(T5)討論Ssl(SecureSocketsLayer)、EnableCors(CrossOriginResourceSharing)、Https、SelfSigned的Certificate  
CourseGUID 4c5822ff-7111-4e25-a336-ef18d48d54bd  
=======================================================================  
(T5)討論Ssl(SecureSocketsLayer)、EnableCors(CrossOriginResourceSharing)、Https、SelfSigned的Certificate

(T5-1)自動生成ApiController

(T5-2)討論Ssl(SecureSocketsLayer)、Https、SelfSigned的Certificate

(T5-3)討論EnableCors(CrossOriginResourceSharing)把Http轉址到Https  
=======================================================================  
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

-----------

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. Web.config : Add Connection String

4.3. Add Reference

4.4. Controllers/Api/GamerController.cs

-----------

5. Enable SSL (Secure Sockets Layer) and Create self-signed certificate

5.1. OnlineGame.WebApi Enable SSL via Visual Studio 2017

5.2. Manually place the certificate issued by visual studio into the Trusted Root Certificates folder

-----------

6. Enables CORS(Cross Origin Resource Sharing) and HTTP Redirect to HTTPS

6.1. Install WebApi Cors

6.2. OnlineGame.WebApi/WebShared/HttpsAuthorizationFilterAttribute.cs

6.3. OnlineGame.WebApi/App\_Start/WebApiConfig.cs

6.4. Decrypt HTTPS traffic via Fiddler  
=======================================================================

1. OnlineGame DB

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

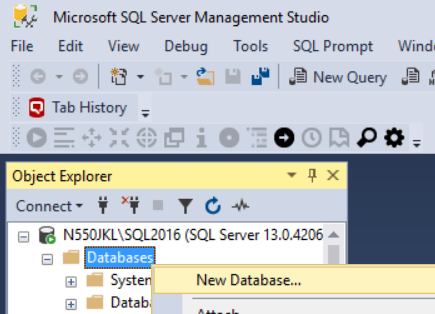
In SQL server Management Studio (SSMS)

Database --> Right Click --> New Database -->

In General Tab -->

Name: **OnlineGame**

In options Tab --> Recovery model : **Simple**



Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

--1 ----------------------------------------------------------

--Drop Table if it exists.

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 ,

      Score INT NOT NULL ,

      GameMoney INT 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

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

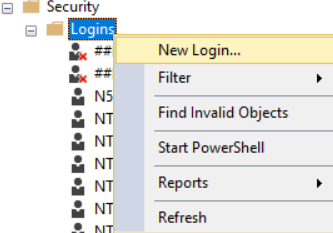
**sysadmin**

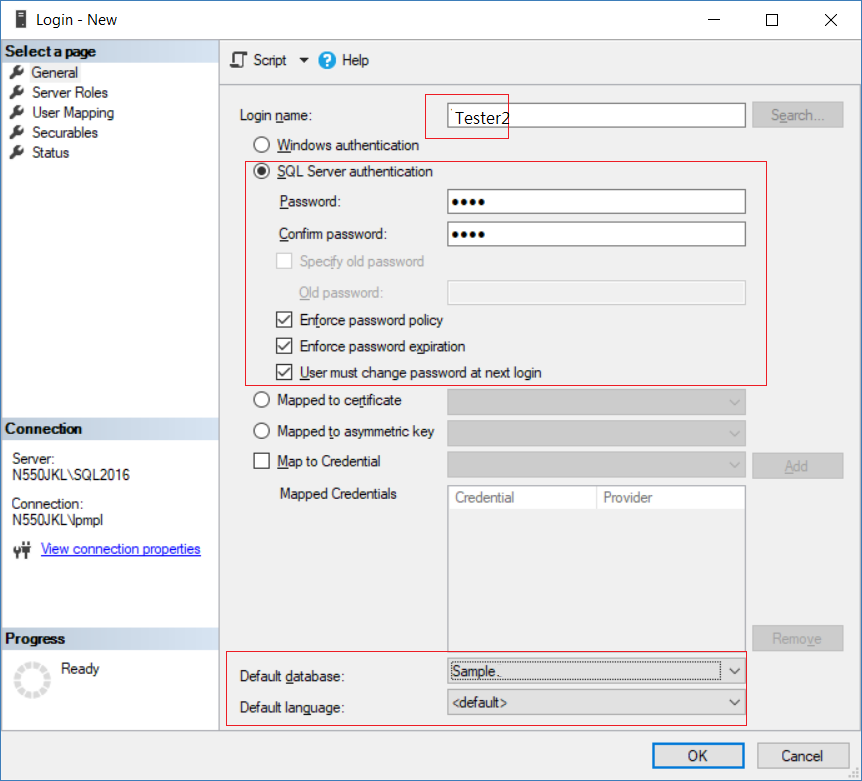
-->

User Mapping Tab

Select **OnlineGame**

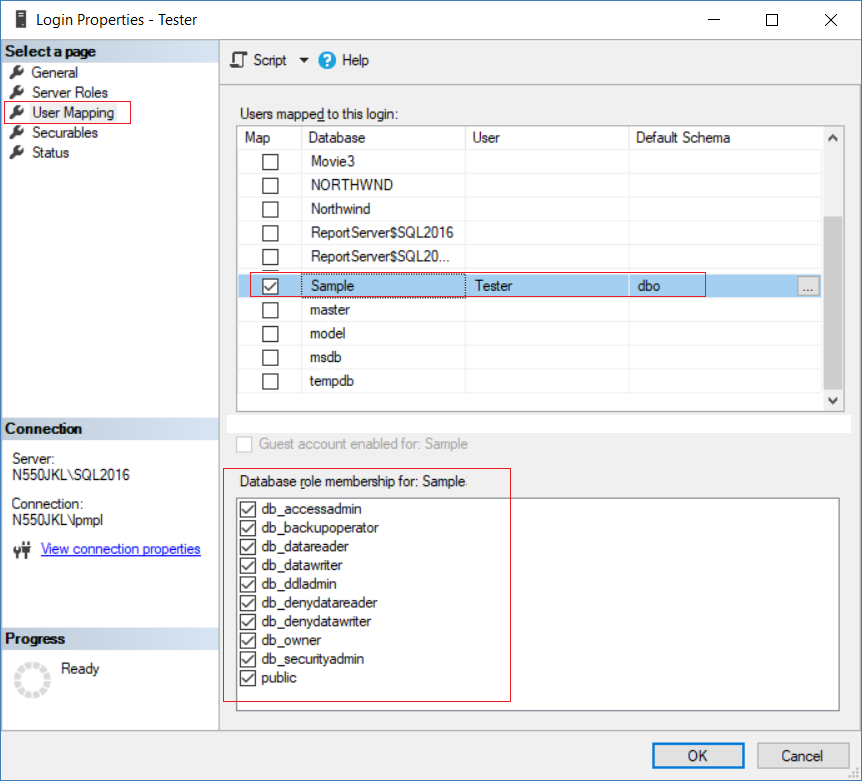
Select every single role.





