(T8)討論RoutePrefixAttribute、RouteAttribute、RouteName、RouteConstraints。比較IHttpActionResult、HttpResponseMessage  
CourseGUID 4c5822ff-7111-4e25-a336-ef18d48d54bd  
=======================================================================  
(T8)討論RoutePrefixAttribute、RouteAttribute、RouteName、RouteConstraints。比較IHttpActionResult、HttpResponseMessage

(T8-1)前置設定。討論TSQL、EF

(T8-2)討論RouteAttribute

(T8-3)討論RoutePrefixAttribute、RouteAttribute

(T8-4)討論RouteConstraints

(T8-5)討論RouteName

(T8-6)比較IHttpActionResult、HttpResponseMessage  
=======================================================================  
0. What to Learn

-----------

1. OnlineGame2 DB

1.0. Some points

1.1. TSQL

1.2. Security login

-----------

2. OnlineGame Solution

2.1. OnlineGame Solution

2.2. OnlineGame.WebApi

-----------

3. OnlineGame.WebApi - Entity Framework

3.1. Install Entity Framework

3.2. ADO.Net Entity Data Model - Entity Framework

-----------

4. OnlineGame.WebApi - API Controller

4.1. OnlineGame.WebApi/App\_Start/WebApiConfig.cs - JSON Formatter

4.2. OnlineGame.WebApi/Controllers/Api/GamerController.cs - Attribute routing

4.2.1. OnlineGame.WebApi/Controllers/Api/GamerController.cs - Attribute routing

4.2.2. OnlineGame.WebApi/Controllers/Api/GamerController.cs - Attribute routing

4.3. OnlineGame.WebApi/Controllers/Api/GamerTwoController.cs - RoutePrefix and Route attribute

4.3.1. OnlineGame.WebApi/Controllers/Api/GamerTwoController.cs - RoutePrefix and Route attribute

4.3.2. OnlineGame.WebApi/Controllers/Api/GamerTwoController.cs - RoutePrefix and Route attribute

4.4. OnlineGame.WebApi/Controllers/Api/GamerThreeController.cs - attribute routing constraints

4.4.1. OnlineGame.WebApi/Controllers/Api/GamerThreeController.cs - attribute routing constraints

4.4.2. OnlineGame.WebApi/Controllers/Api/GamerThreeController.cs - attribute routing constraints

4.5. OnlineGame.WebApi/Controllers/Api/GamerFourController.cs - Route names

4.5.1. OnlineGame.WebApi/Controllers/Api/GamerFourController.cs - Route names

4.5.2. OnlineGame.WebApi/Controllers/Api/GamerFourController.cs - Route names

4.5.3. Post Request

4.5.3.1. Post Request - public async Task<IHttpActionResult> PostGamer(Gamer gamer)

4.5.3.2. Post Request - public async Task<HttpResponseMessage> AddGamer(Gamer gamer) - Bug

4.5.3.3. Post Request - public async Task<IHttpActionResult> AddGamer2(Gamer gamer) - Bug

4.5.3.4. Post Request - public async Task<HttpResponseMessage> AddGamer3(Gamer gamer) - Fix Bug

4.5.3.5. Post Request - public async Task<IHttpActionResult> AddGamer4(Gamer gamer) - Fix Bug

4.6. OnlineGame.WebApi/Controllers/Api/GamerFiveController.cs - Route names

4.6.1. OnlineGame.WebApi/Controllers/Api/GamerFiveController.cs

4.6.2. OnlineGame.WebApi/Controllers/Api/GamerFiveController.cs  
=======================================================================

0. What to Learn

The tutorial will discuss ...

How to use RoutePrefixAttribute, RouteAttribute, RouteName, RouteConstraints.

IHttpActionResult V.S. HttpResponseMessage

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

本堂課討論

關於RoutePrefixAttribute、RouteAttribute、RouteName、RouteConstraints。

比較IHttpActionResult和HttpResponseMessage

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

6.

Attribute routing

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

6.1.

E.g.

//public async Task<IHttpActionResult> GetGamer(int id){...}

....

//[Route("api/gamer/{id}/skills")]

//public async Task<IHttpActionResult> GetGamerSkills(int id){...}

When we call "api/gamer/1" and if we don't have Route attribute,

the API will be confused,

because both GetGamerSkills() and GetGamer() can map to "api/gamer/1".

Thus, we need Route attribute

[Route("api/gamer/{id}/skills")] will make GetGamerSkills() map to something  like "api/gamer/1/skills".

Thus, GetGamer() can map to something like "api/gamer/1".

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

6.2.

In this case,

GetGamer() is using Convention-based routing.

GetGamerSkills() is using Attribute Routing.

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

6.3.

In

OnlineGame.WebApi/WebApiConfig.cs/WebApiConfig.cs

//config.MapHttpAttributeRoutes();

It enables Attribute Routing.

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

7.

RoutePrefix and Route attribute

//[RoutePrefix("api/gamer2")]

RoutePrefix attribute is for route prefix at the controller level.

Route attribute use that route prefix plus its own route value.

//[Route("~/api/getGamerSkillsByGamerId/{gamerId}")]

if you want to  override the route prefix,

just use ~ (tilde) symbol

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

8.

attribute routing constraints

Reference:

https://docs.microsoft.com/en-us/aspnet/web-api/overview/web-api-routing-and-actions/attribute-routing-in-web-api-2#route-constraints

Routing constraints can apply to decimal, double, float, long, bool...etc.

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

8.1.

//// GET: api/gamer3/GetGamerBySomething/2

//[Route("GetGamerBySomething/{gamerId:int}")]

//public async Task<IHttpActionResult> GetGamerBySomething(int gamerId)

int means integer

...

//// GET: api/gamer3/GetGamerBySomething/male

////[Route("GetGamerBySomething/{gender:string}")]    //Error, string type is  not valid

//[Route("GetGamerBySomething/{gender:alpha}")]

//public async Task<IHttpActionResult> GetGamerBySomething(string gender)

alpha means uppercase or lowercase alphabet.

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

8.2.

//[Route("getGamerById/{gamerId:int:min(2)}")]

//public async Task<IHttpActionResult> GetGamerById(int gamerId)

GET: api/gamer3/getGamerById/1

gamerId must be int and min is 2

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

8.3.

//[Route("getGamerById2/{gamerId:int:min(2):max(5)}")]

//public async Task<IHttpActionResult> GetGamerById2(int gamerId)

GET: api/gamer3/getGamerById2/1

gamerId must be int and min is 2, max is 5

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

8.4.

//[Route("getGamerById3/{gamerId:range(2,5)}")]

//public async Task<IHttpActionResult> GetGamerById3(int gamerId)

GET: api/gamer3/getGamerById3/1

gamerId must be int and min is 2, max is 5

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

8.5.

////[Route("getGamersByGender/{gender:string}")]    //Error, string type is not  valid

//[Route("getGamersByGender/{gender:alpha}")]

//public async Task<IHttpActionResult> GetGamersByGender(string gender)

alpha means uppercase or lowercase alphabet characters.

GET: api/gamer3/getGamersByGender/female

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

8.7.

//[Route("getGamersByGender3/{gender:alpha:maxlength(5)}")]

//public async Task<IHttpActionResult> GetGamersByGender3(string gender)

GET: api/gamer3/getGamersByGender3/female        //404

GET: api/gamer3/getGamersByGender3/male

alpha means uppercase or lowercase alphabet characters.

max alpha length is 5

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

8.8.

//[Route("getGamersByGender3/{gender:alpha:maxlength(5)}")]

//public async Task<IHttpActionResult> GetGamersByGender3(string gender)

GET: api/gamer3/getGamersByGender3/female        //404

GET: api/gamer3/getGamersByGender3/male

alpha means uppercase or lowercase alphabet characters.

max alpha length is 5

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

8.9.

//[Route("getGamersByGender4/{gender:alpha:minlength(5):maxlength(7)}")]

//public async Task<IHttpActionResult> GetGamersByGender4(string gender)

GET: api/gamer3/getGamersByGender4/female

GET: api/gamer3/getGamersByGender4/male      //404

alpha means uppercase or lowercase alphabet characters.

max alpha length is 7, and min length is 5.

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

9.

Route names

9.1.

E.g.

//[Route("{id:int}", Name = "GetGamerById")]

//public async Task<IHttpActionResult> GetGamerById(int id)

...

//HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.Created);

//response.Headers.Location = new

//    Uri(Url.Link("GetGamerById", new { id = gamer.Id }));

...

//return CreatedAtRoute("GetGamerById", new { id = gamer.Id }, gamer);     //Created/201

9.2.

//return CreatedAtRoute("DefaultApi", new { id = gamer.Id }, gamer);     //Created/201

...

//HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.Created);

//response.Headers.Location = new Uri(Request.RequestUri + "/" + gamer.Id);

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

10.

IHttpActionResult vs HttpResponseMessage

10.1.

IHttpActionResult

10.1.1.

HttpResponseMessage is from Web API 1

IHttpActionResult is from Web API 2

10.1.2.

IHttpActionResult make code cleaner.

10.1.3.

The following type implements IHttpActionResult interface.

Unauthorized()

BadRequest()

NotFound()

Created()

OK()

InternalServerError()

1. OnlineGame2 DB

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

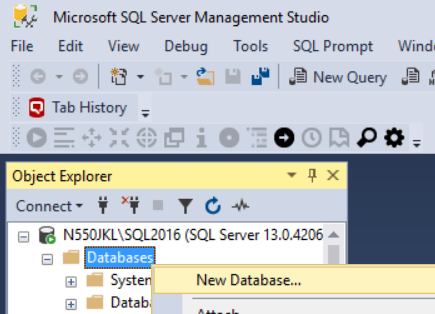
In SQL server Management Studio (SSMS)

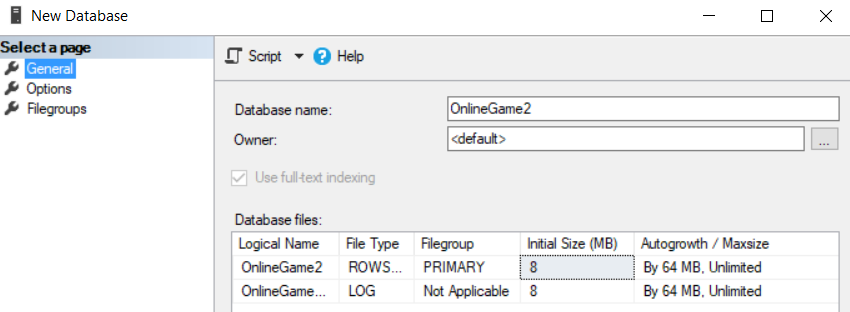
Database --> Right Click --> New Database -->

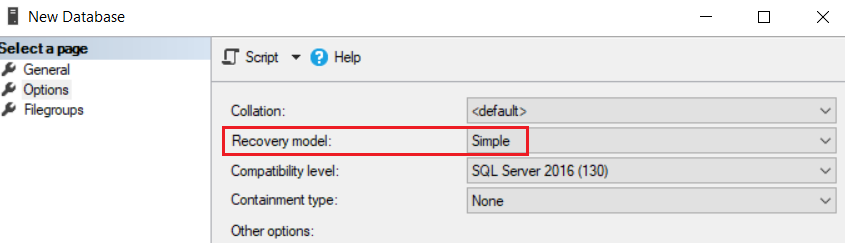
In General Tab -->

Name: **OnlineGame2**

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







--1.1 ----------------------------------------------------------

--Drop Table if it exists.

