(T4)在不同的origin透過Json、Padding和EnableCors(CrossOriginResourceSharing)的方式用Ajax呼叫Api  
CourseGUID 4c5822ff-7111-4e25-a336-ef18d48d54bd  
=======================================================================  
(T4)在不同的origin透過Json、Padding和EnableCors(CrossOriginResourceSharing)的方式用Ajax呼叫Api

(T4-1)自動生成ApiController、MvcController。用Ajax呼叫Api

(T4-2)討論ApiController的Get、Put

(T4-3)討論ApiController的Post、Delete

(T4-4)討論MvcController

(T4-5)在相同的origin用Ajax呼叫Api

(T4-6)討論在不同的origin用Ajax呼叫Api的問題

(T4-7)在不同的origin透過Json、Padding的方式用Ajax呼叫Api

(T4-8)在不同的origin透過EnableCors(CrossOriginResourceSharing)的方式用Ajax呼叫Api  
=======================================================================  
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. Web.config : Add Connection String

4.3. Add Reference

4.4. Controllers/Api/GamerController.cs

4.5. Controllers/GamerController.cs

4.6. Views/Gamer/Index2.cshtml - Jquery AJAX call Web API

-----------

5. OnlineGame.Mvc

5.1. Jquery AJAX may call Web API in the same origin

5.2. Install Entity Framework

5.3. Web.config : Add Connection String

5.4. Controllers/GamerController.cs

5.5. Views/Gamer/IndexWebApi.cshtml

-----------

6. OnlineGame.Mvc

6.1. JSONP allows Jquery AJAX may call Web API in the different origins

6.2. Install JSONP

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

6.4. OnlineGame.Mvc/Views/Gamer/IndexWebApiJsonp.cshtml

6.5. OnlineGame.WebApi/Views/Gamer/Index2.cshtml

6.6. Fiddler test Jsonp

-----------

7. OnlineGame.Mvc

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

7.2. Install WebApi Cors

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

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

7.5. OnlineGame.WebApi/Views/Gamer/Index2.cshtml

7.6. OnlineGame.Mvc/Views/Gamer/IndexWebApiCors.cshtml

7.7. Fiddler test CORS (Cross Origin Resource Sharing)  
=======================================================================

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

Jquery AJAX call Web API in the different origins

JSONP(JSON with Padding) allows Jquery AJAX call Web API in the different origins

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

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

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

Jquery AJAX呼叫Web API在相同的origin

使用JSONP (JSON with Padding)讓Jquery AJAX呼叫Web API在不同的origin

使用Enable CORS (Cross Origin Resource Sharing)讓Jquery AJAX呼叫Web API在不同的origin

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

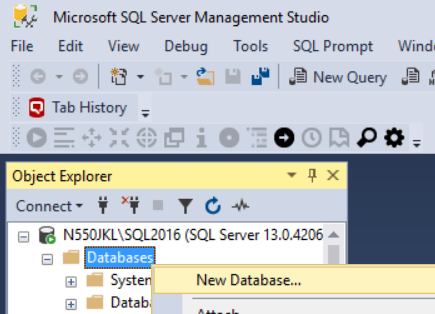
In SQL server Management Studio (SSMS)

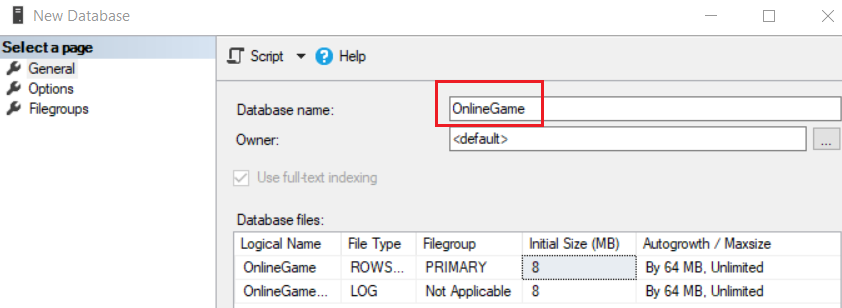
Database --> Right Click --> New Database -->

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**

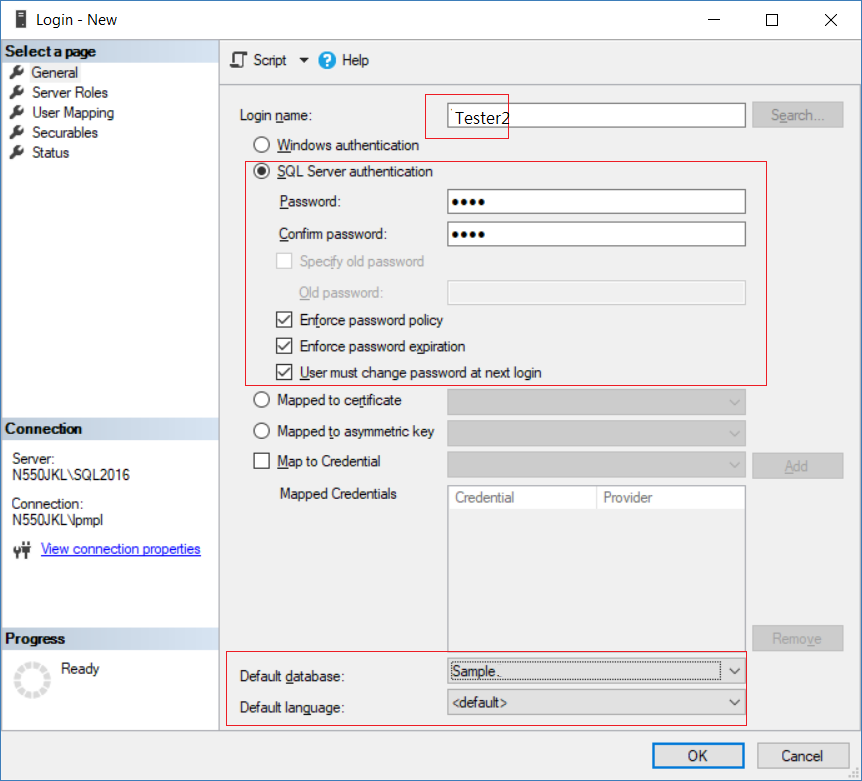
-->

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.Data

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

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

-->

Name:

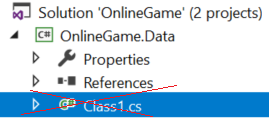
**OnlineGame.Data**

-->

Delete Class1.cs

Graphical user interface, application

Description automatically generated



2.3. OnlineGame.WebApi

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

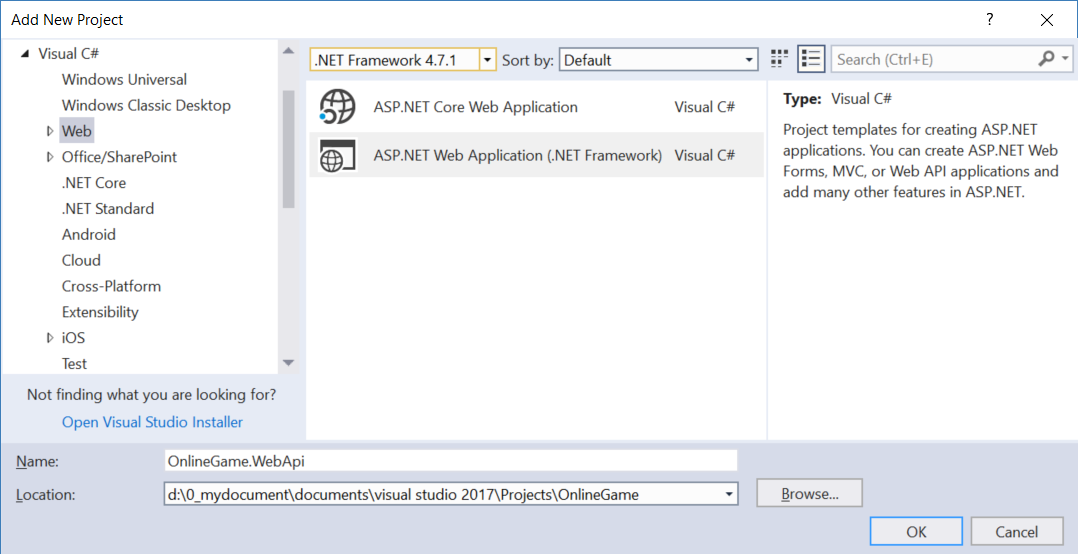
Visual C# --> Web --> [ASP.NET](http://asp.net/)Web Application (.Net Framework)