Graphical user interface, text, application

Description automatically generated



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)**

-->

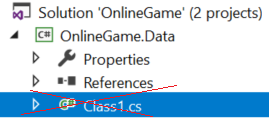
Name:

**OnlineGame.Data**

-->

Delete Class1.cs





2.3. OnlineGame.WebApi

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

Visual C# --> Web --> [ASP.NET](http://asp.net/)Web Application (.Net Framework)

-->

Name: **OnlineGame.WebApi**

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

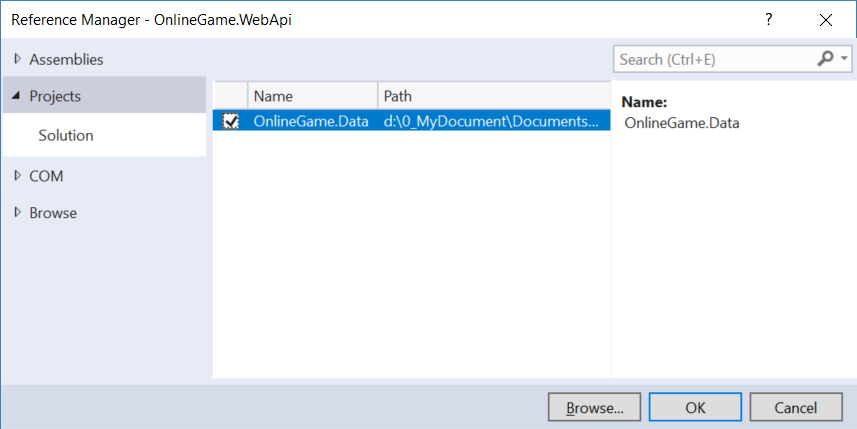
--> Add Reference

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated



3. OnlineGame.Data

3.1. Install Entity Framework

Tools --> NuGet Package Manager --> Manage NuGet Packages for Solutions...

--> Browse tab --> Search  :  **EntityFramework**

--> Install it

Graphical user interface, application, website

Description automatically generated

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**

-->

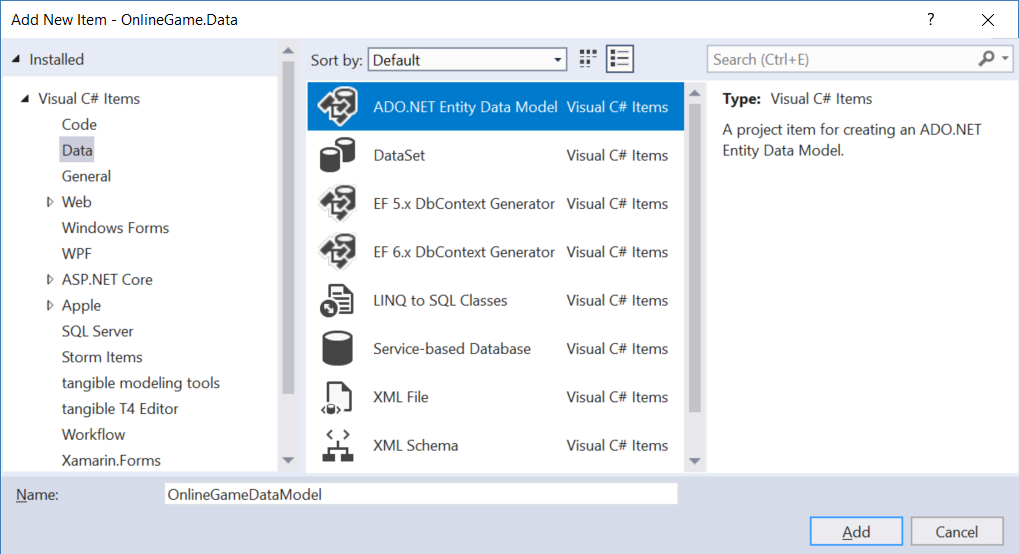
EF Designer from database

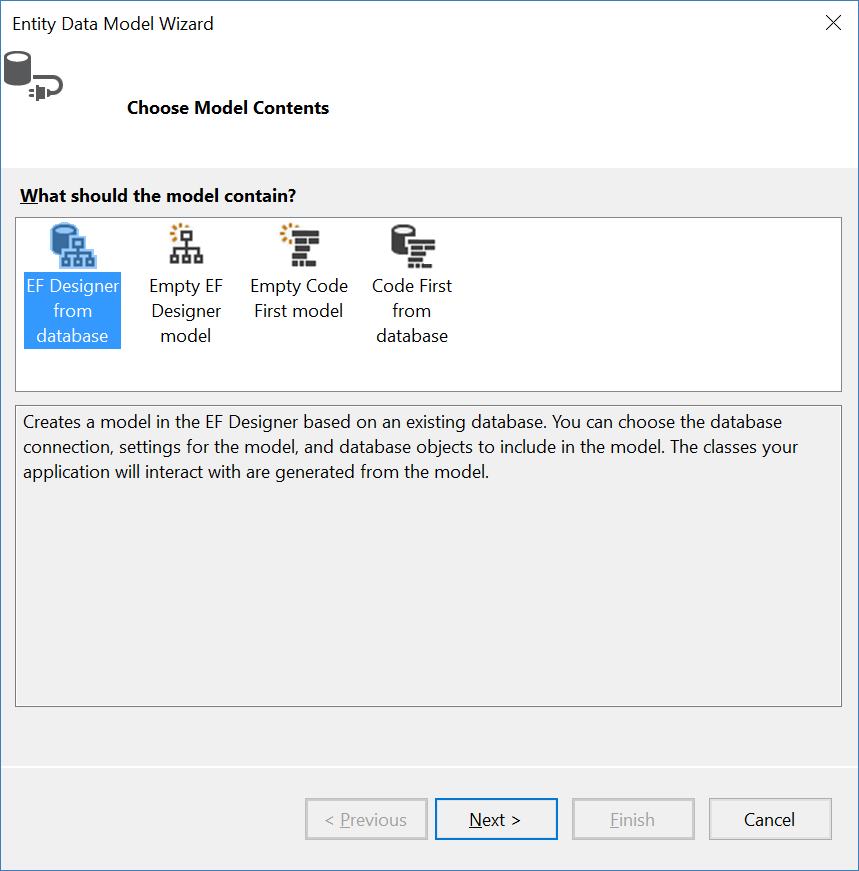
....

-->

Save Connection settings in Web.Config as:

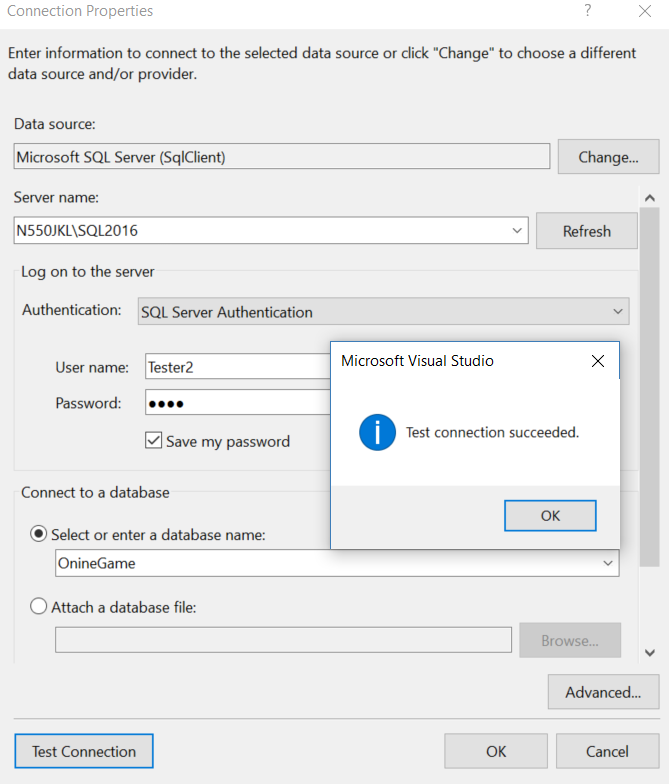
**OnlineGameContext**

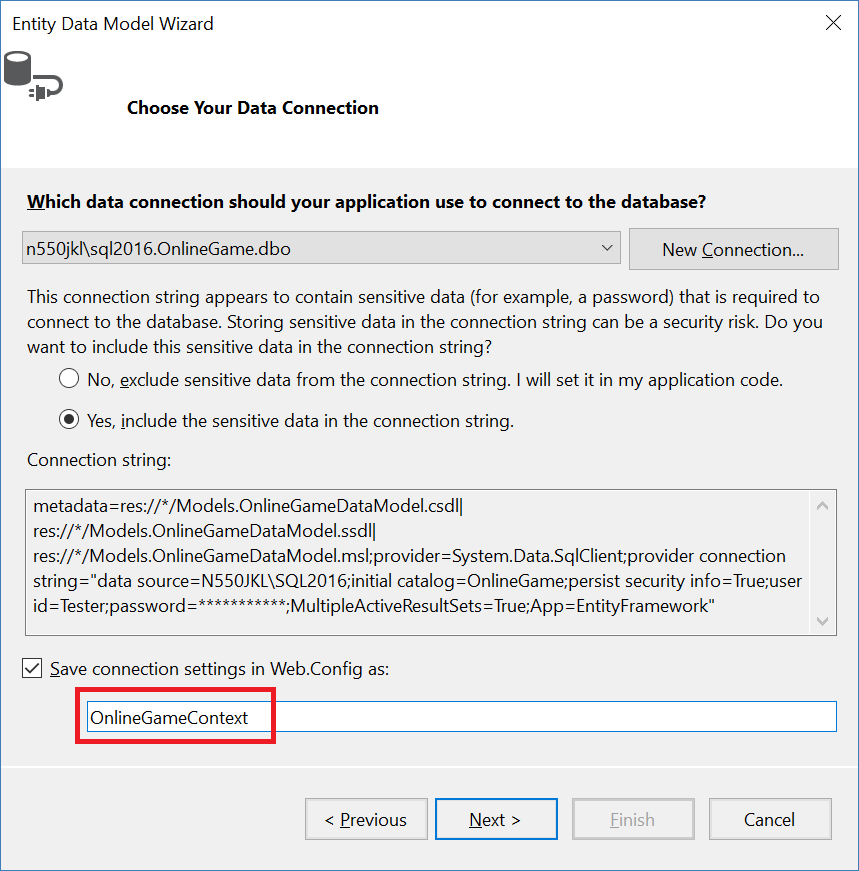


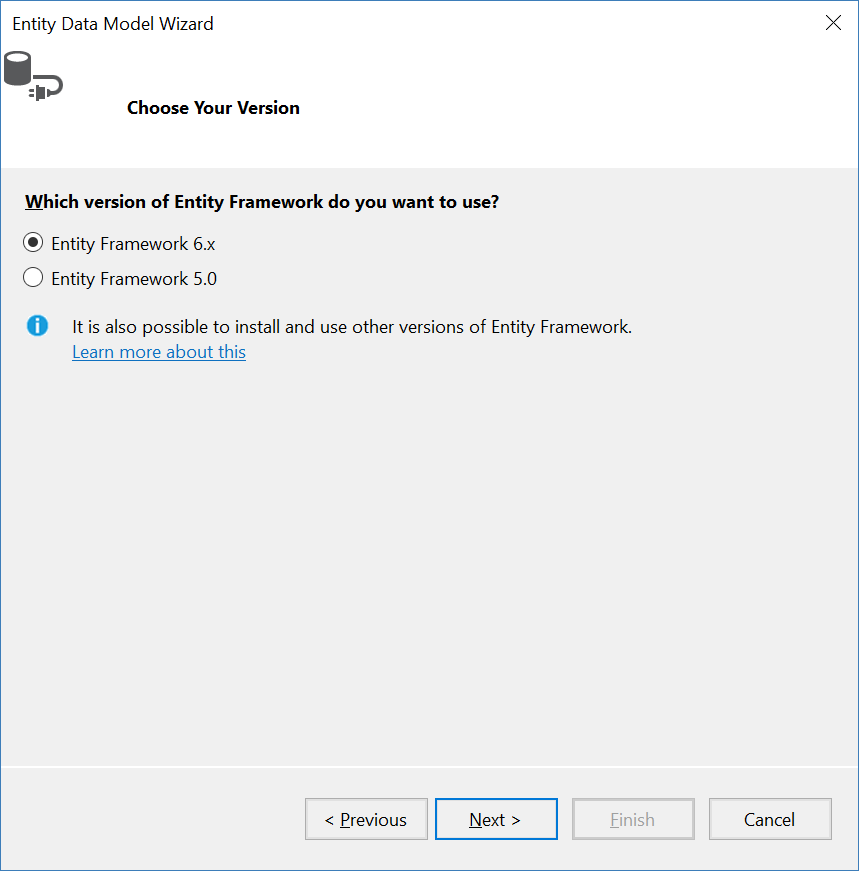


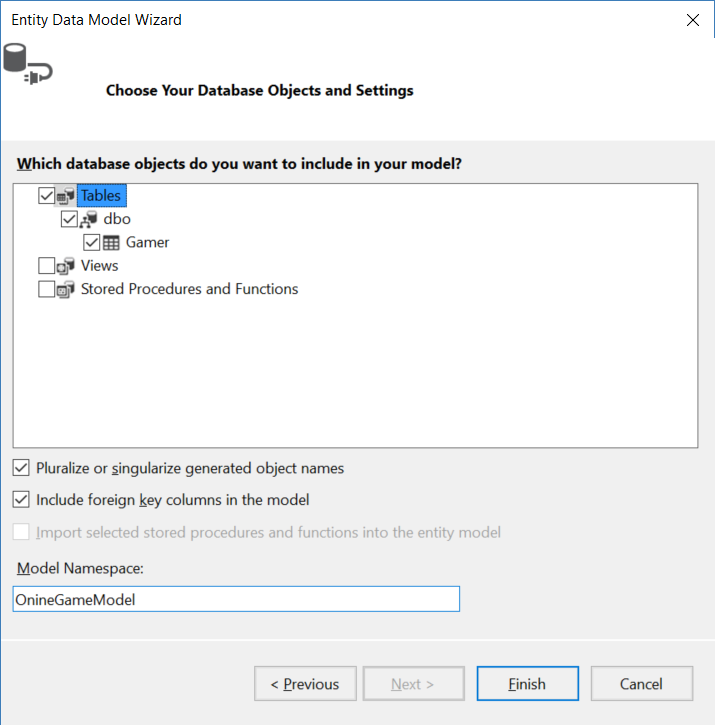
Graphical user interface, text, application

