controller inherits from the Controller class.  
  
2. Notice in the **ValuesController** class we have methods (Get, Put, Post & Delete) that map to the HTTP verbs (GET, PUT, POST, DELETE) respectively. We have 2 overloaded versions of Get() method - One without any parameters and the other with id parameter. Both of these methods respond to the **GET** http verb depending on whether the id parameter is specified in the URI or not.  
  
3. Now let's look at the default route that is in place for our Web API project. We have the Application\_Start() event handler In Global.asax file. This event is raised when the application starts. In the Application\_Start() event handler method we have configuration for Filters, Bundles etc. The one that we are interested in is the configuration for our Web API project, which is in WebApiConfig.Register() method. Right click on WebApiConfig.Register and select "Go To Definition" from the context menu. This will take you to the Register() method in the WebApiConfig class. This class is in App\_Start folder.  
  
4. In the Register() method we have the default route configured for our Web API project. Web API routes are different from the MVC routes. You can find the MVC routes in RouteConfig.cs file in App\_Start folder.  
  
5. The default Web API route starts with the word api and then / and then the name of the controller and another / and an optiontion id parameter.  
"api/{controller}/{id}"  
  
6. At this point if we use the following URI in the browser, we get an error - Authorization has been denied for this request.  
http://localhost/api/values  
  
7. To get rid of this error, comment Authorize attribute on the **ValuesController** class. This is related to security which we will discuss in a later video.  
  
8. Now if you visit, http://localhost/api/values, you should see the following XML as the result

<ArrayOfstring xmlns:i="http://www.w3.org/2001/XMLSchema-instance"

               xmlns="http://schemas.microsoft.com/2003/10/Serialization/Arrays">

<string>value1</string>

<string>value2</string>

</ArrayOfstring>

9. Let us understand what is going on here. The name of the controller is values. So if we use a URI http://localhost:portnumber/api/values, then the web api is going to look for a controller with name **Values + the word controller** in your project. So if you have specified values in the URI it is going to look for **ValuesController**, if you specify **Products**, then it is going to look for **ProductsController**. This is all by convention and works this way out of the box.  
  
10. In a real world application this might be the domain name, for example  
http://pragimtech.com/api/values  
  
11. The browser is issuing a GET request which maps to the Get() method in the **ValuesController** class. The GET() in the values controller is returning value1 and value2 which is what we see in the browser.  
  
12. We have another overload of GET() method which takes Id parameter. Remember with the default route, the id parameter is optional. That is the reason we are able to call the GET method with or without the Id parameter. If the id parameter is specified in the URI, then the Get() method with the parameter in values controller is called   
  
13. If a controller with the specified name is not found you will get an error. For example, in your project if you comment the ValuesController class in your project and then use the URI /api/values you will get the following error  
  
No HTTP resource was found that matches the request URI 'http://localhost:15648/api/values'. No type was found that matches the controller named 'values'.  
  
In our next video we will discuss how to perform the rest of the actions PUT, POST and DELETE. 