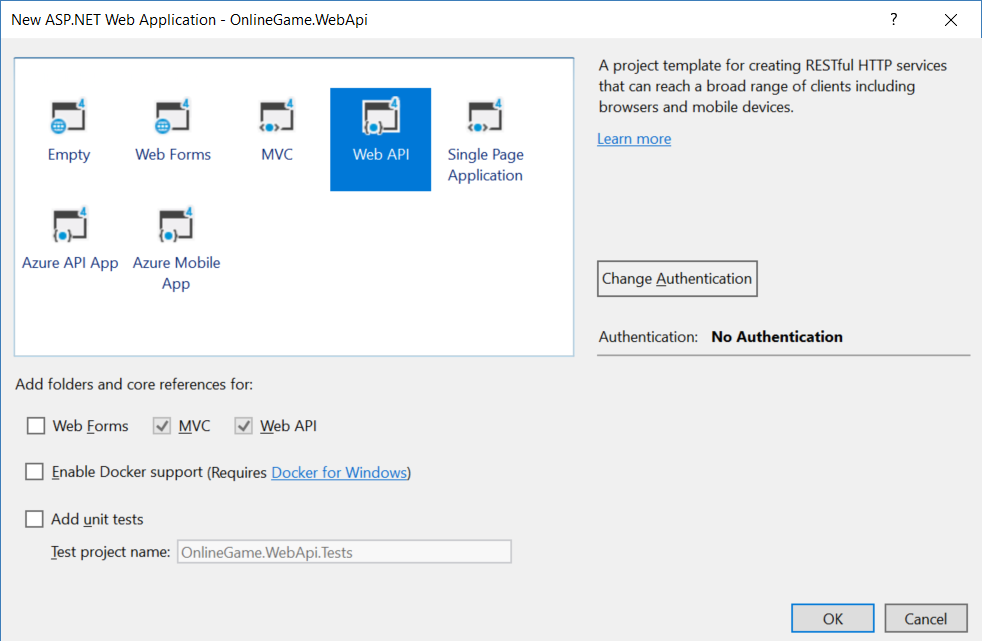
-->

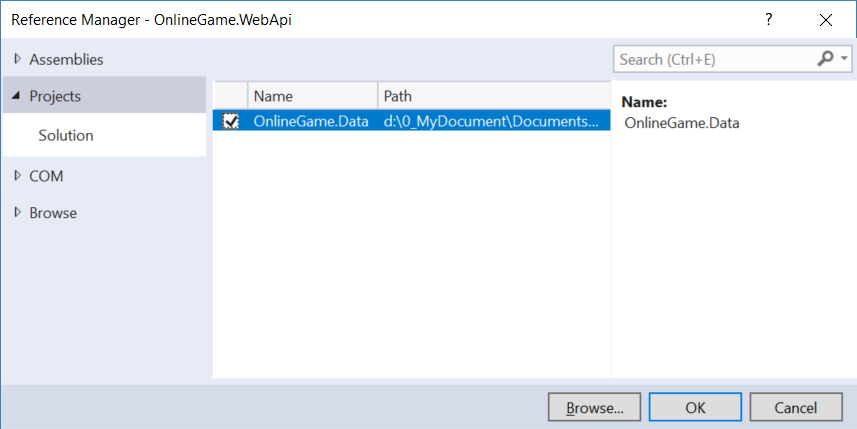
Name: **OnlineGame.WebApi**

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

--> Add Reference







2.4. OnlineGame.Mvc

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

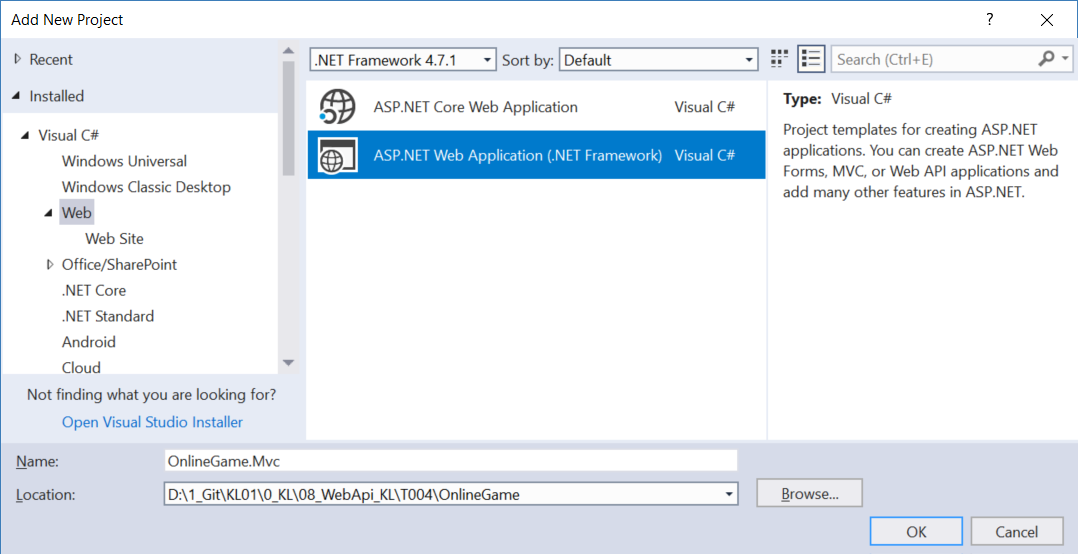
Visual C# --> Web --> [ASP.NET](http://asp.net/)Web Application (.Net Framework)

