(T6)討論 Authentication 登入登出。實作最簡單的 AuthorizationFilterAttribute、Token
CourseGUID 4c5822ff-7111-4e25-a336-ef18d48d54bd
==========
(T6)討論 Authentication 登入登出。實作最簡單的 AuthorizationFilterAttribute、Token
(T6-1)複習 Ssl(SecureSocketsLayer)、Https、EnableCors(CrossOriginResourceSharing)
(T6-2)討論 Authentication 登入登出。實作最簡單的 AuthorizationFilterAttribute、Token
(T6-3)用 Postman、Fiddler 測試 AuthorizationFilterAttribute、Token
(T6-4)用 MVC 寫 Login(登入)Logout(登出)頁面
==========
1. OnlineGame DB
1.0. Some points
1.1. TSQL
1.2. Security login
---
2. OnlineGame Solution
2.1. OnlineGame Solution
2.2. OnlineGame.Data
2.3. OnlineGame.WebApi
2.4. OnlineGame.Mvc
---
3. OnlineGame.Data
3.1. Install Entity Framework
3.2. ADO.Net Entity Data Model - Entity Framework
---
4. OnlineGame.WebApi
4.1. Install Entity Framework
4.2. OnlineGame.WebApi/Web.config : Add Connection String
4.3. Add Reference
4.4. OnlineGame.WebApi/Controllers/Api/GamerController.cs (Bug)
4.5. OnlineGame.WebApi/App_Start/WebApiConfig.cs (Fix Bug)
---
5. OnlineGame.WebApi - WebApi Cors (Cross Origin Resource Sharing)
5.1. WebApi Cors (Cross Origin Resource Sharing) allows Jquery AJAX may call Web API in the different origins
5.2. Install NuGet Package
5.3. OnlineGame.WebApi/App_Start/WebApiConfig.cs

6.OnlineGame.WebApi - Enable SSL (Secure Sockets Layer) and Create self-signed certificate
6.1. OnlineGame.WebApi Enable SSL via Visual Studio 2017
6.2. OnlineGame.WebApi/WebShared/HttpsAuthorizationFilterAttribute.cs
6.3. OnlineGame.WebApi/App_Start/WebApiConfig.cs

7.OnlineGame.WebApi - Basic Authentication
7.1. OnlineGame.WebApi/Account/Authentication.cs
7.2. OnlineGame.WebApi/Account/BasicAuthorizationFilterAttribute.cs
7.3. OnlineGame.WebApi/App_Start/WebApiConfig.cs
7.4. OnlineGame.WebApi/Controllers/Api/GamerController.cs
7.5. base64 encode
7.6. Fiddler test Basic login
7.7. Postman test Basic login
---
8. OnlineGame.Mvc
8.1. Install Entity Framework
8.2. OnlineGame.Mvc/Web.config : Add Connection String
8.3. Add Reference

8.4. OnlineGame.Mvc/Controllers/GamerController.cs
8.5. OnlineGame.Mvc/Controllers/GamerController.cs
8.6. OnlineGame.WebApi/Controllers/Api/GamerController.cs
8.7. OnlineGame.Mvc/Views/Gamer/IndexWebApi.cshtml
==========

1. OnlineGame DB

The tutorial will discuss
Build a very simple login function.
It is for understanding the basic concept, not for real-world practice.
Create your own "Basic Token"

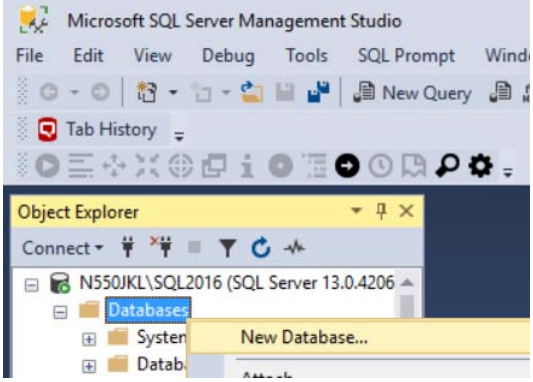
本堂課討論
建議一個不實用但是最基本的 Login。只是要闡述 login 觀念
建立手寫"Basic Token"

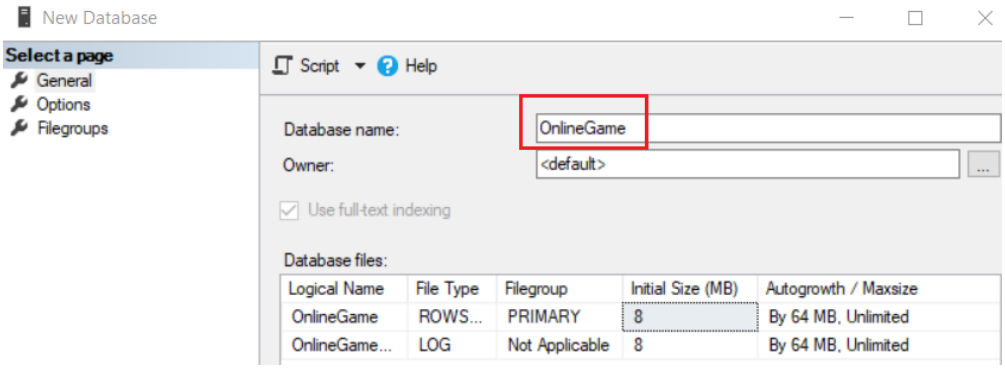
1.0. Some points

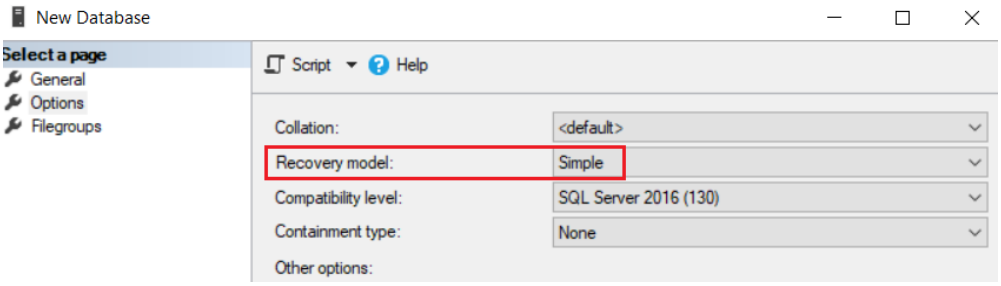
Reference:
保哥的 Certificate 的觀念補充
https://blog.miniasp.com/post/2018/04/21/PKI-Digital-Certificate-Format-Convertion-Notes.aspx
1.
Regular expression
https://regexr.com/
2.
Calling Stored Procedure from Entity Framework 6 Code First
http://www.dotnetodyssey.com/2015/03/12/calling-stored-procedure-from-entity-framework-6-code-first/

1.1. TSQL

In SQL server Management Studio (SSMS)
Database --> Right Click --> New Database -->
In General Tab -->
Name: **OnlineGame**
In options Tab --> Recovery model : **Simple**







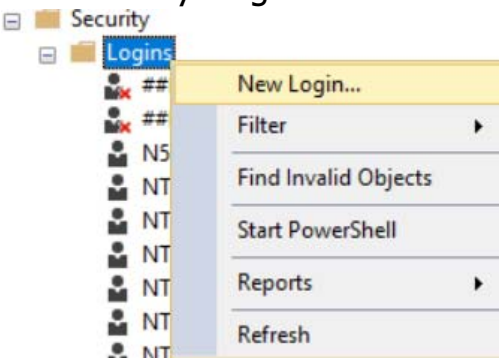

```
--1 -------------------------------------------------------
--Drop Table if it exists.
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'GamerIdentity' ) )
    BEGIN
        TRUNCATE TABLE GamerIdentity;
        DROP TABLE GamerIdentity;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'Gamer' ) )
    BEGIN
        TRUNCATE TABLE Gamer;
        DROP TABLE Gamer;
    END;
GO -- Run the previous command and begins new batch
--2 -------------------------------------------------------
CREATE TABLE Gamer
    (
       Id INT PRIMARY KEY
              IDENTITY(1, 1)
              NOT NULL ,
      Name NVARCHAR(50) NOT NULL ,
       Gender NVARCHAR(50) NOT NULL ,
```

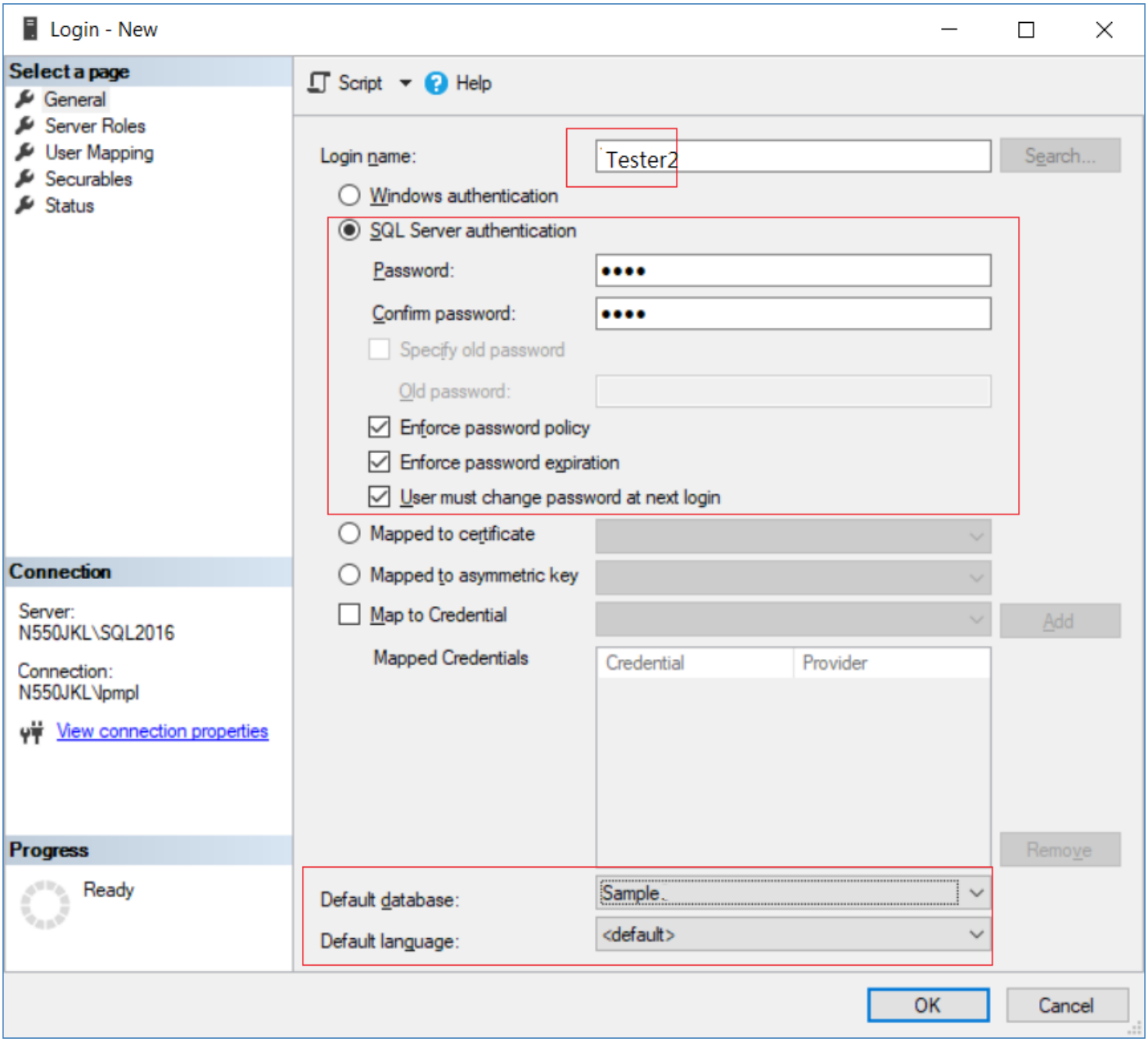
```sql
      Score INT NOT NULL ,
      GameMoney INT NOT NULL
   );
GO -- Run the previous command and begins new batch
CREATE TABLE GamerIdentity
   (
      Id INT PRIMARY KEY
            FOREIGN KEY REFERENCES Gamer ( Id ) ,
      UserName NVARCHAR(50) UNIQUE NOT NULL ,
      [Password] NVARCHAR(50) NOT NULL,
   );
GO -- Run the previous command and begins new batch
--3 -------------------------------------------------------
INSERT  INTO Gamer
VALUES  ( 'NameOne ABC', 'Male', 5000, 550 );
INSERT  INTO Gamer
VALUES  ( 'NameTwo ABCDE', 'Female', 4500, 1200 );
INSERT  INTO Gamer
VALUES  ( 'NameThree EFGH', 'Male', 6500, 3050 );
INSERT  INTO Gamer
VALUES  ( 'NameFour HIJKLMN', 'Female', 45000, 450 );
INSERT  INTO Gamer
VALUES  ( 'NameFive NOP', 'Male', 3000, 200 );
INSERT  INTO Gamer
VALUES  ( 'NameSix PQRSTUVW', 'Male', 4000, 700 );
INSERT  INTO Gamer
VALUES  ( 'NameSeven XYZ', 'Male', 450, 1500 );
GO -- Run the previous command and begins new batch
INSERT  INTO GamerIdentity
VALUES  ( 1, 'One', '1111' );
INSERT  INTO GamerIdentity
VALUES  ( 2, 'Two', '2222' );
INSERT  INTO GamerIdentity
VALUES  ( 3, 'Three', '3333' );
INSERT  INTO GamerIdentity
VALUES  ( 4, 'Four', '4444' );
INSERT  INTO GamerIdentity
VALUES  ( 5, 'Five', '5555' );
INSERT  INTO GamerIdentity
VALUES  ( 6, 'Six', '6666' );
INSERT  INTO GamerIdentity
VALUES  ( 7, 'Seven', '7777' );
GO -- Run the previous command and begins new batch
```

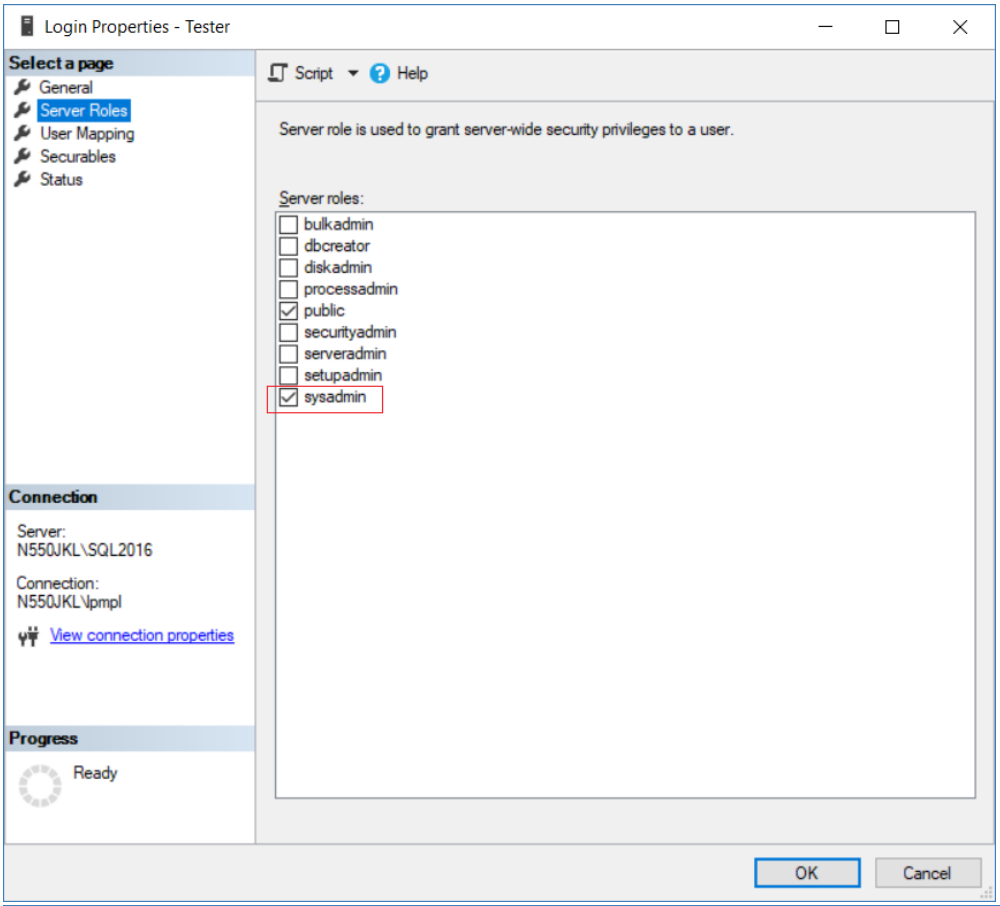
1.2. Security login

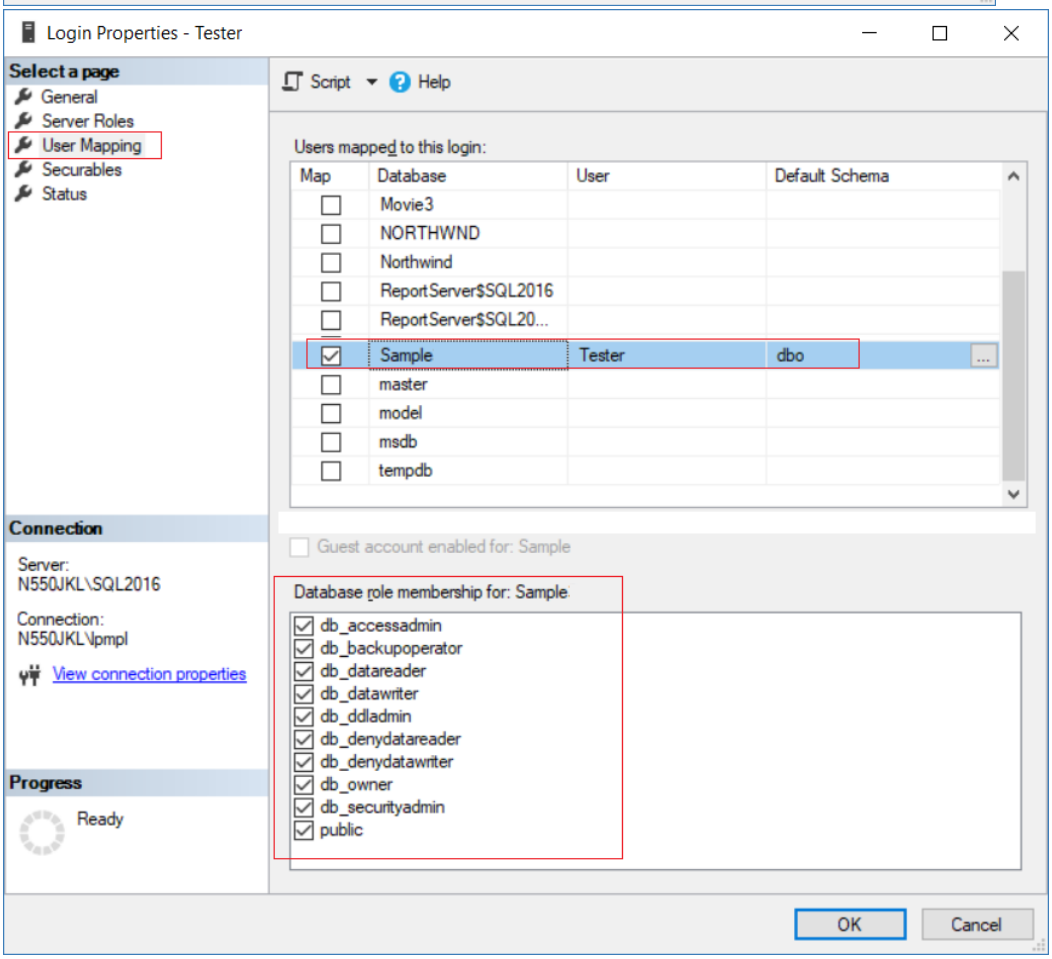
In SQL server
Object Explorer --> Security --> Logins --> New Logins
-->
General Tab
Login Name :

Tester2
Password:
1234
Default Database:
OnlineGame
-->
Server Roles Tab
Select
sysadmin
-->
User Mapping Tab
Select **OnlineGame**
Select every single role.