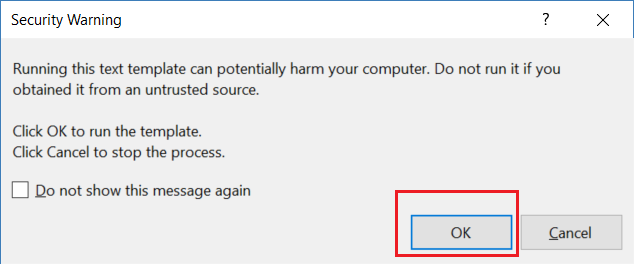
Description automatically generated

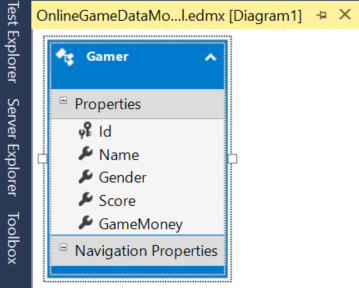












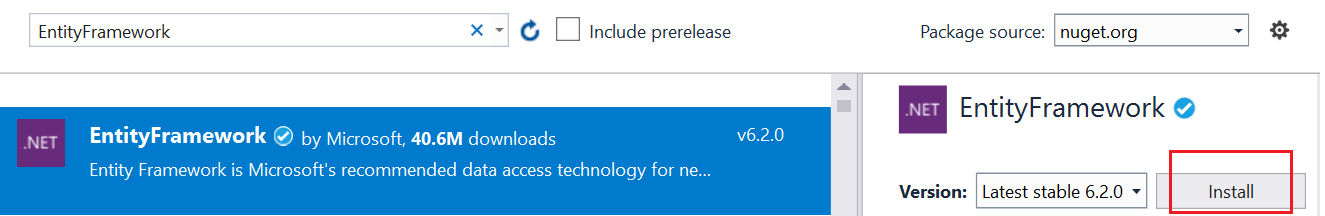
4. OnlineGame.WebApi

4.1. Install Entity Framework

Tools --> NuGet Package Manager --> Manage NuGet Packages for Solutions...

--> Browse tab --> Search  :  **EntityFramework**

--> Install it



4.2. Web.config : Add Connection String

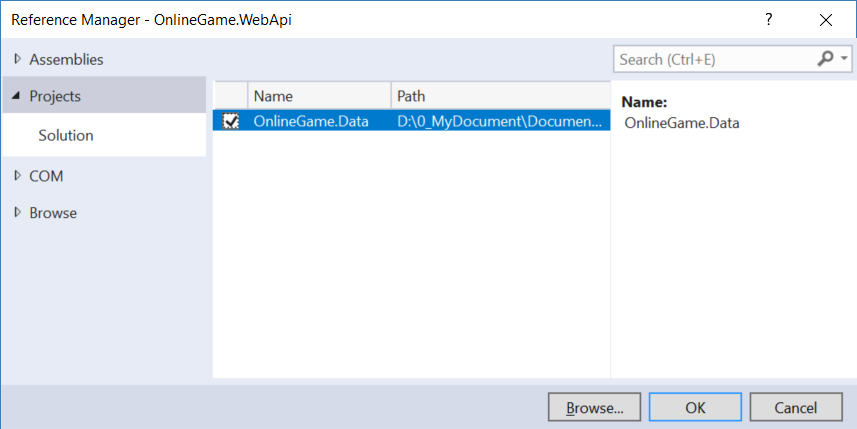


<connectionStrings>

    <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>

4.3. Add Reference



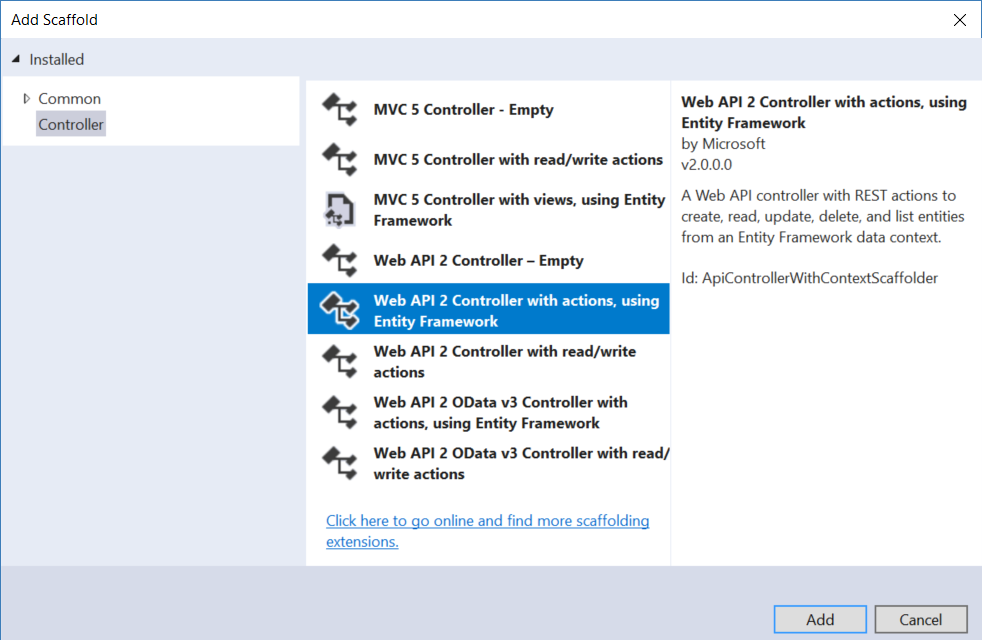
4.4. Controllers/Api/GamerController.cs

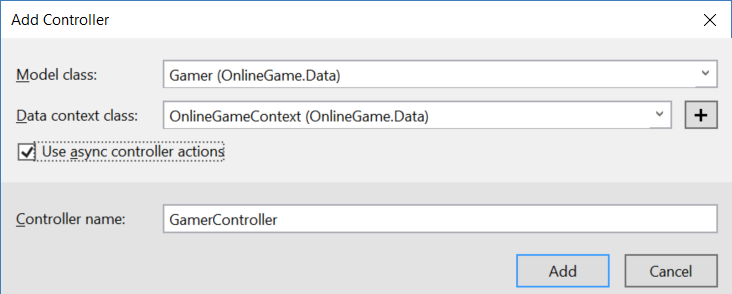
Controllers/Api  folder --> Right Click --> Add --> Controller

--> **Web API 2 Controller with actions, using Entity Framework**

--> **GamerController**

if you have any error message, please ensure re-build whole solutions.





using System.Collections.Generic;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

using System.Linq;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using OnlineGame.Data;

//using System.Web.Http.Cors;

//using OnlineGame.WebApi.WebShared;

namespace OnlineGame.WebApi.Controllers.Api

{

    //[EnableCors("\*", "\*", "\*")]

    //[EnableCors("[https://ithandyguytutorial.blogspot.com.au](https://ithandyguytutorial.blogspot.com.au/)", "\*", "\*")]

    //[EnableCors("[http://localhost:49804](http://localhost:49804/)", "\*", "\*")]

    //[HttpsAuthorizationFilter]

    public class GamerController : ApiController

    {

        private OnineGameContext \_db = new OnineGameContext();

        //// 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]

        [HttpGet]

        public async Task<IHttpActionResult> GetGamers(string gender = "all")

        {

            List<Gamer> gamers;

            switch (gender.ToLower())

            {

                case "all":

                    gamers = await \_db.Gamers.ToListAsync();

                    break;

                case "male":

                    gamers = await \_db.Gamers.Where(g => g.Gender.ToLower() == "male").ToListAsync();

                    break;

                case "female":

                    gamers = await \_db.Gamers.Where(g => g.Gender.ToLower() == "female").ToListAsync();

                    break;

                default:

                    return BadRequest($"{gender} is invalid. Gender must be male, female or all.");

            }

            return Ok(gamers);   //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](http://localhost:49804/),[https://ithandyguytutorial.blogspot.com.au](https://ithandyguytutorial.blogspot.com.au/)"

That means only [http://localhost:49804](http://localhost:49804/) and [https://ithandyguytutorial.blogspot.com.au](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](https://ithandyguytutorial.blogspot.com.au/)", "\*", "\*")]

//[EnableCors("[http://localhost:49804](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](https://ithandyguytutorial.blogspot.com.au/)", "\*", "\*")]

EnableCorsAttribute(origins, headers, methods)

It allows the resource to be accessed by [https://ithandyguytutorial.blogspot.com.au](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](http://localhost:49804/)", "\*", "\*")]

It allows the resource to be accessed by [http://localhost:49804](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());

HTTP will redirect to HTTPS request.