-->

Name: **OnlineGame.Mvc**

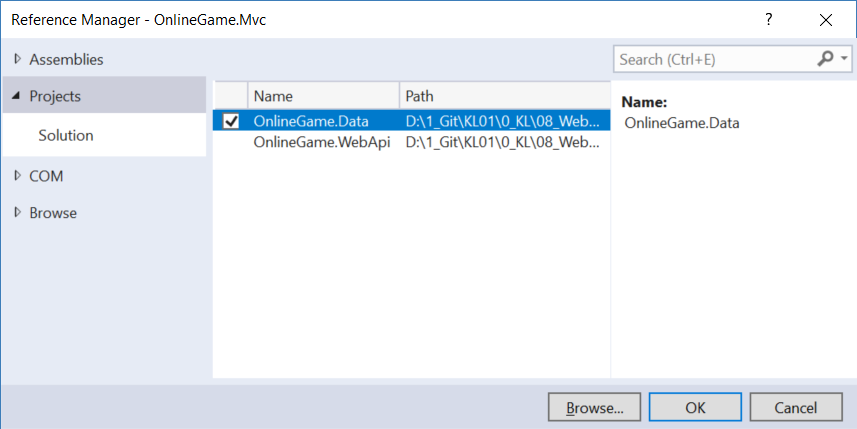
--> Select "**MVC**" --> OK

--> Add Reference



Graphical user interface, application

Description automatically generated



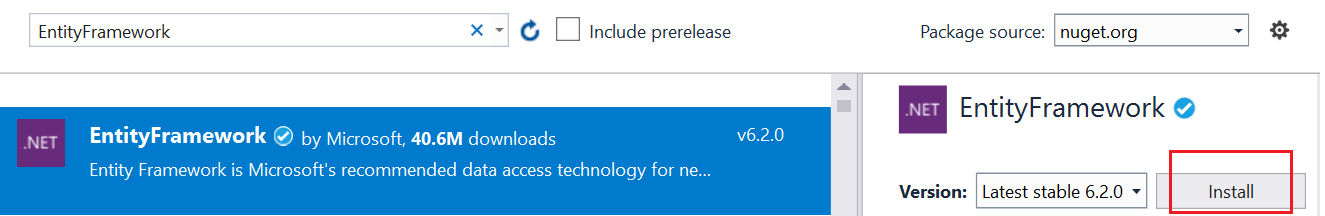
3. OnlineGame.Data

3.1. Install Entity Framework

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

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

--> Install it



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**

-->

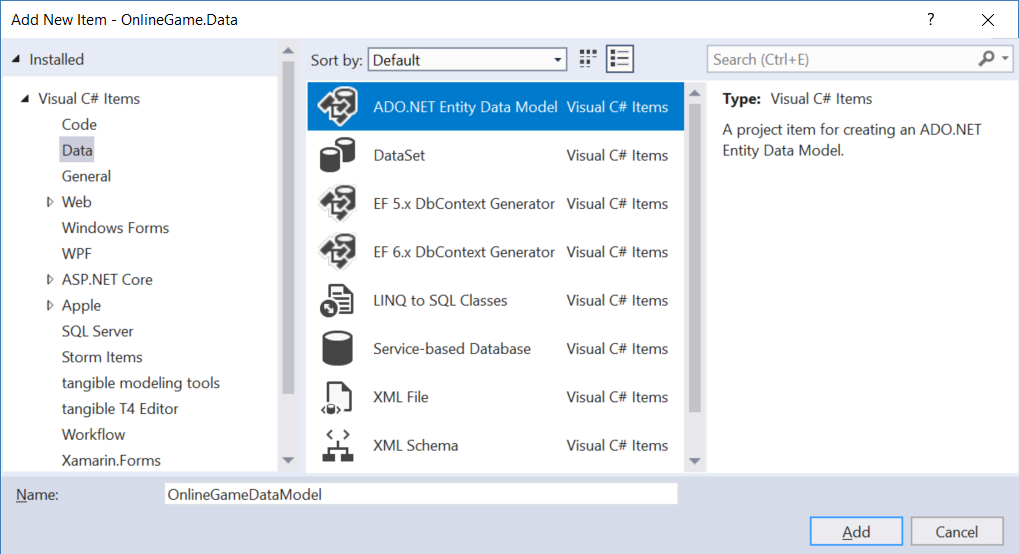
EF Designer from database

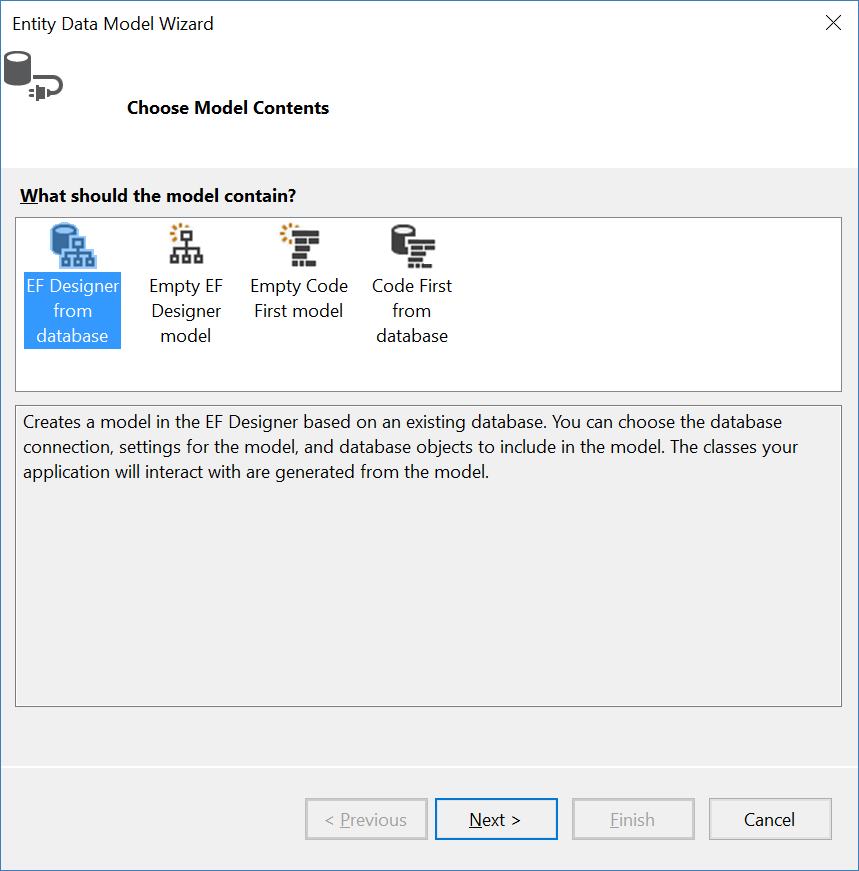
....