Login - New
Select a page
General
Server Roles
User Mapping
Securables
Status
Script
Help
Login name:
Tester2
Search...
Windows authentication
SQL Server authentication
Password:
Confirm password:
Specify old password
Old password:
Enforce password policy
Enforce password expiration
User must change password at next login
Mapped to certificate
Mapped to asymmetric key
Map to Credential
Add
Mapped Credentials
Credential
Provider
Remove
Connection
Server:
N550JKL\SQL2016
Connection:
N550JKL\lpmpl
View connection properties
Progress
Ready
Default database:
Sample
Default language:
<default>
OK
Cancel





2. OnlineGame Solution

2.1. OnlineGame Solution

File --> New --> Project... -->

Other Project Types --> Visual Studio Solutions --> Blank Solution
-->

Name: **OnlineGame**

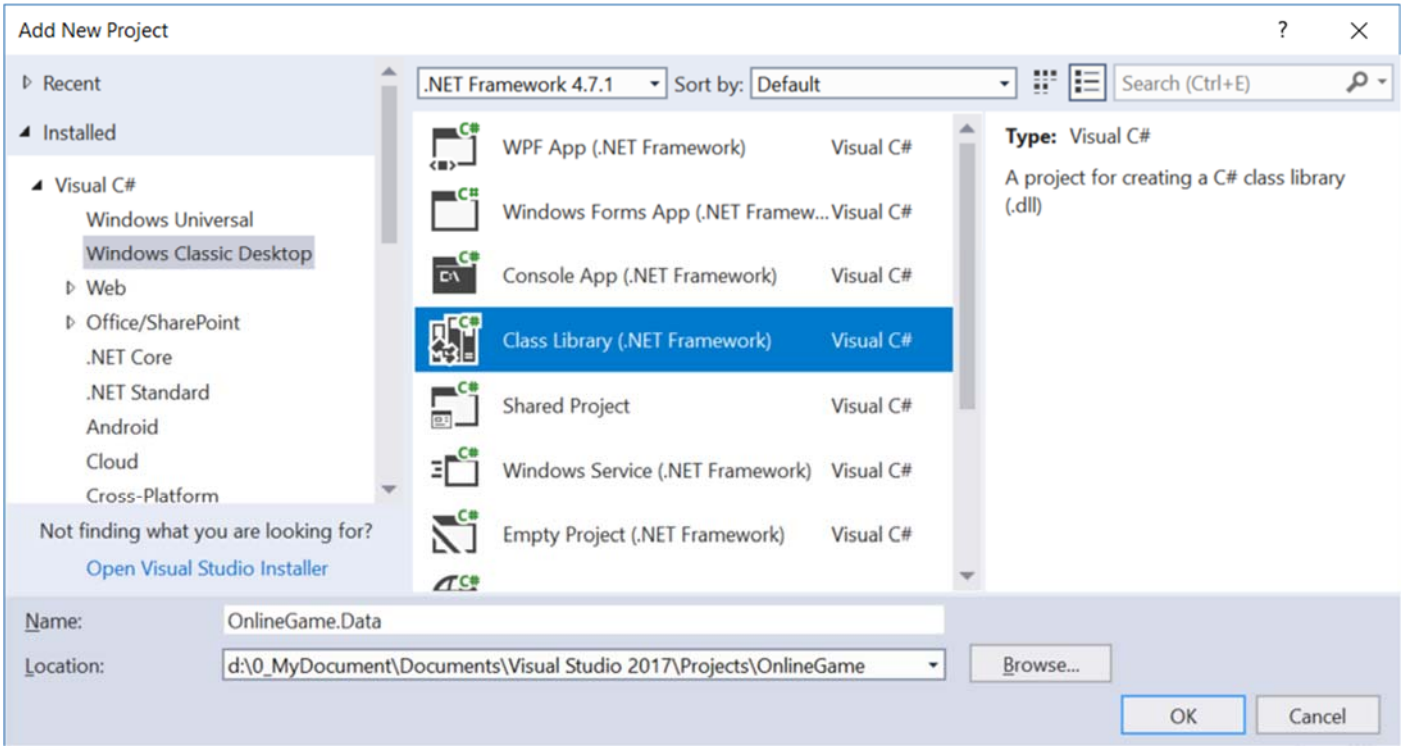
2.2. OnlineGame.Data

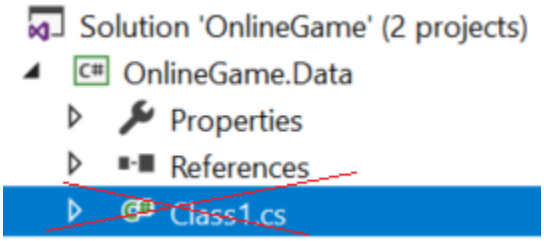
Solutions Name --> Add --> New Project -->

Visual C# --> **Class Library (.NET Framework)**
-->

Name:
OnlineGame.Data
-->

Delete Class1.cs





2.3. OnlineGame.WebApi

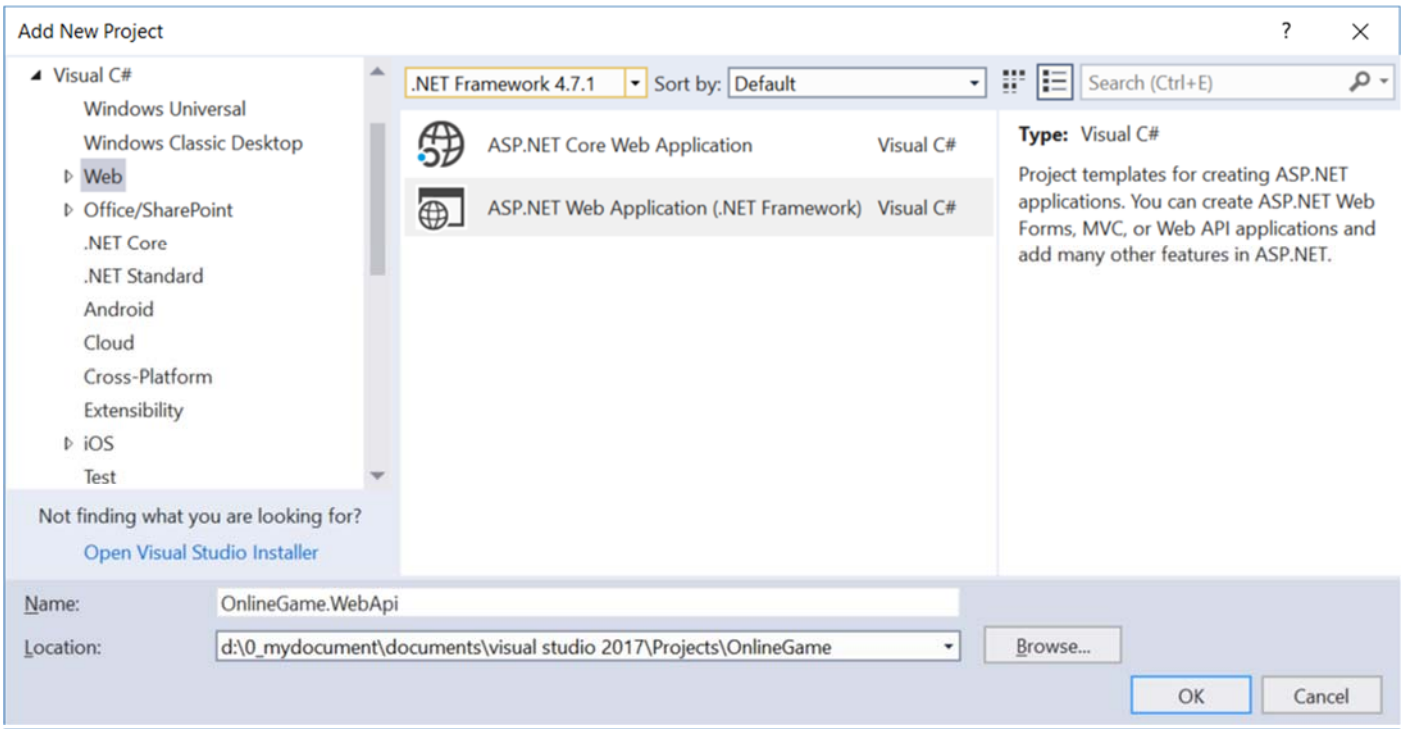
Solutions Name --> Add --> New Project -->

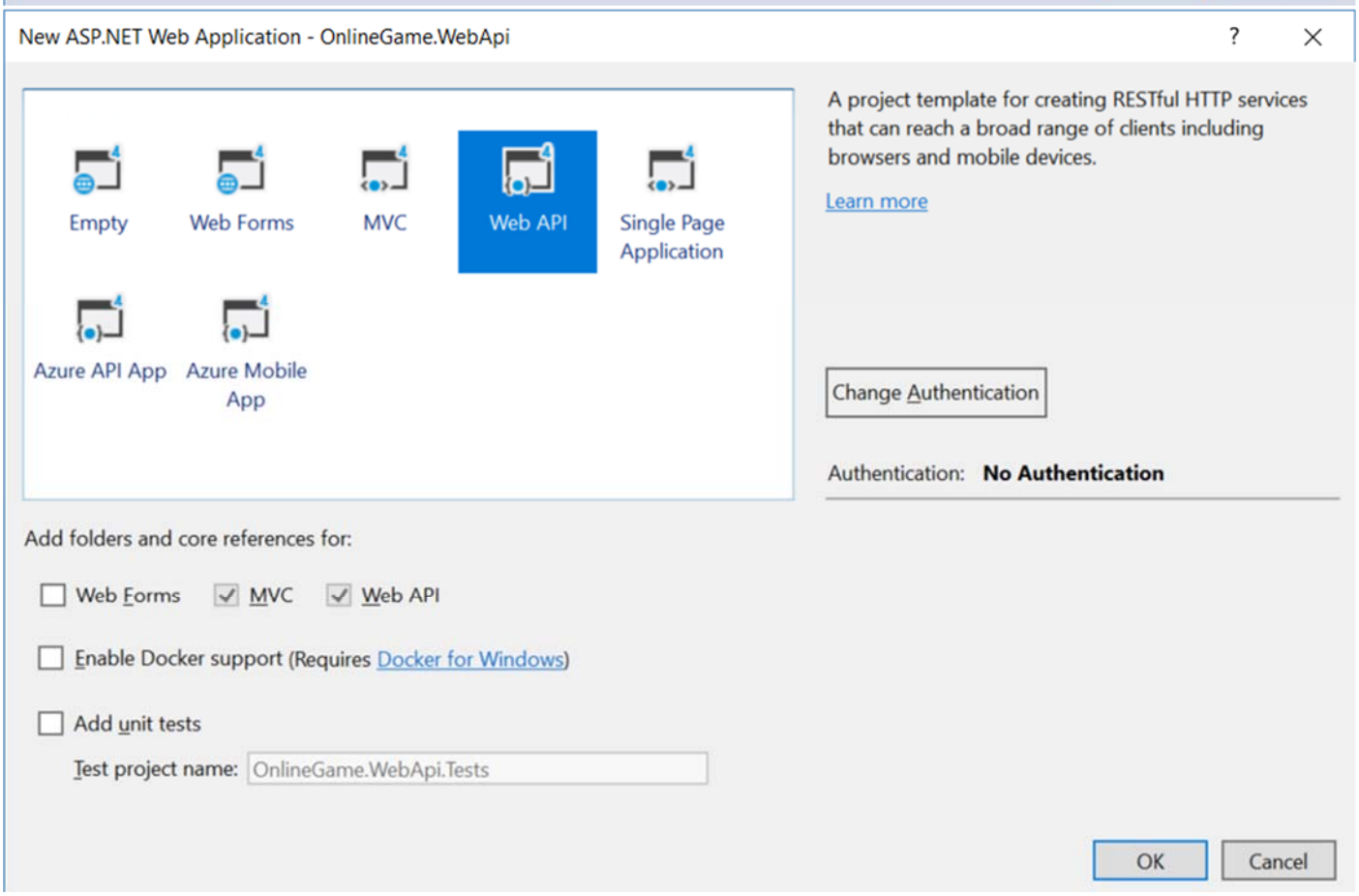
Visual C# --> Web --> ASP.NETWeb Application (.Net Framework)
-->

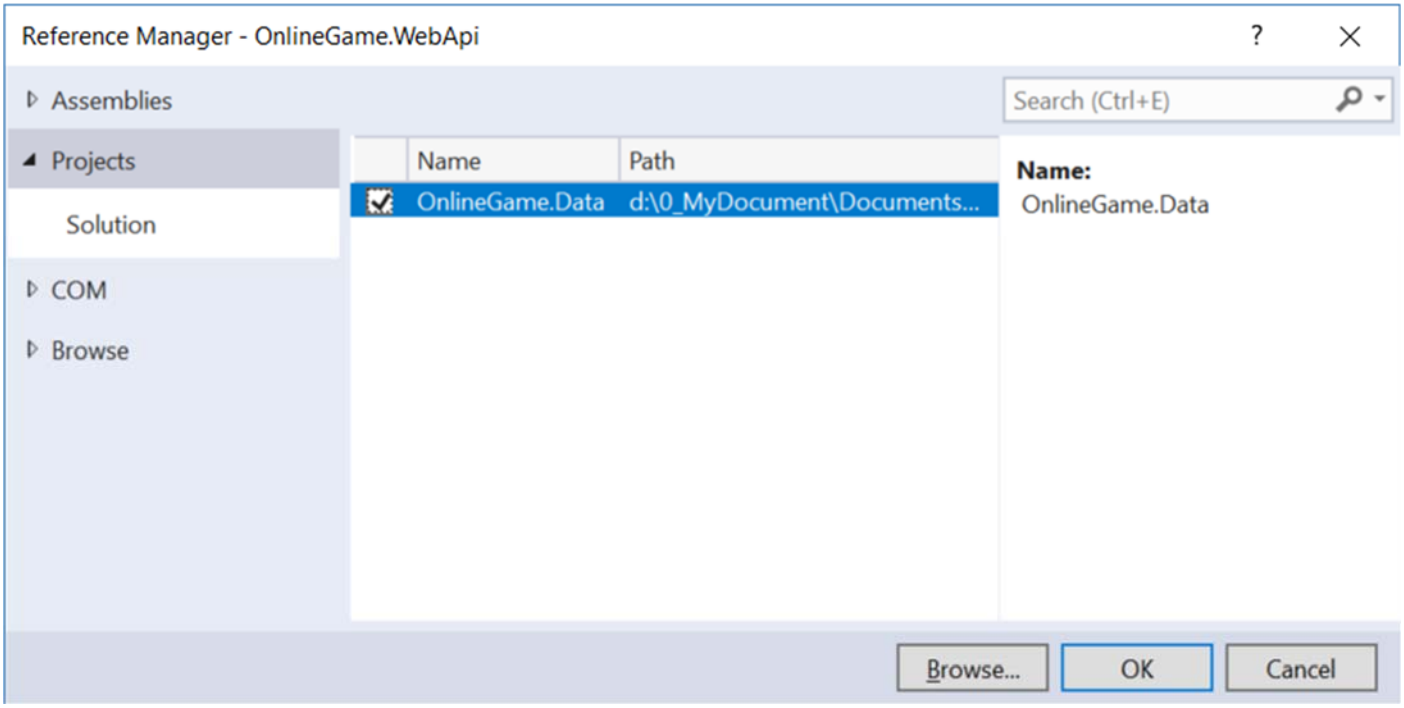
Name: **OnlineGame.WebApi**

--> Select "**Web API**" --> OK

--> Add Reference





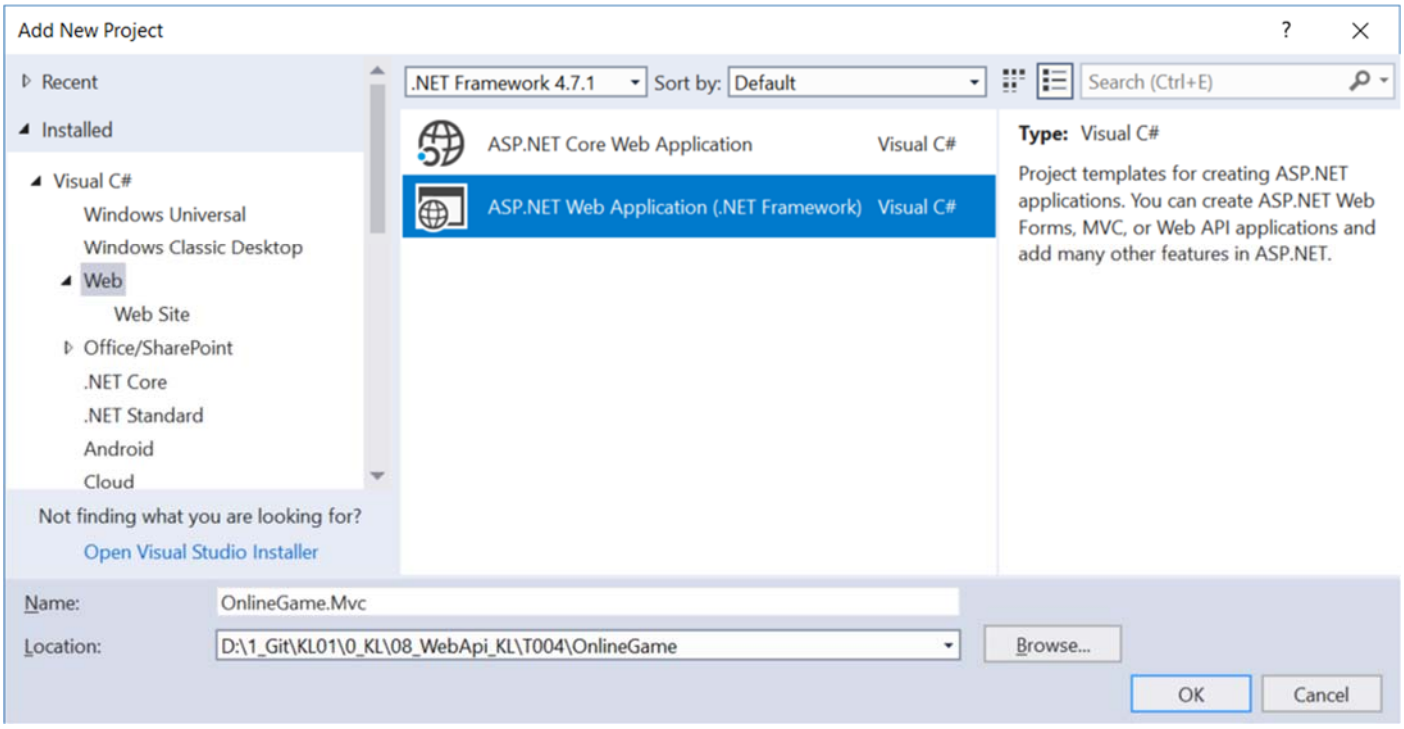


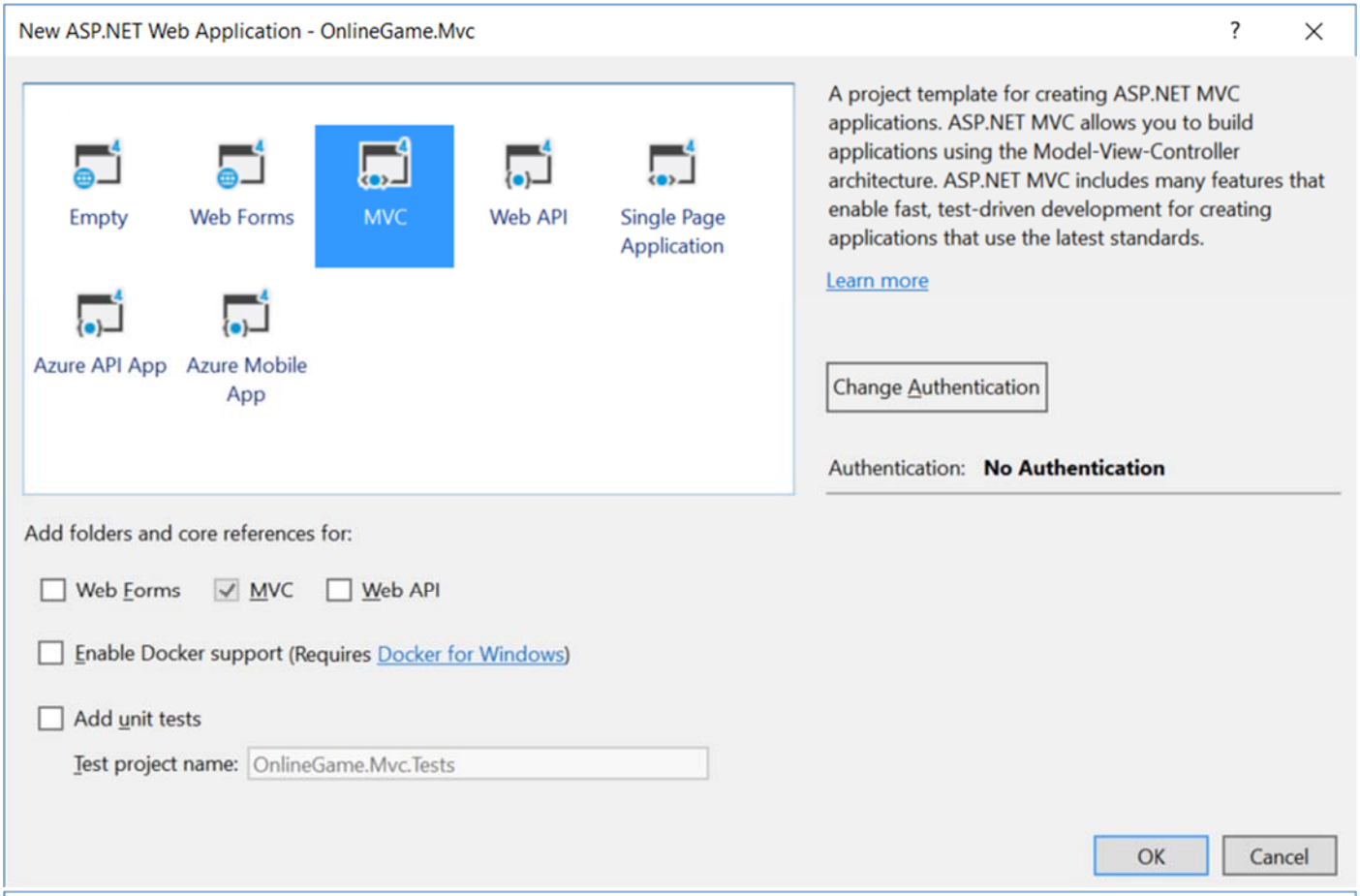
2.4. OnlineGame.Mvc

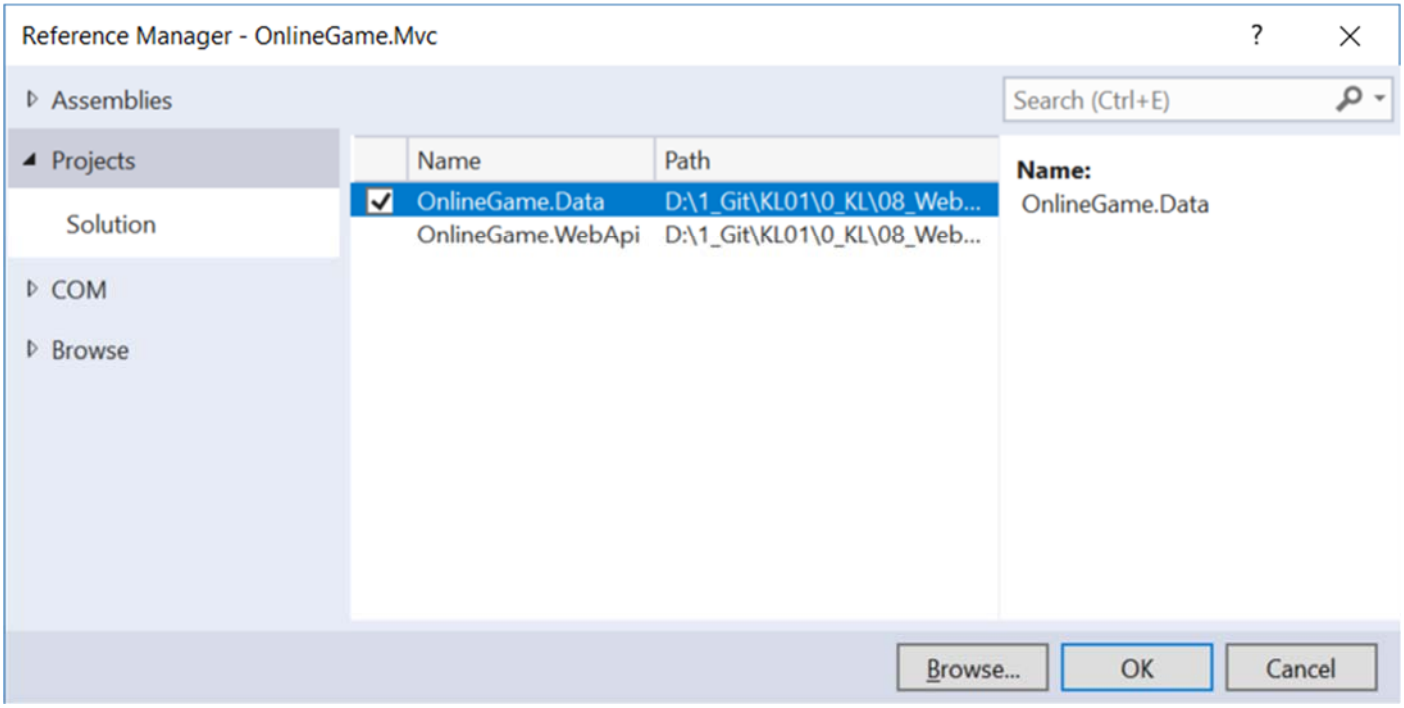
Solutions Name --> Add --> New Project -->