If you add HttpsAuthorizationFilterAttribute 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 [HttpsAuthorizationFilter] attribute at controller level or action level.

E.g.

//[HttpsAuthorizationFilter]

//public class GamerController : ApiController

...

//[HttpsAuthorizationFilter]

//public async Task<IHttpActionResult> GetGamers(string gender = "all")

\*/

5. Enable SSL (Secure Sockets Layer) and Create self-signed certificate

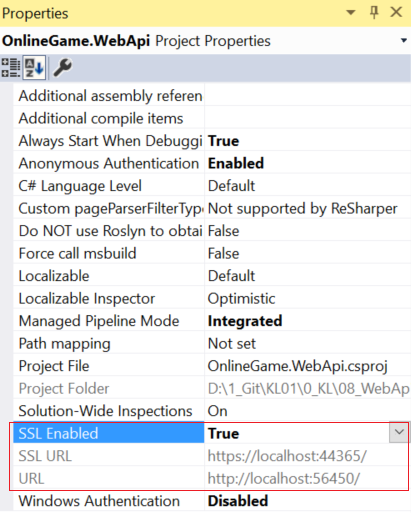
5.1. OnlineGame.WebApi Enable SSL via Visual Studio 2017

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

-->

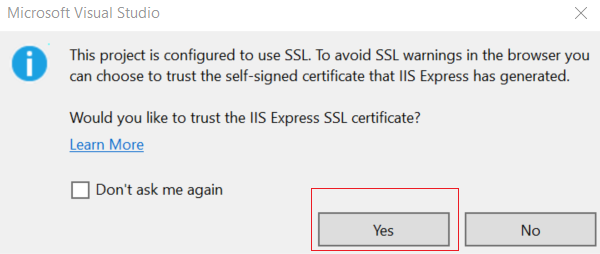
Would you like to trust the IIS Express SSL certificate.

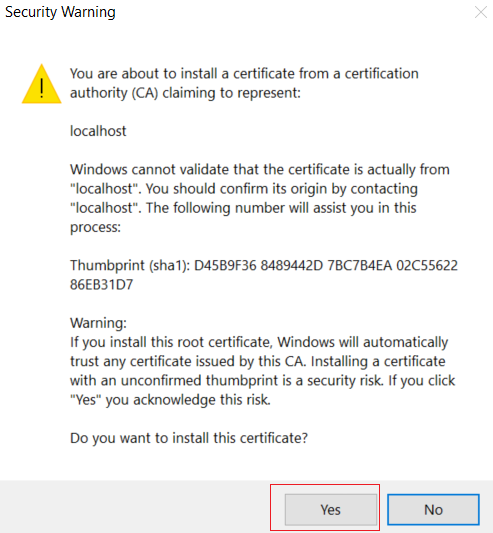
Yes

-->

Do you want to install this certificate?

Yes





[https:](https://localhost:44365/api/gamer)//localhost:[44365](https://localhost:44365/api/gamer)/api/gamer

Graphical user interface, text, application, email

Description automatically generated

[http](http://localhost:56450/api/gamer)://localhost:[56450](http://localhost:56450/api/gamer)/api/gamer



5.2. Manually place the certificate issued by visual studio into the Trusted Root Certificates folder

In previous steps,

Would you like to trust the IIS Express SSL certificate.

Yes

-->

Do you want to install this certificate?

Yes

If you choose "No", then you have to manually place the certificate issued by visual studio into the Trusted Root Certificates folder

Configure the certificate that Visual Studio issued to local and by local in the **Trusted Root Certificates** folder.

**Windows Key  + R -->**

**mmc.exe**

Open the Microsoft Management Console

Graphical user interface

Description automatically generated with medium confidence

File --> Add/Remove Snap-in... (snap-in 管理單元)

Graphical user interface, application

Description automatically generated

We are going to create a "Certificate" snap-in (憑證管理單元).

Thus, in the left list-box, select "Certificate", press "Add"

--> Computer account

--> Local computer

Graphical user interface, application

Description automatically generated

Reference:

<https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/service-accounts>

Here are more details regarding service account.

We want to create a **certificate** issued **by local computer**.

Thus, we select "**Computer account**"

Graphical user interface, text, application

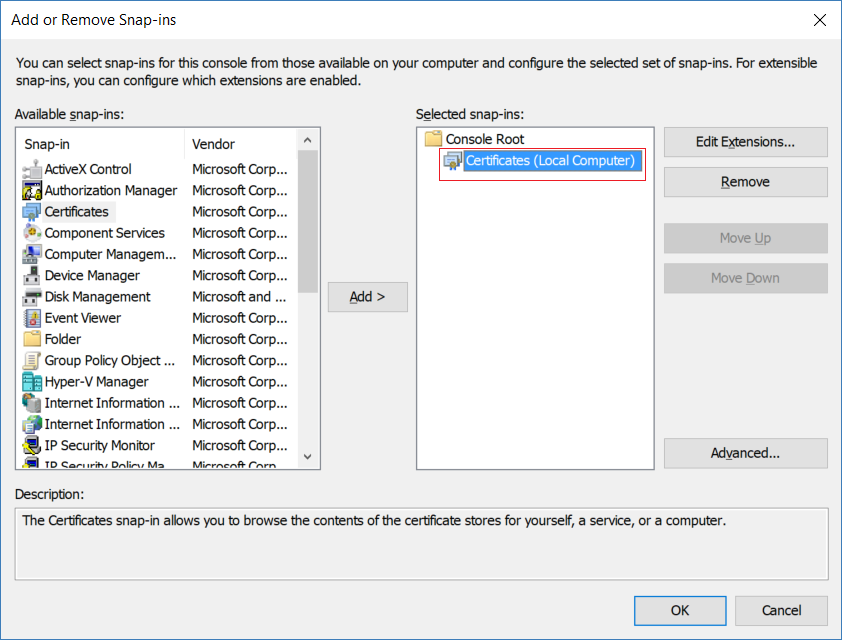
Description automatically generated

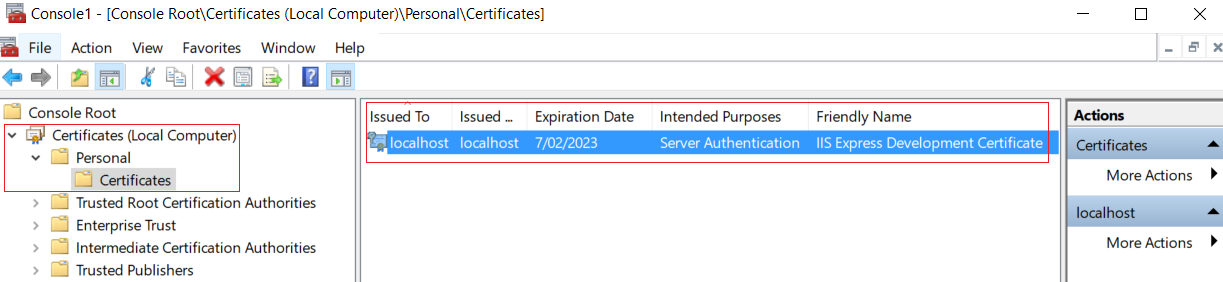
We want to create a **certificate** issued **to local computer**.

