(T3)討論Api的HttpPost、HttpGet、HttpDelete屬性。討論Api的Put的HttpPut、FromBody、FromURI屬性  
CourseGUID 4c5822ff-7111-4e25-a336-ef18d48d54bd  
=======================================================================  
(T3)討論Api的HttpPost、HttpGet、HttpDelete屬性。討論Api的Put的HttpPut、FromBody、FromURI屬性

(T3-1)自動生成Api的Post、Get、Put、Delete對應到資料庫的CRUD

(T3-2)討論Api的Get的HttpGet屬性

(T3-3)討論Api的Post的HttpPost屬性

(T3-4)討論Api的Put的HttpPut、FromBody、FromURI屬性

(T3-5)討論Api的Delete的HttpDelete屬性  
=======================================================================  
1. OnlineGame DB

1.0. Some points

1.1. TSQL

1.2. Security login

-----------

2. OnlineGame Solution

2.1. OnlineGame Solution

2.2. OnlineGame.WebApi

2.3. OnlineGame.Data

-----------

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/GamerController.cs

4.5. Controllers/GamerController.cs

4.6. [FromBody] attribute and [FromUri] attribute  
=======================================================================

1. OnlineGame DB

The tutorial will discuss

Auto-generate the API with Get、Post、Put、Delete

and then Read, Insert, Update, Delete data from the database

About HttpGet、HttpPost、HttpPut、HttpDelete.

About FromBody and FromURI

----------------------------

本堂課討論

建立一個API with Get、Post、Put、Delete並且Read, Insert, Update, Delete data from the database。

關於HttpGet、HttpPost、HttpPut、HttpDelete四大屬性

關於FromBody和FromURI

1.0. Some points

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

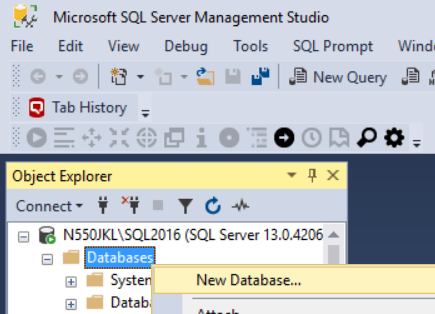
In SQL server Management Studio (SSMS)

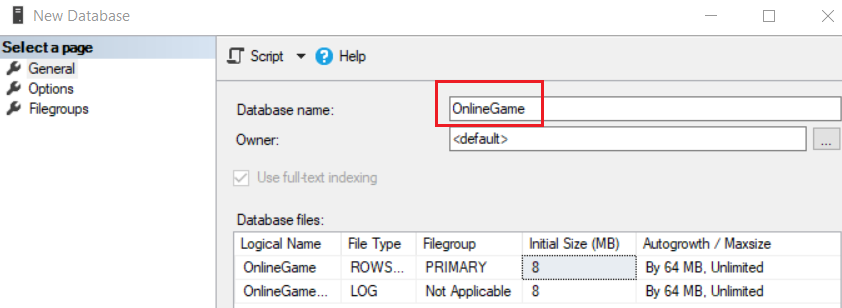
Database --> Right Click --> New Database -->

In General Tab -->

Name: **OnlineGame**

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





Graphical user interface, text, application

Description automatically generated

--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**

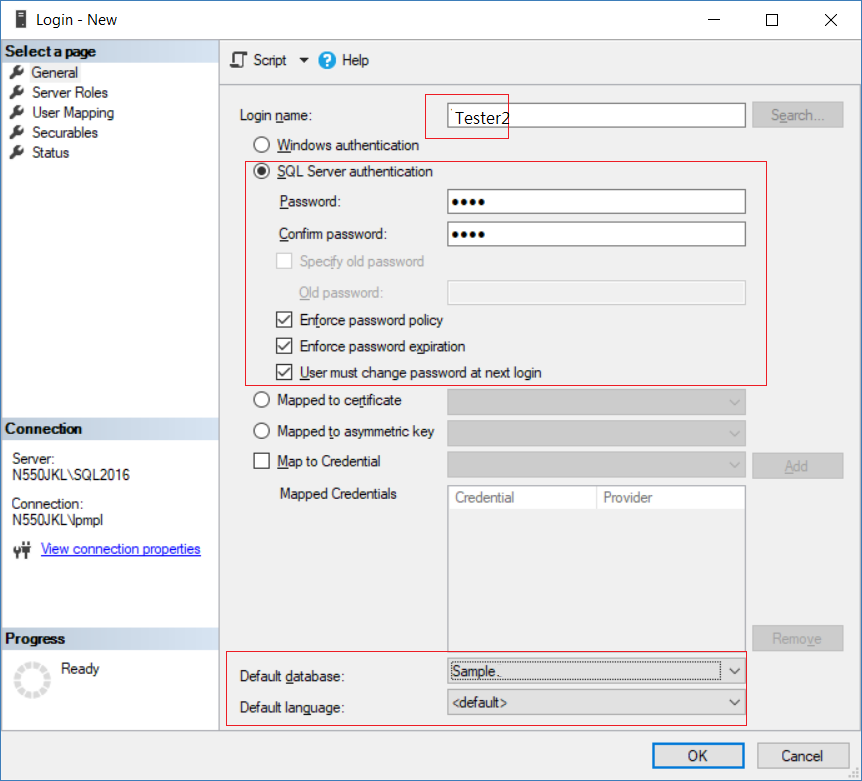
-->

User Mapping Tab

Select **OnlineGame**

Select every single role.