Visual C# --> Web --> ASP.NETWeb Application (.Net Framework)
-->

Name: **OnlineGame.Mvc**
--> Select "**MVC**" --> OK
--> Add Reference



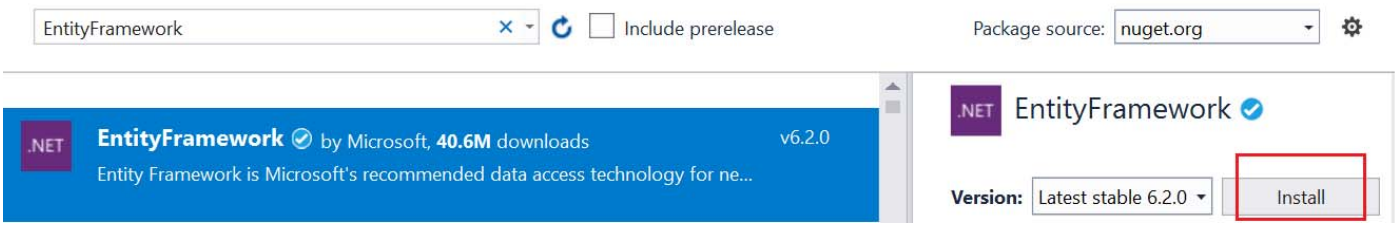




3. OnlineGame.Data

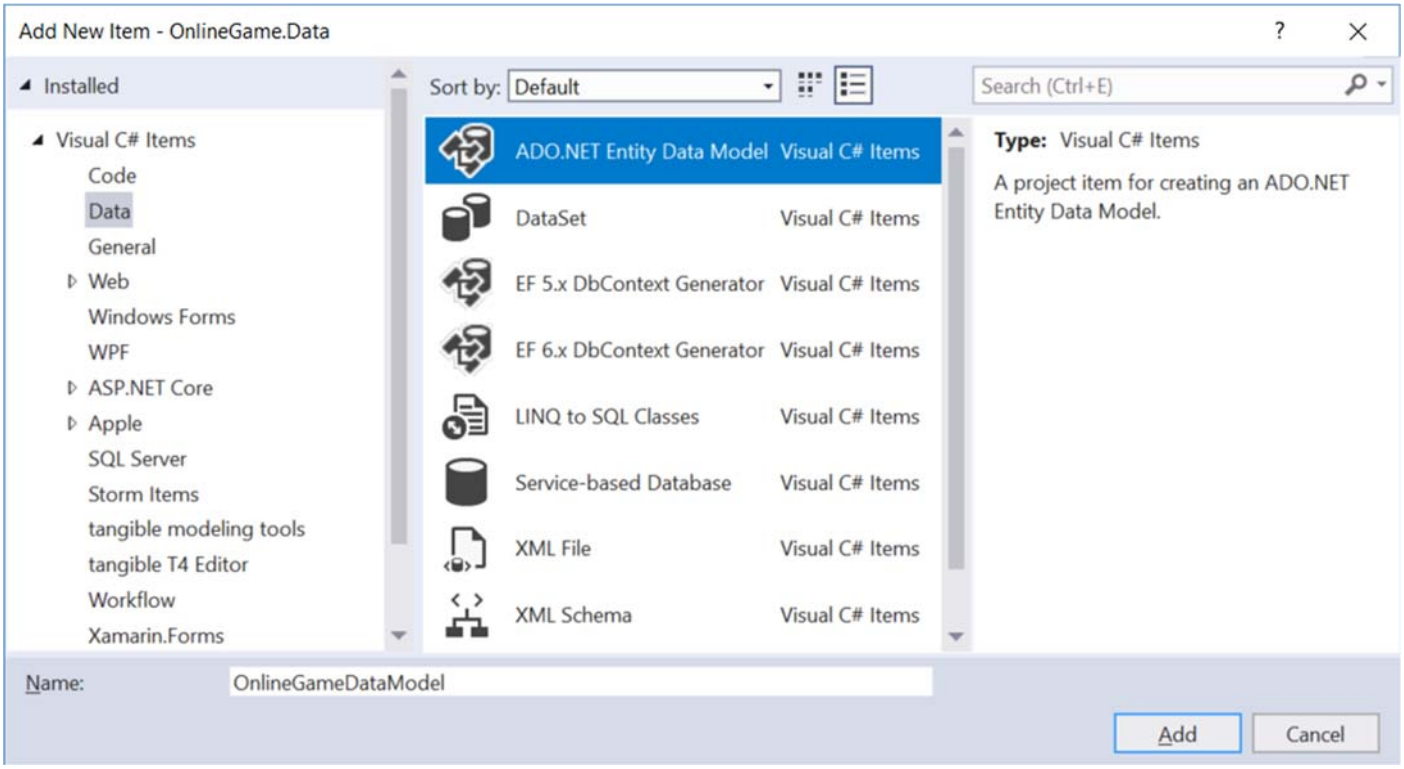
3.1. Install Entity Framework

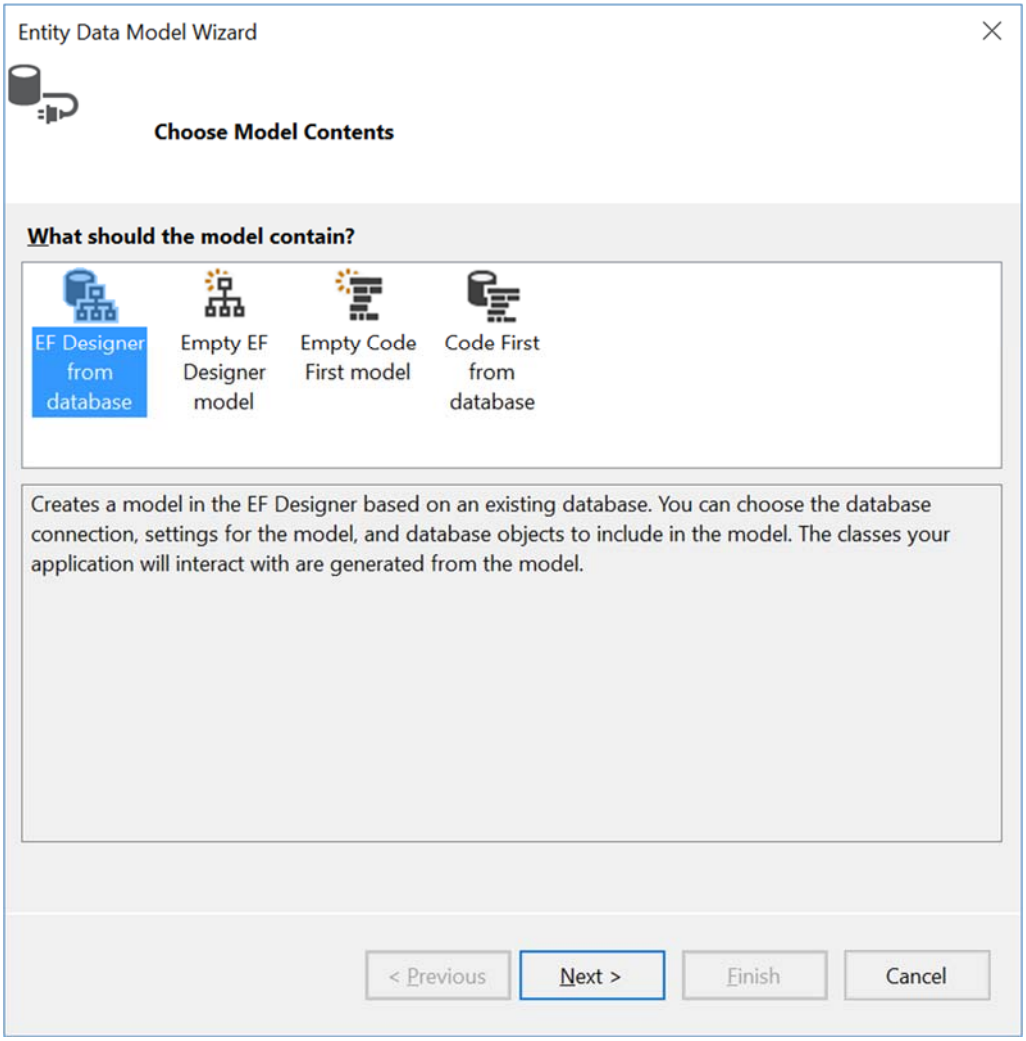
Tools --> NuGet Package Manager --> Manage NuGet Packages for Solutions...
--> Browse tab --> Search : **EntityFramework**
--> Install it

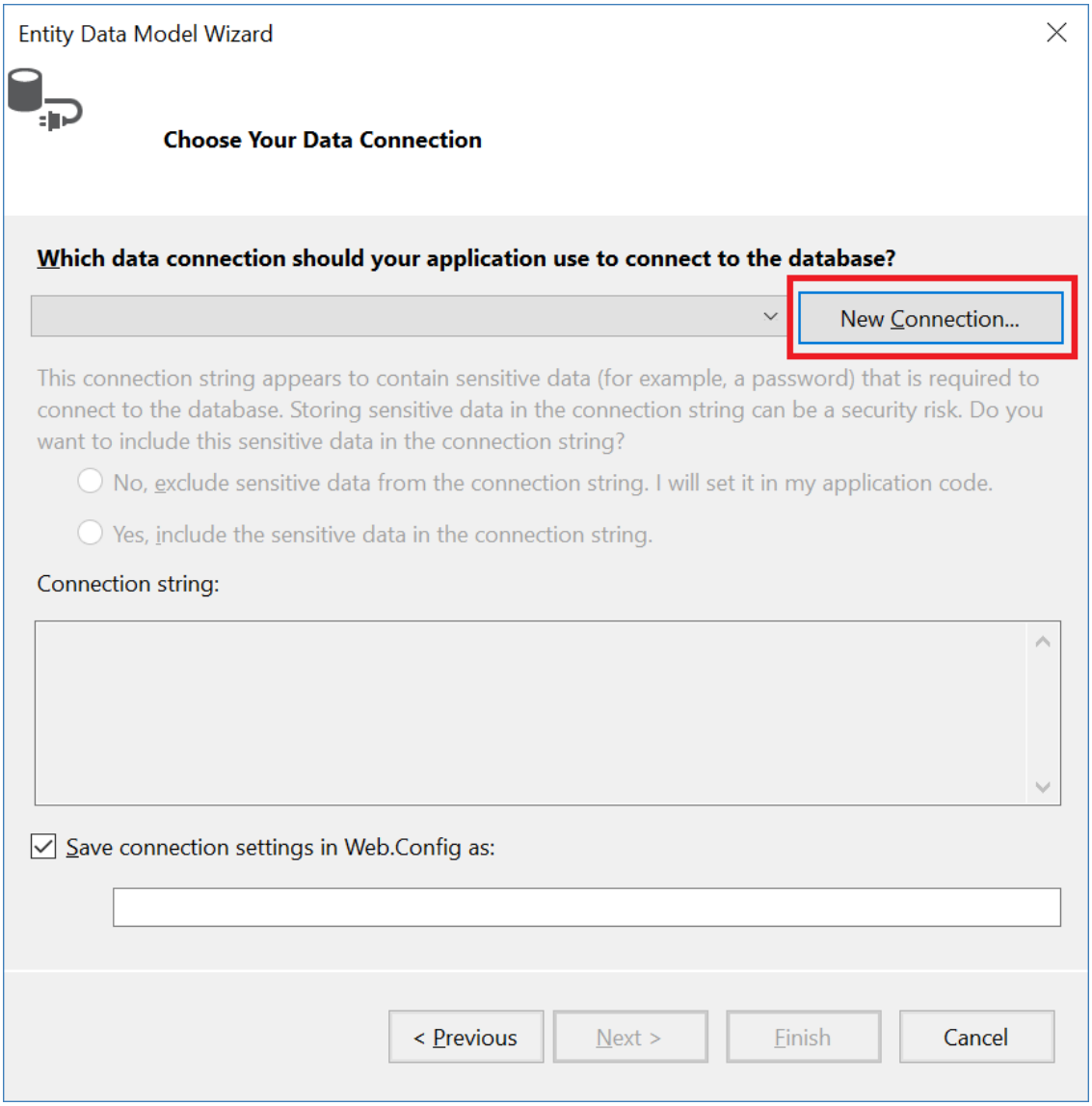


3.2. ADO.Net Entity Data Model - Entity Framework

In Visual Studio 2017
Project Name --> Right Click --> Add --> New Item
--> Visual C# --> Data --> ADO.Net Entity Data Model
Name:
OnlineGameDataModel
-->
EF Designer from database
....
-->
Save Connection settings in Web.Config as:
OnlineGameContext




Entity Data Model Wizard
Choose Model Contents
What should the model contain?
EF Designer from database
Empty EF Designer model
Empty Code First model
Code First from database
Creates a model in the EF Designer based on an existing database. You can choose the database connection, settings for the model, and database objects to include in the model. The classes your application will interact with are generated from the model.
< Previous
Next >
Finish
Cancel


Entity Data Model Wizard
Choose Your Data Connection
Which data connection should your application use to connect to the database?
New Connection...
This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?
No, exclude sensitive data from the connection string. I will set it in my application code.
Yes, include the sensitive data in the connection string.
Connection string:
Save connection settings in Web.Config as:
< Previous
Next >
Finish
Cancel

Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:

Microsoft SQL Server (SqlClient) | Change...

Server name:

N550JKL\SQL2016 | Refresh

Log on to the server

Authentication: SQL Server Authentication

User name: Tester2

Password: ••••

☑ Save my password

Microsoft Visual Studio

Test connection succeeded.

OK

Connect to a database

◉ Select or enter a database name:

OnineGame

○ Attach a database file:

Browse...

Advanced...

Test Connection | OK | Cancel

Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

n550jkl\sql2016.OnlineGame.dbo

New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

- ○ No, exclude sensitive data from the connection string. I will set it in my application code.
- ◉ Yes, include the sensitive data in the connection string.

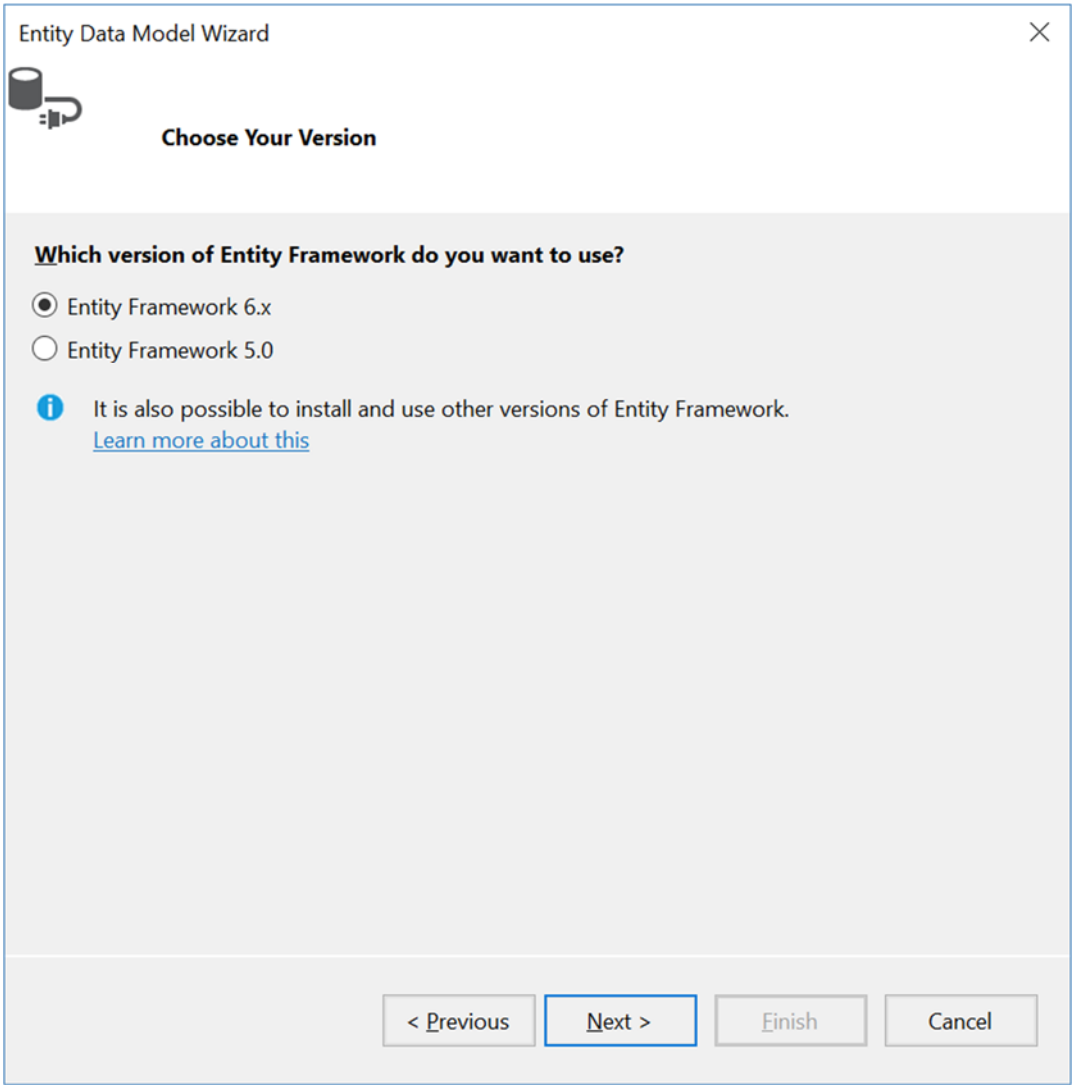
Connection string:

```
metadata=res://*/Models.OnlineGameDataModel.csdl|
res://*/Models.OnlineGameDataModel.ssdl|
res://*/Models.OnlineGameDataModel.msl;provider=System.Data.SqlClient;provider connection
string="data source=N550JKL\SQL2016;initial catalog=OnlineGame;persist security info=True;user
id=Tester;password=***********;MultipleActiveResultSets=True;App=EntityFramework"
```

☑ Save connection settings in Web.Config as:

OnlineGameContext

< Previous | Next > | Finish | Cancel


Entity Data Model Wizard
Choose Your Version
Which version of Entity Framework do you want to use?
Entity Framework 6.x
Entity Framework 5.0
It is also possible to install and use other versions of Entity Framework.
Learn more about this
< Previous
Next >
Finish
Cancel

Entity Data Model Wizard

Choose Your Database Objects and Settings

Which database objects do you want to include in your model?