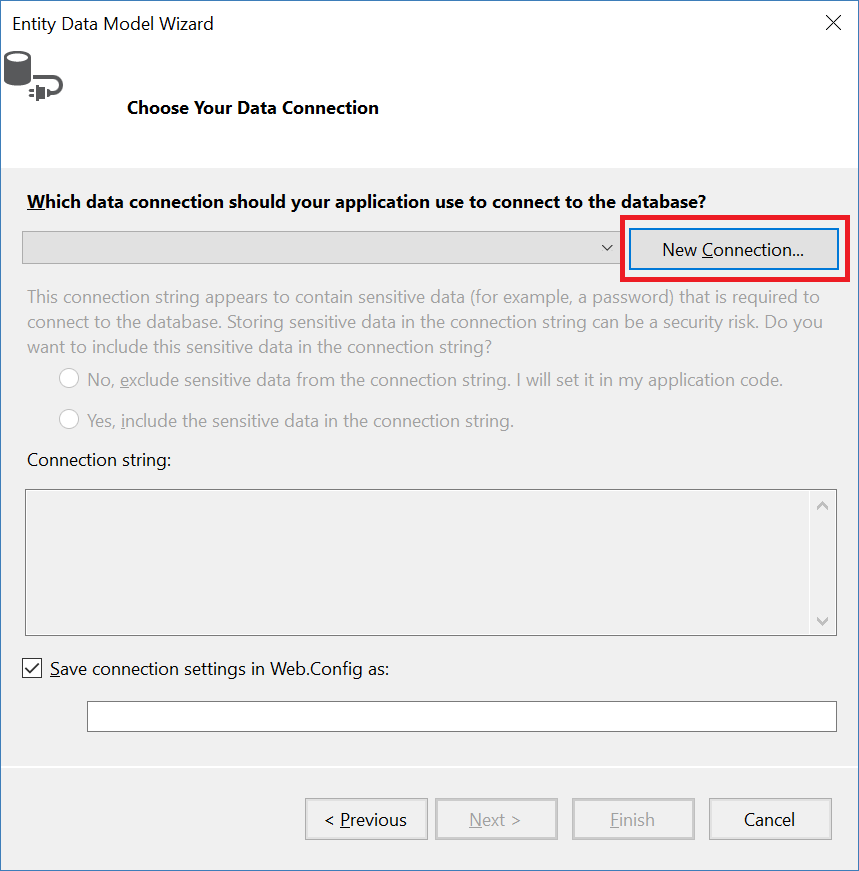
-->

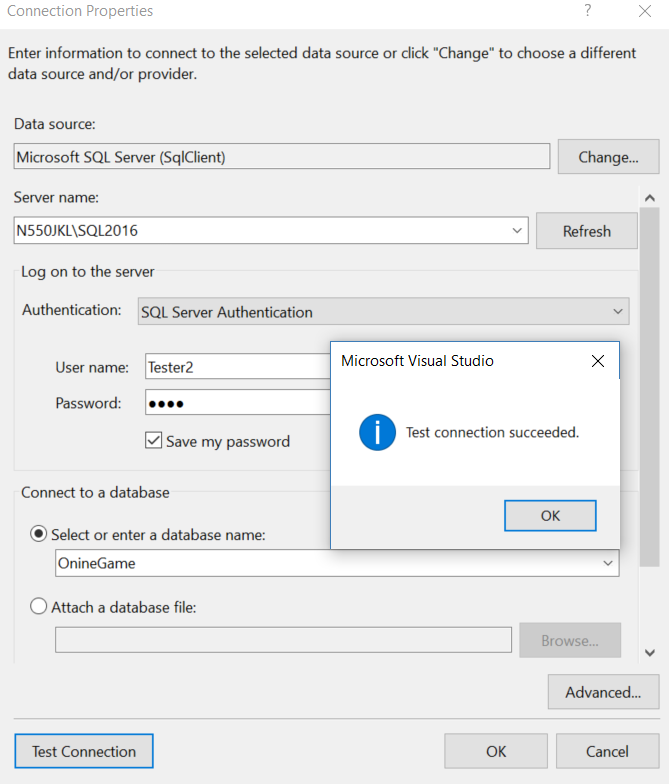
Save Connection settings in Web.Config as:

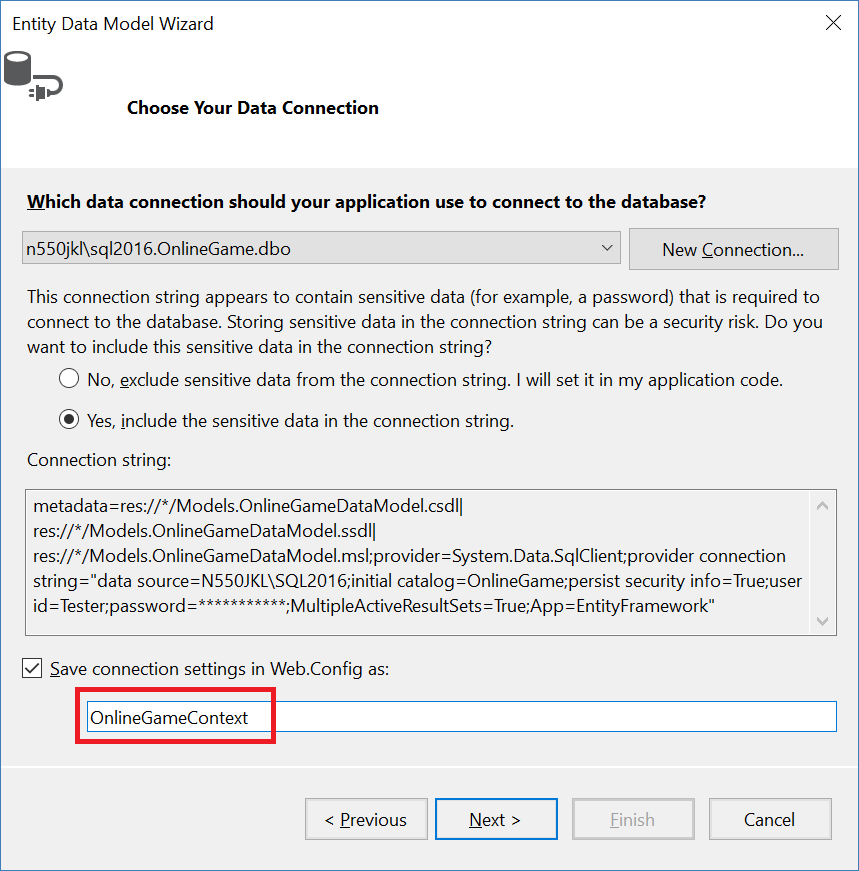
**OnlineGameContext**

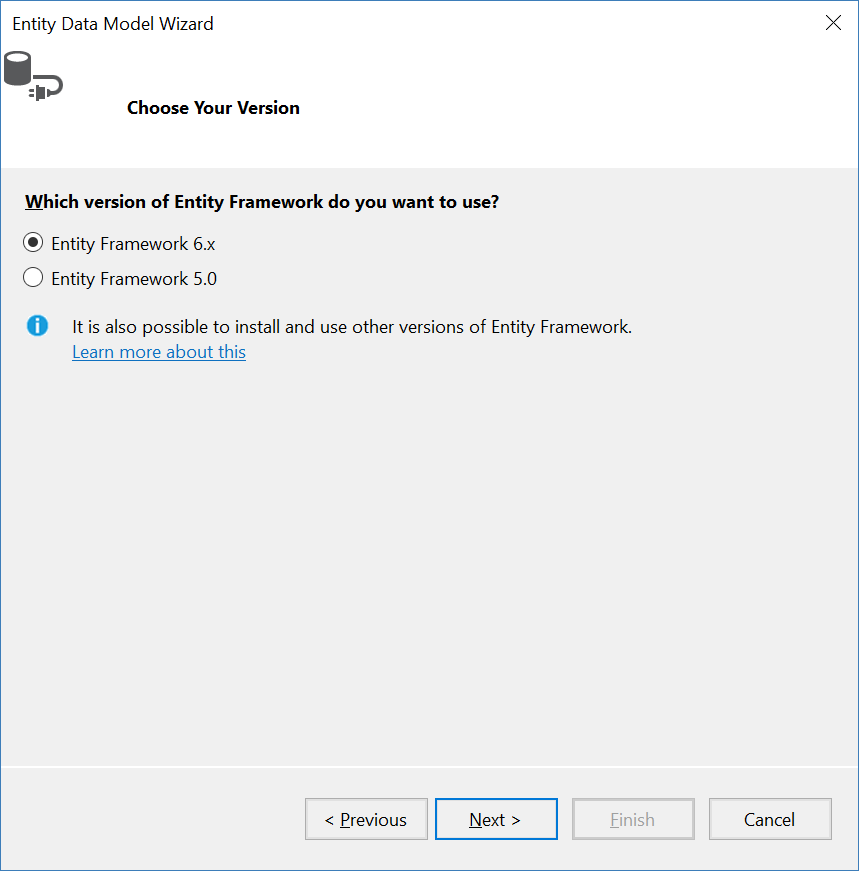


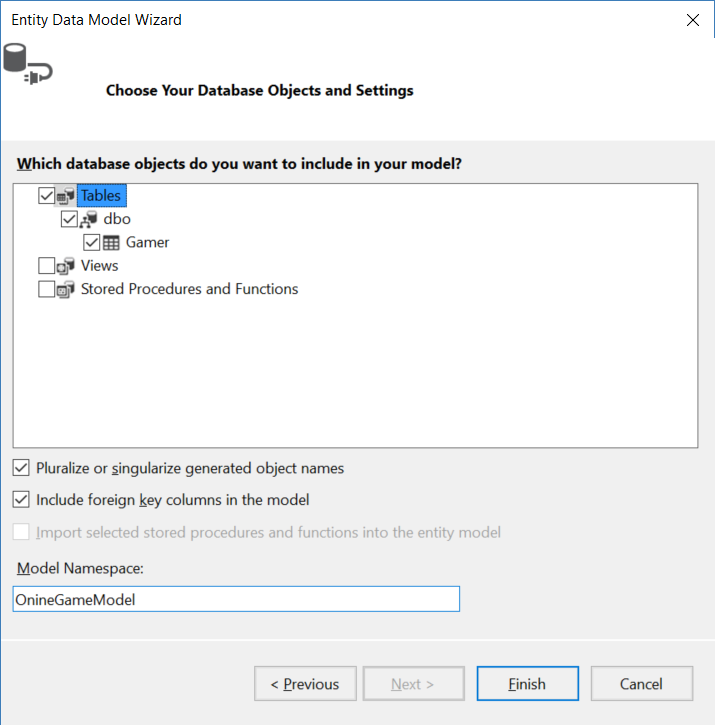






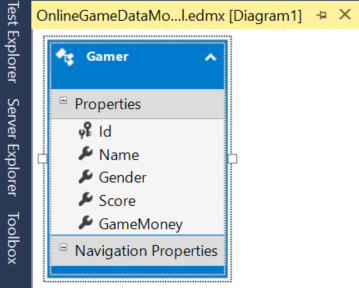






Graphical user interface, text, application, Word

Description automatically generated



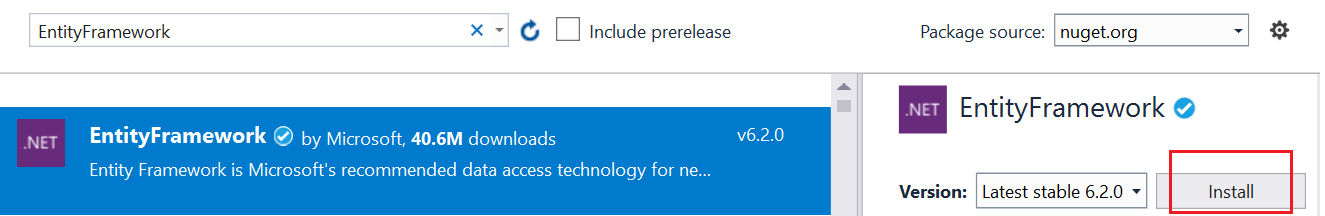
4. OnlineGame.WebApi

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

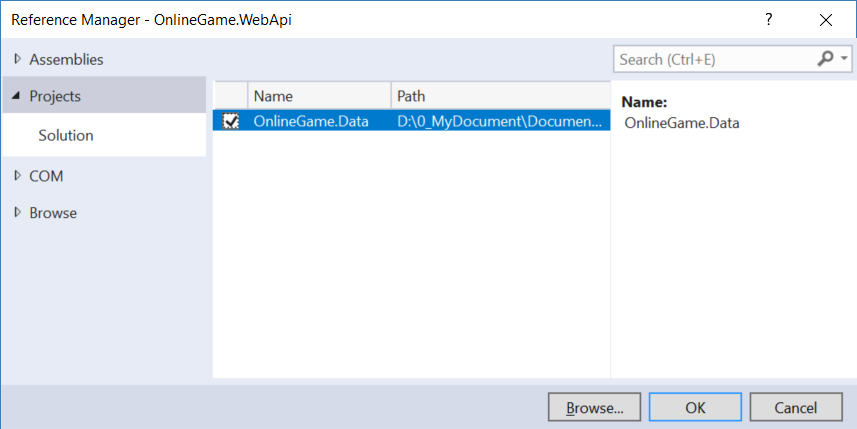


<connectionStrings>

    <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



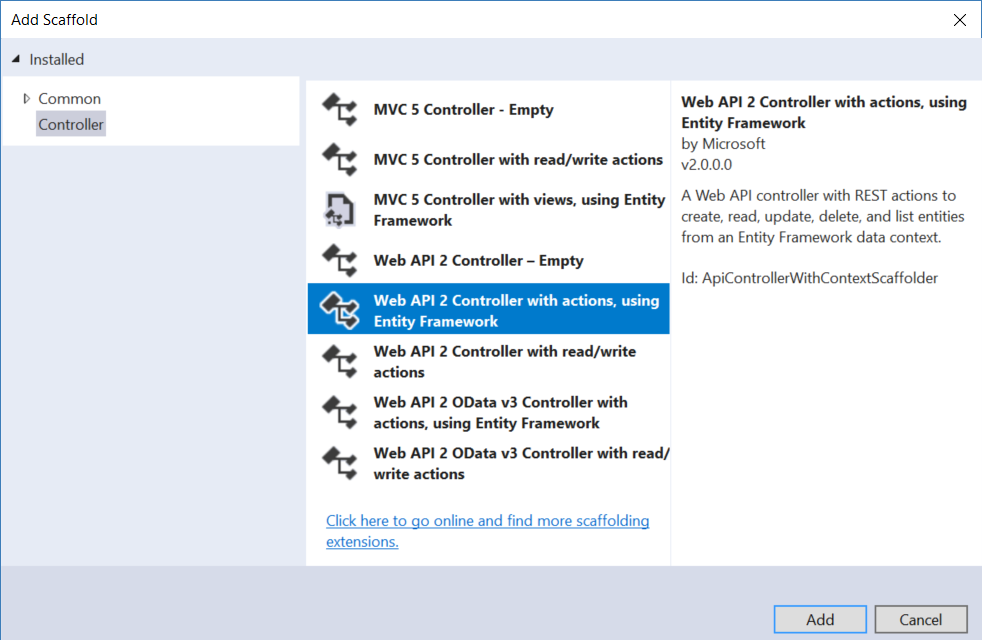
4.4. Controllers/Api/GamerController.cs

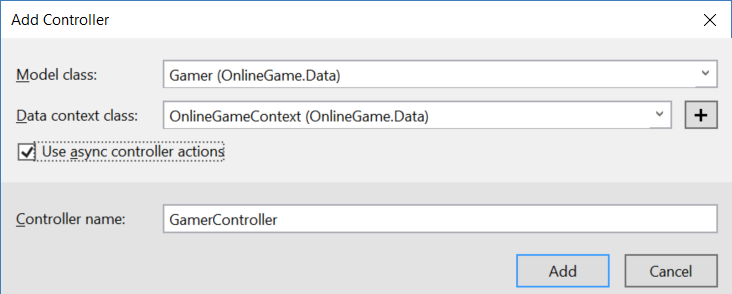
Controllers/Api  folder --> Right Click --> Add --> Controller

--> **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;

namespace OnlineGame.WebApi.Controllers.Api

{

    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.

[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

\*/

4.5. Controllers/GamerController.cs

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

-->

**MVC 5 Controller with views, using Entity Framework**

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

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.

Graphical user interface, text, application

Description automatically generated with medium confidence

using System.Data.Entity;

using System.Threading.Tasks;

using[System.Net](http://system.net/);

using System.Web.Mvc;

using OnlineGame.Data;

namespace OnlineGame.WebApi.Controllers

{

    public class GamerController : Controller

    {

        private OnlineGameContext \_db = new OnlineGameContext();

        // GET: Gamer

        [HttpGet]

        public async Task<ActionResult> Index()

        {

            return View(await \_db.Gamers.ToListAsync());

        }

        // GET: Gamer

        [HttpGet]

        public ActionResult Index2()

        {

            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);

        }

    }

}

4.6. Views/Gamer/Index2.cshtml - Jquery AJAX call Web API

@{

    ViewBag.Title = "Index2";

}

<h2>Index2</h2>

<div>

    <input id="btnGamerList" type="button" value="Gamer List" />

    <input id="btnGamerTable" type="button" value="Gamer Table" />

    <input id="btnClear" type="button" value="Clear" />

    <ul id="ulGamers"></ul>

    <table id="tblGamers"></table>

</div>

<script src="~/Scripts/jquery-1.10.2.min.js"></script>

<script type="text/javascript">

    $(document).ready(function () {

        var ulGamers = $('#ulGamers');

        var tblGamers = $('#tblGamers');

        var gamerApiUrl = '/api/gamer/';