2. OnlineGame Solution

2.1. OnlineGame Solution

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

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

-->

Name: **OnlineGame**

2.2. OnlineGame.WebApi

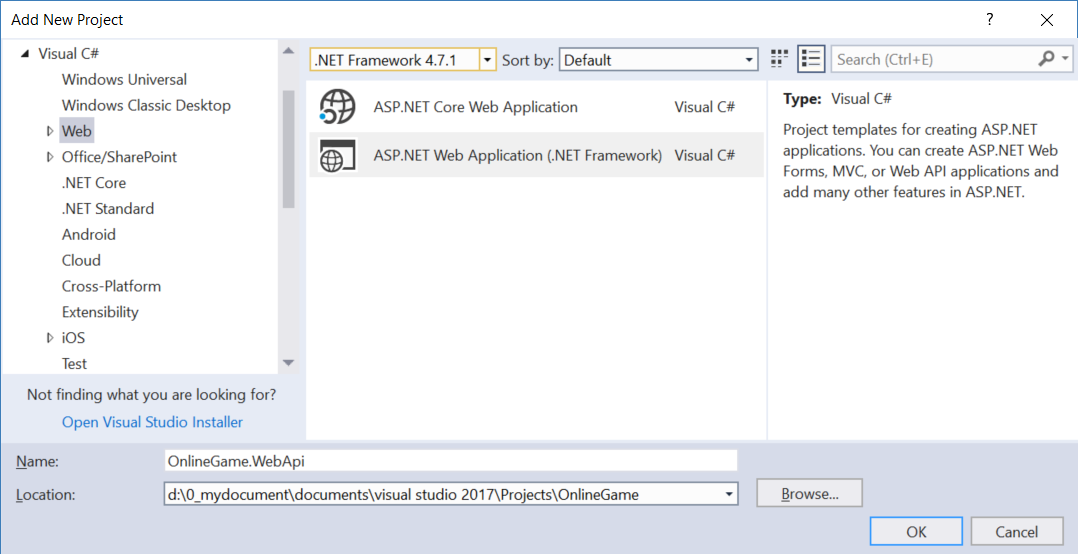
Solutions Name --> Add --> New Project -->

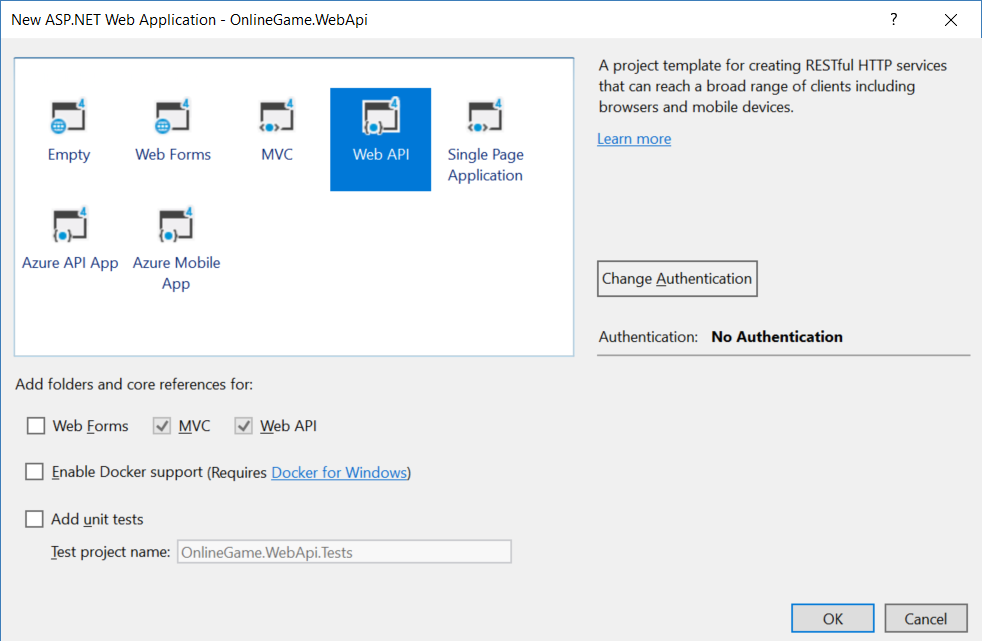
Visual C# --> Web --> [ASP.NET](http://asp.net/)Web Application (.Net Framework)

-->

Name: **OnlineGame.WebApi**

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





2.3. OnlineGame.Data

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

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

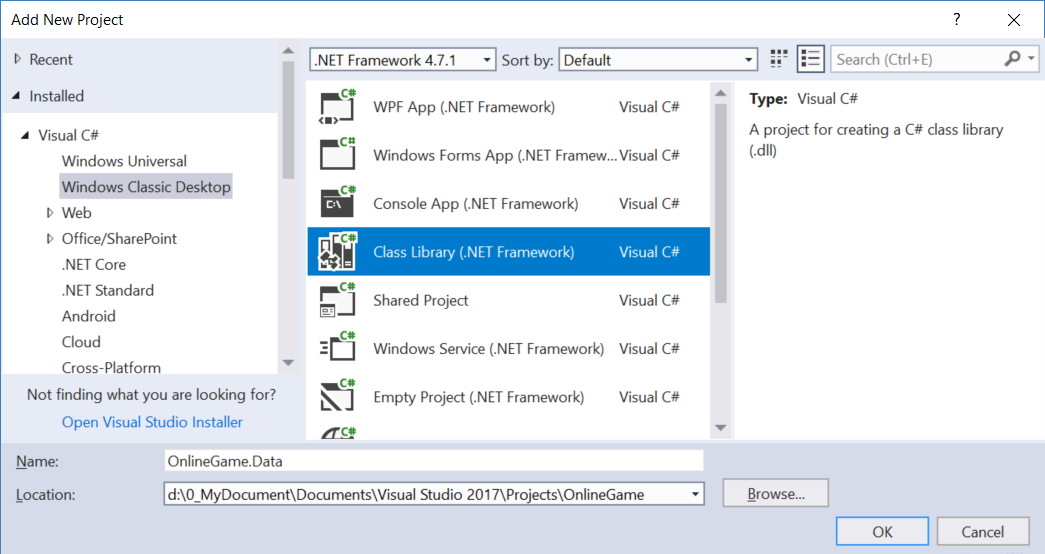
-->

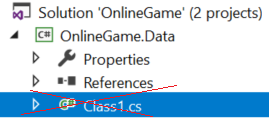
Name:

**OnlineGame.Data**

-->

Delete Class1.cs





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**

-->

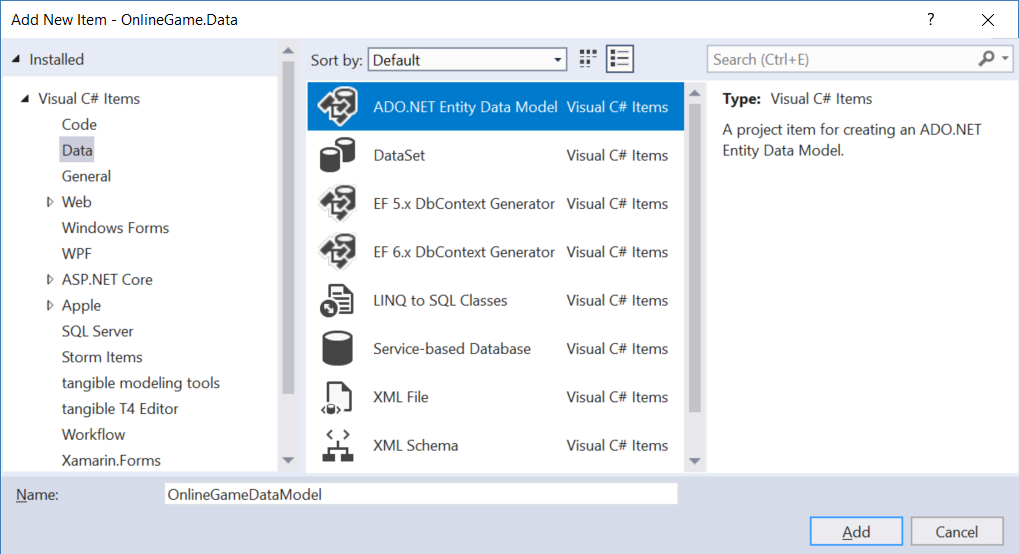
EF Designer from database

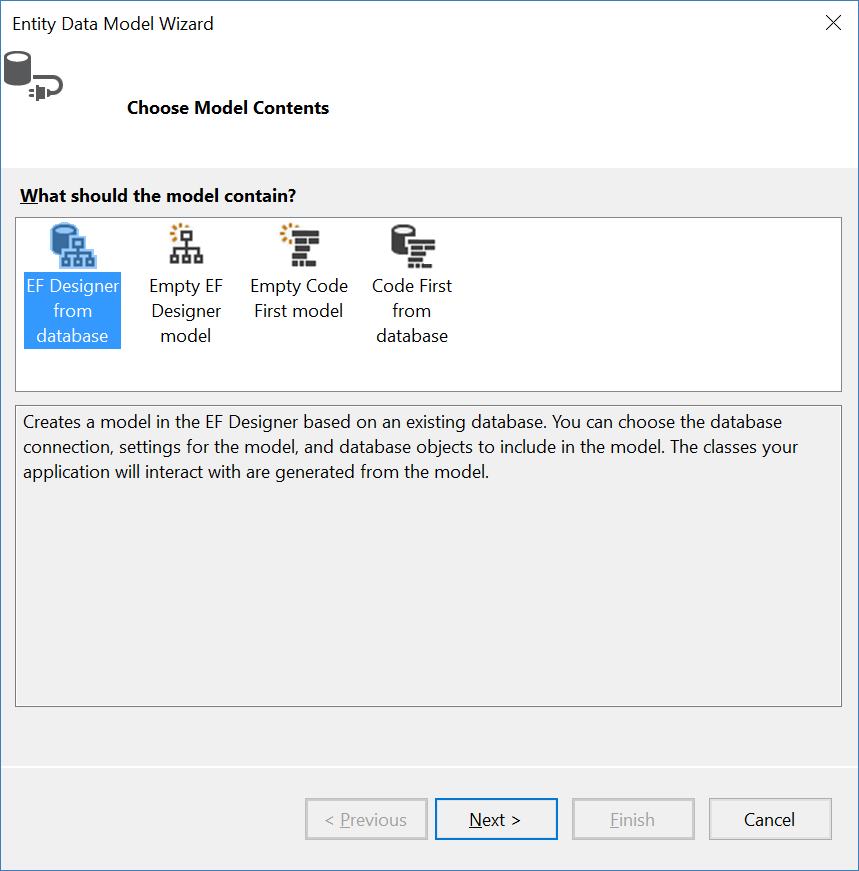
....