- [x] Tables
  - [x] dbo
    - [x] Gamer
    - [x] GamerIdentity
  - [ ] Views
  - [ ] Stored Procedures and Functions

- [x] Pluralize or singularize generated object names
- [x] Include foreign key columns in the model
- [ ] Import selected stored procedures and functions into the entity model

Model Namespace:

OnineGameModel

< Previous | Next > | Finish | Cancel

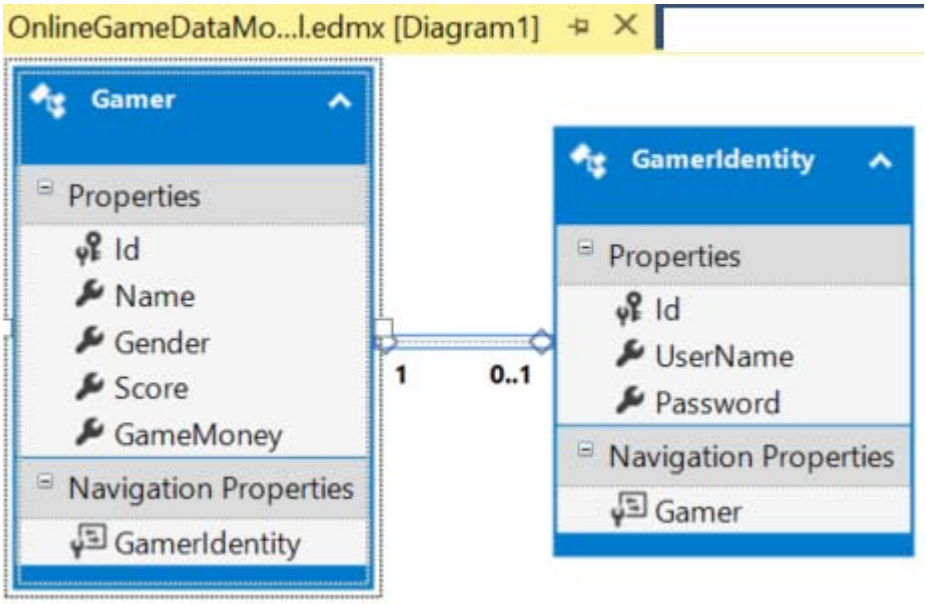
Security Warning

Running this text template can potentially harm your computer. Do not run it if you obtained it from an untrusted source.

Click OK to run the template.
Click Cancel to stop the process.

- [ ] Do not show this message again

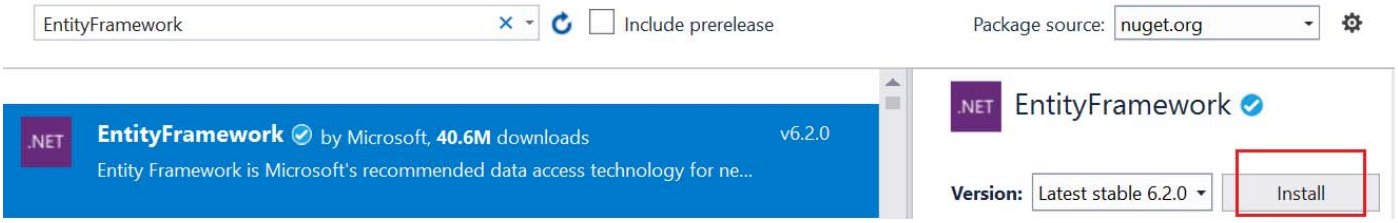
OK | Cancel



4. OnlineGame.WebApi

4.1. Install Entity Framework

Tools --> NuGet Package Manager --> Manage NuGet Packages for Solutions...
--> Browse tab --> Search : **EntityFramework**
--> Install it



4.2. OnlineGame.WebApi/Web.config : Add Connection String

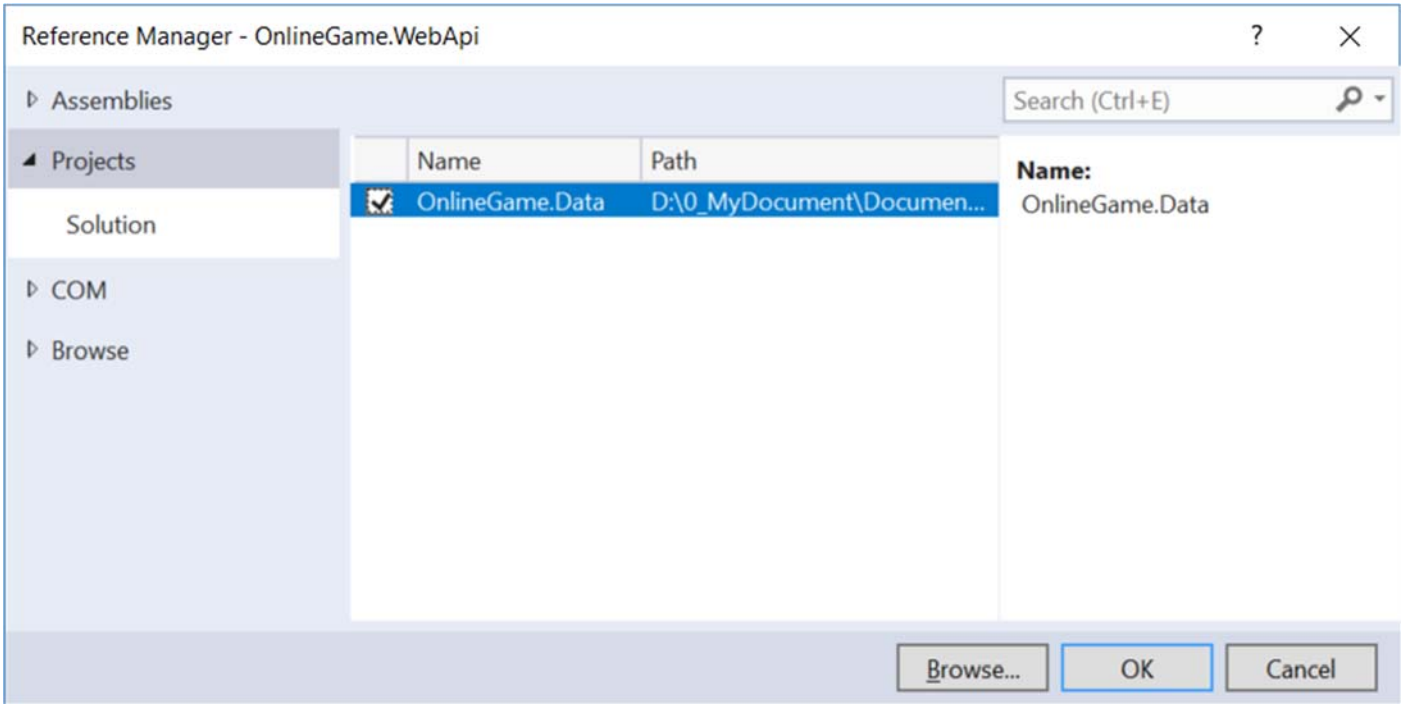
Web.config

```
69      </compilers>
70    </system.codedom>
71    <entityFramework>
72      <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">
73        <parameters>
74          <parameter value="mssqllocaldb" />
75        </parameters>
76      </defaultConnectionFactory>
77      <providers>
78        <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices,
           EntityFramework.SqlServer" />
79      </providers>
80    </entityFramework>
81    <connectionStrings>
82      <add name="OnlineGameContext" connectionString="metadata=res://*/OnlineGameDataModel.csdl|res://*/
       OnlineGameDataModel.ssdl|res://*/OnlineGameDataModel.msl;provider=System.Data.SqlClient;provider connection
       string=&quot;data source=N550JKL\SQL2016;initial catalog=OnineGame;persist security info=True;user
       id=Tester2;password=1234;MultipleActiveResultSets=True;App=EntityFramework&quot;"
       providerName="System.Data.EntityClient" />
83    </connectionStrings>
84  </configuration>
```

<connectionStrings>

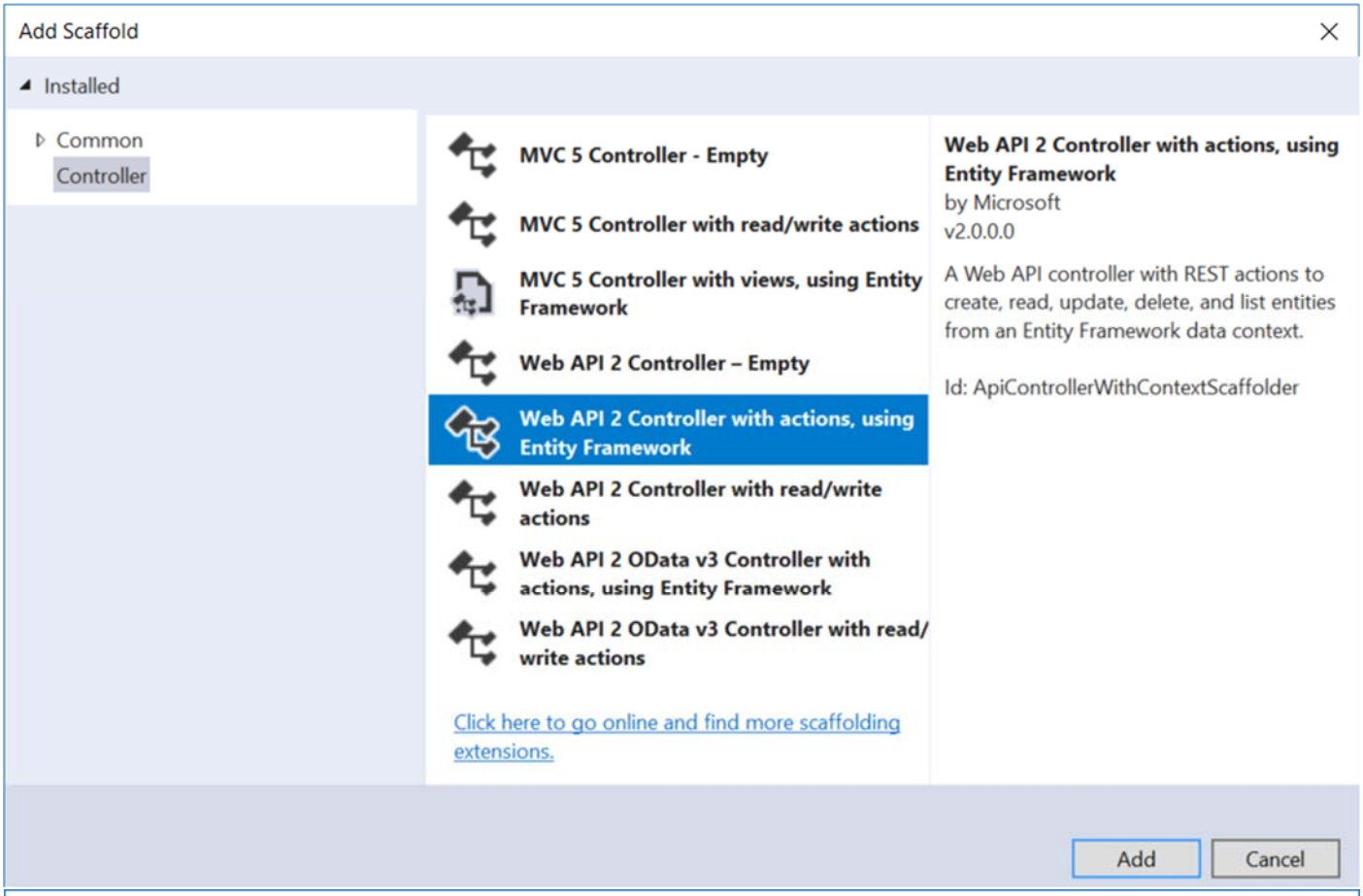
```
    <add name="OnlineGameContext" connectionString="metadata=res://*/OnlineGameDataModel.csdl|res://*/Onli
neGameDataModel.ssdl|res://*/OnlineGameDataModel.msl;provider=System.Data.SqlClient;provider connection
string=&quot;data source=N550JKL\SQL2016;initial catalog=OnineGame;persist security info=True;user
id=Tester2;password=1234;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Da
ta.EntityClient" />
  </connectionStrings>
```

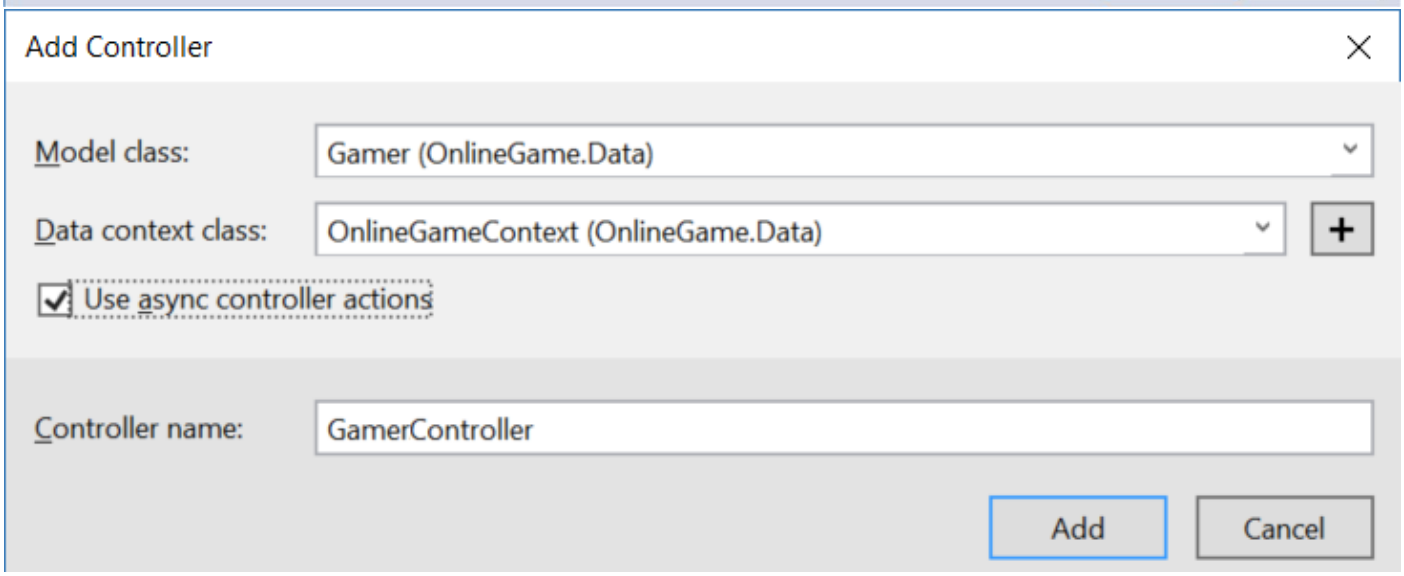
4.3. Add Reference



4.4. OnlineGame.WebApi/Controllers/Api/GamerController.cs (Bug)

Controllers/Api folder --> Right Click --> Add --> Controller
--> **Web API 2 Controller with actions, using Entity Framework**
--> **GamerController**
if you have any error message, please ensure re-build whole solutions.





http://localhost:55402/api/gamer

localhost:55402/api/gam

localhost:55402/api/gamer

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
▼<Error>
  <Message>An error has occurred.</Message>
  ▼<ExceptionMessage>
    The 'ObjectContent`1' type failed to serialize the response body for content type 'application/xml; charset=utf-8'.
  </ExceptionMessage>
  <ExceptionType>System.InvalidOperationException</ExceptionType>
  <StackTrace/>
  ▼<InnerException>
    <Message>An error has occurred.</Message>
    ▼<ExceptionMessage>
      Type 'System.Data.Entity.DynamicProxies.Gamer_49A750658E225C4BB9C50A0D5A93739D7629E2092FB8F7BBC49BE4D84D02B297' with
      name
      'Gamer_49A750658E225C4BB9C50A0D5A93739D7629E2092FB8F7BBC49BE4D84D02B297:http://schemas.datacontract.org/2004/07/Syst
      is not expected. Consider using a DataContractResolver if you are using DataContractSerializer or add any types not
      to the list of known types - for example, by using the KnownTypeAttribute attribute or by adding them to the list of
      passed to the serializer.
    </ExceptionMessage>
    ▼<ExceptionType>
      System.Runtime.Serialization.SerializationException
    </ExceptionType>
```


In the previous tutorial, when we have only one table, both XML formatter and JSON formatter work perfectly.
However, when we have 2 tables, xml formatter starts to give me some problems.
I have done some research.
One way to fix this issue is to remove XML formatter, and enforce to use JSON formatter.
JSON formatter is very popular in many API nowadays.
Reference:
https://forums.asp.net/t/1983286.aspx?Web+API+error+The+ObjectContent+1+type+failed+to+serialize+the+response+body+for+content+type+application+xml+charset+utf+8+
The second way is to remove formatters.
https://stackoverflow.com/questions/23098191/failed-to-serialize-the-response-in-web-api-with-json
I am not sure how it works, so I will use the first way.

4.5. OnlineGame.WebApi/App_Start/WebApiConfig.cs (Fix Bug)

```
using System.Net.Http.Formatting;
using System.Web.Http;
namespace OnlineGame.WebApi
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            // Web API configuration and services
            // Web API routes
            config.MapHttpAttributeRoutes();
            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
            //Use JSON formatter as a PreserveReferencesHandling.
            JsonMediaTypeFormatter json = config.Formatters.JsonFormatter;
            json.SerializerSettings.PreserveReferencesHandling =
Newtonsoft.Json.PreserveReferencesHandling.Objects;
            //Remove Xml Formatter
            config.Formatters.Remove(config.Formatters.XmlFormatter);
        }
    }
```

```
}
/*
//JsonMediaTypeFormatter json = config.Formatters.JsonFormatter;
//json.SerializerSettings.PreserveReferencesHandling =
Newtonsoft.Json.PreserveReferencesHandling.Objects;
//config.Formatters.Remove(config.Formatters.XmlFormatter);
Use JSON formatter as a PreserveReferencesHandling.
Remove Xml Formatter
Reference:
A.
https://forums.asp.net/t/1983286.aspx?Web+API+error+The+ObjectContent+1+type+failed+to+serialize+the+response+body+for+content+type+application+xml+charset+utf+8+
B.
https://stackoverflow.com/questions/23098191/failed-to-serialize-the-response-in-web-api-with-json
*/
```

http://localhost:55402/api/gamer


```
[{"$id":"1","GamerIdentity":{"$id":"2","Gamer":
{"$ref":"1"},"Id":1,"UserName":"One","Password":"1111"},"Id":1,"Name":"NameOne
ABC","Gender":"Male","Score":5000,"GameMoney":550},{"$id":"3","GamerIdentity":
{"$id":"4","Gamer":
{"$ref":"3"},"Id":2,"UserName":"Two","Password":"2222"},"Id":2,"Name":"NameTwo
ABCDE","Gender":"Female","Score":4500,"GameMoney":1200},
{"$id":"5","GamerIdentity":{"$id":"6","Gamer":
{"$ref":"5"},"Id":3,"UserName":"Three","Password":"3333"},"Id":3,"Name":"NameThree
EFGH","Gender":"Male","Score":6500,"GameMoney":3050},{"$id":"7","GamerIdentity":
{"$id":"8","Gamer":
{"$ref":"7"},"Id":4,"UserName":"Four","Password":"4444"},"Id":4,"Name":"NameFour
HIJKLMN","Gender":"Female","Score":45000,"GameMoney":450},
{"$id":"9","GamerIdentity":{"$id":"10","Gamer":
{"$ref":"9"},"Id":5,"UserName":"Five","Password":"5555"},"Id":5,"Name":"NameFive
NOP","Gender":"Male","Score":3000,"GameMoney":200},{"$id":"11","GamerIdentity":
{"$id":"12","Gamer":
{"$ref":"11"},"Id":6,"UserName":"Six","Password":"6666"},"Id":6,"Name":"NameSix
PQRSTUVW","Gender":"Male","Score":4000,"GameMoney":700},
{"$id":"13","GamerIdentity":{"$id":"14","Gamer":
{"$ref":"13"},"Id":7,"UserName":"Seven","Password":"7777"},"Id":7,"Name":"NameSeve
n XYZ","Gender":"Male","Score":450,"GameMoney":1500}]
```


5. OnlineGame.WebApi - WebApi Cors (Cross Origin Resource Sharing)

5.1. WebApi Cors (Cross Origin Resource Sharing) allows Jquery AJAX may call Web API in the different origins

Reference:
https://docs.microsoft.com/en-us/aspnet/web-api/overview/security/enabling-cross-origin-requests-in-web-api
https://www.nuget.org/packages/Microsoft.AspNet.WebApi.Cors/
For security reason, web browsers do not allow Jquery AJAX call Web API in the different origin.
There are 2 popular ways to fix it.
1.
JSONP (JSON with Padding) will wrap the JSON data in a function

Install-Package WebApiContrib.Formatting.Jsonp
E.g.1.1. JSON
```
{
   "Name":"KL",
    "Gender":"Male"
}
```
E.g.1.2. JSONP
```
CallbackFunction({
   "Name":"KL",
    "Gender":"Male"
})
```

2.
Enable CORS (Cross Origin Resource Sharing)
Install-Package Microsoft.AspNet.WebApi.Cors
The following examples have the **same origin**.
http://localhost:1234/api/gamer
http://localhost:1234/gamer/Index2

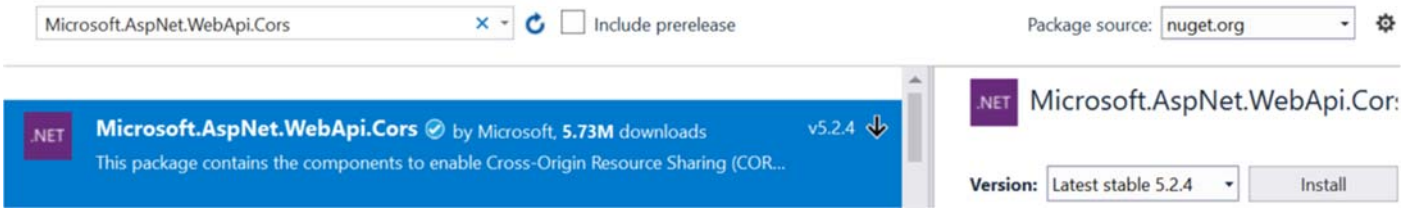
The following examples have **different port** numbers, so they are **different origins**.
http://localhost:**1234**/api/gamer
http://localhost:**4321**/gamer/Index2

The following examples have **different domains**, so they are **different origins**.
http://AAAA.**com**/api/gamer
http://AAAA.**net**/gamer/Index2

The following examples have **different schemes**, so they are **different origins**.
https://AAAA.com/api/gamer
http://AAAA.com/gamer/Index2

5.2. Install NuGet Package

Install-Package Microsoft.AspNet.WebApi.Cors
Tools --> NuGet Package Manager --> Manage NuGet Packages for Solutions...
--> Browse tab --> Search : **Microsoft.AspNet.WebApi.Cors**
--> Install it



5.3. OnlineGame.WebApi/App_Start/WebApiConfig.cs