        $('#btnGamerList').click(function () {

            $.ajax({

                type: 'GET',

                url: gamerApiUrl,

                dataType: 'json',

                success: function (data) {

                    ulGamers.empty();

                    $.each(data, function (index, val) {

                        var name = val.Name;

                        ulGamers.append('<li>' + name + '</li>');

                    });

                }

            });

        });

        $('#btnGamerTable').click(function () {

            $.ajax({

                type: 'GET',

                url: gamerApiUrl,

                dataType: 'json',

                success: function (data) {

                    tblGamers.empty();

                    tblGamers.append('<tr><th>Id</th><th>Name</th><th>Gender</th><th>Score</th><th>GameMoney</th></tr>');

                    $.each(data, function (index, val) {

                        tblGamers.append('<tr>' +

                            '<td>' + val.Id + '</td>' +

                            '<td>' + val.Name + '</td>' +

                            '<td>' + val.Gender + '</td>' +

                            '<td>' + val.Score + '</td>' +

                            '<td>' + val.GameMoney + '</td>' +

                            '</tr>');

                    });

                }

            });

        });

        $('#btnClear').click(function () {

            ulGamers.empty();

            tblGamers.empty();

        });

    });

</script>

Graphical user interface, table

Description automatically generated

5. OnlineGame.Mvc

5.1. Jquery AJAX may call Web API in the same origin

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 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

5.3. 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>

5.4. Controllers/GamerController.cs

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

-->

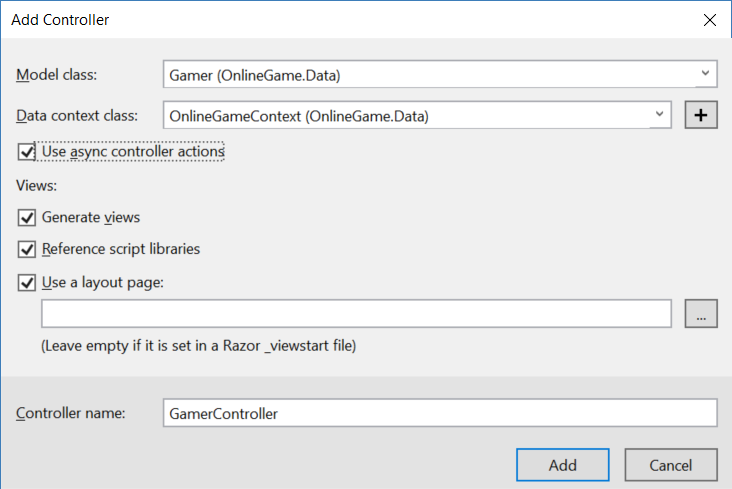
**MVC 5 Controller with views, using Entity Framework**

Graphical user interface, application

Description automatically generated

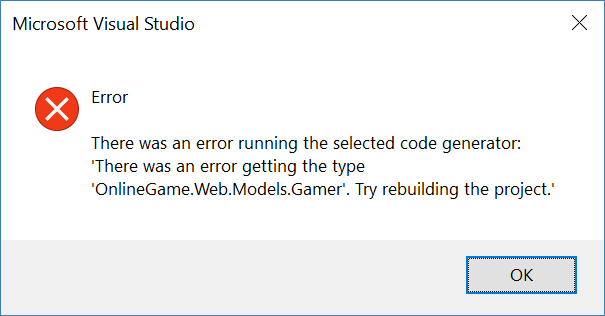
Graphical user interface, text, application

Description automatically generated



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.



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();

        }

        [HttpGet]

        public ActionResult IndexWebApiJsonp()

        {

            return View();

        }

        [HttpGet]

        public ActionResult IndexWebApiCors()

        {

            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);

        }

    }

}

5.5. Views/Gamer/IndexWebApi.cshtml

@{

    ViewBag.Title = "IndexWebApi";

}

<h2>IndexWebApi</h2>

<div>

    <input id="btnGamerList" type="button" value="Gamer List" />

    <input id="btnGamerTable" type="button" value="Gamer Table" />

    <input id="btnClear" type="button" value="Clear" />

    <ul id="ulGamers"></ul>

    <table id="tblGamers"></table>

</div>

<script src="~/Scripts/jquery-1.10.2.min.js"></script>

<script type="text/javascript">

    $(document).ready(function () {

        var ulGamers = $('#ulGamers');

        var tblGamers = $('#tblGamers');

        var gamerApiUrl = '<http://localhost:49789/api/gamer>';

        //[http://localhost:49789](http://localhost:49789/) 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.

        $('#btnGamerList').click(function () {

            $.ajax({

                type: 'GET',

                url: gamerApiUrl,

                dataType: 'json',

                success: function (data) {

                    ulGamers.empty();

                    $.each(data, function (index, val) {

                        var name = val.Name;

                        ulGamers.append('<li>' + name + '</li>');

                    });

                }

            });

        });

        $('#btnGamerTable').click(function () {

            $.ajax({

                type: 'GET',

                url: gamerApiUrl,

                dataType: 'json',

                success: function (data) {

                    tblGamers.empty();

                    tblGamers.append('<tr><th>Id</th><th>Name</th><th>Gender</th><th>Score</th><th>GameMoney</th></tr>');

                    $.each(data, function (index, val) {

                        tblGamers.append('<tr>' +

                            '<td>' + val.Id + '</td>' +

                            '<td>' + val.Name + '</td>' +

                            '<td>' + val.Gender + '</td>' +

                            '<td>' + val.Score + '</td>' +

                            '<td>' + val.GameMoney + '</td>' +

                            '</tr>');

                    });

                }

            });

        });

        $('#btnClear').click(function () {

            ulGamers.empty();

            tblGamers.empty();

        });

    });

</script>

6. OnlineGame.Mvc

6.1. JSONP allows Jquery AJAX may call Web API in the different origins

Reference:

<https://github.com/WebApiContrib/WebApiContrib.Formatting.Jsonp>

<https://www.nuget.org/packages/WebApiContrib.Formatting.Jsonp/>

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.

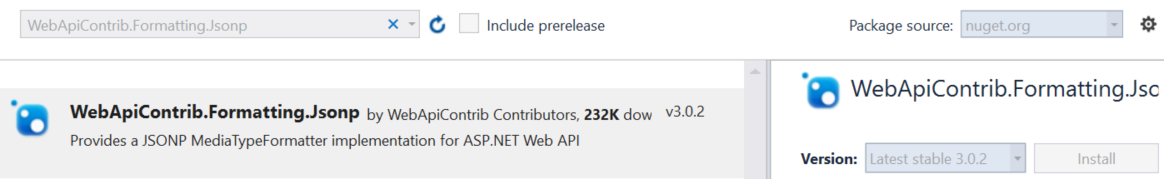
Enable CORS (Cross Origin Resource Sharing)

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

6.2. Install JSONP

In OnlineGame.WebApi project

**Install-Package** **WebApiContrib.Formatting.Jsonp**



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

using System.Web.Http;

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 }

            );

            //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);

        }

    }

}

/\*

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"

})

\*/

6.4. OnlineGame.Mvc/Views/Gamer/IndexWebApiJsonp.cshtml

@{

    ViewBag.Title = "IndexWebApiJsonp";

}

<h2>IndexWebApiJsonp</h2>

<div>

    <input id="btnGamerList" type="button" value="Gamer List" />

    <input id="btnGamerTable" type="button" value="Gamer Table" />

    <input id="btnClear" type="button" value="Clear" />

    <ul id="ulGamers"></ul>

    <table id="tblGamers"></table>

</div>

<script src="~/Scripts/jquery-1.10.2.min.js"></script>

<script type="text/javascript">

    $(document).ready(function () {

        var ulGamers = $('#ulGamers');

        var tblGamers = $('#tblGamers');

        var gamerdatatype = 'jsonp';

        var gamerApiUrl = '<http://localhost:49789/api/gamer>';

        //[http://localhost:49789](http://localhost:49789/) 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)

        //Here, we will use JSONP to fix the issue.