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'GamerSkill' ) )

    BEGIN

        TRUNCATE TABLE GamerSkill;

        DROP TABLE GamerSkill;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Skill' ) )

    BEGIN

        TRUNCATE TABLE Skill;

        DROP TABLE Skill;

    END;

GO -- Run the previous command and begins new batch

--IF OBJECT\_ID('Gamer') IS NOT NULL

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

--IF OBJECT\_ID('Gamer') IS NOT NULL

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Team' ) )

    BEGIN

        TRUNCATE TABLE Team;

        DROP TABLE Team;

    END;

GO -- Run the previous command and begins new batch

--1.2 ----------------------------------------------------------

--Drop Stored Procedure if it exists.

--IF OBJECT\_ID('spSearchGamer') IS NOT NULL

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spInsertGamerSkill' ) )

    BEGIN

        DROP PROCEDURE spInsertGamerSkill;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spDeleteGamerSkill' ) )

    BEGIN

        DROP PROCEDURE spDeleteGamerSkill;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spSelectGamerSkill' ) )

    BEGIN

        DROP PROCEDURE spSelectGamerSkill;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spSkillsAssignToTheGamer' ) )

    BEGIN

        DROP PROCEDURE spSkillsAssignToTheGamer;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spSkillsNotAssignToTheGamer' ) )

    BEGIN

        DROP PROCEDURE spSkillsNotAssignToTheGamer;

    END;

GO -- Run the previous command and begins new batch

--2 ----------------------------------------------------------

CREATE TABLE Team

    (

      Id INT PRIMARY KEY

             IDENTITY(1, 1)

             NOT NULL ,

      Name NVARCHAR(50) NOT NULL

    );

GO -- Run the previous command and begins new batch

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 ,

      TeamId INT FOREIGN KEY REFERENCES Team ( Id )

    );

GO -- Run the previous command and begins new batch

CREATE TABLE Skill

    (

      Id INT PRIMARY KEY

             IDENTITY(1, 1)

             NOT NULL ,

      Name NVARCHAR(50) NOT NULL

    );

GO -- Run the previous command and begins new batch

CREATE TABLE GamerSkill

    (

      GamerId INT FOREIGN KEY REFERENCES Gamer ( Id )

                  NOT NULL ,

      SkillId INT FOREIGN KEY REFERENCES Skill ( Id )

                  NOT NULL ,

      CreatedDate DATETIME DEFAULT ( GETUTCDATE() )

                           NOT NULL ,

      PRIMARY KEY ( GamerId, SkillId )

    );

GO -- Run the previous command and begins new batch

--3 ----------------------------------------------------------

INSERT  Team

VALUES  ( 'TeamOne' );

INSERT  Team

VALUES  ( 'TeamTwo' );

INSERT  Team

VALUES  ( 'TeamThree' );

GO -- Run the previous command and begins new batch

INSERT  INTO Gamer

VALUES  ( 'NameOne ABC', 'Male', 5000, 1 );

INSERT  INTO Gamer

VALUES  ( 'NameTwo ABCDE', 'Female', 4500, 1 );

INSERT  INTO Gamer

VALUES  ( 'NameThree EFGH', 'Male', 6500, 3 );

INSERT  INTO Gamer

VALUES  ( 'NameFour HIJKLMN', 'Female', 45000, 2 );

INSERT  INTO Gamer

VALUES  ( 'NameFive NOP', 'Male', 3000, 3 );

INSERT  INTO Gamer

VALUES  ( 'NameSix PQRSTUVW', 'Male', 4000, 3 );

INSERT  INTO Gamer

VALUES  ( 'NameSeven XYZ', 'Male', 4500, 1 );

GO -- Run the previous command and begins new batch

INSERT  INTO Skill

VALUES  ( 'SkillA Play Dead' );

INSERT  INTO Skill

VALUES  ( 'SkillB Flame Punch' );

INSERT  INTO Skill

VALUES  ( 'SkillC Steal' );

INSERT  INTO Skill

VALUES  ( 'SkillD Fly' );

INSERT  INTO Skill

VALUES  ( 'SkillE Super Speed' );

INSERT  INTO Skill

VALUES  ( 'SkillF Forzen' );

INSERT  INTO Skill

VALUES  ( 'SkillG Invisible' );

GO -- Run the previous command and begins new batch

INSERT  INTO GamerSkill

        ( GamerId, SkillId )

VALUES  ( 1, 2 );

INSERT  INTO GamerSkill

        ( GamerId, SkillId )

VALUES  ( 1, 3 );

INSERT  INTO GamerSkill

        ( GamerId, SkillId )

VALUES  ( 2, 2 );

INSERT  INTO GamerSkill

        ( GamerId, SkillId )

VALUES  ( 2, 1 );

INSERT  INTO GamerSkill

        ( GamerId, SkillId )

VALUES  ( 2, 4 );

GO -- Run the previous command and begins new batch

--4 SP ----------------------------------------------------------

CREATE PROCEDURE spInsertGamerSkill

    (

      @GamerId INT ,

      @SkillId INT

    )

AS

    BEGIN

        INSERT  INTO GamerSkill

                ( GamerId, SkillId )

        VALUES  ( @GamerId, -- GamerId - int

                  @SkillId  -- SkillId - int

                  );

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spDeleteGamerSkill

    (

      @GamerId INT ,

      @SkillId INT

    )

AS

    BEGIN

        DELETE  FROM GamerSkill

        WHERE   GamerId = @GamerId

                AND SkillId = @SkillId;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spSelectGamerSkill

AS

    BEGIN

        SELECT  gs.GamerId ,

                g.Name ,

                g.Gender ,

                g.Score ,

                gs.SkillId ,

                s.Name

        FROM    Gamer g

                INNER JOIN GamerSkill gs ON g.Id = gs.GamerId

                INNER JOIN Skill s ON s.Id = gs.SkillId;

    END;

GO -- Run the previous command and begins new batch

--This is for test purpose

--If you want to use it in EF, you have to return a view or table function.

CREATE PROCEDURE spSkillsAssignToTheGamer ( @GamerId INT )

AS

    BEGIN

        SELECT  \*

        FROM    GamerSkill gs

                INNER JOIN Skill s ON s.Id = gs.SkillId

        WHERE   GamerId = @GamerId;

    END;

GO -- Run the previous command and begins new batch

--This is for test purpose

--If you want to use it in EF, you have to return a view or table function.

CREATE PROCEDURE spSkillsNotAssignToTheGamer ( @GamerId INT )

AS

    BEGIN

        SELECT  \*

        FROM    Skill s

        WHERE   s.Id NOT IN (

                SELECT  s.Id

                FROM    GamerSkill gs

                        INNER JOIN Skill s ON s.Id = gs.SkillId

                WHERE   GamerId = @GamerId );

    END;

GO -- Run the previous command and begins new batch

--This is for test purpose

--If you want to use it in EF, you have to return a view or table function.

--EXEC spInsertGamerSkill @GamerId = 100, @SkillId = 1;

--EXEC spDeleteGamerSkill @GamerId = 100, @SkillId = 1;

--EXEC spSelectGamerSkill

--EXEC spSkillsAssignToTheGamer @GamerId=1

--EXEC spSkillsNotAssignToTheGamer @GamerId=1

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**

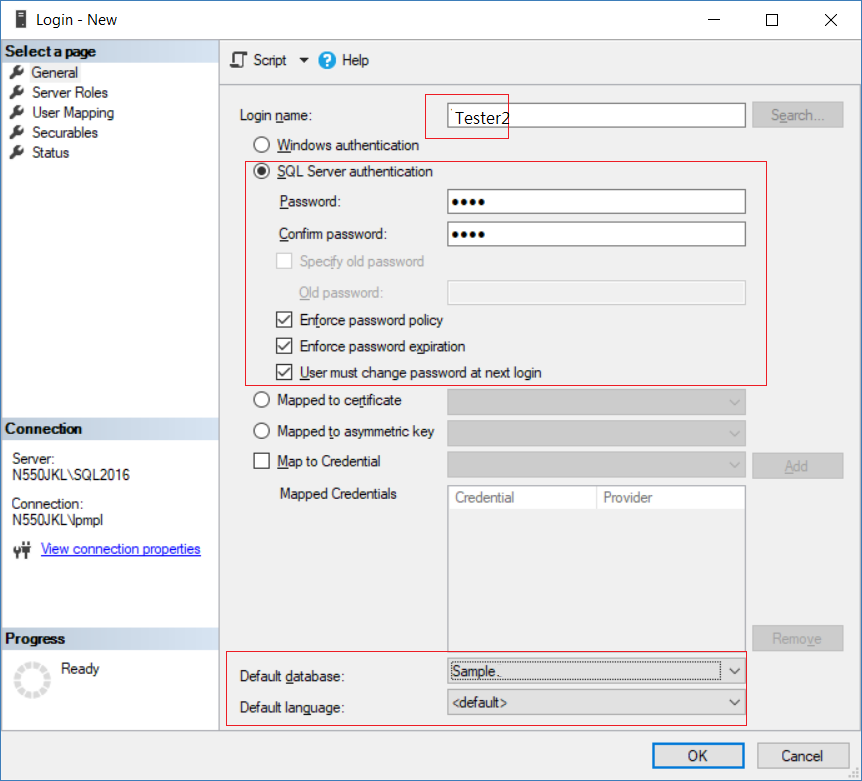
-->

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**"