```
using System.Net.Http.Formatting;
using System.Web.Http;
using System.Web.Http.Cors;
namespace OnlineGame.WebApi
{
```

```
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            // Web API configuration and services
            // Web API routes
            config.MapHttpAttributeRoutes();
            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
            //----------------------------
            ////1.
            ////JSONP allows Jquery AJAX may call Web API in the different origins
            ////Create a new JSON media type formatter,
            ////and insert it into first position of HttpConfiguration formatter.
            ////It will allow you to use JSONP formatter which
            ////can wrap the JSON data in a function
            //JsonpMediaTypeFormatter jsonpFormatter =
            //    new JsonpMediaTypeFormatter(config.Formatters.JsonFormatter);
            //config.Formatters.Insert(0, jsonpFormatter);
            //2.
            //WebApi Cors(Cross Origin Resource Sharing)
            //allows Jquery AJAX may call Web API in the different origins
            ////2.1.
            //config.EnableCors();
            //2.2.
            //EnableCorsAttribute(origins, headers, methods)
            //It allows the resource to be accessed by all origins,
            //and it accepts any request header ("accept,content-type,origin...etc"),
            //and it accepts all methods ("GET,POST...etc")
            EnableCorsAttribute cors = new EnableCorsAttribute("*", "*", "*");
            config.EnableCors(cors);

            //----------------------------
            //6.
            //Use JSON formatter as a PreserveReferencesHandling.
            JsonMediaTypeFormatter json = config.Formatters.JsonFormatter;
            json.SerializerSettings.PreserveReferencesHandling =
Newtonsoft.Json.PreserveReferencesHandling.Objects;
            //Remove Xml Formatter
            config.Formatters.Remove(config.Formatters.XmlFormatter);
        }
    }
}
/*
1.
JSONP allows Jquery AJAX may call Web API in the different origins
//JsonpMediaTypeFormatter jsonpFormatter =
//    new JsonpMediaTypeFormatter(config.Formatters.JsonFormatter);
//config.Formatters.Insert(0, jsonpFormatter);
Create a new JSON media type formatter,
and insert it into first position of HttpConfiguration formatter.
It will allow you to use JSONP formatter which
can wrap the JSON data in a function
E.g.1.1. JSON
{
```

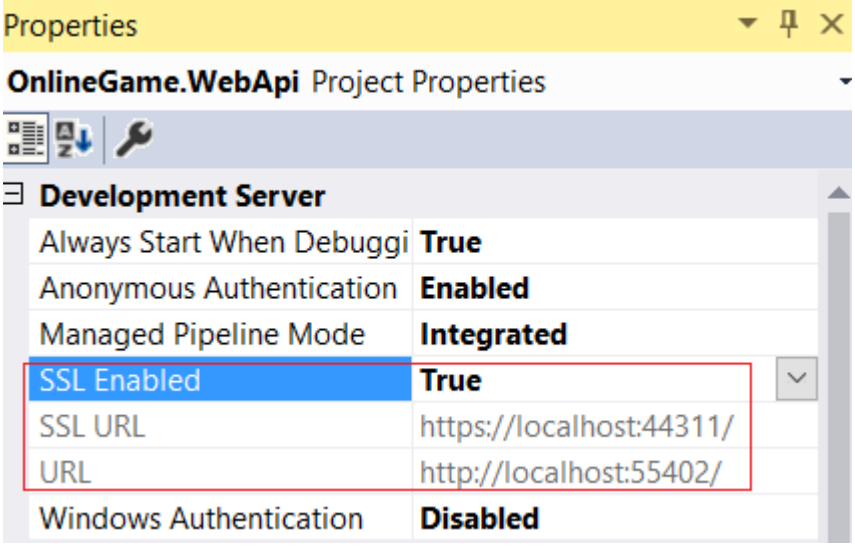
```
    "Name":"KL",
     "Gender":"Male"
}
E.g.1.2. JSONP
CallbackFunction({
    "Name":"KL",
     "Gender":"Male"
})
-------------------------------------
3.
WebApi Cors (Cross Origin Resource Sharing)
allows Jquery AJAX may call Web API in the different origins
------------------------------
3.1.
new EnableCorsAttribute(origins, headers, methods)
//EnableCorsAttribute cors = new EnableCorsAttribute("*", "*", "*");
//config.EnableCors(cors);
It allows the resource to be accessed by all origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")
----------------
3.1.1.
origins:
It is a Comma-separated whitelist which are allowed to access the web api by Ajax call.
E.g.3.1.1.1.
"http://localhost:49804,https://ithandyguytutorial.blogspot.com.au"
That means only http://localhost:49804 and https://ithandyguytutorial.blogspot.com.au
can access the web api by Ajax call.
E.g.3.1.1.2.
"*"
It means allows all origins to access the web api by Ajax call.
----------------
3.1.2.
headers:
It is a Comma-separated whitelist of request headers which are supported by the resource.
E.g.3.1.2.1.
"accept,content-type,origin" means only these 3 things can be used in request header.
E.g.3.1.2.2.
"*"
It means allows all request headers to the web api by Ajax call.
----------------
3.1.3.
methods:
It is a Comma-separated whitelist of methods which are supported by the resource.
E.g.3.1.3.1.
"GET,POST" means only these 2 methods can be used in request.
E.g.3.1.3.2.
"*"
It means allows all request methods to the web api by Ajax call.
------------------------------
3.2.
In OnlineGame.WebApi/App_Start/WebApiConfig.cs
//config.EnableCors();
In OnlineGame.WebApi/Controllers/Api/GamerController.cs
////[EnableCors("*", "*", "*")]
////[EnableCors("https://ithandyguytutorial.blogspot.com.au", "*", "*")]
//[EnableCors("http://localhost:49804", "*", "*")]
//public class GamerController : ApiController
...
//[DisableCors]
//[HttpGet]
//public async Task<IHttpActionResult> LoadGamers(string gender = "")
3.2.1.
If you don't want to enable Cors globally,
then you may enable Cors in api controller level or method level.
When you enable Cors, in api controller level,
//[EnableCors("*", "*", "*")]
```

```
it will apply to all methods in that controller.
If you want to exclude any method, then you may use
//[DisableCors]
3.2.2.
3.2.2.1.
//[EnableCors("*", "*", "*")]
EnableCorsAttribute(origins, headers, methods)
It allows the resource to be accessed by all origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")
3.2.2.2.
//[EnableCors("https://ithandyguytutorial.blogspot.com.au", "*", "*")]
EnableCorsAttribute(origins, headers, methods)
It allows the resource to be accessed by https://ithandyguytutorial.blogspot.com.au origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")
3.2.2.3.
//[EnableCors("http://localhost:49804", "*", "*")]
It allows the resource to be accessed by http://localhost:49804 origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")
-----------------------------------------
6.
//JsonMediaTypeFormatter json = config.Formatters.JsonFormatter;
//json.SerializerSettings.PreserveReferencesHandling =
Newtonsoft.Json.PreserveReferencesHandling.Objects;
//config.Formatters.Remove(config.Formatters.XmlFormatter);
Use JSON formatter as a PreserveReferencesHandling.
Remove Xml Formatter
Reference:
A.
https://forums.asp.net/t/1983286.aspx?Web+API+error+The+ObjectContent+1+type+failed+to+serialize+the+response+body+for+content+type+application+xml+charset+utf+8+
B.
https://stackoverflow.com/questions/23098191/failed-to-serialize-the-response-in-web-api-with-json
*/
```

6.OnlineGame.WebApi - Enable SSL (Secure Sockets Layer) and Create self-signed certificate

6.1. OnlineGame.WebApi Enable SSL via Visual Studio 2017

In the Visual Studio "Solution Explorer" windows
--> Select API Project
--> Go to Properties window
--> SSL Enabled: **true**



Set the API project as the start up project

Run the project

-->

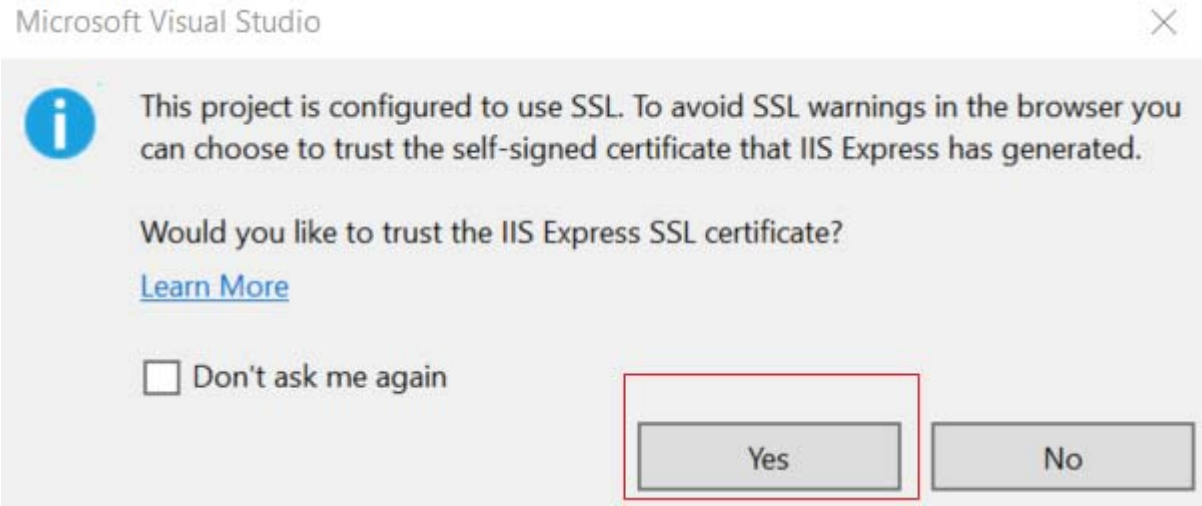
Would you like to trust the IIS Express SSL certificate.

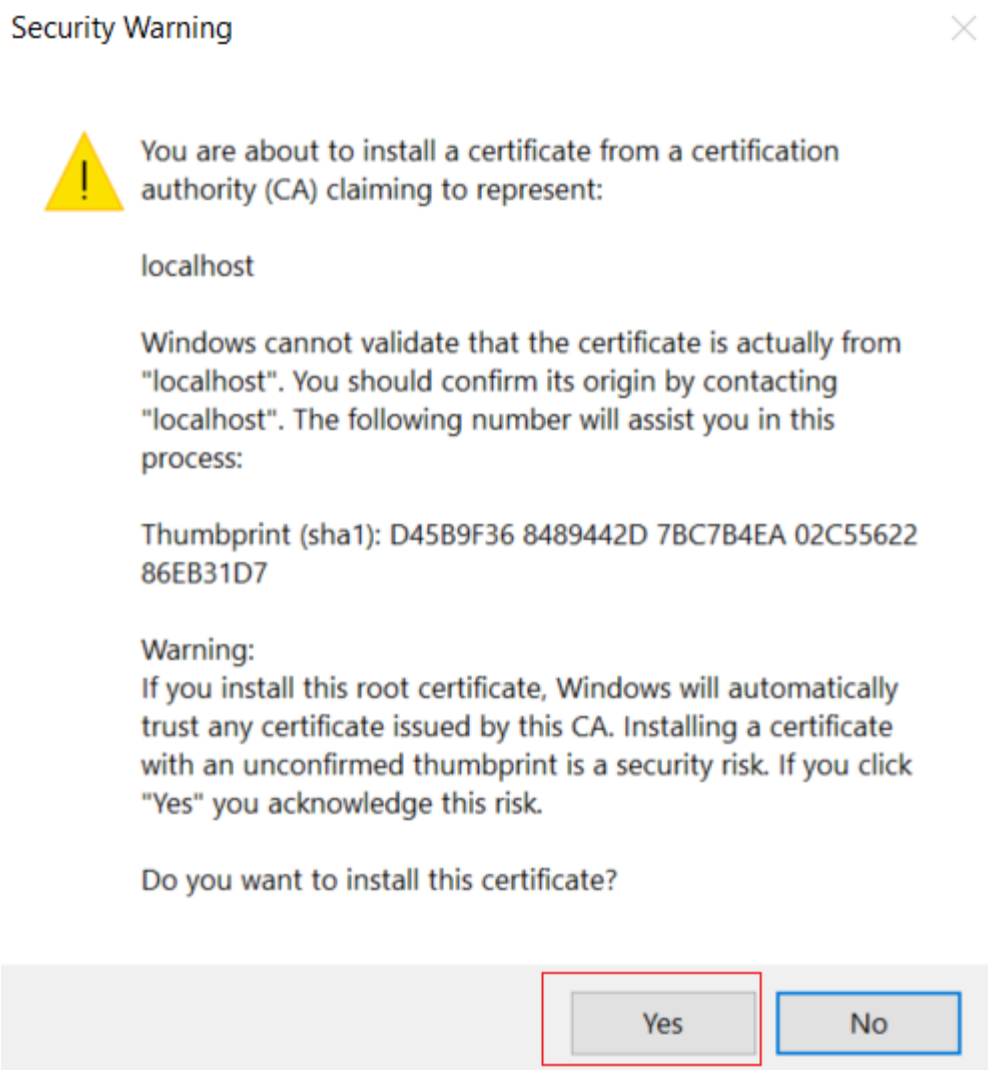
Yes

-->

Do you want to install this certificate?

Yes





6.2. OnlineGame.WebApi/WebShared/HttpsAuthorizationFilterAttribute.cs

```
using System;
using System.Net;
using System.Net.Http;
using System.Text;
using System.Web.Http.Controllers;
using System.Web.Http.Filters;
namespace OnlineGame.WebApi.WebShared
{
    public class HttpsAuthorizationFilterAttribute : AuthorizationFilterAttribute
    {
        public override void OnAuthorization(HttpActionContext actionContext)
        {
            //If the request is not HTTPS request.
            if (actionContext.Request.RequestUri.Scheme != Uri.UriSchemeHttps)
            {
                //If the resourece is found, then create a response with HttpStatusCode.Found/302.
                actionContext.Response = actionContext.Request
                    .CreateResponse(HttpStatusCode.Found);
                //Create a response content that encoding is UTF8 and mediaType is html.
                actionContext.Response.Content = new StringContent
                    ("<p>HTTPS is required.</p>", Encoding.UTF8, "text/html");
                //Create a new URI by current requested URI.
```

```
                //The new URI will redirect HTTPS. 44365 is the SSL URL port.
                UriBuilder uriBuilder = new UriBuilder(actionContext.Request.RequestUri)
                {
                    Scheme = Uri.UriSchemeHttps,
                    Port = 44311    //*************Change to your port
                };
                //Set the Response.Headers.Location to new URI,
                //It will redirect to new URI that is HTTPS URI
                actionContext.Response.Headers.Location = uriBuilder.Uri;
            }
            else
            {
                //If the request is the HTTPS request,
                //then do what it supposed to do.
                base.OnAuthorization(actionContext);
            }
        }
    }
}
```

6.3. OnlineGame.WebApi/App_Start/WebApiConfig.cs

In this simple tutorial, we don't use HTTPS and SSL, to keep it simple.

```
using System.Net.Http.Formatting;
using System.Web.Http;
using System.Web.Http.Cors;
namespace OnlineGame.WebApi
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            // Web API configuration and services
            // Web API routes
            config.MapHttpAttributeRoutes();
            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
            //----------------------------
            ////1.
            ////JSONP allows Jquery AJAX may call Web API in the different origins
            ////Create a new JSON media type formatter,
            ////and insert it into first position of HttpConfiguration formatter.
            ////It will allow you to use JSONP formatter which
            ////can wrap the JSON data in a function
            //JsonpMediaTypeFormatter jsonpFormatter =
            //    new JsonpMediaTypeFormatter(config.Formatters.JsonFormatter);
            //config.Formatters.Insert(0, jsonpFormatter);
            //2.
            //WebApi Cors(Cross Origin Resource Sharing)
            //allows Jquery AJAX may call Web API in the different origins
            ////2.1.
            //config.EnableCors();
```

```
            //2.2.
            //EnableCorsAttribute(origins, headers, methods)
            //It allows the resource to be accessed by all origins,
            //and it accepts any request header ("accept,content-type,origin...etc"),
            //and it accepts all methods ("GET,POST...etc")
            EnableCorsAttribute cors = new EnableCorsAttribute("*", "*", "*");
            config.EnableCors(cors);

            //----------------------------
            ////4.
            ////HTTP request will redirect to HTTPS request
            //config.Filters.Add(new HttpsAuthorizationFilterAttribute());

            //----------------------------
            //6.
            //Use JSON formatter as a PreserveReferencesHandling.
            JsonMediaTypeFormatter json = config.Formatters.JsonFormatter;
            json.SerializerSettings.PreserveReferencesHandling =
Newtonsoft.Json.PreserveReferencesHandling.Objects;
            //Remove Xml Formatter
            config.Formatters.Remove(config.Formatters.XmlFormatter);
        }
    }
}

/*
1.
JSONP allows Jquery AJAX may call Web API in the different origins
//JsonpMediaTypeFormatter jsonpFormatter =
//    new JsonpMediaTypeFormatter(config.Formatters.JsonFormatter);
//config.Formatters.Insert(0, jsonpFormatter);
Create a new JSON media type formatter,
and insert it into first position of HttpConfiguration formatter.
It will allow you to use JSONP formatter which
can wrap the JSON data in a function
E.g.1.1. JSON
{
    "Name":"KL",
     "Gender":"Male"
}
E.g.1.2. JSONP
CallbackFunction({
    "Name":"KL",
     "Gender":"Male"
})
-------------------------------------
3.
WebApi Cors (Cross Origin Resource Sharing)
allows Jquery AJAX may call Web API in the different origins
------------------------------
3.1.
new EnableCorsAttribute(origins, headers, methods)
//EnableCorsAttribute cors = new EnableCorsAttribute("*", "*", "*");
//config.EnableCors(cors);
It allows the resource to be accessed by all origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")
----------------
3.1.1.
origins:
It is a Comma-separated whitelist which are allowed to access the web api by Ajax call.
E.g.3.1.1.1.
"http://localhost:49804,https://ithandyguytutorial.blogspot.com.au"
```

That means only http://localhost:49804 and https://ithandyguytutorial.blogspot.com.au
can access the web api by Ajax call.
E.g.3.1.1.2.
"*"
It means allows all origins to access the web api by Ajax call.

3.1.2.
headers:
It is a Comma-separated whitelist of request headers which are supported by the resource.
E.g.3.1.2.1.
"accept,content-type,origin" means only these 3 things can be used in request header.
E.g.3.1.2.2.
"*"
It means allows all request headers to the web api by Ajax call.

3.1.3.
methods:
It is a Comma-separated whitelist of methods which are supported by the resource.
E.g.3.1.3.1.
"GET,POST" means only these 2 methods can be used in request.
E.g.3.1.3.2.
"*"
It means allows all request methods to the web api by Ajax call.

3.2.
In OnlineGame.WebApi/App_Start/WebApiConfig.cs
//config.EnableCors();
In OnlineGame.WebApi/Controllers/Api/GamerController.cs
////[EnableCors("*", "*", "*")]
////[EnableCors("https://ithandyguytutorial.blogspot.com.au", "*", "*")]
//[EnableCors("http://localhost:49804", "*", "*")]
//public class GamerController : ApiController
...
//[DisableCors]
//[HttpGet]
//public async Task<IHttpActionResult> LoadGamers(string gender = "")
3.2.1.
If you don't want to enable Cors globally,
then you may enable Cors in api controller level or method level.
When you enable Cors, in api controller level,
//[EnableCors("*", "*", "*")]
it will apply to all methods in that controller.
If you want to exclude any method, then you may use
//[DisableCors]
3.2.2.
3.2.2.1.
//[EnableCors("*", "*", "*")]
EnableCorsAttribute(origins, headers, methods)
It allows the resource to be accessed by all origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")
3.2.2.2.
//[EnableCors("https://ithandyguytutorial.blogspot.com.au", "*", "*")]
EnableCorsAttribute(origins, headers, methods)
It allows the resource to be accessed by https://ithandyguytutorial.blogspot.com.au origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")
3.2.2.3.
//[EnableCors("http://localhost:49804", "*", "*")]
It allows the resource to be accessed by http://localhost:49804 origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")

4.
HTTP redirect to HTTPS
4.1.