Thus, we select "**Local Computer**"

Graphical user interface, text, application, email

Description automatically generated





Console Root --> Certificates (Local Computer) --> Personal --> Certificates

You may find the **certificate** that we have just created.

The **certificate** is issued **by local computer** and **to local computer**.

In Next Step, we have to manually place the certificate issued by visual studio into the Trusted Root Certificates folder.

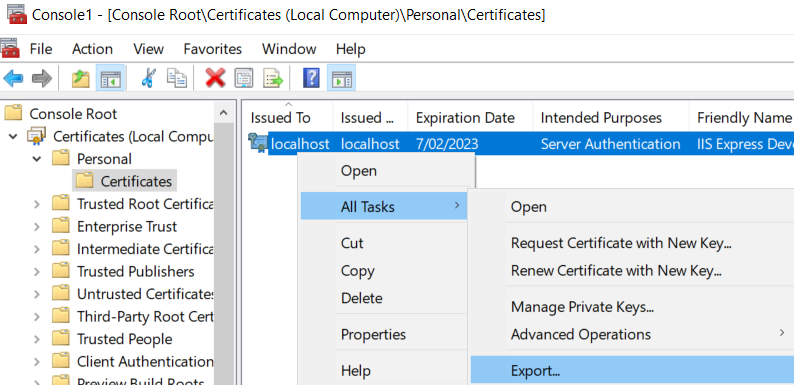
Firstly, we have to **export** the **certificate** your local computer folder.

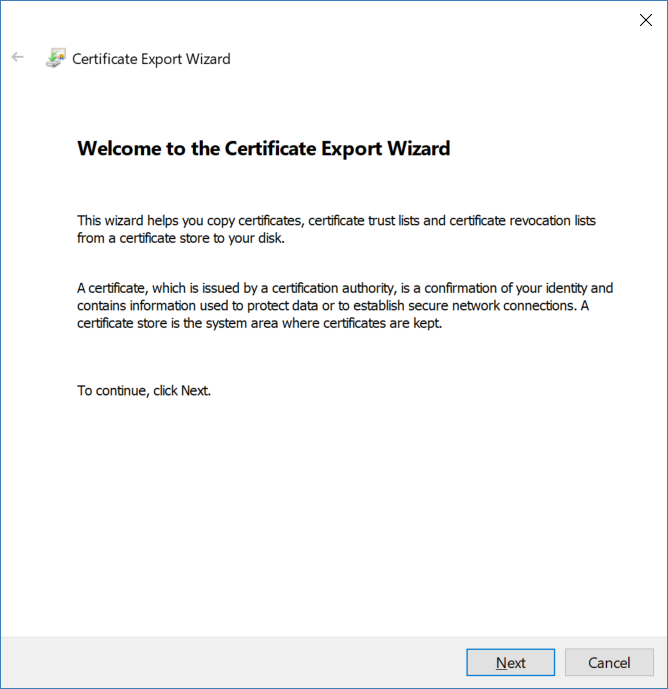
Secondly, we have to **import** that **certificate into Trusted Root Certificates folder**.

The Certificate --> Right Click --> All Tasks --> Export --> Next

-->

 No, do not export the private key.

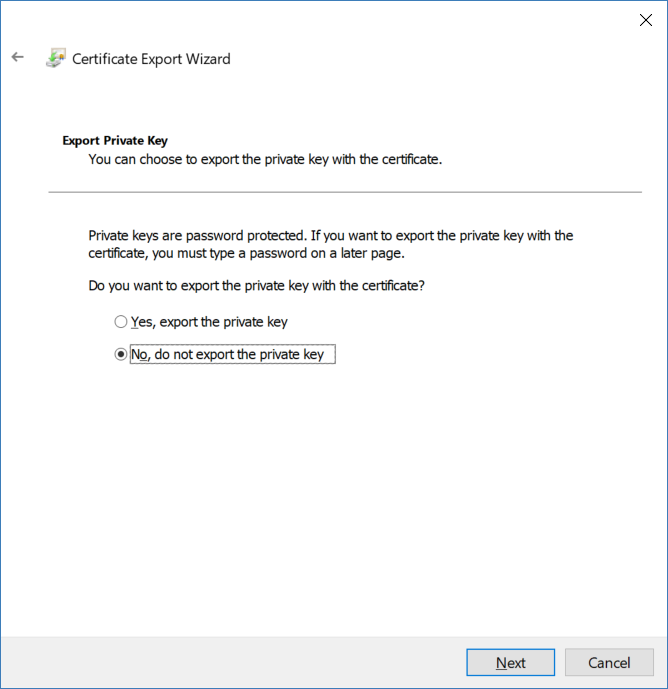




We don't want to expose my private key.

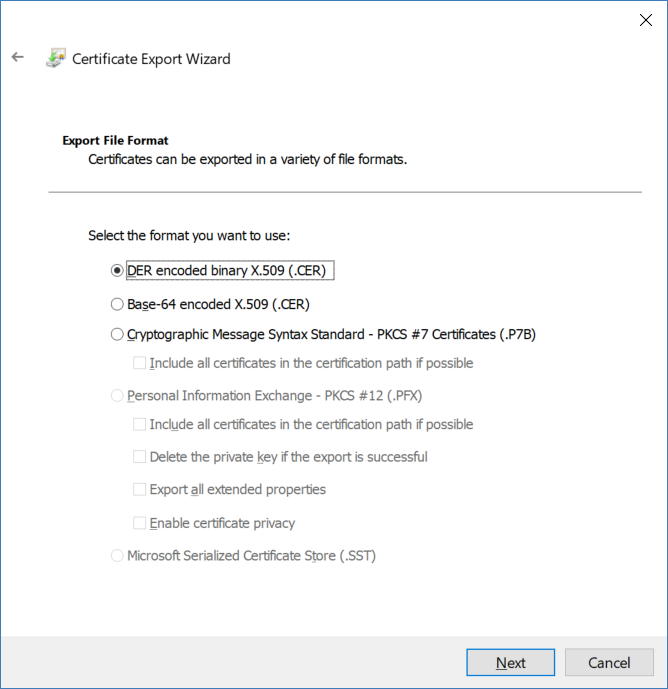
-->

**No, do not export the private key.**



We use the default format

**DER encoded binary X.509 (.CER)**



Where do you want to save the certificate?

-->

**D:\1\_TestCertificate\localhost.cer**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Secondly, we have to **import** that **certificate into Trusted Root Certificates folder**.