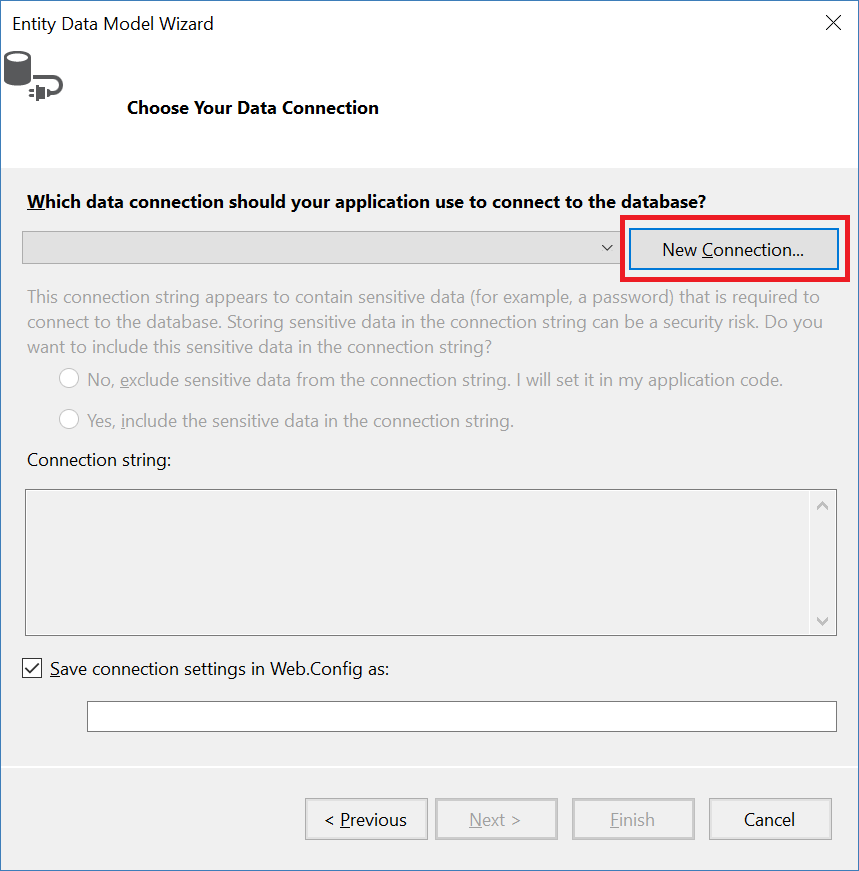
-->

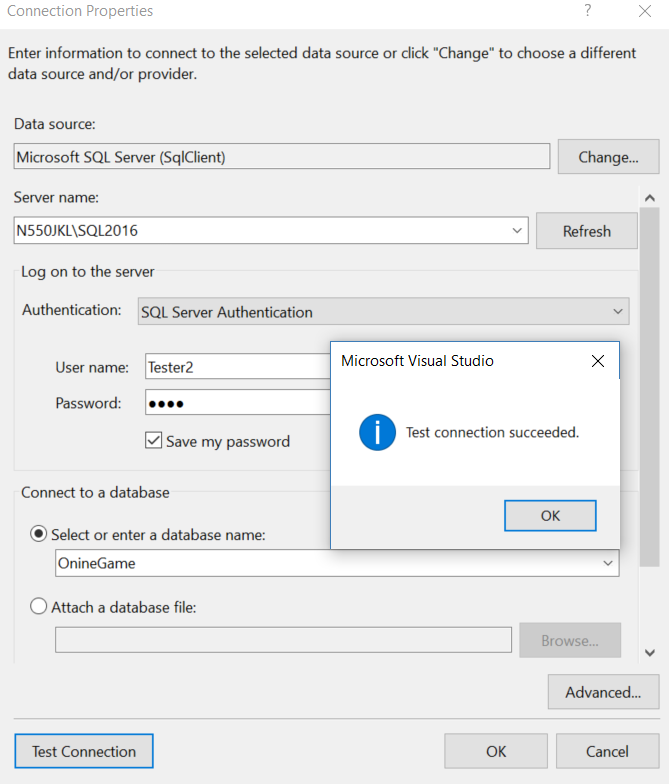
Save Connection settings in Web.Config as:

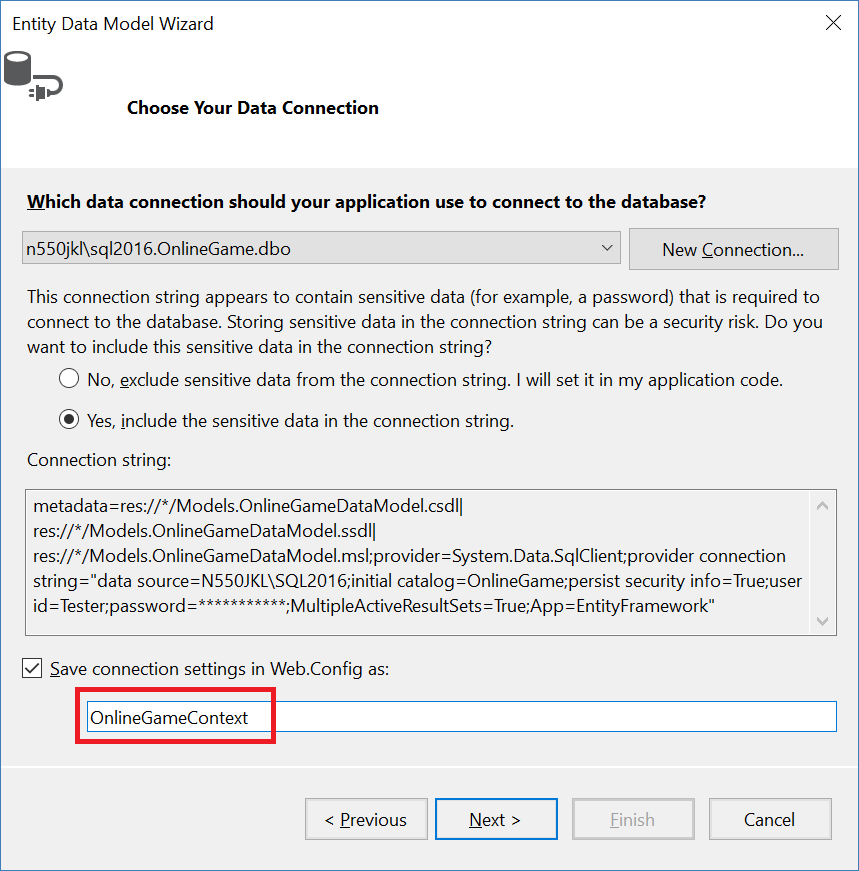
**OnlineGameContext**

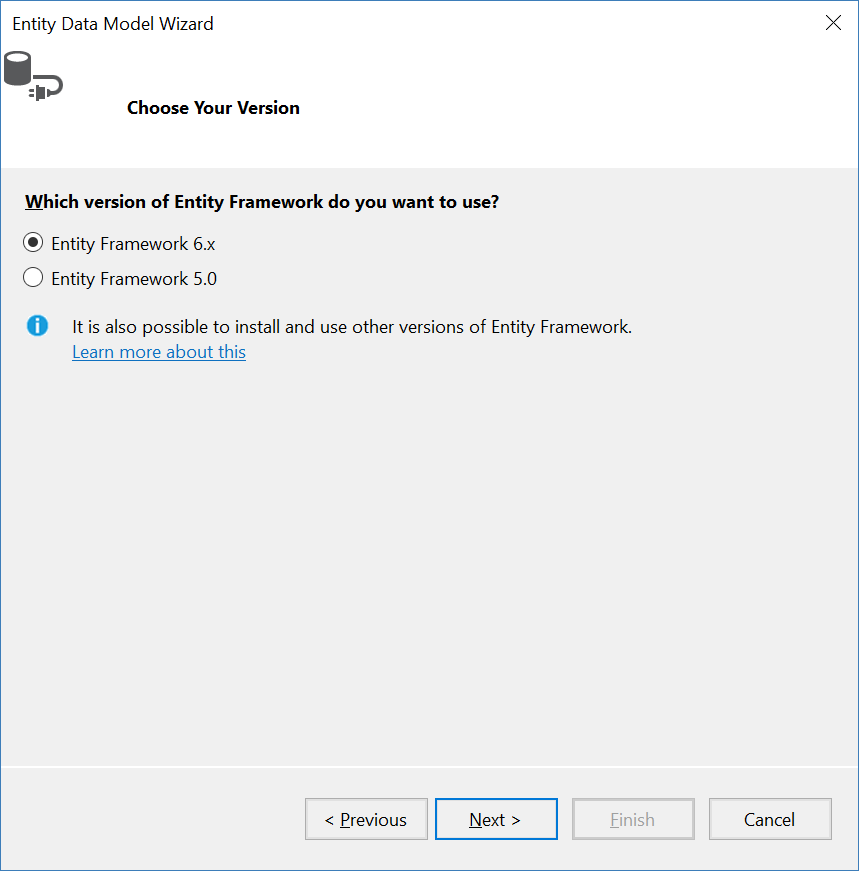


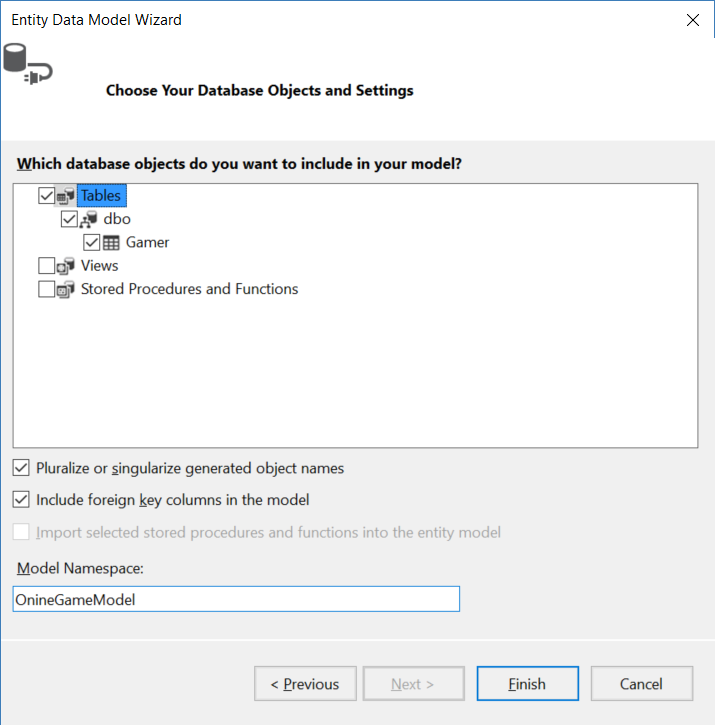


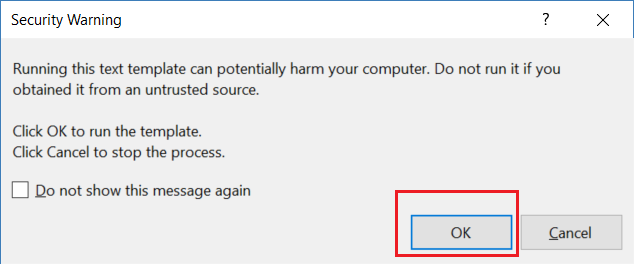


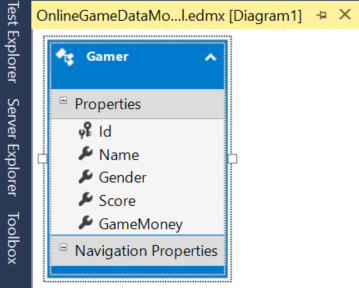












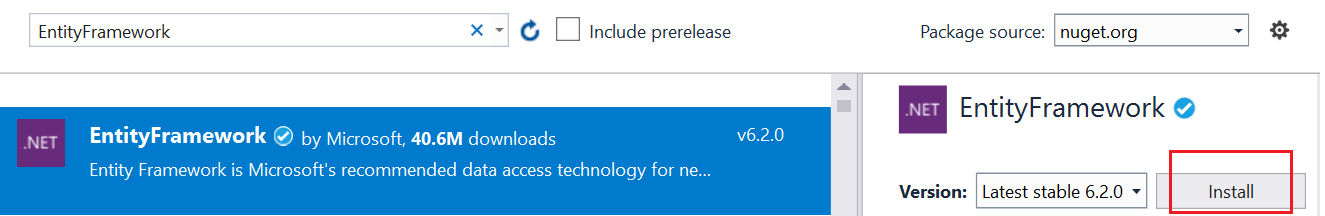
4. OnlineGame.WebApi

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

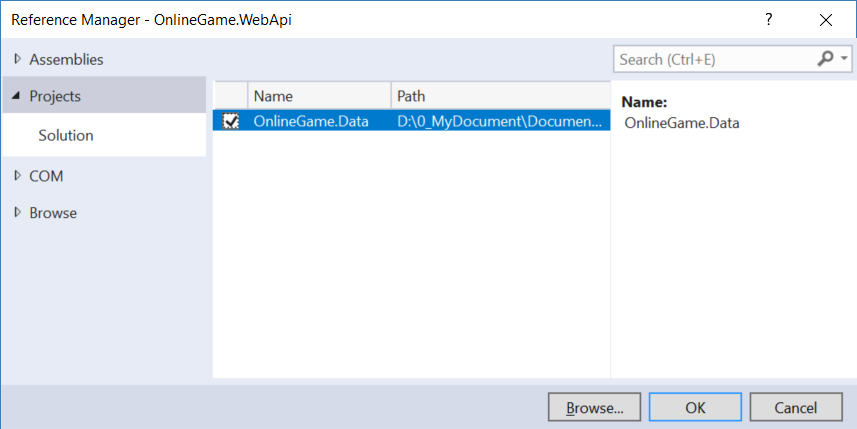


<connectionStrings>

    <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



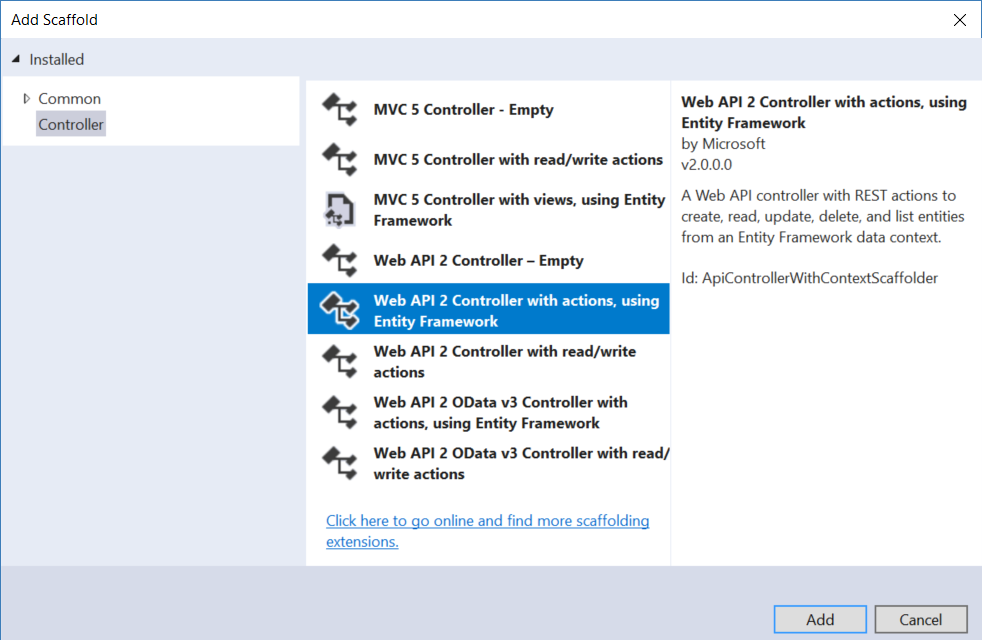
4.4. Controllers/GamerController.cs

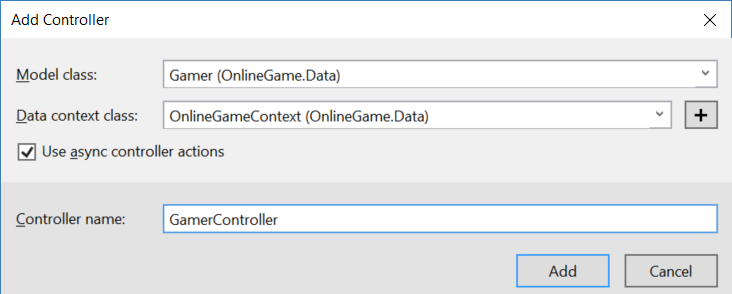
Controllers  folder --> Right Click --> Add --> Controller

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

--> **GamerController**

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





4.5. Controllers/GamerController.cs

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using OnlineGame.Data;

namespace OnlineGame.WebApi.Controllers

{

    public class GamerController : ApiController

    {

        private OnlineGameContext \_db = new OnlineGameContext();

        ////GET: api/Gamer

        //[HttpGet]

        //public IQueryable<Gamer> LoadGamers()

        ////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

        [HttpGet]

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

        //public IQueryable<Gamer> GetGamers()

        {

            List<Gamer> gamers;

            switch (gender.ToLower())

            {

                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:

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

                    break;

            }

            return Ok(gamers);   //200

        }

        // GET: api/Gamer/5

        [ResponseType(typeof(Gamer))]

        [HttpGet]

        public async Task<IHttpActionResult> LoadGamer(int id)

        //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))]

        //public async Task<IHttpActionResult> PutGamer(int id, Gamer gamer)

        [HttpPut]