```
In OnlineGame.WebApi/App_Start/WebApiConfig.cs
//config.Filters.Add(new HttpsAuthorizationFilterAttribute());
If you add the attribute in WebApiConfig.cs,
it will apply to the entire application.
4.2.
If you don't want to apply to the entire application,
You may apply the attribute at controller level or action level.
E.g.
//[HttpsAuthorizationFilter]
//public class GamerController : ApiController
...
//[HttpsAuthorizationFilter]
//public async Task<IHttpActionResult> GetGamers(string gender = "all")

------------------------------------
6.
//JsonMediaTypeFormatter json = config.Formatters.JsonFormatter;
//json.SerializerSettings.PreserveReferencesHandling =
Newtonsoft.Json.PreserveReferencesHandling.Objects;
//config.Formatters.Remove(config.Formatters.XmlFormatter);
Use JSON formatter as a PreserveReferencesHandling.
Remove Xml Formatter
Reference:
A.
https://forums.asp.net/t/1983286.aspx?Web+API+error+The+ObjectContent+1+type+failed+to+serialize+the+response+body+for+content+type+application+xml+charset+utf+8+
B.
https://stackoverflow.com/questions/23098191/failed-to-serialize-the-response-in-web-api-with-json
*/
```

7.OnlineGame.WebApi - Basic Authentication

7.1. OnlineGame.WebApi/Account/Authentication.cs

```
using System;
using System.Linq;
using OnlineGame.Data;
namespace OnlineGame.WebApi.Account
{
    public class Authentication
    {
        public static bool IsAuthentic(string username, string password)
        {
            using (var db = new OnlineGameContext())
            {
                return db.GamerIdentities.Any(user =>
                        user.UserName.Equals(username, StringComparison.OrdinalIgnoreCase)
                                        && user.Password == password);
            }
        }
    }
}
```

7.2. OnlineGame.WebApi/Account/BasicAuthorizationFilterAttribute.cs

```
using System;
using System.Net;
```

```
using System.Net.Http;
using System.Security.Principal;
using System.Text;
using System.Threading;
using System.Web.Http.Controllers;
using System.Web.Http.Filters;
namespace OnlineGame.WebApi.Account
{
    public class BasicAuthorizationFilterAttribute : AuthorizationFilterAttribute
    {
        public override void OnAuthorization(HttpActionContext actionContext)
        {
            //if there is no userName and password parameter from
actionContext.Request.Headers.Authorization,
            //then response Unauthorized/401
            if (actionContext.Request.Headers.Authorization == null)
            {
                actionContext.Response = actionContext.Request
                    .CreateResponse(HttpStatusCode.Unauthorized);
            }
            else
            {
                //if there is a parameter from actionContext.Request.Headers.Authorization.
                //the Authorization.parameter is the token Base 64 String
                //which includes user name and password and they are separate by colon(:)
                //E.g. "username:password"
                string authToken =
                    actionContext.Request.Headers.Authorization.Parameter;
                //convert the string authToken from Base64String to UTF8 string.
                string decodedAuthToken = Encoding.UTF8.GetString(
                    Convert.FromBase64String(authToken));
                string[] usernamePasswordArray = decodedAuthToken.Split(':');
                string username = usernamePasswordArray[0];
                string password = usernamePasswordArray[1];
                //if the username and password is correct, then create a GenericPrincipal.
                if (Authentication.IsAuthentic(username, password))
                {
                    //GenericPrincipal has 2 parameters.
                    //The first parameter is a user IIdentity, in this case, username.
                    //The second parameter is string[] roles, in this case, null.
                    Thread.CurrentPrincipal = new GenericPrincipal(
                        new GenericIdentity(username), null);
                }
                else
                {
                    //if the username and password is not correct
                    //then response Unauthorized/401
                    actionContext.Response = actionContext.Request
                        .CreateResponse(HttpStatusCode.Unauthorized);
                }
            }
        }
    }
}
```

7.3. OnlineGame.WebApi/App_Start/WebApiConfig.cs

In this simple tutorial, we don't use BasicAuthorizationFilterAttribute in the entire application

```
using System.Net.Http.Formatting;
using System.Web.Http;
using System.Web.Http.Cors;
namespace OnlineGame.WebApi
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            // Web API configuration and services
            // Web API routes
            config.MapHttpAttributeRoutes();
            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
            //-----------------------------
            ////1.
            ////JSONP allows Jquery AJAX may call Web API in the different origins
            ////Create a new JSON media type formatter,
            ////and insert it into first position of HttpConfiguration formatter.
            ////It will allow you to use JSONP formatter which
            ////can wrap the JSON data in a function
            //JsonpMediaTypeFormatter jsonpFormatter =
            //    new JsonpMediaTypeFormatter(config.Formatters.JsonFormatter);
            //config.Formatters.Insert(0, jsonpFormatter);
            //2.
            //WebApi Cors(Cross Origin Resource Sharing)
            //allows Jquery AJAX may call Web API in the different origins
            ////2.1.
            //config.EnableCors();
            //2.2.
            //EnableCorsAttribute(origins, headers, methods)
            //It allows the resource to be accessed by all origins,
            //and it accepts any request header ("accept,content-type,origin...etc"),
            //and it accepts all methods ("GET,POST...etc")
            EnableCorsAttribute cors = new EnableCorsAttribute("*", "*", "*");
            config.EnableCors(cors);

            //-----------------------------
            ////4.
            ////HTTP request will redirect to HTTPS request
            //config.Filters.Add(new HttpsAuthorizationFilterAttribute());

            ////-----------------------------
            ////5.
            ////Use BasicAuthorizationFilterAttribute.
            ////It is for understanding basic concept, not for real world practice.
            //config.Filters.Add(new BasicAuthorizationFilterAttribute());

            //-----------------------------
```

```
            //6.
            //Use JSON formatter as a PreserveReferencesHandling.
             JsonMediaTypeFormatter json = config.Formatters.JsonFormatter;
             json.SerializerSettings.PreserveReferencesHandling =
Newtonsoft.Json.PreserveReferencesHandling.Objects;
            //Remove Xml Formatter
             config.Formatters.Remove(config.Formatters.XmlFormatter);
        }
    }
}

/*
1.
JSONP allows Jquery AJAX may call Web API in the different origins
//JsonpMediaTypeFormatter jsonpFormatter =
//    new JsonpMediaTypeFormatter(config.Formatters.JsonFormatter);
//config.Formatters.Insert(0, jsonpFormatter);
Create a new JSON media type formatter,
and insert it into first position of HttpConfiguration formatter.
It will allow you to use JSONP formatter which
can wrap the JSON data in a function
E.g.1.1. JSON
{
    "Name":"KL",
     "Gender":"Male"
}
E.g.1.2. JSONP
CallbackFunction({
    "Name":"KL",
     "Gender":"Male"
})
-------------------------------------
3.
WebApi Cors (Cross Origin Resource Sharing)
allows Jquery AJAX may call Web API in the different origins
------------------------------
3.1.
new EnableCorsAttribute(origins, headers, methods)
//EnableCorsAttribute cors = new EnableCorsAttribute("*", "*", "*");
//config.EnableCors(cors);
It allows the resource to be accessed by all origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")
----------------
3.1.1.
origins:
It is a Comma-separated whitelist which are allowed to access the web api by Ajax call.
E.g.3.1.1.1.
"http://localhost:49804,https://ithandyguytutorial.blogspot.com.au"
That means only http://localhost:49804 and https://ithandyguytutorial.blogspot.com.au
can access the web api by Ajax call.
E.g.3.1.1.2.
"*"
It means allows all origins to access the web api by Ajax call.
----------------
3.1.2.
headers:
It is a Comma-separated whitelist of request headers which are supported by the resource.
E.g.3.1.2.1.
"accept,content-type,origin" means only these 3 things can be used in request header.
E.g.3.1.2.2.
"*"
It means allows all request headers to the web api by Ajax call.
----------------
3.1.3.
methods:
```

It is a Comma-separated whitelist of methods which are supported by the resource.
E.g.3.1.3.1.
"GET,POST" means only these 2 methods can be used in request.
E.g.3.1.3.2.
"*"
It means allows all request methods to the web api by Ajax call.

3.2.
In OnlineGame.WebApi/App_Start/WebApiConfig.cs
//config.EnableCors();
In OnlineGame.WebApi/Controllers/Api/GamerController.cs
////[EnableCors("*", "*", "*")]
////[EnableCors("https://ithandyguytutorial.blogspot.com.au", "*", "*")]
//[EnableCors("http://localhost:49804", "*", "*")]
//public class GamerController : ApiController
...
//[DisableCors]
//[HttpGet]
//public async Task<IHttpActionResult> LoadGamers(string gender = "")
3.2.1.
If you don't want to enable Cors globally,
then you may enable Cors in api controller level or method level.
When you enable Cors, in api controller level,
//[EnableCors("*", "*", "*")]
it will apply to all methods in that controller.
If you want to exclude any method, then you may use
//[DisableCors]
3.2.2.
3.2.2.1.
//[EnableCors("*", "*", "*")]
EnableCorsAttribute(origins, headers, methods)
It allows the resource to be accessed by all origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")
3.2.2.2.
//[EnableCors("https://ithandyguytutorial.blogspot.com.au", "*", "*")]
EnableCorsAttribute(origins, headers, methods)
It allows the resource to be accessed by https://ithandyguytutorial.blogspot.com.au origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")
3.2.2.3.
//[EnableCors("http://localhost:49804", "*", "*")]
It allows the resource to be accessed by http://localhost:49804 origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")

4.
HTTP redirect to HTTPS
4.1.
In OnlineGame.WebApi/App_Start/WebApiConfig.cs
//config.Filters.Add(new HttpsAuthorizationFilterAttribute());
If you add the attribute in WebApiConfig.cs,
it will apply to the entire application.
4.2.
If you don't want to apply to the entire application,
You may apply the attribute at controller level or action level.
E.g.
//[HttpsAuthorizationFilter]
//public class GamerController : ApiController
...
//[HttpsAuthorizationFilter]
//public async Task<IHttpActionResult> GetGamers(string gender = "all")

5.
Use BasicAuthorizationFilterAttribute.
It is for understanding basic concept, not for real world practice.
5.1.

```
In OnlineGame.WebApi/App_Start/WebApiConfig.cs
//config.Filters.Add(new BasicAuthorizationFilterAttribute());
If you add the attribute in WebApiConfig.cs,
it will apply to the entire application.
5.2.
If you don't want to apply to the entire application,
You may apply the attribute at controller level or action level.
E.g.
//[BasicAuthorizationFilter]
//public class GamerController : ApiController
...
//[BasicAuthorizationFilter]
//public async Task<IHttpActionResult> GetGamers()
-------------------------------------
6.
//JsonMediaTypeFormatter json = config.Formatters.JsonFormatter;
//json.SerializerSettings.PreserveReferencesHandling =
Newtonsoft.Json.PreserveReferencesHandling.Objects;
//config.Formatters.Remove(config.Formatters.XmlFormatter);
Use JSON formatter as a PreserveReferencesHandling.
Remove Xml Formatter
Reference:
A.
https://forums.asp.net/t/1983286.aspx?Web+API+error+The+ObjectContent+1+type+failed+to+serialize+the+response+body+for+content+type+application+xml+charset+utf+8+
B.
https://stackoverflow.com/questions/23098191/failed-to-serialize-the-response-in-web-api-with-json
*/
```

7.4. OnlineGame.WebApi/Controllers/Api/GamerController.cs

```
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using System.Web.Http;
using System.Web.Http.Description;
using OnlineGame.Data;
using OnlineGame.WebApi.Account;
//using System.Web.Http.Cors;
//using OnlineGame.WebApi.WebShared;
namespace OnlineGame.WebApi.Controllers.Api
{
    //[EnableCors("*", "*", "*")]
    //[EnableCors("https://ithandyguytutorial.blogspot.com.au", "*", "*")]
    //[EnableCors("http://localhost:49804", "*", "*")]
    //[HttpsAuthorizationFilter]
    //[BasicAuthorizationFilter]
    public class GamerController : ApiController
    {
        private OnlineGameContext _db = new OnlineGameContext();

        //// GET: api/Gamer
        //[HttpGet]
        //public IQueryable<Gamer> GetGamers()
        //{
        //    return _db.Gamers;
        //}
```

```
        //GET: api/gamer?gender=female  --> Only Female Gamer
        //GET: api/gamer? gender = male-- > Only Male Gamer
        //GET: api/gamer --> All Gamers
        //[DisableCors]
        //[HttpsAuthorizationFilter]
        [BasicAuthorizationFilter]
        [HttpGet]
        public async Task<IHttpActionResult> GetGamers()
        {
            string username = Thread.CurrentPrincipal.Identity.Name;
            if (string.IsNullOrEmpty(username))
                return BadRequest($"{username} is null or empty.");
            GamerIdentity gamerIdentity =
                await _db.GamerIdentities
                 .Where(gi => gi.UserName.Equals(username))
                 .FirstOrDefaultAsync();
            if (gamerIdentity == null) return NotFound();  //404
            Gamer gamer =
                await _db.Gamers
                 .Where(g => g.Id == gamerIdentity.Id)
                 .FirstOrDefaultAsync();
            if (gamer == null) return NotFound();  //404
            return Ok(gamer);   //200
        }

        // GET: api/Gamer/5
        [HttpGet]
        [ResponseType(typeof(Gamer))]
        public async Task<IHttpActionResult> GetGamer(int id)
        {
            Gamer gamer = await _db.Gamers.FindAsync(id);
            if (gamer == null) return NotFound();  //404
            return Ok(gamer);   //200
        }

        // PUT: api/Gamer/5
        [ResponseType(typeof(void))]
        [HttpPut]
        public async Task<IHttpActionResult> PutGamer(int id, Gamer gamer)
        {
            if (!ModelState.IsValid) return BadRequest(ModelState);  //400
            //if (id != gamer.Id)   return BadRequest();
            //1.
            gamer.Id = id;
            _db.Entry(gamer).State = EntityState.Modified;  //update the gamer
            //2.
            //Gamer currentGamer = await _db.Gamers.FirstOrDefaultAsync(g => g.Id == id);
            //if (currentGamer == null) return NotFound();  //404
            //currentGamer.Name = gamer.Name;
            //currentGamer.Gender = gamer.Gender;
            //currentGamer.Score = gamer.Score;
            //currentGamer.GameMoney = gamer.GameMoney;
            try
            {
                await _db.SaveChangesAsync();
                return Ok();    //200
```

```
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!GamerExists(id)) return NotFound();  //404
                throw;
            }
        }

        // POST: api/Gamer
        [ResponseType(typeof(Gamer))]
        [HttpPost]
        public async Task<IHttpActionResult> PostGamer(Gamer gamer)
        {
            if (!ModelState.IsValid) return BadRequest(ModelState); //400
            _db.Gamers.Add(gamer);
            await _db.SaveChangesAsync();
            //Return Created/201.
            //1.
            return CreatedAtRoute("DefaultApi", new { id = gamer.Id }, gamer);    //Created/201
        }

        // DELETE: api/Gamer/5
        [ResponseType(typeof(Gamer))]
        [HttpDelete]
        public async Task<IHttpActionResult> DeleteGamer(int id)
        {
            Gamer gamer = await _db.Gamers.FindAsync(id);
            if (gamer == null) return NotFound();   //404
            _db.Gamers.Remove(gamer);
            await _db.SaveChangesAsync();
            return Ok(gamer);   //200
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing) _db.Dispose();   //Dispose DBContext
            base.Dispose(disposing);
        }

        private bool GamerExists(int id)
        {
            return _db.Gamers.Count(e => e.Id == id) > 0;
        }
    }
}


/*
1.
1.1.
By default, the HTTP verb GET maps to a method that has the name Get() or "Get" prefix.
E.g. Get(), GetGamers, GetXXX()
If you want the HTTP verb GET maps to the method name without "Get" prefix.
You can use [HttpGet] attribute.
1.2.
[HttpGet] attribute maps HTTP verb GET.
[HttpPost] attribute maps HTTP verb POST.
[HttpPut] attribute maps HTTP verb PUT.
[HttpDelete] attribute maps HTTP verb DELETE.
```

2.
[FromUri] V.S. [FromBody]
Web Api default binding parameter convention
2.1.
By default, if the parameter is a simple type,
Web Api will try to get value from uri.
E.g. int, double, bool, ...etc.
2.2.
By default, if the parameter is a complex type,
Web Api will try to get value from the request body.
E.g. Gamer

2.3.
//[HttpPut]
//public async Task<IHttpActionResult> UpdateGamer(int id, Gamer gamer)
By Default, the Web Api will try to get id from uri, and gamer from request body as below code.
//[HttpPut]
//public async Task<IHttpActionResult> UpdateGamer([FromUri]int id, [FromBody]Gamer gamer)
E.g.
A.
PUT
http://localhost:58302/api/Gamer/8
B.
Request Header
Host: localhost:58302
Content-Type: application/json
B.1.
Accept: application/json
means we request JSON format response.
B.2.
Content-Type: application/json
The client will post a data to the server, the data format is JSON
C.
Request Body
{
"Name":"NameEight XYZ222",
"Gender":"Male",
"Score":450,
"GameMoney":1500
}

2.4.
//[HttpPut]
//public async Task<IHttpActionResult> UpdateGamer([FromBody]int id, [FromUri]Gamer gamer)
[FromBody] will enfroce to get id from request body
[FromUri] will enforce to get gamer from uri
E.g.
A.
PUT
http://localhost:58302/api/Gamer?Name=NameEight%20XYZ333&Gender=Male&Score=450&GameMoney=1500
B.
Request Header
Host: localhost:58302
Content-Type: application/json
B.1.
Accept: application/json
means we request JSON format response.
B.2.
Content-Type: application/json
The client will post a data to the server, the data format is JSON
C.
Request Body
8

3.

WebApi Cors (Cross Origin Resource Sharing)
allows Jquery AJAX may call Web API in the different origins

3.1.
new EnableCorsAttribute(origins, headers, methods)
//EnableCorsAttribute cors = new EnableCorsAttribute("*", "*", "*");
//config.EnableCors(cors);
It allows the resource to be accessed by all origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")

3.1.1.
origins:
It is a Comma-separated whitelist which are allowed to access the web api by Ajax call.
E.g.3.1.1.1.
"http://localhost:49804,https://ithandyguytutorial.blogspot.com.au"
That means only http://localhost:49804 and https://ithandyguytutorial.blogspot.com.au
can access the web api by Ajax call.
E.g.3.1.1.2.
"*"
It means allows all origins to access the web api by Ajax call.

3.1.2.
headers:
It is a Comma-separated whitelist of request headers which are supported by the resource.
E.g.3.1.2.1.
"accept,content-type,origin" means only these 3 things can be used in request header.
E.g.3.1.2.2.
"*"
It means allows all request headers to the web api by Ajax call.

3.1.3.
methods:
It is a Comma-separated whitelist of methods which are supported by the resource.
E.g.3.1.3.1.
"GET,POST" means only these 2 methods can be used in request.
E.g.3.1.3.2.
"*"
It means allows all request methods to the web api by Ajax call.

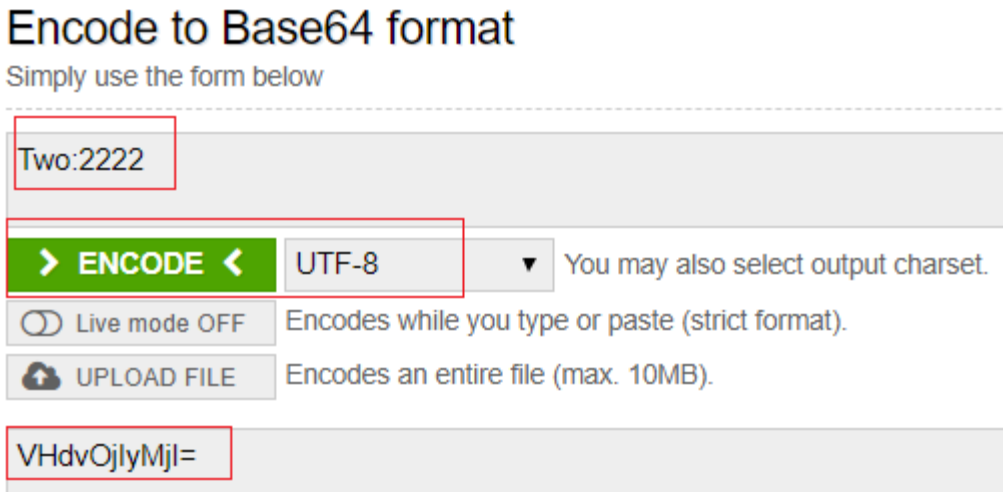
3.2.
In OnlineGame.WebApi/App_Start/WebApiConfig.cs
//config.EnableCors();
In OnlineGame.WebApi/Controllers/Api/GamerController.cs
////[EnableCors("*", "*", "*")]
////[EnableCors("https://ithandyguytutorial.blogspot.com.au", "*", "*")]
//[EnableCors("http://localhost:49804", "*", "*")]
//public class GamerController : ApiController
...
//[DisableCors]
//[HttpGet]
//public async Task<IHttpActionResult> LoadGamers(string gender = "")
3.2.1.
If you don't want to enable Cors globally,
then you may enable Cors in api controller level or method level.
When you enable Cors, in api controller level,
//[EnableCors("*", "*", "*")]
it will apply to all methods in that controller.
If you want to exclude any method, then you may use
//[DisableCors]
3.2.2.
3.2.2.1.
//[EnableCors("*", "*", "*")]
EnableCorsAttribute(origins, headers, methods)
It allows the resource to be accessed by all origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")