Console Root

--> Certificates (Local Computer)

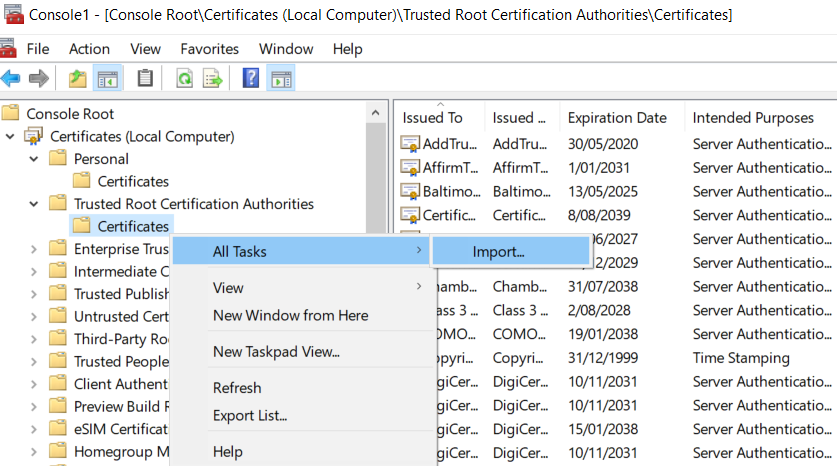
--> Trusted Root Certification Authorities

--> Certificates

--> All Tasks

--> Import

--> Next



Graphical user interface, text, application

Description automatically generated

Where is the certificate you want to import?

-->

**D:\1\_TestCertificate\localhost.cer**

-->

Next

Graphical user interface, text, application, email

Description automatically generated

What is the destination you want to import?

-->

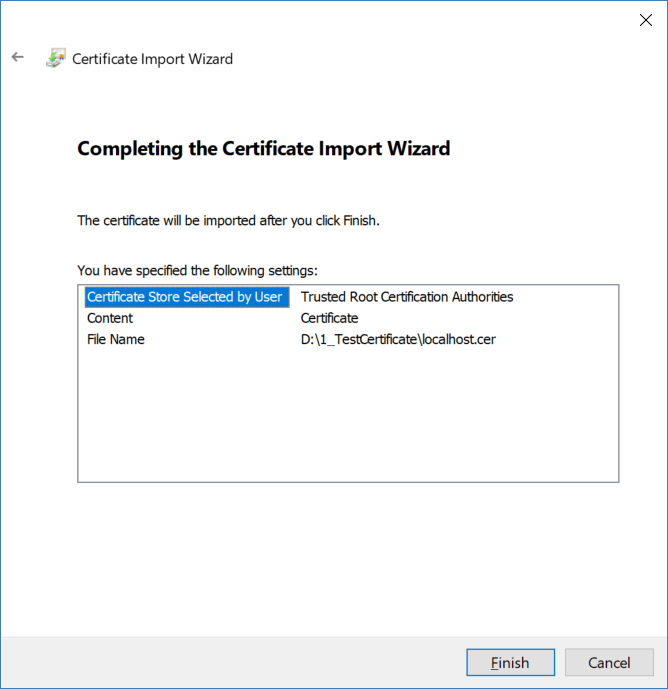
We want to import the certificate into the "**Trusted Root Certification Authorities**".

-->

Next

Graphical user interface, text, application

Description automatically generated



Graphical user interface, text, application, chat or text message

Description automatically generated

"**localhost**" is the name of the certificate which you have just imported into "**Trusted Root Certification Authorities**".

Graphical user interface, text, application, email

Description automatically generated

[https:](https://localhost:44365/api/gamer)//localhost:[44365](https://localhost:44365/api/gamer)/api/gamer

In Google Chrome

Graphical user interface, text, application, email

Description automatically generated

[https:](https://localhost:44365/api/gamer)//localhost:[44365](https://localhost:44365/api/gamer)/api/gamer

In Internet Explorer Edge

Text

Description automatically generated

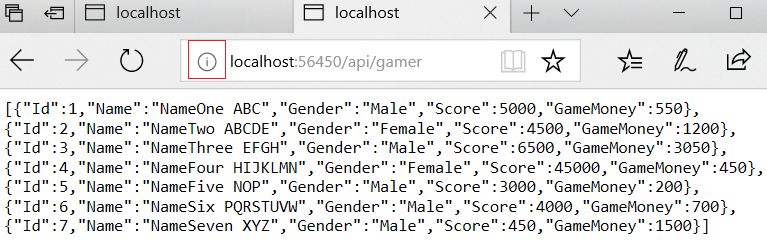
[http](http://localhost:56450/api/gamer)://localhost:[56450](http://localhost:56450/api/gamer)/api/gamer

In Google Chrome



[http](http://localhost:56450/api/gamer)://localhost:[56450](http://localhost:56450/api/gamer)/api/gamer

In Internet Explorer Edge



6. Enables CORS(Cross Origin Resource Sharing) and HTTP Redirect to HTTPS

6.1. Install WebApi Cors

**Install-Package** **Microsoft.AspNet.WebApi.Cors**

Graphical user interface, text, application

Description automatically generated

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 = 44365    //\*\*\*\*\*\*\*\*\*\*\*\*\*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

using System.Web.Http;

using System.Web.Http.Cors;

using OnlineGame.WebApi.WebShared;

//using WebApiContrib.Formatting.Jsonp;

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());

        }

    }

}

/\*

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](http://localhost:49804/),[https://ithandyguytutorial.blogspot.com.au](https://ithandyguytutorial.blogspot.com.au/)"

That means only [http://localhost:49804](http://localhost:49804/) and [https://ithandyguytutorial.blogspot.com.au](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](https://ithandyguytutorial.blogspot.com.au/)", "\*", "\*")]

//[EnableCors("[http://localhost:49804](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](https://ithandyguytutorial.blogspot.com.au/)", "\*", "\*")]

EnableCorsAttribute(origins, headers, methods)

It allows the resource to be accessed by [https://ithandyguytutorial.blogspot.com.au](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](http://localhost:49804/)", "\*", "\*")]

It allows the resource to be accessed by [http://localhost:49804](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());

HTTP will redirect to HTTPS request.

If you add HttpsAuthorizationFilterAttribute in WebApiConfig.cs,

it will apply to the entire application.

4.2.

If you don't want to apply to entire application.

You may apply [HttpsAuthorizationFilter] attribute in controller level or action level.

E.g.

//[HttpsAuthorizationFilter]

//public class GamerController : ApiController

...

//[HttpsAuthorizationFilter]

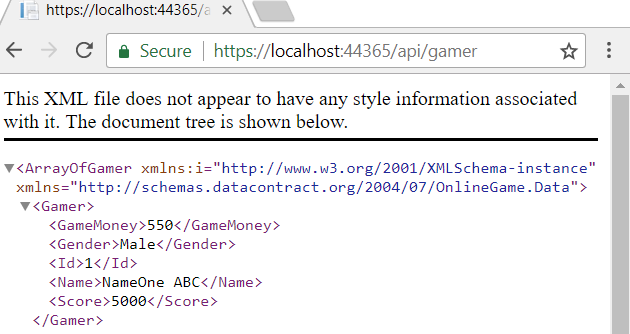
//public async Task<IHttpActionResult> GetGamers(string gender = "all")

\*/

[http](http://localhost:56450/api/gamer)://localhost:[56450](http://localhost:56450/api/gamer)/api/gamer

It will redirect to

[https](https://localhost:44365/api/gamer)://localhost:[44365](https://localhost:44365/api/gamer)/api/gamer



**Ctrl + Shift + I**

to see the Google Chrome Debug window

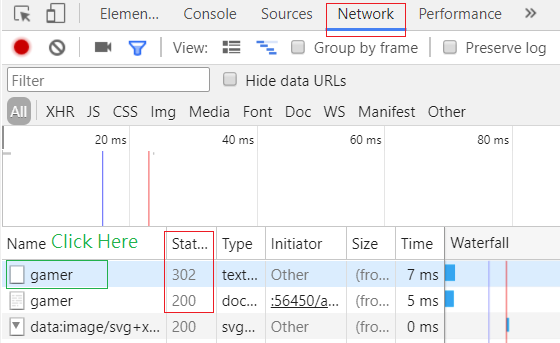
You may see the first gamer status code is 302/Found,