-->

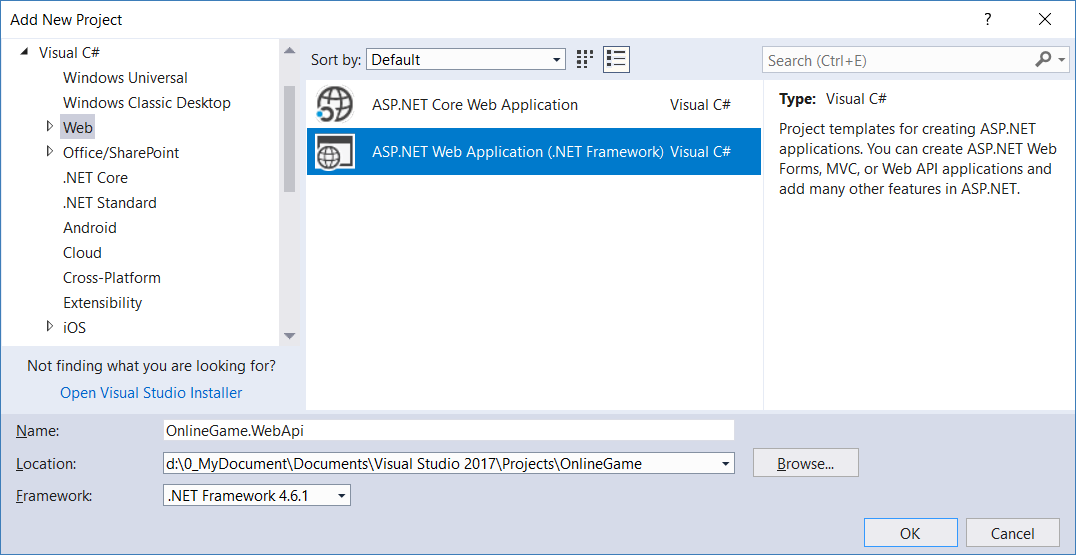
Change Authentication

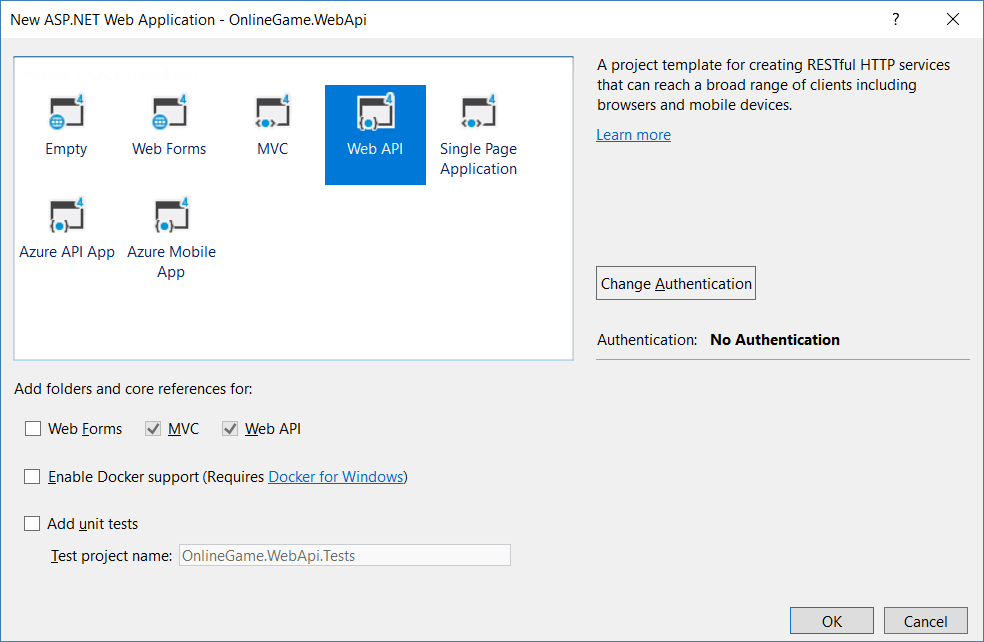
-->

**Individual User Accounts**

-->

OK





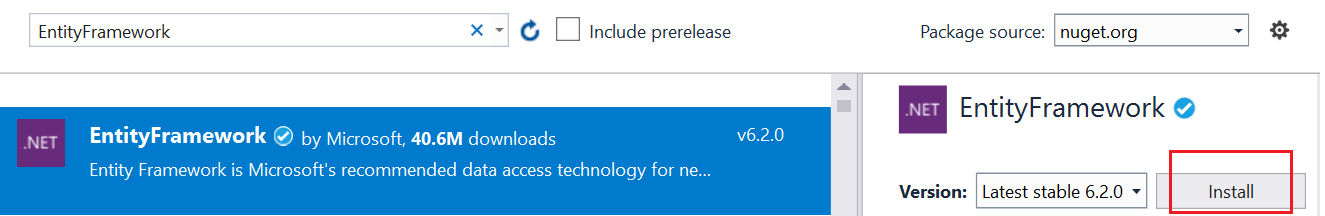
3. OnlineGame.WebApi - Entity Framework

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

**Models Folder** --> Right Click --> Add --> New Item

--> Visual C# --> Data  -->  ADO.Net Entity Data Model

Name:

**OnlineGameDataModel**

-->

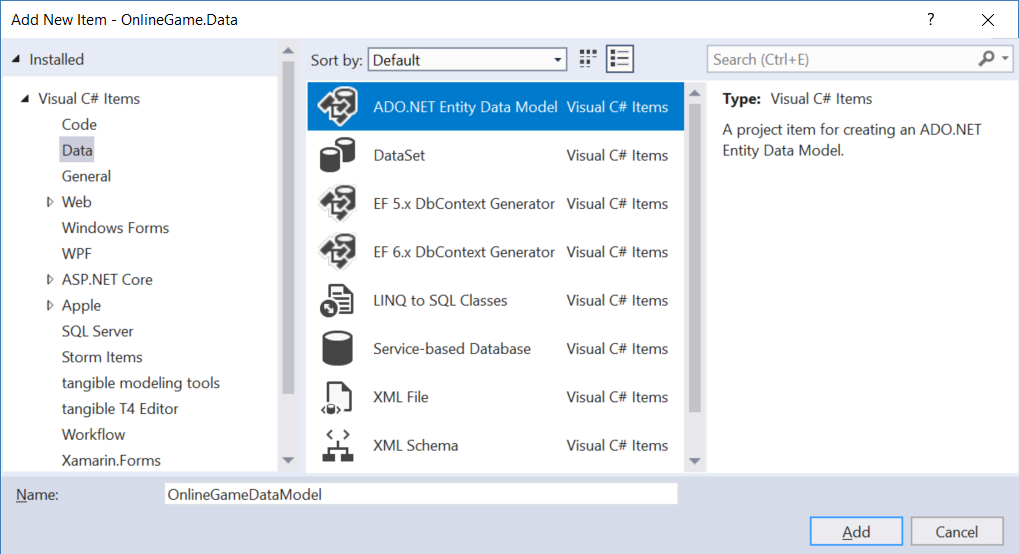
EF Designer from database

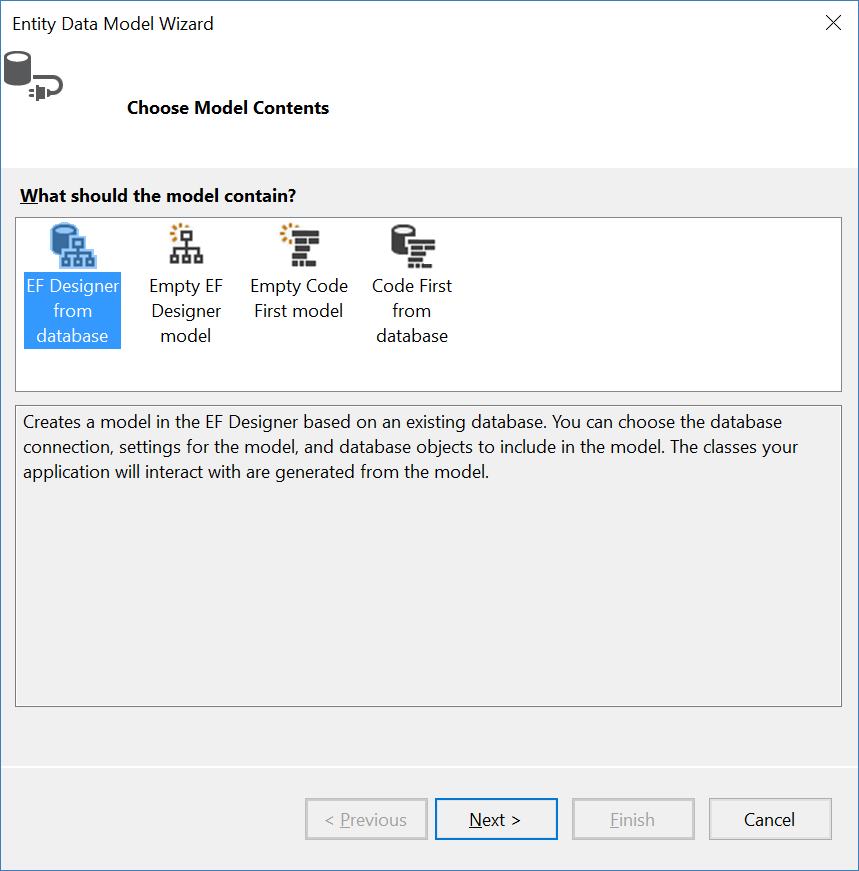
....