```
3.2.2.2.
//[EnableCors("https://ithandyguytutorial.blogspot.com.au", "*", "*")]
EnableCorsAttribute(origins, headers, methods)
It allows the resource to be accessed by https://ithandyguytutorial.blogspot.com.au origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")
3.2.2.3.
//[EnableCors("http://localhost:49804", "*", "*")]
It allows the resource to be accessed by http://localhost:49804 origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")

-------------------------------------
4.
HTTP redirect to HTTPS
4.1.
In OnlineGame.WebApi/App_Start/WebApiConfig.cs
//config.Filters.Add(new HttpsAuthorizationFilterAttribute());
If you add the attribute in WebApiConfig.cs,
it will apply to the entire application.
4.2.
If you don't want to apply to the entire application,
You may apply the attribute at controller level or action level.
E.g.
//[HttpsAuthorizationFilter]
//public class GamerController : ApiController
...
//[HttpsAuthorizationFilter]
//public async Task<IHttpActionResult> GetGamers(string gender = "all")

-------------------------------------
5.
Use BasicAuthorizationFilterAttribute.
It is for understanding basic concept, not for real world practice.
5.1.
In OnlineGame.WebApi/App_Start/WebApiConfig.cs
//config.Filters.Add(new BasicAuthorizationFilterAttribute());
If you add the attribute in WebApiConfig.cs,
it will apply to the entire application.
5.2.
If you don't want to apply to the entire application,
You may apply the attribute at controller level or action level.
E.g.
//[BasicAuthorizationFilter]
//public class GamerController : ApiController
...
//[BasicAuthorizationFilter]
//public async Task<IHttpActionResult> GetGamers()
*/
```

7.5. base64 encode

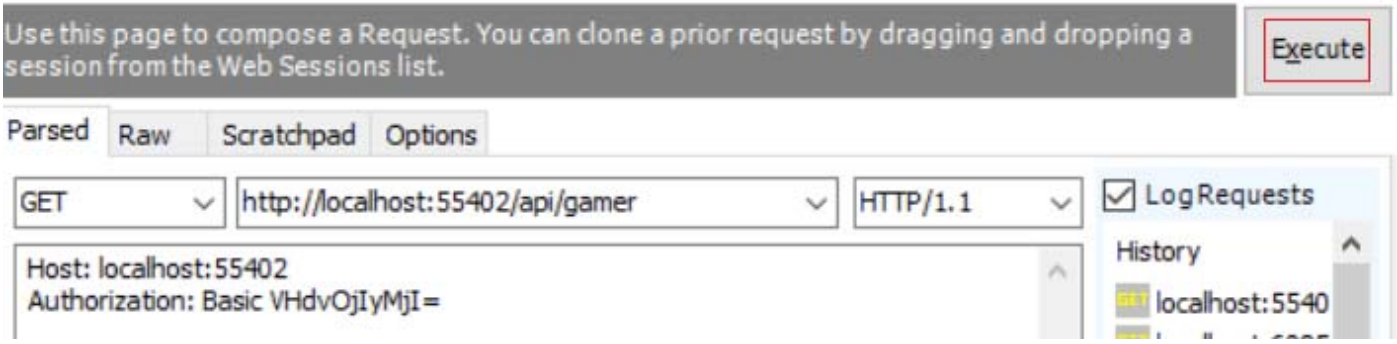
Google key word "base64 encode"
E.g.
https://www.base64encode.org/
We know the username and password are the following formats.
"username:password"
We know we have a user that the username is "**Two**" and password "**2222**",
so the token string will be "**Two:2222**"
But the token string must be base64 encode, so we have to convert the UTF8 to base64.
"**One:1111**" in UTF8 is "**T25lOjExMTE=**" in base64.
"**Two:2222**" in UTF8 is "**VHdvOjIyMjI=**" in base64.

"**Three:3333**" in UTF8 is "**VGhyZWU6MzMzMw==**" in base64.
"**Four:4444**" in UTF8 is "**Rm91cjo0NDQ0**" in base64.
"**Five:5555**" in UTF8 is "**Rml2ZTo1NTU1**" in base64.
"**Six:6666**" in UTF8 is "**U2l4OjY2NjY=**" in base64.
"**Seven:7777**" in UTF8 is "**U2V2ZW46Nzc3Nw==**" in base64.



7.6. Fiddler test Basic login

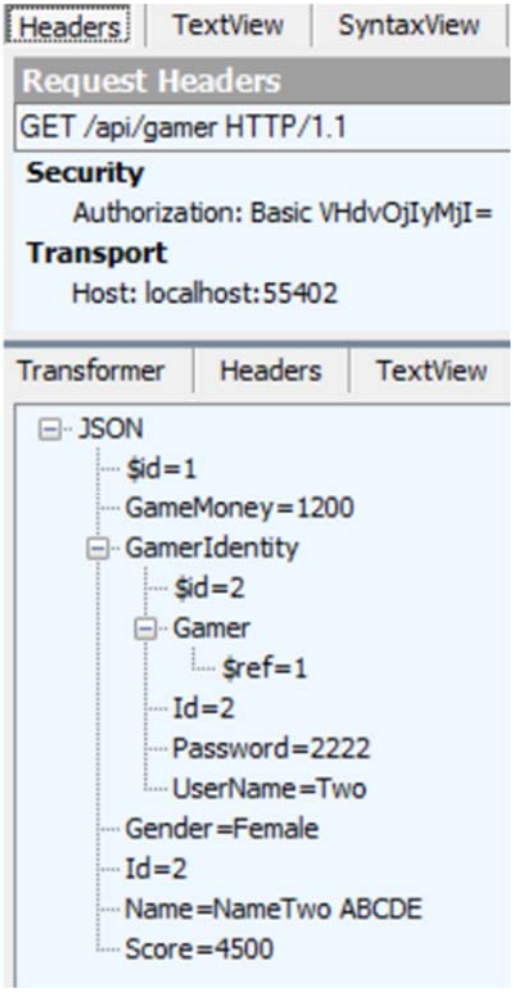
"**Two:2222**" in UTF8 is "**VHdvOjIyMjI=**" in base64.
-->
GET
http://localhost:55402/api/gamer
-->
Request Header:
Host: localhost:55402
Authorization: Basic VHdvOjIyMjI=



-->

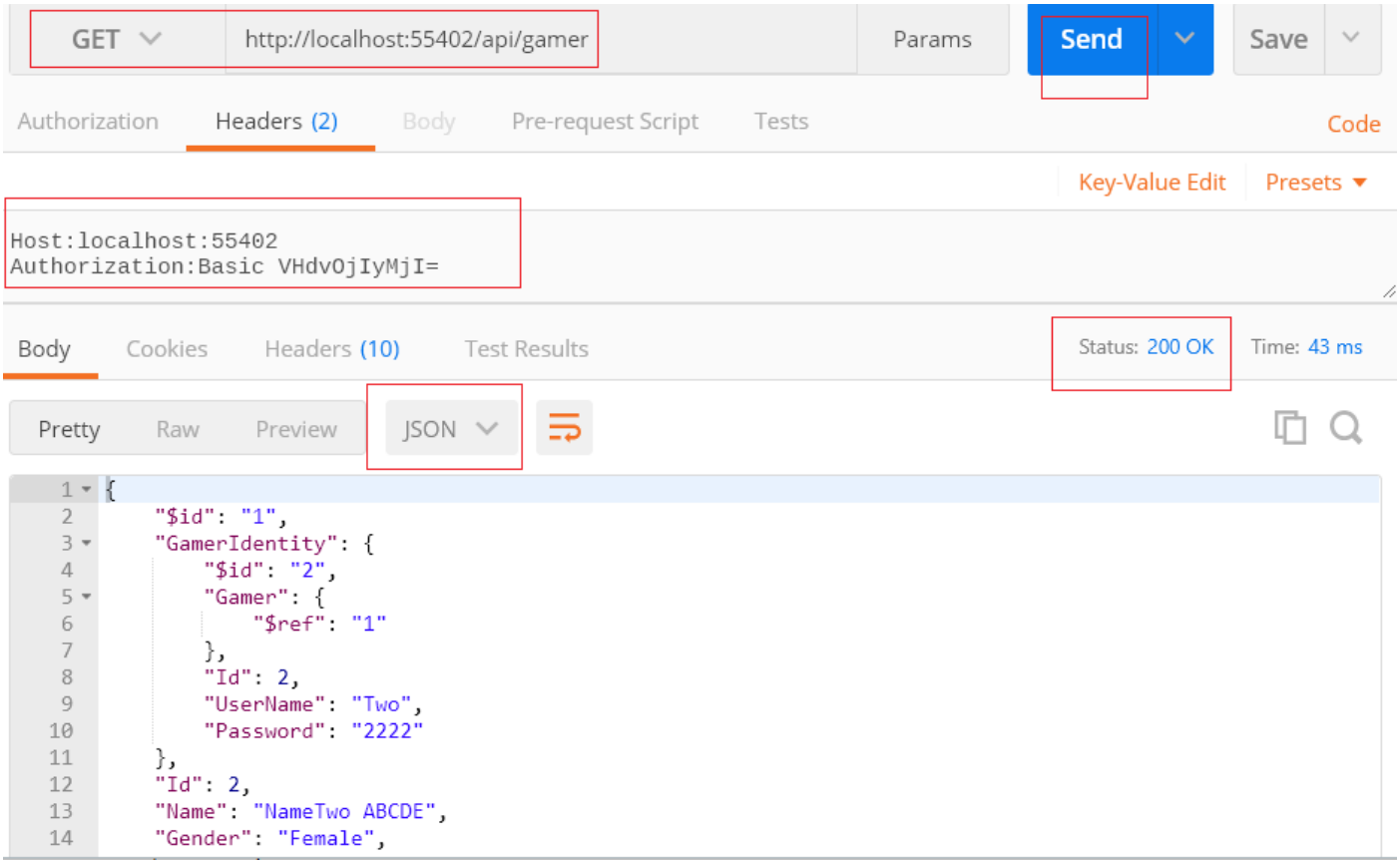
#	Result	Protocol	Host	URL
125	200	HTTP	localhost:55402	/api/gamer

-->



7.7. Postman test Basic login

"**Two:2222**" in UTF8 is "**VHdvOjIyMjI=**" in base64.
-->
GET
http://localhost:55402/api/gamer
-->
Request Header:
Host: localhost:55402
Authorization: Basic VHdvOjIyMjI=



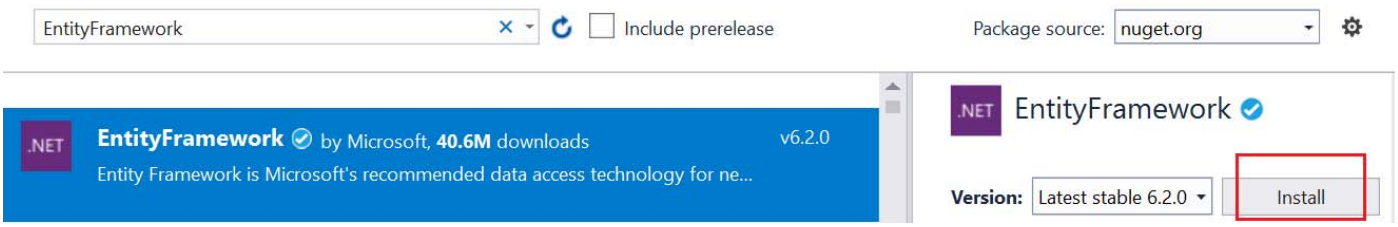
-->


```
{
  "$id": "1",
  "GamerIdentity": {
    "$id": "2",
    "Gamer": {
      "$ref": "1"
    },
    "Id": 2,
    "UserName": "Two",
    "Password": "2222"
  },
  "Id": 2,
  "Name": "NameTwo ABCDE",
  "Gender": "Female",
  "Score": 4500,
  "GameMoney": 1200
}
```


8. OnlineGame.Mvc

8.1. Install Entity Framework

Tools --> NuGet Package Manager --> Manage NuGet Packages for Solutions...
--> Browse tab --> Search : **EntityFramework**
--> Install it

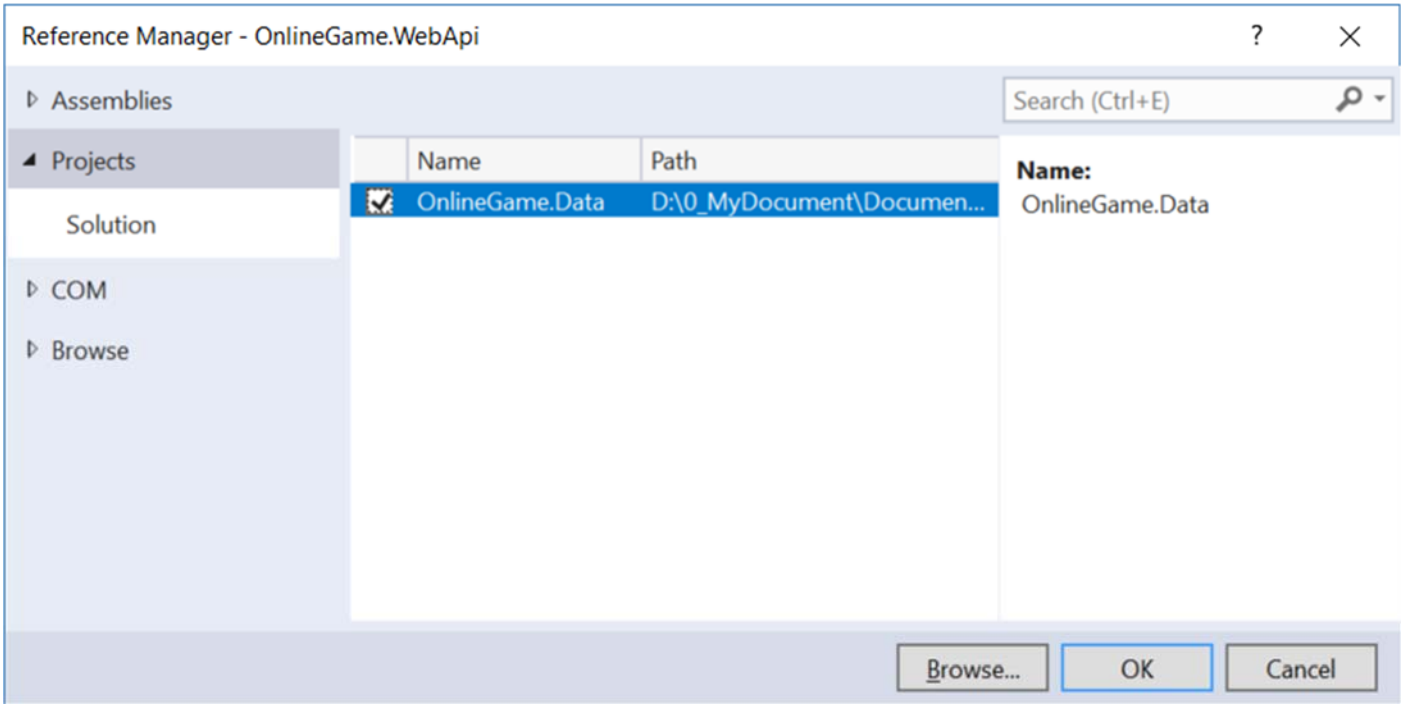


8.2. OnlineGame.Mvc/Web.config : Add Connection String

```
Web.config
69      </compilers>
70    </system.codedom>
71    <entityFramework>
72      <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">
73        <parameters>
74          <parameter value="mssqllocaldb" />
75        </parameters>
76      </defaultConnectionFactory>
77      <providers>
78        <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices,
           EntityFramework.SqlServer" />
79      </providers>
80    </entityFramework>
81    <connectionStrings>
82      <add name="OnlineGameContext" connectionString="metadata=res://*/OnlineGameDataModel.csdl|res://*/
       OnlineGameDataModel.ssdl|res://*/OnlineGameDataModel.msl;provider=System.Data.SqlClient;provider connection
       string=&quot;data source=N550JKL\SQL2016;initial catalog=OnineGame;persist security info=True;user
       id=Tester2;password=1234;MultipleActiveResultSets=True;App=EntityFramework&quot;"
       providerName="System.Data.EntityClient" />
83    </connectionStrings>
84  </configuration>
```

```
<connectionStrings>
    <add name="OnlineGameContext" connectionString="metadata=res://*/OnlineGameDataModel.csdl|res://*/OnlineGameDataModel.ssdl|res://*/OnlineGameDataModel.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=N550JKL\SQL2016;initial catalog=OnineGame;persist security info=True;user id=Tester2;password=1234;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />
  </connectionStrings>
```

8.3. Add Reference

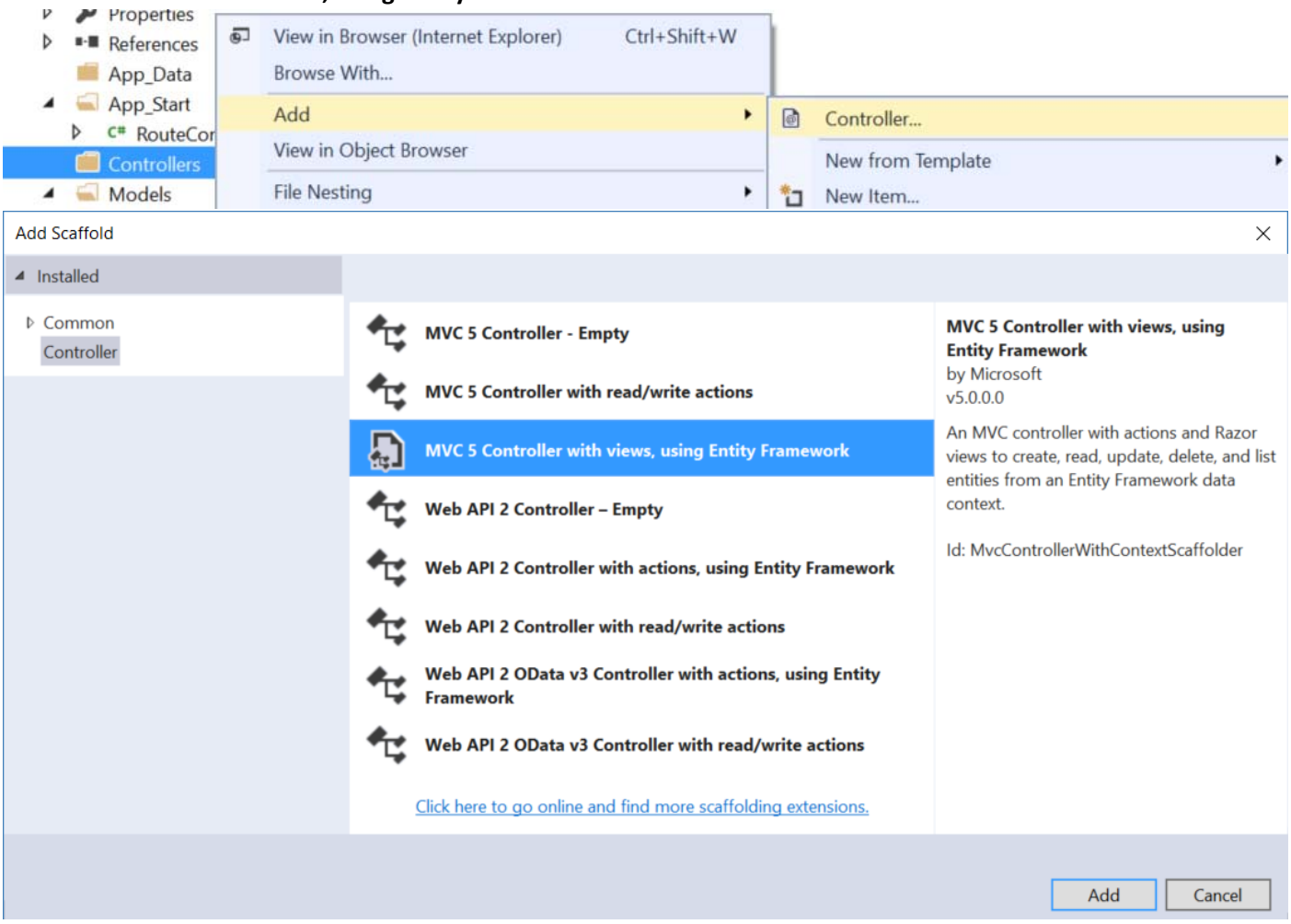


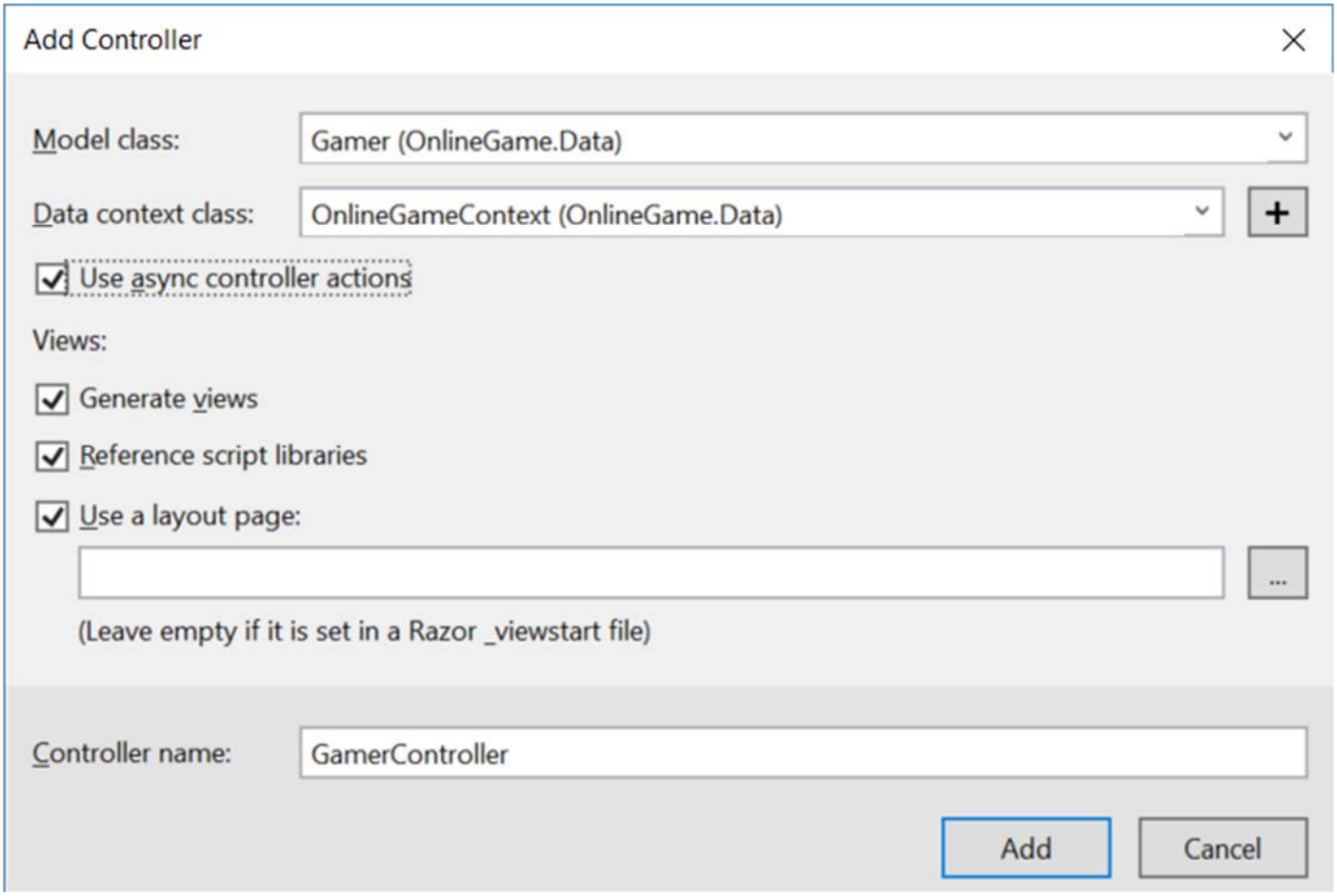
8.4. OnlineGame.Mvc/Controllers/GamerController.cs

Controllers --> Right click --> Add --> Controller

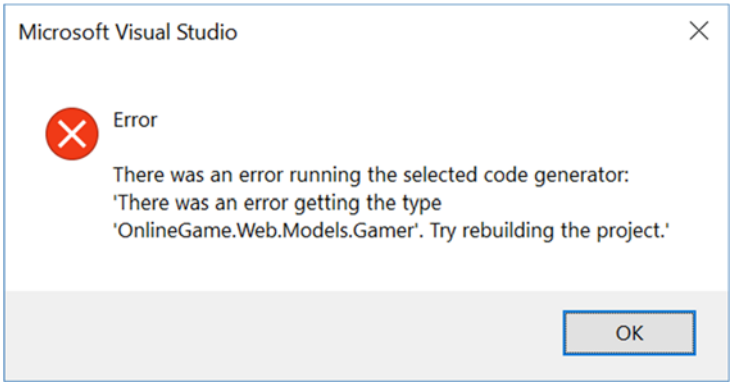
-->

MVC 5 Controller with views, using Entity Framework





It will automatically generate the controller, views, and several javascript and css files.
If you see the following error message, then you have to re-build solution before you create the controller.



8.5. OnlineGame.Mvc/Controllers/GamerController.cs