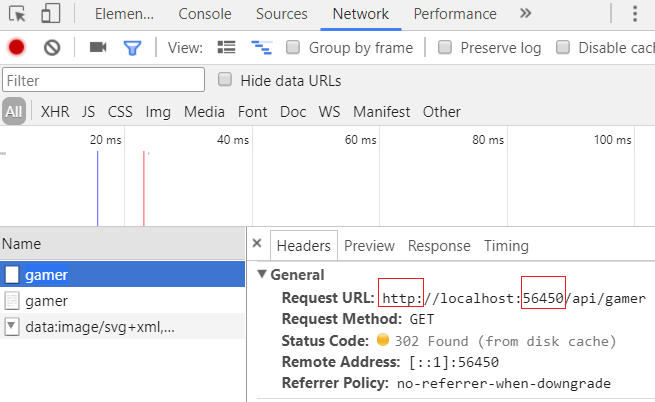
it is the HTTP request.

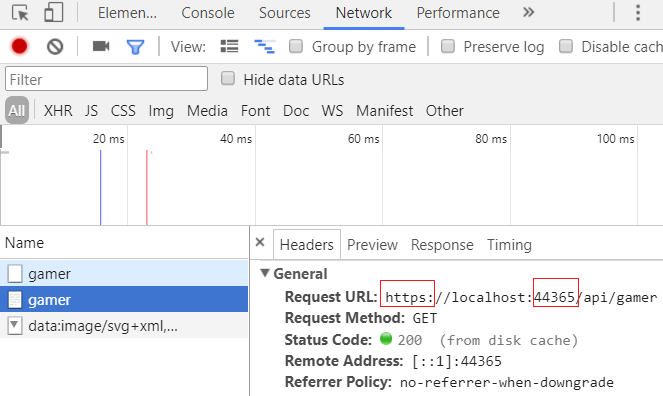
Then HTTP redirect to HTTPS to get the second gamer with status code 200/OK

-->

inspect the first gamer, and then inspect the second gamer.







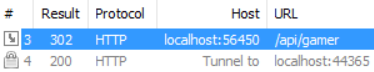
Let's inspect the **fiddler**

You may see the first gamer status code is 302/Found,

it is the HTTP request.

Then HTTP redirect to HTTPS to get the second gamer with status code 200/OK

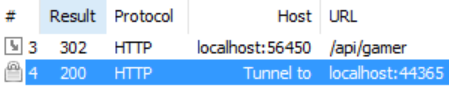
1.



-->



2.



-->

Graphical user interface, text, application, email

Description automatically generated

6.4. Decrypt HTTPS traffic via Fiddler

1.

Tools --> Options

--> HTTPS tab

-->

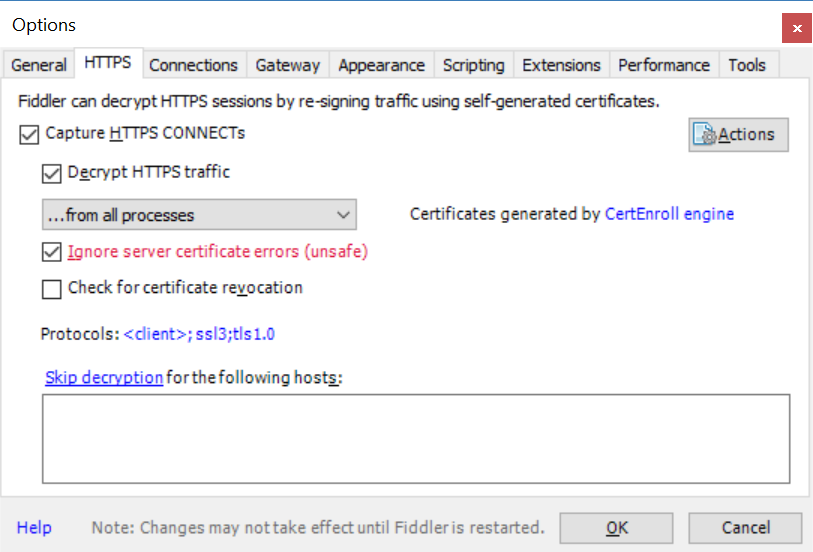
**Decrypt HTTPS traffic**

Select "**Ignore server certificate error(unsafe)**"

Graphical user interface, text, table

Description automatically generated

-->

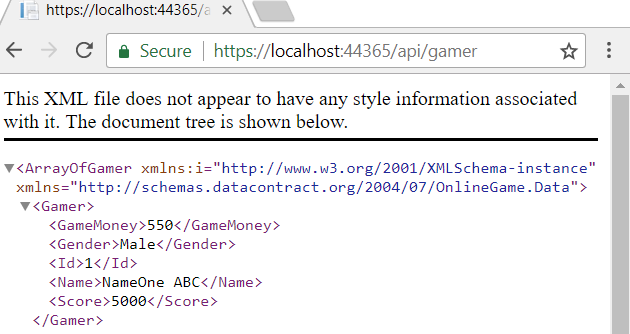


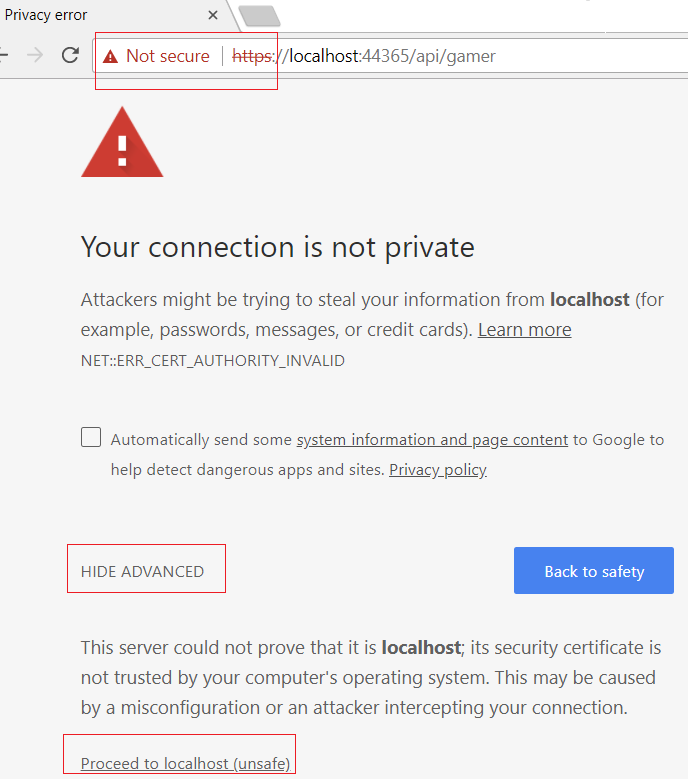
2.

<http://localhost:56450/api/gamer>

It will redirect to

<https://localhost:44365/api/gamer>





3.

Let's inspect the **fiddler**

You may see the first gamer status code is **302/Found**,

it is the HTTP request.

Then HTTP redirect to HTTPS to get the **second** gamer with status code **200/OK**

In the second request,

we can see the data, because the Fiddler can decrypt HTTPS traffic

Graphical user interface, application

Description automatically generated

-->

Graphical user interface, text, application, email

Description automatically generated