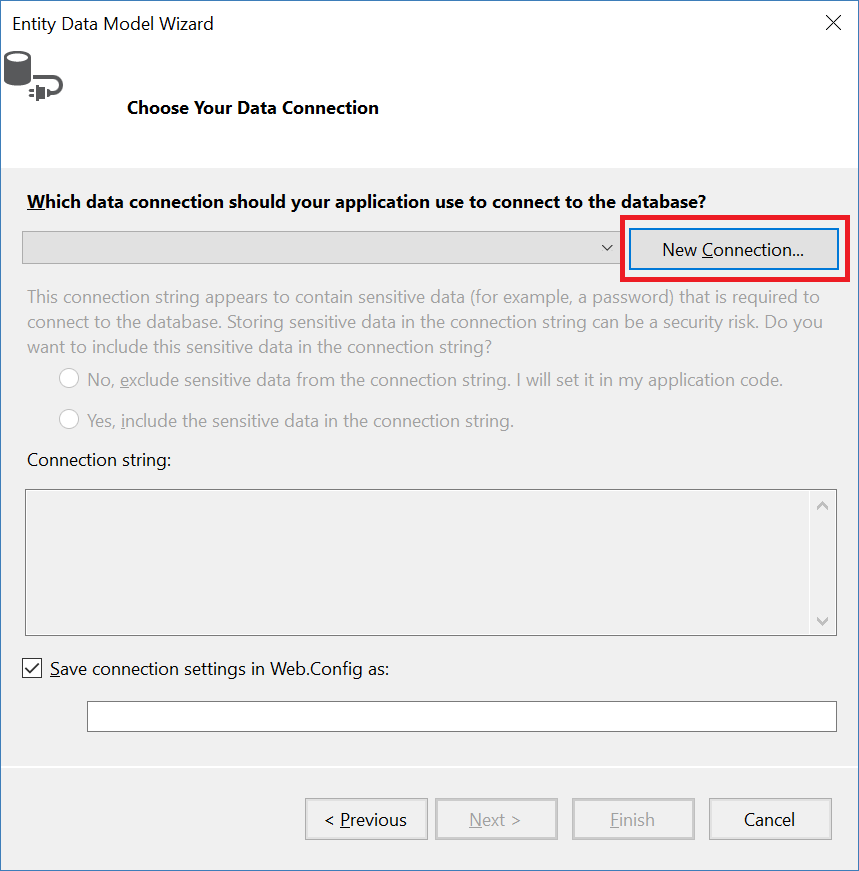
-->

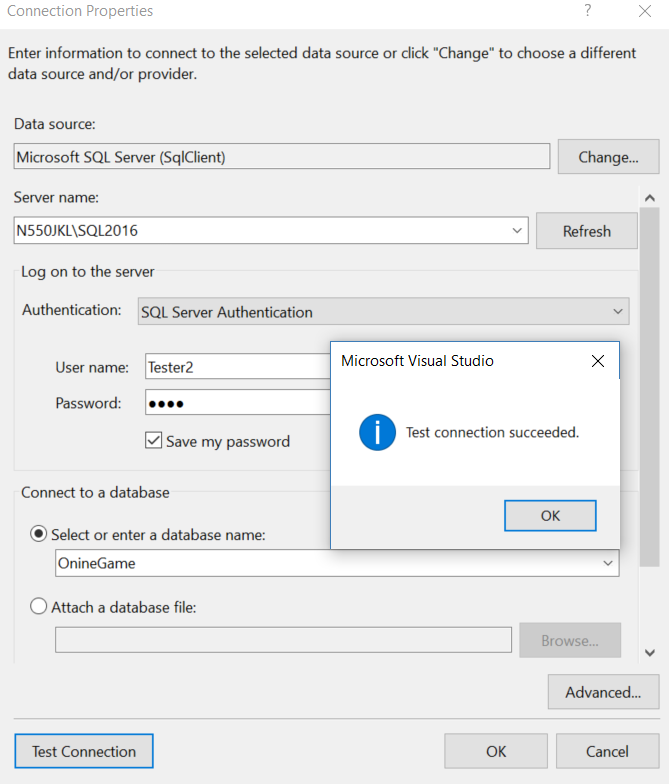
Save Connection settings in Web.Config as:

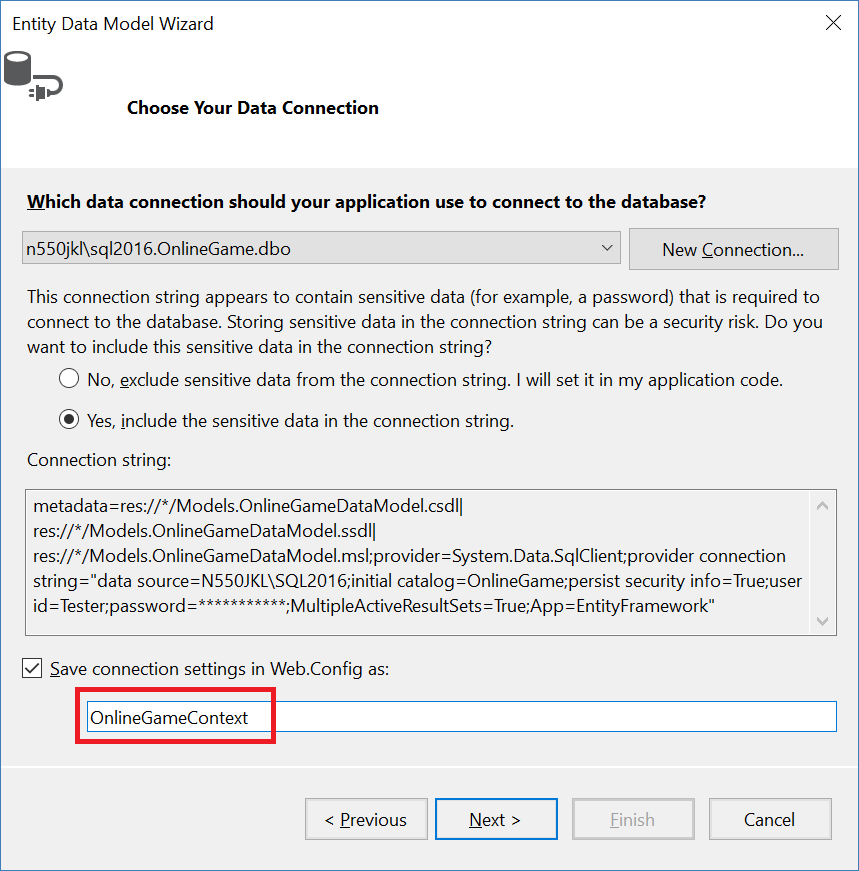
**OnlineGameContext**

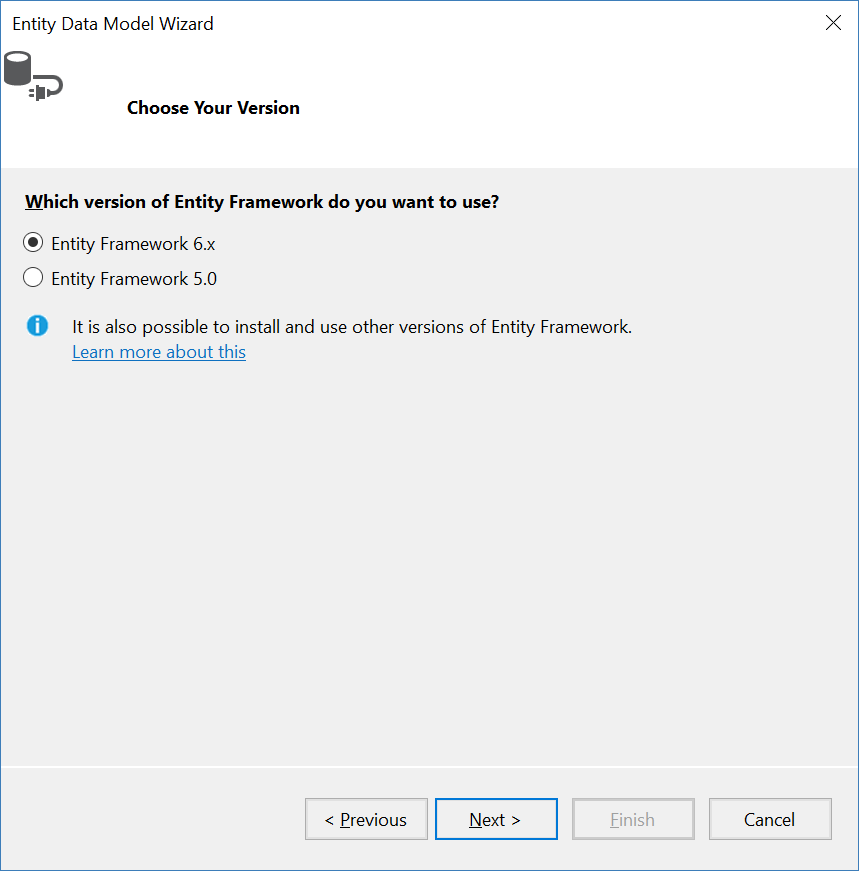


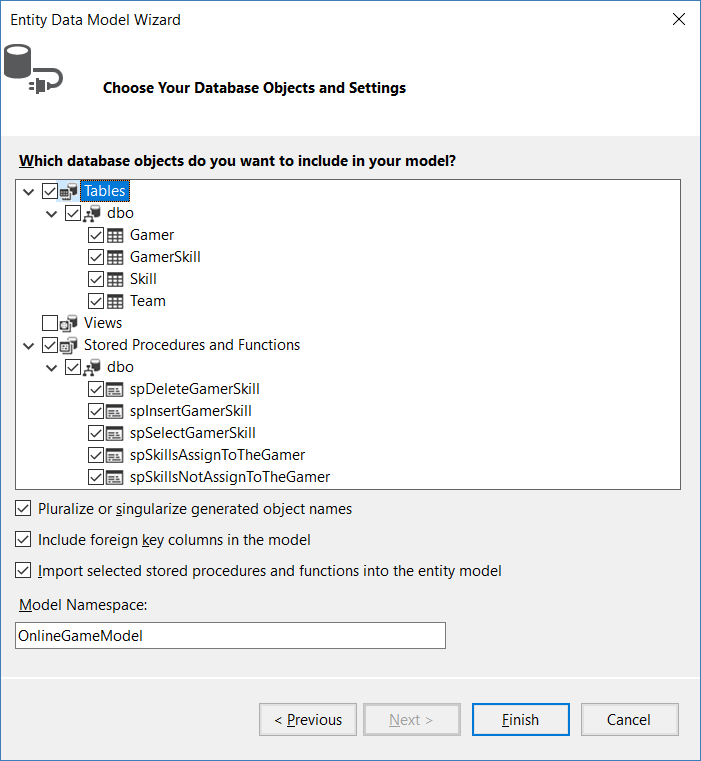


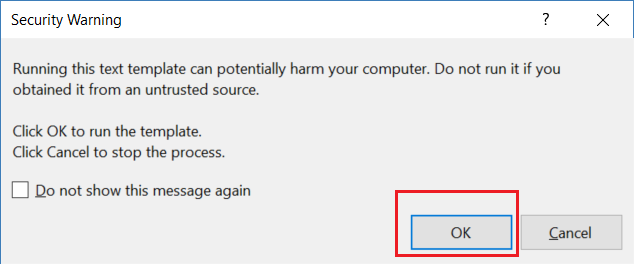


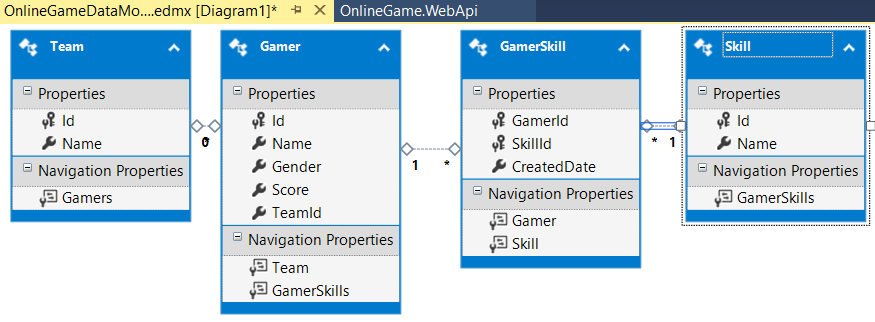












4. OnlineGame.WebApi - API Controller

4.1. OnlineGame.WebApi/App\_Start/WebApiConfig.cs - JSON Formatter

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Http.Formatting;

using System.Web.Http;

namespace OnlineGame.WebApi

{

    public static class WebApiConfig

    {

        public static void Register(HttpConfiguration config)

        {

            // Web API configuration and services

            // Web API routes

            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(

                name: "DefaultApi",

                routeTemplate: "api/{controller}/{id}",

                defaults: new { id = RouteParameter.Optional }

            );

            //Use JSON formatter as a PreserveReferencesHandling.