        $('#btnGamerList').click(function () {

            $.ajax({

                type: 'GET',

                url: gamerApiUrl,

                dataType: gamerdatatype,

                success: function (data) {

                    ulGamers.empty();

                    $.each(data, function (index, val) {

                        var name = val.Name;

                        ulGamers.append('<li>' + name + '</li>');

                    });

                }

            });

        });

        $('#btnGamerTable').click(function () {

            $.ajax({

                type: 'GET',

                url: gamerApiUrl,

                dataType: gamerdatatype,

                success: function (data) {

                    tblGamers.empty();

                    tblGamers.append('<tr><th>Id</th><th>Name</th><th>Gender</th><th>Score</th><th>GameMoney</th></tr>');

                    $.each(data, function (index, val) {

                        tblGamers.append('<tr>' +

                            '<td>' + val.Id + '</td>' +

                            '<td>' + val.Name + '</td>' +

                            '<td>' + val.Gender + '</td>' +

                            '<td>' + val.Score + '</td>' +

                            '<td>' + val.GameMoney + '</td>' +

                            '</tr>');

                    });

                }

            });

        });

        $('#btnClear').click(function () {

            ulGamers.empty();

            tblGamers.empty();

        });

    });

</script>

<http://localhost:49804/Gamer/IndexWebApiJsonp>

Table

Description automatically generated

6.5. OnlineGame.WebApi/Views/Gamer/Index2.cshtml

@{

    ViewBag.Title = "Index2";

}

<h2>Index2</h2>

<div>

    <input id="btnGamerList" type="button" value="Gamer List" />

    <input id="btnGamerTable" type="button" value="Gamer Table" />

    <input id="btnClear" type="button" value="Clear" />

    <ul id="ulGamers"></ul>

    <table id="tblGamers"></table>

</div>

<script src="~/Scripts/jquery-1.10.2.min.js"></script>

<script type="text/javascript">

    $(document).ready(function () {

        var ulGamers = $('#ulGamers');

        var tblGamers = $('#tblGamers');

        var gamerdatatype = 'jsonp';

        var gamerApiUrl = '/api/gamer/';

        $('#btnGamerList').click(function () {

            $.ajax({

                type: 'GET',

                url: gamerApiUrl,

                dataType: gamerdatatype,

                success: function (data) {

                    ulGamers.empty();

                    $.each(data, function (index, val) {

                        var name = val.Name;

                        ulGamers.append('<li>' + name + '</li>');

                    });

                }

            });

        });

        $('#btnGamerTable').click(function () {

            $.ajax({

                type: 'GET',

                url: gamerApiUrl,

                dataType: gamerdatatype,

                success: function (data) {

                    tblGamers.empty();

                    tblGamers.append('<tr><th>Id</th><th>Name</th><th>Gender</th><th>Score</th><th>GameMoney</th></tr>');

                    $.each(data, function (index, val) {

                        tblGamers.append('<tr>' +

                            '<td>' + val.Id + '</td>' +

                            '<td>' + val.Name + '</td>' +

                            '<td>' + val.Gender + '</td>' +

                            '<td>' + val.Score + '</td>' +

                            '<td>' + val.GameMoney + '</td>' +

                            '</tr>');

                    });

                }

            });

        });

        $('#btnClear').click(function () {

            ulGamers.empty();

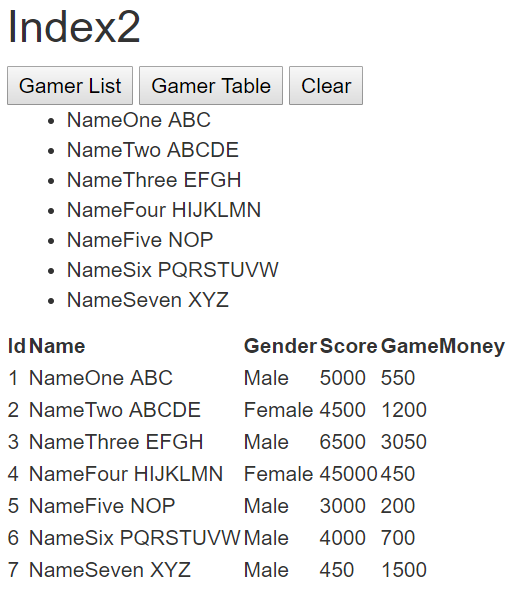
            tblGamers.empty();

        });

    });

</script>

<http://localhost:49789/gamer/index2>



6.6. Fiddler test Jsonp

1.

<http://localhost:49789/api/gamer>

Request header:

Host: localhost:49789

-->

Response

HTTP/1.1 500 Internal Server Error

Graphical user interface, text, application

Description automatically generated

-->

Text

Description automatically generated with medium confidence

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

2.

<http://localhost:49789/api/gamer?callback=AAA>

Request header:

Host: localhost:49789

-->

Response

200

Raw Json data wrapped by AAA function.

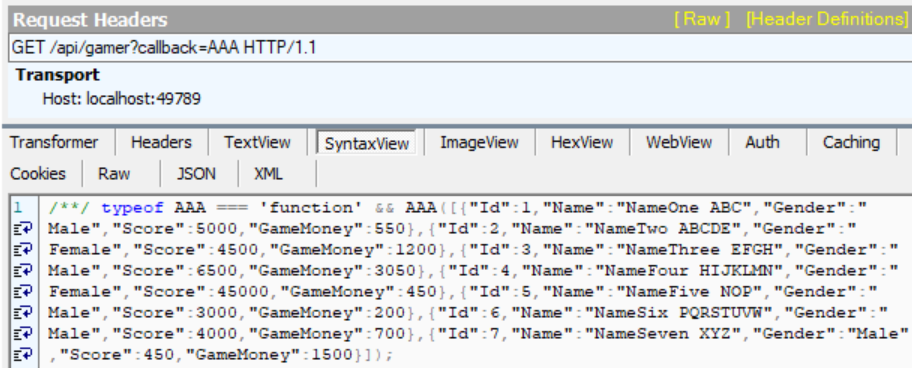
Graphical user interface, text, application

Description automatically generated

-->



-->



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

3.

<http://localhost:49789/api/gamer>

Request header:

Host: localhost:49789

Accept: application/json

-->

Response

200

The json data.

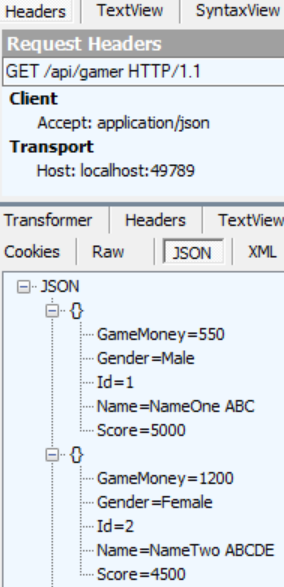
Graphical user interface, text, application

Description automatically generated

-->



-->



7. OnlineGame.Mvc

7.1. 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"

})

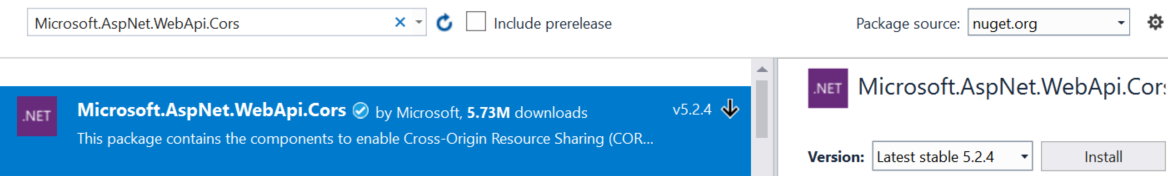
2.