        //public async Task<IHttpActionResult> UpdateGamer(int id, Gamer gamer)

        public async Task<IHttpActionResult> UpdateGamer([FromUri]int id, [FromBody]Gamer gamer)    //By Default

        //public async Task<IHttpActionResult> UpdateGamer([FromBody]int id, [FromUri]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> InsertGamer([FromBody]Gamer gamer)

        //public async Task<IHttpActionResult> PostGamer([FromBody]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

            ////Return Created/201.

            ////2.

            ////If you want to return HttpResponseMessage()

            ////2.

            ////Create a HttpResponseMessage with status code 201 Item Created.

            ////Pass the gamer into 2nd parameter as the created value.

            //HttpResponseMessage message =

            //    Request.CreateResponse(HttpStatusCode.Created, gamer);

            ////The Headers.Location should know the URI of the created item.

            //message.Headers.Location = new Uri(Request.RequestUri +

            //    gamer.Id.ToString());

            //return message;   //Created/201

            ////Return OK/200.

            ////3.

            ////if you want to return OK/200 when item created.

            //return Created(new Uri(Request.RequestUri + gamer.Id.ToString()), gamer);    //OK/200

        }

        // DELETE: api/Gamer/5

        [ResponseType(typeof(Gamer))]

        //[HttpDelete]

        //public async Task<IHttpActionResult> RemoveGamer(Gamer gamer)

        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.

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

\*/

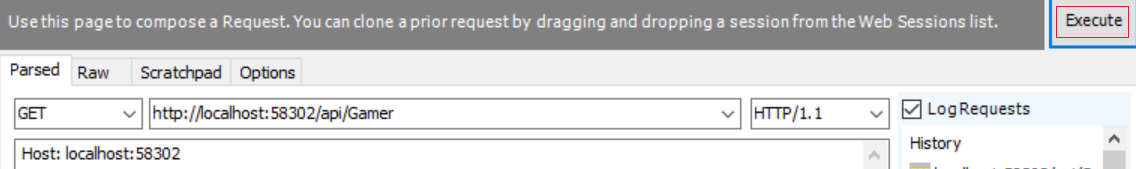
--------------------------------------------------------------------------

E.g.1.1.

Get

<http://localhost:58302/api/Gamer>

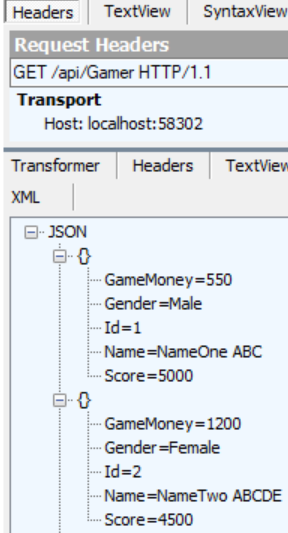
**Host: localhost:58302**



-->



-->

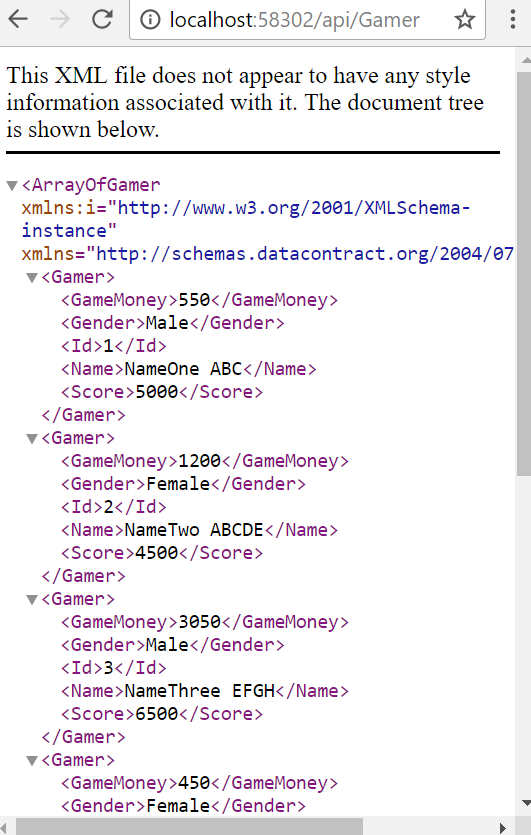


------------------------------------------------------------

E.g.1.2.