            JsonMediaTypeFormatter json = config.Formatters.JsonFormatter;

            json.SerializerSettings.PreserveReferencesHandling = Newtonsoft.Json.PreserveReferencesHandling.Objects;

            //Remove Xml Formatter

            config.Formatters.Remove(config.Formatters.XmlFormatter);

        }

    }

}

/\*

//JsonMediaTypeFormatter json = config.Formatters.JsonFormatter;

//json.SerializerSettings.PreserveReferencesHandling = Newtonsoft.Json.PreserveReferencesHandling.Objects;

//config.Formatters.Remove(config.Formatters.XmlFormatter);

Use JSON formatter as a PreserveReferencesHandling.

Remove Xml Formatter

Reference:

A.

[https://forums.asp.net/t/1983286.aspx?Web+API+error+The+ObjectContent+1+type+failed+to+serialize+the+response+body+for+content+type+application+xml+charset+utf+8](https://forums.asp.net/t/1983286.aspx?Web%2BAPI%2Berror%2BThe%2BObjectContent%2B1%2Btype%2Bfailed%2Bto%2Bserialize%2Bthe%2Bresponse%2Bbody%2Bfor%2Bcontent%2Btype%2Bapplication%2Bxml%2Bcharset%2Butf%2B8)+

B.

<https://stackoverflow.com/questions/23098191/failed-to-serialize-the-response-in-web-api-with-json>

\*/

4.2. OnlineGame.WebApi/Controllers/Api/GamerController.cs - Attribute routing

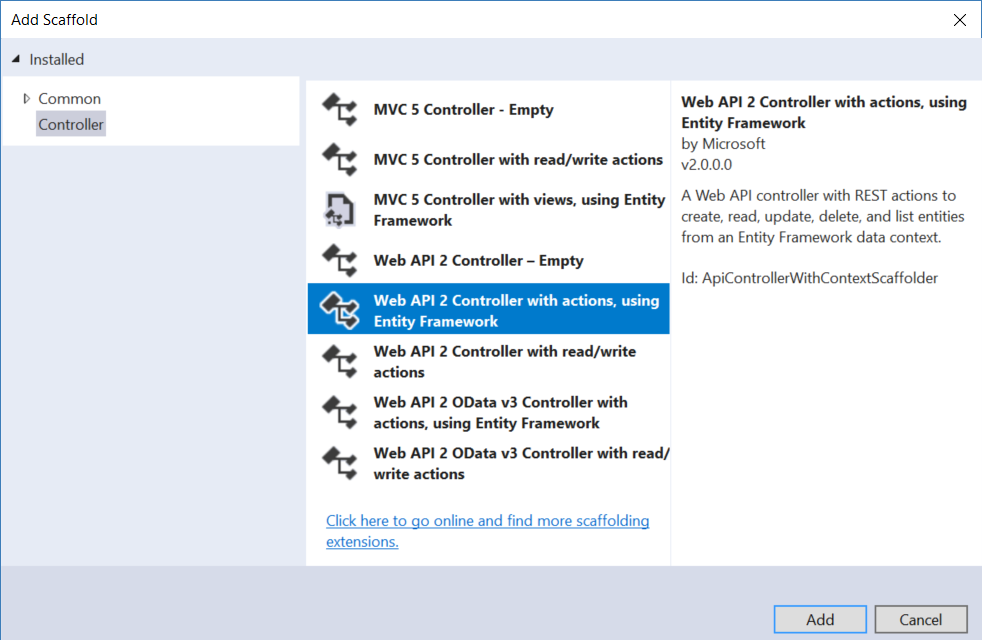
4.2.1. OnlineGame.WebApi/Controllers/Api/GamerController.cs - Attribute routing

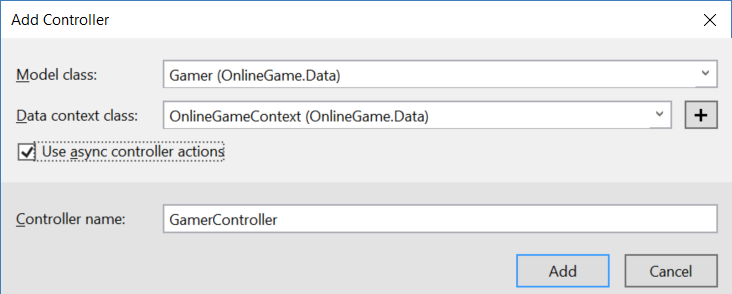
Controllers/Api  folder --> Right Click --> Add --> Controller

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

--> **GamerController**

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





4.2.2. OnlineGame.WebApi/Controllers/Api/GamerController.cs - Attribute routing

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.WebApi.Models;

namespace OnlineGame.WebApi.Controllers.Api

{

    public class GamerController : ApiController

    {

        private OnlineGameContext \_db = new OnlineGameContext();

        // GET: api/Gamer

        [HttpGet]

        public async Task<IEnumerable<Gamer>> GetGamers()

        {

            return await \_db.Gamers.ToListAsync();

        }

        // GET: api/Gamer/1

        //Convention-based routing.

        [HttpGet]

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> GetGamer(int id)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(id);

            if (gamer == null) return NotFound();  //404

            return Ok(gamer);   //200

        }

        [HttpGet]

        //Attribute Routing

        [Route("api/gamer/{id}/skills")]    // GET: api/gamer/1/skills

        public async Task<IHttpActionResult> GetGamerSkills(int id)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(id);

            if (gamer == null) return NotFound();  //404

            List<Skill> skills = await GetSkillsByGamerId(id);

            return Ok(skills);   //200

        }

        [HttpGet]

        [Route("api/gamer/skills/{id}")]    // GET: api/gamer/skills/1

        public async Task<IHttpActionResult> GetGamerSkills2(int id)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(id);

            if (gamer == null) return NotFound();  //404

            List<Skill> skills = await GetSkillsByGamerId(id);

            return Ok(skills);   //200

        }

        // PUT: api/Gamer/1

        [HttpPut]

        [ResponseType(typeof(void))]

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

        {

            if (!ModelState.IsValid) return BadRequest(ModelState);  //400

            //if (id != gamer.Id)   return BadRequest();

            //1.

            gamer.Id = id;

            \_db.Entry(gamer).State = EntityState.Modified;  //update the gamer

            //2.

            //Gamer currentGamer = await \_db.Gamers.FirstOrDefaultAsync(g => g.Id == id);

            //if (currentGamer == null) return NotFound();  //404

            //currentGamer.Name = gamer.Name;

            //currentGamer.Gender = gamer.Gender;

            //currentGamer.Score = gamer.Score;

            //currentGamer.GameMoney = gamer.GameMoney;

            try

            {

                await \_db.SaveChangesAsync();

                return Ok();    //200

            }

            catch (DbUpdateConcurrencyException)

            {

                if (!GamerExists(id)) return NotFound();  //404

                throw;

            }

        }

        // POST: api/Gamer

        [HttpPost]

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> PostGamer(Gamer gamer)

        {

            if (!ModelState.IsValid) return BadRequest(ModelState); //400

            \_db.Gamers.Add(gamer);

            await \_db.SaveChangesAsync();

            //Return Created/201.

            return CreatedAtRoute("DefaultApi", new { id = gamer.Id }, gamer);    //Created/201

        }

        // DELETE: api/Gamer/1

        [HttpDelete]

        [ResponseType(typeof(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

        }

        private async Task<List<Skill>> GetSkillsByGamerId(int gamerId)

        {

            IQueryable<GamerSkill> gamerSkills = \_db.Gamers

                .SelectMany(

                    g => g.GamerSkills, //The source of gamerSkill in second parameter

                    (g, gamerSkill) =>

                        new { GamerId = g.Id, GamerSkill = gamerSkill }) //Projection to a anonymous type

                .Where(gs => gs.GamerId == gamerId) //gamer id==gamerId

                .Select(gs => gs.GamerSkill); // Projection to GamerSkill Type

            List<Skill> skills =

                    await gamerSkills.Select(gamerSkill => \_db.Skills.FirstOrDefault(s => s.Id == gamerSkill.SkillId)).ToListAsync();

            //Projection to Skill

            return skills;

        }

        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

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

6.

Attribute routing

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

6.1.

E.g.

//public async Task<IHttpActionResult> GetGamer(int id){...}

....

//[Route("api/gamer/{id}/skills")]

//public async Task<IHttpActionResult> GetGamerSkills(int id){...}

When we call "api/gamer/1" and if we don't have Route attribute,

the API will be confused,

because both GetGamerSkills() and GetGamer() can map to "api/gamer/1".

Thus, we need Route attribute

[Route("api/gamer/{id}/skills")] will make GetGamerSkills() map to something like "api/gamer/1/skills".

Thus, GetGamer() can map to something like "api/gamer/1".

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

6.2.

In this case,

GetGamer() is using Convention-based routing.

GetGamerSkills() is using Attribute Routing.

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

6.3.

In

OnlineGame.WebApi/WebApiConfig.cs/WebApiConfig.cs

//config.MapHttpAttributeRoutes();

It enables Attribute Routing.

\*/

4.3. OnlineGame.WebApi/Controllers/Api/GamerTwoController.cs - RoutePrefix and Route attribute

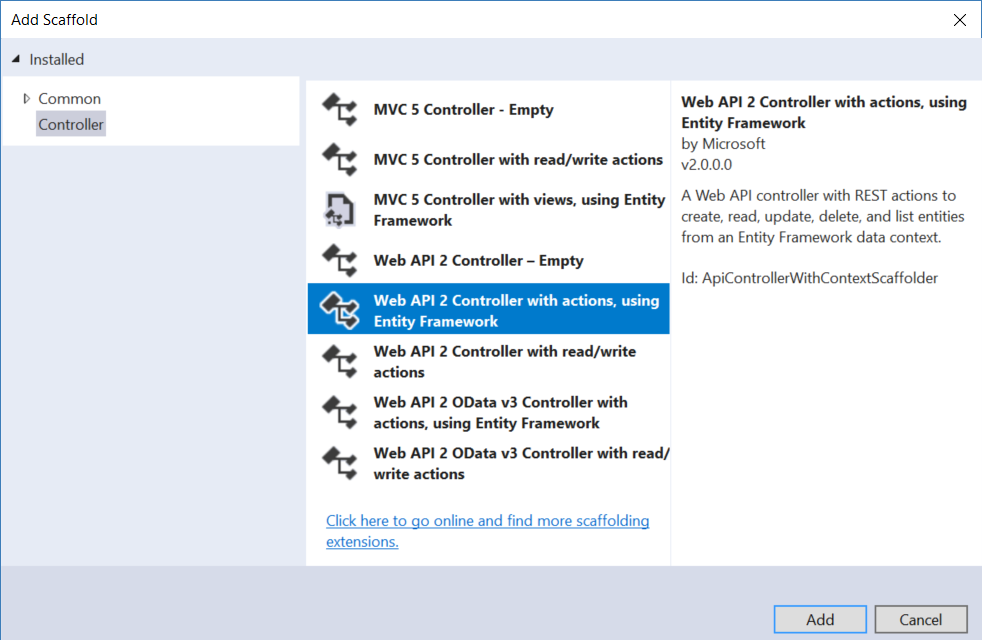
4.3.1. OnlineGame.WebApi/Controllers/Api/GamerTwoController.cs - RoutePrefix and Route attribute