Enable CORS (Cross Origin Resource Sharing)

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

7.2. Install WebApi Cors

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



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

using System.Web.Http;

using System.Web.Http.Cors;

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.

            ////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);

            //2.2.

            config.EnableCors();

        }

    }

}

/\*

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")

\*/

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

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.Cors;

using System.Web.Http.Description;

using OnlineGame.Data;

namespace OnlineGame.WebApi.Controllers.Api

{

    //[EnableCors("\*", "\*", "\*")]

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

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

    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

        //[DisableCors]

        [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.

[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")

\*/

7.5. OnlineGame.WebApi/Views/Gamer/Index2.cshtml

@{

    ViewBag.Title = "Index2";

}

<h2>Index2</h2>

<div>

    <input id="btnGamerList" type="button" value="Gamer List" />

    <input id="btnGamerTable" type="button" value="Gamer Table" />

    <input id="btnClear" type="button" value="Clear" />

    <ul id="ulGamers"></ul>

    <table id="tblGamers"></table>

</div>

<script src="~/Scripts/jquery-1.10.2.min.js"></script>

<script type="text/javascript">

    $(document).ready(function () {

        var ulGamers = $('#ulGamers');

        var tblGamers = $('#tblGamers');

        //var gamerdatatype = 'jsonp';

        var gamerdatatype = 'json';

        var gamerApiUrl = '/api/gamer/';

        $('#btnGamerList').click(function () {

            $.ajax({

                type: 'GET',

                url: gamerApiUrl,

                dataType: gamerdatatype,

                success: function (data) {

                    ulGamers.empty();

                    $.each(data, function (index, val) {

                        var name = val.Name;

                        ulGamers.append('<li>' + name + '</li>');

                    });

                }

            });

        });

        $('#btnGamerTable').click(function () {

            $.ajax({

                type: 'GET',

                url: gamerApiUrl,

                dataType: gamerdatatype,

                success: function (data) {

                    tblGamers.empty();

                    tblGamers.append('<tr><th>Id</th><th>Name</th><th>Gender</th><th>Score</th><th>GameMoney</th></tr>');

                    $.each(data, function (index, val) {

                        tblGamers.append('<tr>' +

                            '<td>' + val.Id + '</td>' +

                            '<td>' + val.Name + '</td>' +

                            '<td>' + val.Gender + '</td>' +

                            '<td>' + val.Score + '</td>' +

                            '<td>' + val.GameMoney + '</td>' +

                            '</tr>');

                    });

                }

            });

        });

        $('#btnClear').click(function () {

            ulGamers.empty();

            tblGamers.empty();

        });

    });

</script>

7.6. OnlineGame.Mvc/Views/Gamer/IndexWebApiCors.cshtml

@{

    ViewBag.Title = "IndexWebApiJsonp";

}

<h2>IndexWebApiJsonp</h2>

<div>

    <input id="btnGamerList" type="button" value="Gamer List" />

    <input id="btnGamerTable" type="button" value="Gamer Table" />

    <input id="btnClear" type="button" value="Clear" />

    <ul id="ulGamers"></ul>

    <table id="tblGamers"></table>

</div>

<script src="~/Scripts/jquery-1.10.2.min.js"></script>

<script type="text/javascript">

    $(document).ready(function () {

        var ulGamers = $('#ulGamers');

        var tblGamers = $('#tblGamers');

        //var gamerdatatype = 'jsonp';

        var gamerdatatype = 'json';

        var gamerApiUrl = '<http://localhost:49789/api/gamer>';

        //[http://localhost:49789](http://localhost:49789/) 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)

        //Here, we will use JSONP to fix the issue.

        $('#btnGamerList').click(function () {

            $.ajax({

                type: 'GET',

                url: gamerApiUrl,

                dataType: gamerdatatype,

                success: function (data) {

                    ulGamers.empty();

                    $.each(data, function (index, val) {

                        var name = val.Name;

                        ulGamers.append('<li>' + name + '</li>');

                    });

                }

            });

        });

        $('#btnGamerTable').click(function () {

            $.ajax({

                type: 'GET',

                url: gamerApiUrl,

                dataType: gamerdatatype,

                success: function (data) {

                    tblGamers.empty();

                    tblGamers.append('<tr><th>Id</th><th>Name</th><th>Gender</th><th>Score</th><th>GameMoney</th></tr>');

                    $.each(data, function (index, val) {

                        tblGamers.append('<tr>' +

                            '<td>' + val.Id + '</td>' +

                            '<td>' + val.Name + '</td>' +

                            '<td>' + val.Gender + '</td>' +

                            '<td>' + val.Score + '</td>' +

                            '<td>' + val.GameMoney + '</td>' +

                            '</tr>');

                    });

                }

            });

        });

        $('#btnClear').click(function () {

            ulGamers.empty();

            tblGamers.empty();

        });

    });

</script>

7.7. Fiddler test CORS (Cross Origin Resource Sharing)

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")

E.g.

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

config.EnableCors();

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

[EnableCors("\*", "\*", "\*")]

public class GamerController : ApiController

<http://localhost:49804/Gamer/IndexWebApiJsonp>

-->

In Fiddler



-->

Graphical user interface, text, application, email

Description automatically generated

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

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")

E.g.

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

config.EnableCors();

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

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

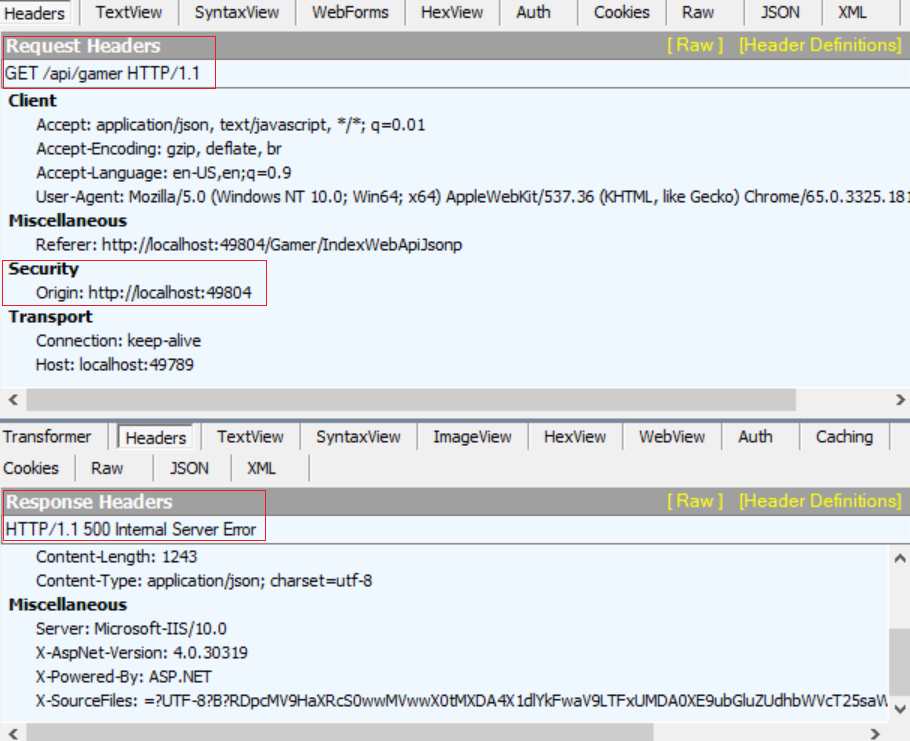
public class GamerController : ApiController

<http://localhost:49804/Gamer/IndexWebApiJsonp>

-->



-->



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

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")

E.g.

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

config.EnableCors();

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

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

public class GamerController : ApiController

<http://localhost:49804/Gamer/IndexWebApiJsonp>

-->

Text

Description automatically generated

-->

Graphical user interface, text, application, email

Description automatically generated