```
using System.Data.Entity;
using System.Threading.Tasks;
using System.Net;
using System.Web.Mvc;
using OnlineGame.Data;

namespace OnlineGame.Mvc.Controllers
{
    public class GamerController : Controller
    {
        private OnlineGameContext _db = new OnlineGameContext();

        // GET: Gamer
        [HttpGet]
```

```
        public async Task<ActionResult> Index()
        {
            return View(await _db.Gamers.ToListAsync());
        }

        [HttpGet]
        public ActionResult IndexWebApi()
        {
            return View();
        }

        // GET: Gamer/Details/5
        [HttpGet]
        public async Task<ActionResult> Details(int? id)
        {
            if (id == null) return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            Gamer gamer = await _db.Gamers.FindAsync(id);
            if (gamer == null) return HttpNotFound();
            return View(gamer);
        }

        // GET: Gamer/Create
        [HttpGet]
        public ActionResult Create()
        {
            return View();
        }

        // POST: Gamer/Create
        // To protect from overposting attacks, please enable the specific properties you want to bind to,
for
        // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Create([Bind(Include = "Id,Name,Gender,Score,GameMoney")] Gamer
gamer)
        {
            if (!ModelState.IsValid) return View(gamer);
            _db.Gamers.Add(gamer);
            await _db.SaveChangesAsync();
            return RedirectToAction("Index");
        }

        // GET: Gamer/Edit/5
        [HttpGet]
        public async Task<ActionResult> Edit(int? id)
        {
            if (id == null) return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            Gamer gamer = await _db.Gamers.FindAsync(id);
            if (gamer == null) return HttpNotFound();
            return View(gamer);
        }

        // POST: Gamer/Edit/5
        // To protect from overposting attacks, please enable the specific properties you want to bind to,
for
        // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
```

```
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Edit([Bind(Include = "Id,Name,Gender,Score,GameMoney")] Gamer
gamer)
        {
            if (!ModelState.IsValid) return View(gamer);
            _db.Entry(gamer).State = EntityState.Modified;
            await _db.SaveChangesAsync();
            return RedirectToAction("Index");
        }

        // GET: Gamer/Delete/5
        [HttpGet]
        public async Task<ActionResult> Delete(int? id)
        {
            if (id == null) return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            Gamer gamer = await _db.Gamers.FindAsync(id);
            if (gamer == null) return HttpNotFound();
            return View(gamer);
        }

        // POST: Gamer/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> DeleteConfirmed(int id)
        {
            Gamer gamer = await _db.Gamers.FindAsync(id);
            if (gamer != null) _db.Gamers.Remove(gamer);
            await _db.SaveChangesAsync();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing) _db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

8.6. OnlineGame.WebApi/Controllers/Api/GamerController.cs

```
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using System.Web.Http;
using System.Web.Http.Description;
using OnlineGame.Data;
using OnlineGame.WebApi.Account;
//using System.Web.Http.Cors;
//using OnlineGame.WebApi.WebShared;

namespace OnlineGame.WebApi.Controllers.Api
{
    //[EnableCors("*", "*", "*")]
```

```
//[EnableCors("https://ithandyguytutorial.blogspot.com.au", "*", "*")]
//[EnableCors("http://localhost:49804", "*", "*")]
//[HttpsAuthorizationFilter]
//[BasicAuthorizationFilter]
public class GamerController : ApiController
 {
    private OnlineGameContext _db = new OnlineGameContext();

    //// GET: api/Gamer
    //[HttpGet]
    //public IQueryable<Gamer> GetGamers()
    //{
    //    return _db.Gamers;
    //}
    //GET: api/gamer?gender=female  --> Only Female Gamer
    //GET: api/gamer? gender = male-- > Only Male Gamer
    //GET: api/gamer --> All Gamers
    //[DisableCors]
    //[HttpsAuthorizationFilter]
     [BasicAuthorizationFilter]
     [HttpGet]
    public async Task<IHttpActionResult> GetGamers()
     {
        string username = Thread.CurrentPrincipal.Identity.Name;
        if (string.IsNullOrEmpty(username))
            return BadRequest($"{username} is null or empty.");

        //1.
         GamerIdentity gamerIdentity =
            await _db.GamerIdentities
             .Where(gi => gi.UserName.Equals(username))
             .FirstOrDefaultAsync();
        if (gamerIdentity == null) return NotFound();  //404
         Gamer gamer =
            await _db.Gamers
             .Where(g => g.Id == gamerIdentity.Id)
             .FirstOrDefaultAsync();
        if (gamer == null) return NotFound();  //404
        return Ok(gamer);   //200

        ////-------------------------
        ////2.
        ////Inner Join - Lambda expression query
        //var gamerJoinGamerIdentity =
        //     _db.Gamers.ToList().Join(_db.GamerIdentities.ToList(),
        //     g => g.Id,
        //     gi => gi.Id,
        //     (gamer, gamerIdentity) => new
        //     {
        //         GamerIdentity = gamerIdentity,
        //         Gamer = gamer
        //     }).FirstOrDefault(g => g.GamerIdentity.UserName == username);
        //if (gamerJoinGamerIdentity == null) return NotFound();  //404
        //return Ok(gamerJoinGamerIdentity.Gamer);   //200
```

```
            ////-------------------------
            ////3.
            ////Inner Join - Sql like query
            //var gamerJoinGamerIdentity =
            //   (from g in _db.Gamers.ToList()
            //    join gi in _db.GamerIdentities.ToList()
            //    on g.Id equals gi.Id
            //    select new
            //    {
            //        GamerIdentity = gi,
            //        Gamer = g
            //    }).FirstOrDefault(g => g.GamerIdentity.UserName == username);
            //if (gamerJoinGamerIdentity == null) return NotFound();  //404
            //return Ok(gamerJoinGamerIdentity.Gamer);   //200
        }

        // GET: api/Gamer/5
        [HttpGet]
        [ResponseType(typeof(Gamer))]
        public async Task<IHttpActionResult> GetGamer(int id)
        {
            Gamer gamer = await _db.Gamers.FindAsync(id);
            if (gamer == null) return NotFound();  //404
            return Ok(gamer);   //200
        }

        // PUT: api/Gamer/5
        [ResponseType(typeof(void))]
        [HttpPut]
        public async Task<IHttpActionResult> PutGamer(int id, Gamer gamer)
        {
            if (!ModelState.IsValid) return BadRequest(ModelState);  //400
            //if (id != gamer.Id)   return BadRequest();
            //1.
            gamer.Id = id;
            _db.Entry(gamer).State = EntityState.Modified;  //update the gamer
            //2.
            //Gamer currentGamer = await _db.Gamers.FirstOrDefaultAsync(g => g.Id == id);
            //if (currentGamer == null) return NotFound();  //404
            //currentGamer.Name = gamer.Name;
            //currentGamer.Gender = gamer.Gender;
            //currentGamer.Score = gamer.Score;
            //currentGamer.GameMoney = gamer.GameMoney;
            try
            {
                await _db.SaveChangesAsync();
                return Ok();    //200
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!GamerExists(id)) return NotFound();  //404
                throw;
            }
        }

        // POST: api/Gamer
```

```
        [ResponseType(typeof(Gamer))]
        [HttpPost]
        public async Task<IHttpActionResult> PostGamer(Gamer gamer)
        {
            if (!ModelState.IsValid) return BadRequest(ModelState); //400
            _db.Gamers.Add(gamer);
            await _db.SaveChangesAsync();
            //Return Created/201.
            //1.
            return CreatedAtRoute("DefaultApi", new { id = gamer.Id }, gamer);    //Created/201
        }

        // DELETE: api/Gamer/5
        [ResponseType(typeof(Gamer))]
        [HttpDelete]
        public async Task<IHttpActionResult> DeleteGamer(int id)
        {
            Gamer gamer = await _db.Gamers.FindAsync(id);
            if (gamer == null) return NotFound();   //404
            _db.Gamers.Remove(gamer);
            await _db.SaveChangesAsync();
            return Ok(gamer);   //200
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing) _db.Dispose();   //Dispose DBContext
            base.Dispose(disposing);
        }

        private bool GamerExists(int id)
        {
            return _db.Gamers.Count(e => e.Id == id) > 0;
        }
    }
}

/*
1.
1.1.
By default, the HTTP verb GET maps to a method that has the name Get() or "Get" prefix.
E.g. Get(), GetGamers, GetXXX()
If you want the HTTP verb GET maps to the method name without "Get" prefix.
You can use [HttpGet] attribute.
1.2.
[HttpGet] attribute maps HTTP verb GET.
[HttpPost] attribute maps HTTP verb POST.
[HttpPut] attribute maps HTTP verb PUT.
[HttpDelete] attribute maps HTTP verb DELETE.
----------------------------
2.
[FromUri] V.S. [FromBody]
Web Api default binding parameter convention
2.1.
By default, if the parameter is a simple type,
Web Api will try to get value from uri.
E.g. int, double, bool, ...etc.
2.2.
By default, if the parameter is a complex type,
Web Api will try to get value from the request body.
E.g. Gamer
-----------------
```

2.3.
//[HttpPut]
//public async Task<IHttpActionResult> UpdateGamer(int id, Gamer gamer)
By Default, the Web Api will try to get id from uri, and gamer from request body as below code.
//[HttpPut]
//public async Task<IHttpActionResult> UpdateGamer([FromUri]int id, [FromBody]Gamer gamer)
E.g.
A.
PUT
http://localhost:58302/api/Gamer/8
B.
Request Header
Host: localhost:58302
Content-Type: application/json
B.1.
Accept: application/json
means we request JSON format response.
B.2.
Content-Type: application/json
The client will post a data to the server, the data format is JSON
C.
Request Body
{
"Name":"NameEight XYZ222",
"Gender":"Male",
"Score":450,
"GameMoney":1500
}

2.4.
//[HttpPut]
//public async Task<IHttpActionResult> UpdateGamer([FromBody]int id, [FromUri]Gamer gamer)
[FromBody] will enfroce to get id from request body
[FromUri] will enforce to get gamer from uri
E.g.
A.
PUT
http://localhost:58302/api/Gamer?Name=NameEight%20XYZ333&Gender=Male&Score=450&GameMoney=1500
B.
Request Header
Host: localhost:58302
Content-Type: application/json
B.1.
Accept: application/json
means we request JSON format response.
B.2.
Content-Type: application/json
The client will post a data to the server, the data format is JSON
C.
Request Body
8

3.
WebApi Cors (Cross Origin Resource Sharing)
allows Jquery AJAX may call Web API in the different origins

3.1.
new EnableCorsAttribute(origins, headers, methods)
//EnableCorsAttribute cors = new EnableCorsAttribute("*", "*", "*");
//config.EnableCors(cors);
It allows the resource to be accessed by all origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")

3.1.1.
origins:
It is a Comma-separated whitelist which are allowed to access the web api by Ajax call.

E.g.3.1.1.1.
"http://localhost:49804,https://ithandyguytutorial.blogspot.com.au"
That means only http://localhost:49804 and https://ithandyguytutorial.blogspot.com.au
can access the web api by Ajax call.
E.g.3.1.1.2.
"*"
It means allows all origins to access the web api by Ajax call.

3.1.2.
headers:
It is a Comma-separated whitelist of request headers which are supported by the resource.
E.g.3.1.2.1.
"accept,content-type,origin" means only these 3 things can be used in request header.
E.g.3.1.2.2.
"*"
It means allows all request headers to the web api by Ajax call.

3.1.3.
methods:
It is a Comma-separated whitelist of methods which are supported by the resource.
E.g.3.1.3.1.
"GET,POST" means only these 2 methods can be used in request.
E.g.3.1.3.2.
"*"
It means allows all request methods to the web api by Ajax call.

3.2.
In OnlineGame.WebApi/App_Start/WebApiConfig.cs
//config.EnableCors();
In OnlineGame.WebApi/Controllers/Api/GamerController.cs
////[EnableCors("*", "*", "*")]
////[EnableCors("https://ithandyguytutorial.blogspot.com.au", "*", "*")]
//[EnableCors("http://localhost:49804", "*", "*")]
//public class GamerController : ApiController
...
//[DisableCors]
//[HttpGet]
//public async Task<IHttpActionResult> LoadGamers(string gender = "")
3.2.1.
If you don't want to enable Cors globally,
then you may enable Cors in api controller level or method level.
When you enable Cors, in api controller level,
//[EnableCors("*", "*", "*")]
it will apply to all methods in that controller.
If you want to exclude any method, then you may use
//[DisableCors]
3.2.2.
3.2.2.1.
//[EnableCors("*", "*", "*")]
EnableCorsAttribute(origins, headers, methods)
It allows the resource to be accessed by all origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")
3.2.2.2.
//[EnableCors("https://ithandyguytutorial.blogspot.com.au", "*", "*")]
EnableCorsAttribute(origins, headers, methods)
It allows the resource to be accessed by https://ithandyguytutorial.blogspot.com.au origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")
3.2.2.3.
//[EnableCors("http://localhost:49804", "*", "*")]
It allows the resource to be accessed by http://localhost:49804 origins,
and it accepts any request header ("accept,content-type,origin...etc"),
and it accepts all methods ("GET,POST...etc")

4.

```
HTTP redirect to HTTPS
4.1.
In OnlineGame.WebApi/App_Start/WebApiConfig.cs
//config.Filters.Add(new HttpsAuthorizationFilterAttribute());
If you add the attribute in WebApiConfig.cs,
it will apply to the entire application.
4.2.
If you don't want to apply to the entire application,
You may apply the attribute at controller level or action level.
E.g.
//[HttpsAuthorizationFilter]
//public class GamerController : ApiController
...
//[HttpsAuthorizationFilter]
//public async Task<IHttpActionResult> GetGamers(string gender = "all")
-------------------------------------
5.
Use BasicAuthorizationFilterAttribute.
It is for understanding basic concept, not for real world practice.
5.1.
In OnlineGame.WebApi/App_Start/WebApiConfig.cs
//config.Filters.Add(new BasicAuthorizationFilterAttribute());
If you add the attribute in WebApiConfig.cs,
it will apply to the entire application.
5.2.
If you don't want to apply to the entire application,
You may apply the attribute at controller level or action level.
E.g.
//[BasicAuthorizationFilter]
//public class GamerController : ApiController
...
//[BasicAuthorizationFilter]
//public async Task<IHttpActionResult> GetGamers()
*/
```

8.7. OnlineGame.Mvc/Views/Gamer/IndexWebApi.cshtml

```
@{
    ViewBag.Title = "IndexWebApi";
}
<h2>IndexWebApi</h2>
<div>
    Username : <input type="text" id="TextboxUserName" /><br />
    Password : <input type="password" id="TextboxPassword" /><br />
    <br /><br />
    <input id="btnGamerList" type="button" value="Gamer List" />
    <input id="btnClear" type="button" value="Clear" />
    <ul id="ulGamers"></ul>
</div>
<script src="~/Scripts/jquery-1.10.2.min.js"></script>
<script type="text/javascript">
    $(document).ready(function () {
        var ulGamers = $('#ulGamers');
        var gamerDataType = 'json';
        var gamerApiUrl = 'http://localhost:55402/api/gamer';   //*************Change to your port
        //http://localhost:55402 is the domain of OnlineGame.WebApi project.
        //It supposed to call gamer api controller in OnlineGame.WebApi.
        //However, it will fails.
        //For security reason, web browsers do not allow
        //Jquery AJAX call Web API in the different origin/domain.
```

```
        //There are 2 popular ways to fix it.
        //1.
        //JSONP (JSON with Padding) will wrap the JSON data in a function
        //2.
        //Enable CORS (Cross Origin Resource Sharing)
        //In our case, we use CORS in OnlineGame.WebApi/App_Start/WebApiConfig.cs
        ////EnableCorsAttribute cors = new EnableCorsAttribute("*", "*", "*");
        ////config.EnableCors(cors);
        $('#btnGamerList').click(function () {
            // Get the username & password from textboxes
            var username = $('#TextboxUserName').val();
            var password = $('#TextboxPassword').val();
            // btoa() method encodes a string to Base64
            var headersAuthorizationToken = 'Basic ' + btoa(username + ':' + password);
            $.ajax({
                type: 'GET',
                url: gamerApiUrl,
                dataType: gamerDataType,
                // Specify the authentication header
                // btoa() method encodes a string to Base64
                headers: {
                    'Authorization': headersAuthorizationToken
                },
                success: function (data) {
                    ulGamers.empty();
                    $.each(data, function (index, val) {
                        ulGamers.append('<li>' + val + '</li>');
                    });
                },
                //A.
                //No matter the AJAX has been called successfully or not,
                //complete event will always be called when AJAX complete.
                //B.
                //jqXHR is Jquery XML HTTP Request object.
                complete: function (jqXHR) {
                    if (jqXHR.status == '401') {
                        ulGamers.empty();
                        ulGamers.append('<li style="color:red">' + jqXHR.status + ' : ' + jqXHR.statusText
+ '</li>');  //401 : Unauthorized
                    }
                }
            });
        });
        $('#btnClear').click(function () {
            ulGamers.empty();
        });
    });
</script>
```

http://localhost:55415/Gamer/IndexWebApi

IndexWebApi

Username : Two

Password : ••••

Gamer List Clear

- 1
- [object Object]
- 2
- NameTwo ABCDE
- Female
- 4500
- 1200

-->

IndexWebApi

Username : Two2

Password : ••••

Gamer List Clear

- 401 : Unauthorized

-->

GET http://localhost:55402/api/gamer 401 (Unauthorized)