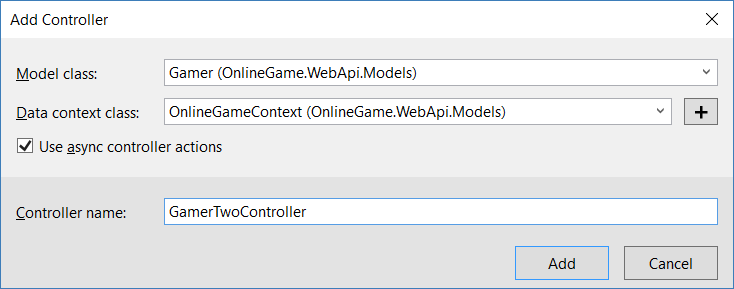
Controllers/Api  folder --> Right Click --> Add --> Controller

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

--> **GamerTwoController**

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





4.3.2. OnlineGame.WebApi/Controllers/Api/GamerTwoController.cs - RoutePrefix and Route attribute

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using OnlineGame.WebApi.Models;

namespace OnlineGame.WebApi.Controllers.Api

{

    [RoutePrefix("api/gamer2")]

    public class GamerTwoController : ApiController

    {

        private OnlineGameContext \_db = new OnlineGameContext();

        // GET: api/Gamertwo

        public IQueryable<Gamer> GetGamers()

        {

            return \_db.Gamers;

        }

        // GET: api/Gamer2

        [Route("")]

        public IQueryable<Gamer> GetGamers2()

        {

            return \_db.Gamers;

        }

        // GET: api/gamer2/api/gamer2

        [Route("api/gamer2")]

        public IQueryable<Gamer> GetGamers3()

        {

            return \_db.Gamers;

        }

        // GET: api/gamer2/api/getGamers

        [Route("api/getGamers")]

        public IQueryable<Gamer> GetGamers4()

        {

            return \_db.Gamers;

        }

        // GET: api/getGamers

        [Route("~/api/getGamers")]

        public IQueryable<Gamer> GetGamers5()

        {

            return \_db.Gamers;

        }

        // GET: api/gamerTwo/1

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> GetGamer(int id)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(id);

            if (gamer == null) return NotFound();  //404

            return Ok(gamer);   //200

        }

        // GET: api/gamer2/1

        [Route("{id}")]

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> GetGamer2(int id)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(id);

            if (gamer == null) return NotFound();  //404

            return Ok(gamer);   //200

        }

        // GET: api/gamer2/api/gamer2/1

        [ResponseType(typeof(Gamer))]

        [Route("api/gamer2/{id}")]

        public async Task<IHttpActionResult> GetGamer3(int id)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(id);

            if (gamer == null) return NotFound();  //404

            return Ok(gamer);   //200

        }

        // GET: api/gamer2GetGamerById/1

        [ResponseType(typeof(Gamer))]

        [Route("~/api/gamer2GetGamerById/{id}")]

        public async Task<IHttpActionResult> GetGamer4(int id)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(id);

            if (gamer == null) return NotFound();  //404

            return Ok(gamer);   //200

        }

        [HttpGet]

        [Route("api/gamer2/{gamerId}/skills")]    // GET: api/gamer2/api/gamer2/1/skills

        public async Task<IHttpActionResult> GetGamerSkills(int gamerId)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(gamerId);

            if (gamer == null) return NotFound();  //404

            List<Skill> skills = await GetSkillsByGamerId(gamerId);

            return Ok(skills);   //200

        }

        [HttpGet]

        [Route("api/gamer2/skills/{gamerId}")]    // GET: api/gamer2/api/gamer2/skills/1

        public async Task<IHttpActionResult> GetGamerSkills2(int gamerId)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(gamerId);

            if (gamer == null) return NotFound();  //404

            List<Skill> skills = await GetSkillsByGamerId(gamerId);

            return Ok(skills);   //200

        }

        [HttpGet]

        [Route("skills/{gamerId}")]    // GET: api/gamer2/skills/1

        public async Task<IHttpActionResult> GetGamerSkills3(int gamerId)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(gamerId);

            if (gamer == null) return NotFound();  //404

            List<Skill> skills = await GetSkillsByGamerId(gamerId);

            return Ok(skills);   //200

        }

        [HttpGet]

        [Route("~/api/getGamerSkillsByGamerId/{gamerId}")]    // GET: api/getGamerSkillsByGamerId/1

        public async Task<IHttpActionResult> GetGamerSkills4(int gamerId)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(gamerId);

            if (gamer == null) return NotFound();  //404

            List<Skill> skills = await GetSkillsByGamerId(gamerId);

            return Ok(skills);   //200

        }

        private async Task<List<Skill>> GetSkillsByGamerId(int gamerId)

        {

            IQueryable<GamerSkill> gamerSkills = \_db.Gamers

                .SelectMany(

                    g => g.GamerSkills, //The source of gamerSkill in second parameter

                    (g, gamerSkill) =>

                        new { GamerId = g.Id, GamerSkill = gamerSkill }) //Projection to a anonymous type

                .Where(gs => gs.GamerId == gamerId) //gamer id==gamerId

                .Select(gs => gs.GamerSkill); // Projection to GamerSkill Type

            List<Skill> skills =

                    await gamerSkills.Select(gamerSkill => \_db.Skills.FirstOrDefault(s => s.Id == gamerSkill.SkillId)).ToListAsync();

            //Projection to Skill

            return skills;

        }

        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;

        }

    }

}

/\*

7.

RoutePrefix and Route attribute

//[RoutePrefix("api/gamer2")]

RoutePrefix attribute is for route prefix at the controller level.

Route attribute use that route prefix plus its own route value.

//[Route("~/api/getGamerSkillsByGamerId/{gamerId}")]

if you want to  override the route prefix,

just use ~ (tilde) symbol

\*/

4.4. OnlineGame.WebApi/Controllers/Api/GamerThreeController.cs - attribute routing constraints

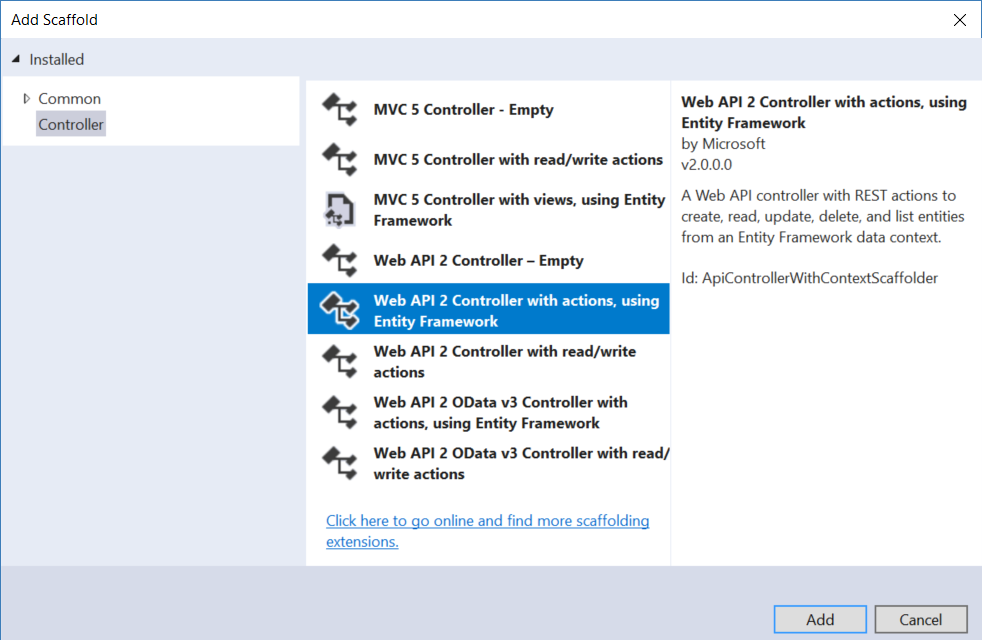
4.4.1. OnlineGame.WebApi/Controllers/Api/GamerThreeController.cs - attribute routing constraints

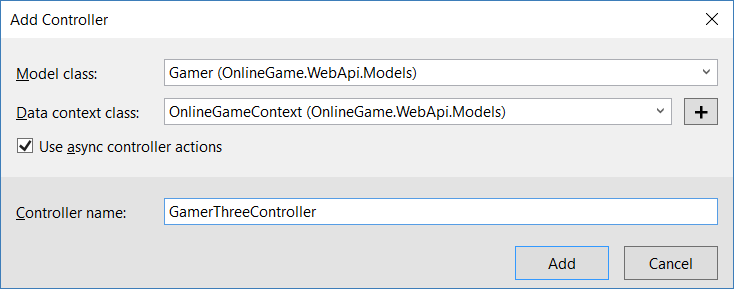
Controllers/Api  folder --> Right Click --> Add --> Controller

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

--> **GamerThreeController**

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





4.4.2. OnlineGame.WebApi/Controllers/Api/GamerThreeController.cs - attribute routing constraints

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using OnlineGame.WebApi.Models;

namespace OnlineGame.WebApi.Controllers.Api

{

    [RoutePrefix("api/gamer3")]

    public class GamerThreeController : ApiController

    {

        private OnlineGameContext \_db = new OnlineGameContext();

        // GET: api/GamerThree

        public IQueryable<Gamer> GetGamers()

        {

            return \_db.Gamers;

        }

        // GET: api/GamerThree/1

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> GetGamer(int id)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(id);

            if (gamer == null) return NotFound();   //404

            return Ok(gamer);

        }

        // GET: api/gamer3/GetGamerBySomething/2

        [Route("GetGamerBySomething/{gamerId:int}")]

        public async Task<IHttpActionResult> GetGamerBySomething(int gamerId)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(gamerId);

            if (gamer == null) return NotFound();   //404

            return Ok(gamer);

        }

        // GET: api/gamer3/GetGamerBySomething/male

        //[Route("GetGamerBySomething/{gender:string}")]    //Error, string type is not valid

        [Route("GetGamerBySomething/{gender:alpha}")]

        //alpha means uppercase or lowercase alphabet.

        public async Task<IHttpActionResult> GetGamerBySomething(string gender)