Get

<http://localhost:58302/api/Gamer>

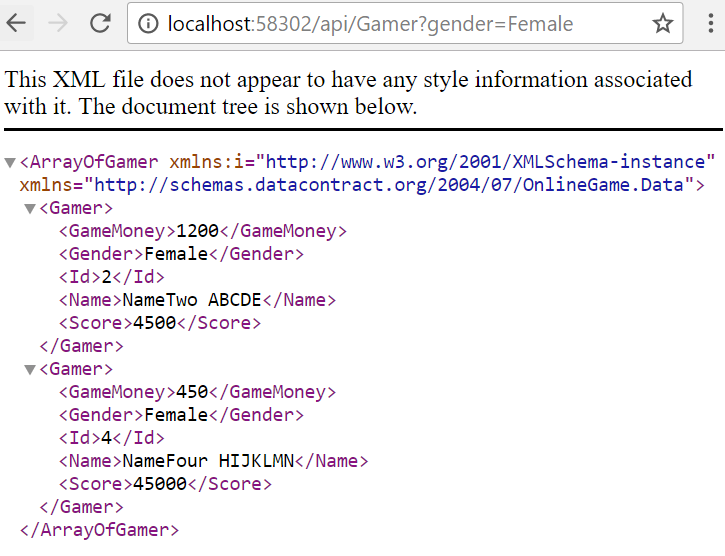


------------------------------------------------------------

E.g.1.3.

Get

<http://localhost:58302/api/Gamer?gender=Female>

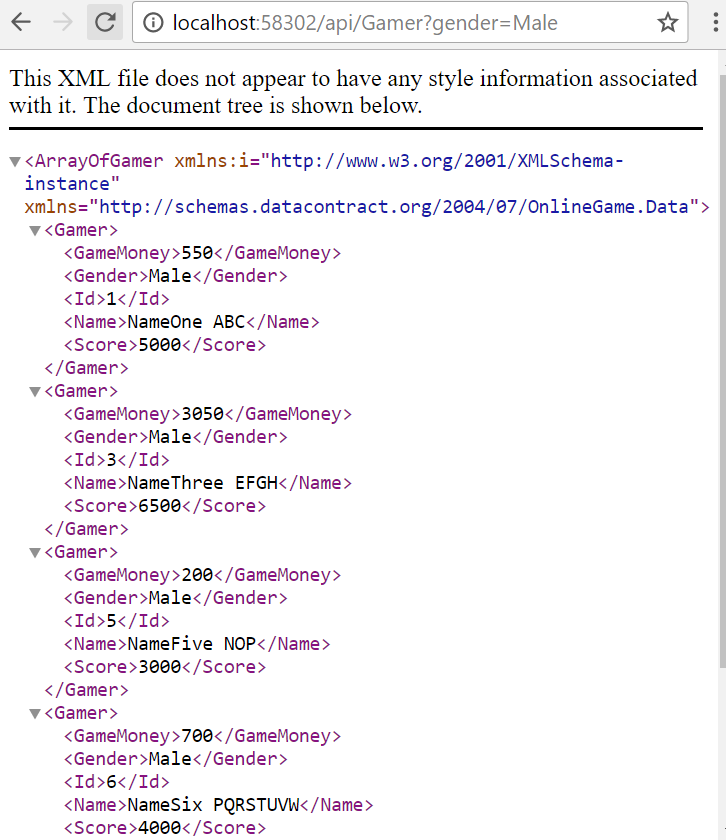


------------------------------------------------------------

E.g.1.4.

Get

<http://localhost:58302/api/Gamer?gender=Male>



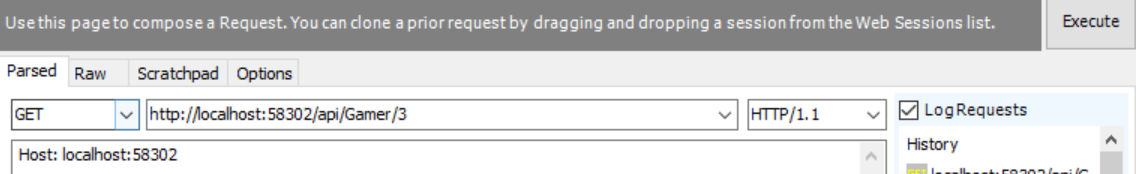
-----------------------------------------------

E.g.2.

Get(int id)

<http://localhost:58302/api/Gamer/3>

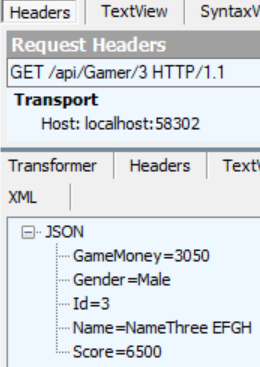
**Host: localhost:58302**



-->



-->



-----------------------------------------------

E.g.3.

A.

Post

<http://localhost:58302/api/Gamer>

B.

Request Header

**Host: localhost:58302**

**Accept: application/json**

**Content-Type: application/json**

B.1.

Accept: application/json

means we request JSON format response.

B.2.

Content-Type: application/json

Content-Length: 80

The client will post a data to the server, the data format is JSON and length is 80.

C.

Request Body

**{**

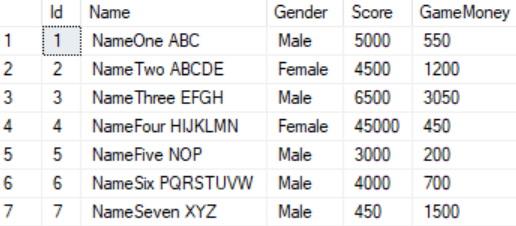
**"Name":"NameEight XYZ",**

**"Gender":"Male",**

**"Score":450,**

**"GameMoney":1500**

**}**



-->

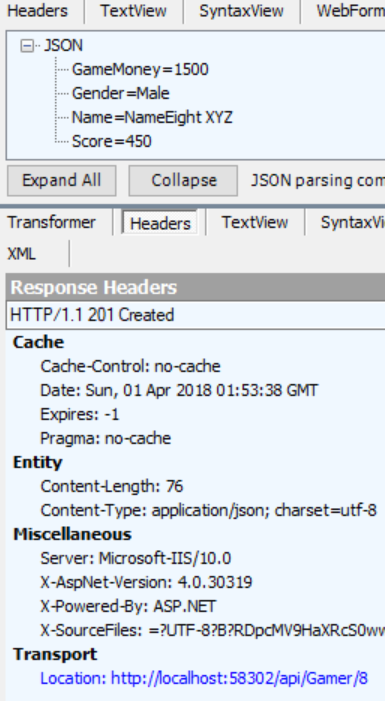
Graphical user interface, text, application, email

Description automatically generated

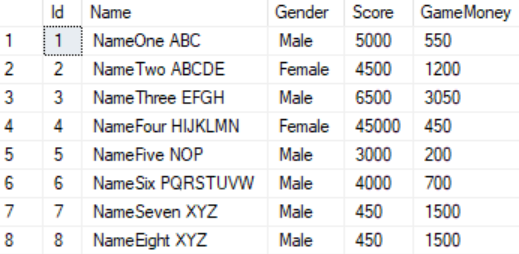
-->



-->



-->



-----------------------------------------------

E.g.4.

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

Content-Length: 80

The client will post a data to the server, the data format is JSON and length is 80.

C.

Request Body

**{**

**"Name":"NameEight XYZ222",**

**"Gender":"Male",**

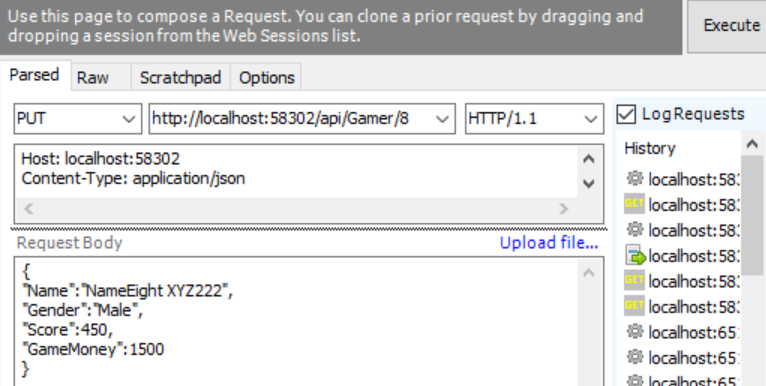
**"Score":450,**

**"GameMoney":1500**

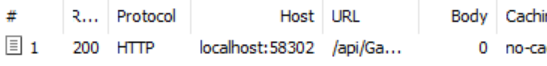
**}**



-->



-->



-->

Graphical user interface, text, application, email

Description automatically generated

-->



-----------------------------------------------

A.

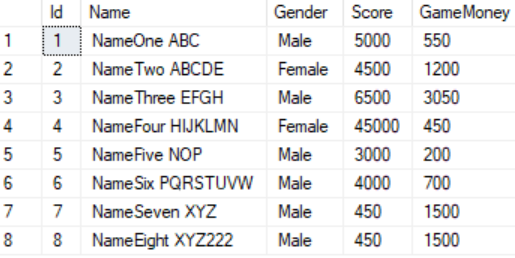
Delete

<http://localhost:58302/api/Gamer/8>

B.

Request Header

**Host: localhost:58302**



-->

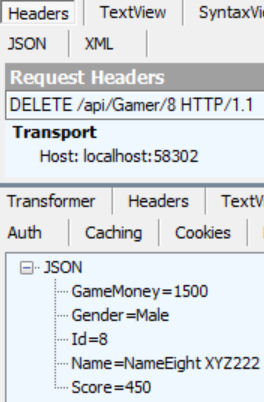
Graphical user interface, text, application

Description automatically generated

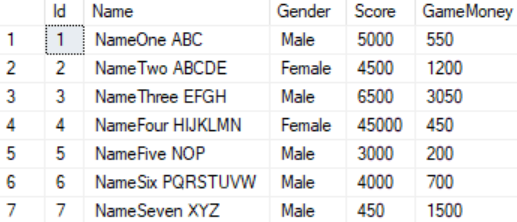
-->



-->



-->



4.6. [FromBody] attribute and [FromUri] attribute

// PUT: api/Gamer/5

[ResponseType(typeof(void))]

[HttpPut]

//public async Task<IHttpActionResult> PutGamer(int id, Gamer gamer)

//public async Task<IHttpActionResult> UpdateGamer(int id, Gamer gamer)

//public async Task<IHttpActionResult> UpdateGamer([FromUri]int id, [FromBody]Gamer gamer)    //By Default

public async Task<IHttpActionResult> UpdateGamer([FromBody]int id, [FromUri]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;

    }

}

2.

Web Api default binding parameter convention

2.1.

By default, if the parameter is 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 complex type,

Web Api will try to get value from 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/8?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.

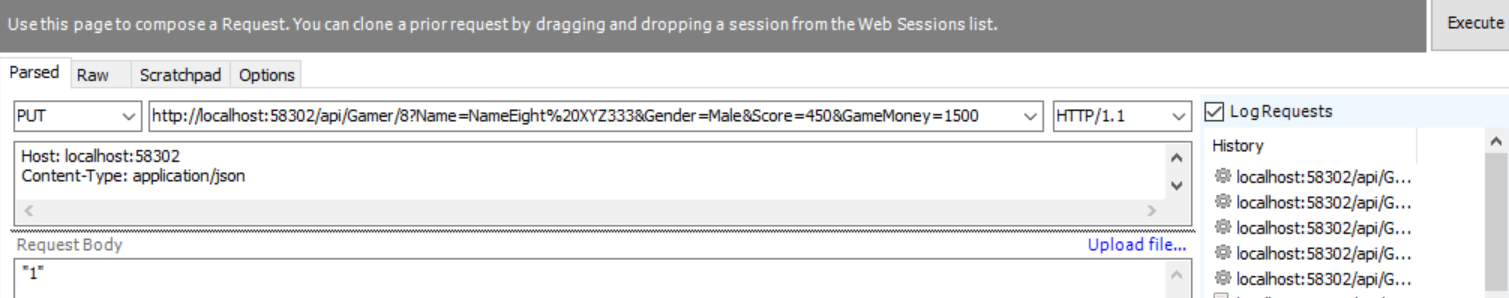
Content-Type: application/json

The client will post a data to the server, the data format is JSON

C.

Request Body

**"1"**



-->