        {

            List<Gamer> gamer =

                await \_db.Gamers.Where(

                    g => g.Gender.ToLower().Equals(gender.ToLower()))   //it is not case sensitive

                    .ToListAsync();

            return Ok(gamer);

        }

        [ResponseType(typeof(Gamer))]

        [Route("{gamerId}")]    // GET: api/gamer3/1

        public async Task<IHttpActionResult> GetGamer2(int gamerId)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(gamerId);

            if (gamer == null) return NotFound();   //404

            return Ok(gamer);

        }

        // GET: api/gamer3/getGamerById/1

        // gamerId must be int and min is 2

        [Route("getGamerById/{gamerId:int:min(2)}")]

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> GetGamerById(int gamerId)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(gamerId);

            if (gamer == null) return NotFound();   //404

            return Ok(gamer);

        }

        // GET: api/gamer3/getGamerById2/1

        // gamerId must be int and min is 2, max is 5

        [Route("getGamerById2/{gamerId:int:min(2):max(5)}")]

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> GetGamerById2(int gamerId)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(gamerId);

            if (gamer == null) return NotFound();   //404

            return Ok(gamer);

        }

        // GET: api/gamer3/getGamerById3/1

        // gamerId must be int and min is 2, max is 5

        [Route("getGamerById3/{gamerId:range(2,5)}")]

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> GetGamerById3(int gamerId)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(gamerId);

            if (gamer == null) return NotFound();   //404

            return Ok(gamer);

        }

        // GET: api/gamer3/getGamersByGender/female

        //[Route("getGamersByGender/{gender:string}")]    //Error, string type is not valid

        [Route("getGamersByGender/{gender:alpha}")] //alpha means uppercase or lowercase alphabet characters.

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> GetGamersByGender(string gender)

        {

            List<Gamer> gamer =

                await \_db.Gamers.Where(

                    g => g.Gender.ToLower().Equals(gender.ToLower()))   //it is not case sensitive

                    .ToListAsync();

            return Ok(gamer);

        }

        // GET: api/gamer3/getGamersByGender2/female     //will return nothing, it is case sensitive

        // GET: api/gamer3/getGamersByGender2/Female

        [Route("getGamersByGender2/{gender:alpha}")]

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> GetGamersByGender2(string gender)

        {

            List<Gamer> gamer =

                await \_db.Gamers.Where(

                    g => g.Gender.Equals(gender))   //it is case sensitive

                    .ToListAsync();

            return Ok(gamer);

        }

        // GET: api/gamer3/getGamersByGender3/female        //404

        // GET: api/gamer3/getGamersByGender3/male

        [Route("getGamersByGender3/{gender:alpha:maxlength(5)}")]

        //alpha means uppercase or lowercase alphabet characters.

        //max alpha length is 5

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> GetGamersByGender3(string gender)

        {

            List<Gamer> gamer =

                await \_db.Gamers.Where(

                    g => g.Gender.ToLower().Equals(gender.ToLower()))

                    .ToListAsync();

            return Ok(gamer);

        }

        // GET: api/gamer3/getGamersByGender4/female

        // GET: api/gamer3/getGamersByGender4/male      //404

        [Route("getGamersByGender4/{gender:alpha:minlength(5):maxlength(7)}")]

        //alpha means uppercase or lowercase alphabet characters.

        //max alpha length is 7, and min length is 5.

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> GetGamersByGender4(string gender)

        {

            List<Gamer> gamer =

                await \_db.Gamers.Where(

                    g => g.Gender.ToLower().Equals(gender.ToLower()))   //it is not case sensitive

                    .ToListAsync();

            return Ok(gamer);

        }

        protected override void Dispose(bool disposing)

        {

            if (disposing) \_db.Dispose();

            base.Dispose(disposing);

        }

        private bool GamerExists(int id)

        {

            return \_db.Gamers.Count(e => e.Id == id) > 0;

        }

    }

}

/\*

8.

attribute routing constraints

Reference:

<https://docs.microsoft.com/en-us/aspnet/web-api/overview/web-api-routing-and-actions/attribute-routing-in-web-api-2#route-constraints>

Routing constraints can apply to decimal, double, float, long, bool...etc.

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

8.1.

//// GET: api/gamer3/GetGamerBySomething/2

//[Route("GetGamerBySomething/{gamerId:int}")]

//public async Task<IHttpActionResult> GetGamerBySomething(int gamerId)

int means integer

...

//// GET: api/gamer3/GetGamerBySomething/male

////[Route("GetGamerBySomething/{gender:string}")]    //Error, string type is not valid

//[Route("GetGamerBySomething/{gender:alpha}")]

//public async Task<IHttpActionResult> GetGamerBySomething(string gender)

alpha means uppercase or lowercase alphabet.

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

8.2.

//[Route("getGamerById/{gamerId:int:min(2)}")]

//public async Task<IHttpActionResult> GetGamerById(int gamerId)

GET: api/gamer3/getGamerById/1

gamerId must be int and min is 2

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

8.3.

//[Route("getGamerById2/{gamerId:int:min(2):max(5)}")]

//public async Task<IHttpActionResult> GetGamerById2(int gamerId)

GET: api/gamer3/getGamerById2/1

gamerId must be int and min is 2, max is 5

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

8.4.

//[Route("getGamerById3/{gamerId:range(2,5)}")]

//public async Task<IHttpActionResult> GetGamerById3(int gamerId)

GET: api/gamer3/getGamerById3/1

gamerId must be int and min is 2, max is 5

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

8.5.

////[Route("getGamersByGender/{gender:string}")]    //Error, string type is not valid

//[Route("getGamersByGender/{gender:alpha}")]

//public async Task<IHttpActionResult> GetGamersByGender(string gender)

alpha means uppercase or lowercase alphabet characters.

GET: api/gamer3/getGamersByGender/female

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

8.7.

//[Route("getGamersByGender3/{gender:alpha:maxlength(5)}")]

//public async Task<IHttpActionResult> GetGamersByGender3(string gender)

GET: api/gamer3/getGamersByGender3/female        //404

GET: api/gamer3/getGamersByGender3/male

alpha means uppercase or lowercase alphabet characters.

max alpha length is 5

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

8.8.

//[Route("getGamersByGender3/{gender:alpha:maxlength(5)}")]

//public async Task<IHttpActionResult> GetGamersByGender3(string gender)

GET: api/gamer3/getGamersByGender3/female        //404

GET: api/gamer3/getGamersByGender3/male

alpha means uppercase or lowercase alphabet characters.

max alpha length is 5

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

8.9.

//[Route("getGamersByGender4/{gender:alpha:minlength(5):maxlength(7)}")]

//public async Task<IHttpActionResult> GetGamersByGender4(string gender)

GET: api/gamer3/getGamersByGender4/female

GET: api/gamer3/getGamersByGender4/male      //404

alpha means uppercase or lowercase alphabet characters.

max alpha length is 7, and min length is 5.

\*/

4.5. OnlineGame.WebApi/Controllers/Api/GamerFourController.cs - Route names

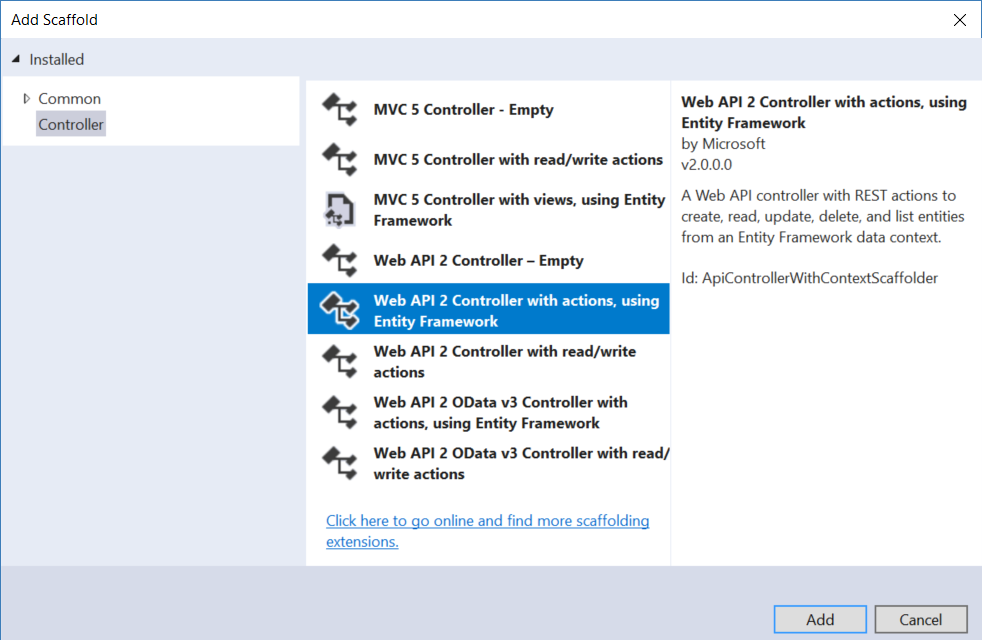
4.5.1. OnlineGame.WebApi/Controllers/Api/GamerFourController.cs - Route names

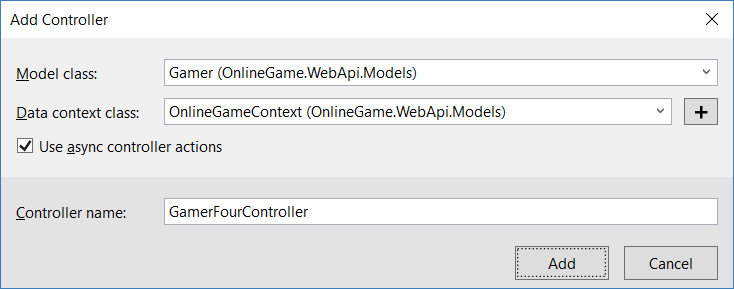
Controllers/Api  folder --> Right Click --> Add --> Controller

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

--> **GamerFourController**

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





4.5.2. OnlineGame.WebApi/Controllers/Api/GamerFourController.cs - Route names

using System;

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.WebApi.Models;

namespace OnlineGame.WebApi.Controllers.Api

{

    [RoutePrefix("api/gamer4")]

    public class GamerFourController : ApiController

    {

        private OnlineGameContext \_db = new OnlineGameContext();

        // GET: api/GamerFour

        public IQueryable<Gamer> GetGamers()

        {

            return \_db.Gamers;

        }

        // GET: api/GamerFour/1

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> GetGamer(int id)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(id);

            if (gamer == null) return NotFound();   //404

            return Ok(gamer);

        }

        // GET: api/Gamer4/1

        [ResponseType(typeof(Gamer))]

        [Route("{id:int}", Name = "GetGamerById")]

        public async Task<IHttpActionResult> GetGamerById(int id)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(id);

            if (gamer == null) return NotFound();   //404

            return Ok(gamer);

        }

        //IHttpActionResult is from Web API 2

        // POST: api/GamerFour

        [HttpPost]

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> PostGamer(Gamer gamer)

        {

            if (!ModelState.IsValid) return BadRequest(ModelState); //400

            \_db.Gamers.Add(gamer);

            await \_db.SaveChangesAsync();

            //Return Created/201.

            return CreatedAtRoute("DefaultApi", new { id = gamer.Id }, gamer);    //Created/201

        }

        // POST: api/Gamer4/AddGamer

        [HttpPost]

        [Route("AddGamer")]

        public async Task<HttpResponseMessage> AddGamer(Gamer gamer)

        {

            if (!ModelState.IsValid)

                return Request.CreateErrorResponse(HttpStatusCode.BadRequest,

                   "ModelState is invalid");    //400

            \_db.Gamers.Add(gamer);

            await \_db.SaveChangesAsync();

            //Return Created/201.

            HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.Created);

            response.Headers.Location = new Uri(Request.RequestUri + "/" + gamer.Id);

            return response;    //Created/201

        }

        //IHttpActionResult is from Web API 2

        // POST: api/Gamer4/AddGamer2

        [Route("AddGamer2")]

        [HttpPost]

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> AddGamer2(Gamer gamer)

        {

            if (!ModelState.IsValid) return BadRequest(ModelState); //400

            \_db.Gamers.Add(gamer);

            await \_db.SaveChangesAsync();

            //Return Created/201.

            return CreatedAtRoute("DefaultApi", new { id = gamer.Id }, gamer);    //Created/201

        }

        // POST: api/Gamer4/AddGamer3

        [HttpPost]

        [Route("AddGamer3")]

        public async Task<HttpResponseMessage> AddGamer3(Gamer gamer)

        {

            if (!ModelState.IsValid)

                return Request.CreateErrorResponse(HttpStatusCode.BadRequest,

                   "ModelState is invalid");    //400

            \_db.Gamers.Add(gamer);

            await \_db.SaveChangesAsync();

            //Return Created/201.

            HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.Created);

            response.Headers.Location = new

                        Uri(Url.Link("GetGamerById", new { id = gamer.Id }));

            return response;    //Created/201

        }

        // POST: api/Gamer4/AddGamer4

        [HttpPost]

        [Route("AddGamer4")]

        public async Task<IHttpActionResult> AddGamer4(Gamer gamer)

        {

            if (!ModelState.IsValid) return BadRequest(ModelState); //400

            \_db.Gamers.Add(gamer);

            await \_db.SaveChangesAsync();

            //Return Created/201.

            return CreatedAtRoute("GetGamerById", new { id = gamer.Id }, gamer);    //Created/201

        }

        protected override void Dispose(bool disposing)

        {

            if (disposing) \_db.Dispose();

            base.Dispose(disposing);

        }

        private bool GamerExists(int id)

        {

            return \_db.Gamers.Count(e => e.Id == id) > 0;

        }

    }

}

/\*

9.

Route names

9.1.

E.g.

//[Route("{id:int}", Name = "GetGamerById")]

//public async Task<IHttpActionResult> GetGamerById(int id)

...

//HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.Created);

//response.Headers.Location = new

//    Uri(Url.Link("GetGamerById", new { id = gamer.Id }));

...

//return CreatedAtRoute("GetGamerById", new { id = gamer.Id }, gamer);    //Created/201

9.2.

//return CreatedAtRoute("DefaultApi", new { id = gamer.Id }, gamer);    //Created/201

...

//HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.Created);

//response.Headers.Location = new Uri(Request.RequestUri + "/" + gamer.Id);

\*/

4.5.3. Post Request

4.5.3.1. Post Request - public async Task<IHttpActionResult> PostGamer(Gamer gamer)

//IHttpActionResult is from Web API 2

// POST: api/GamerFour

[HttpPost]

[ResponseType(typeof(Gamer))]

public async Task<IHttpActionResult> PostGamer(Gamer gamer)

{

    if (!ModelState.IsValid) return BadRequest(ModelState); //400

    \_db.Gamers.Add(gamer);

    await \_db.SaveChangesAsync();

    //Return Created/201.

    return CreatedAtRoute("DefaultApi", new { id = gamer.Id }, gamer);    //Created/201

}

-->

Post: api/GamerFour

<http://localhost:59537/api/GamerFour>

Request Header:

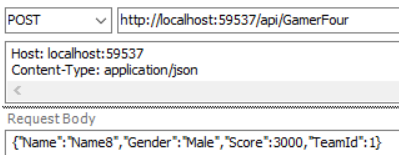
Host: localhost:59537

Content-Type: application/json

Request Body:

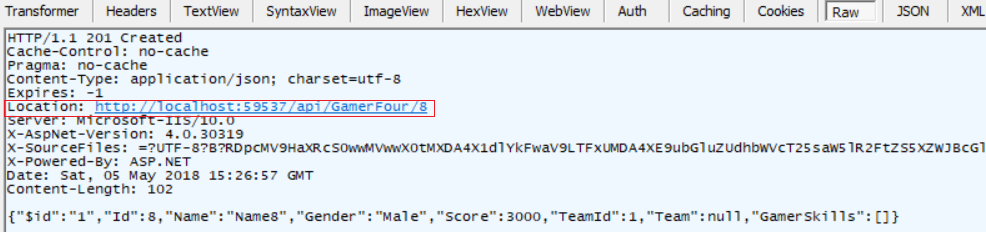
{"Name":"Name8","Gender":"Male","Score":3000,"TeamId":1}

-->



-->





4.5.3.2. Post Request - public async Task<HttpResponseMessage> AddGamer(Gamer gamer) - Bug

// POST: api/Gamer4/AddGamer

[HttpPost]

[Route("AddGamer")]

public async Task<HttpResponseMessage> AddGamer(Gamer gamer)

{

    if (!ModelState.IsValid)

        return Request.CreateErrorResponse(HttpStatusCode.BadRequest,

            "ModelState is invalid");    //400

    \_db.Gamers.Add(gamer);

    await \_db.SaveChangesAsync();

    //Return Created/201.

    HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.Created);

    response.Headers.Location = new Uri(Request.RequestUri + "/" + gamer.Id);

    return response;    //Created/201

}

-->

Post: api/Gamer4/AddGamer

<http://localhost:59537/api/Gamer4/AddGamer>

Request Header:

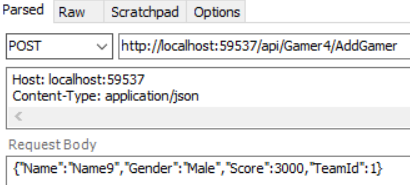
Host: localhost:59537

Content-Type: application/json

Request Body:

{"Name":"Name9","Gender":"Male","Score":3000,"TeamId":1}

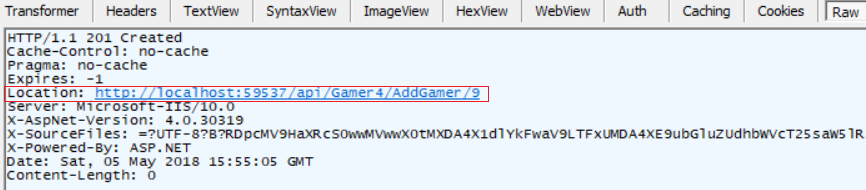
-->



-->



-->



**The location is totally not right.**

4.5.3.3. Post Request - public async Task<IHttpActionResult> AddGamer2(Gamer gamer)  - Bug

//IHttpActionResult is from Web API 2

// POST: api/Gamer4/AddGamer2

[Route("AddGamer2")]

[HttpPost]

[ResponseType(typeof(Gamer))]

public async Task<IHttpActionResult> AddGamer2(Gamer gamer)

{

    if (!ModelState.IsValid) return BadRequest(ModelState); //400

    \_db.Gamers.Add(gamer);

    await \_db.SaveChangesAsync();

    //Return Created/201.

    return CreatedAtRoute("DefaultApi", new { id = gamer.Id }, gamer);    //Created/201

}

-->

Post: api/Gamer4/AddGamer2

<http://localhost:59537/api/Gamer4/AddGamer2>

Request Header:

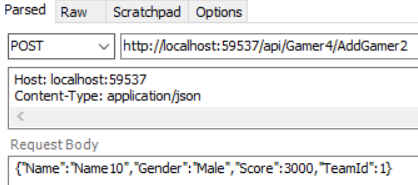
Host: localhost:59537

Content-Type: application/json

Request Body:

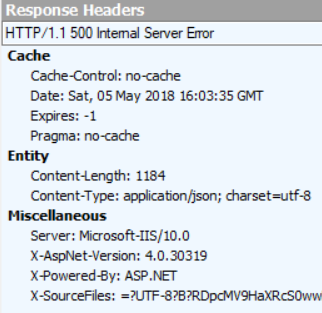
{"Name":"Name10","Gender":"Male","Score":3000,"TeamId":1}

-->



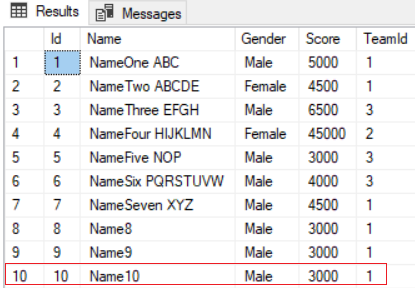
-->





-->

But SQL still has the record



4.5.3.4. Post Request - public async Task<HttpResponseMessage> AddGamer3(Gamer gamer) - Fix Bug

// GET: api/Gamer4/1

[ResponseType(typeof(Gamer))]

[Route("{id:int}", Name = "GetGamerById")]

public async Task<IHttpActionResult> GetGamerById(int id)

{

    Gamer gamer = await \_db.Gamers.FindAsync(id);

    if (gamer == null) return NotFound();   //404

    return Ok(gamer);

}

// POST: api/Gamer4/AddGamer3

[HttpPost]

[Route("AddGamer3")]

public async Task<HttpResponseMessage> AddGamer3(Gamer gamer)

{

    if (!ModelState.IsValid)

        return Request.CreateErrorResponse(HttpStatusCode.BadRequest,

            "ModelState is invalid");    //400

    \_db.Gamers.Add(gamer);

    await \_db.SaveChangesAsync();

    //Return Created/201.

    HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.Created);

    response.Headers.Location = new

                Uri(Url.Link("GetGamerById", new { id = gamer.Id }));

    return response;    //Created/201

}

-->

Post: api/Gamer4/AddGamer3

<http://localhost:59537/api/Gamer4/AddGamer3>

Request Header:

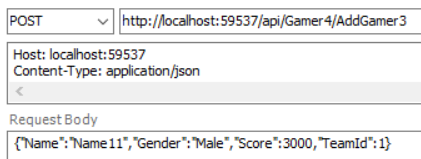
Host: localhost:59537

Content-Type: application/json

Request Body:

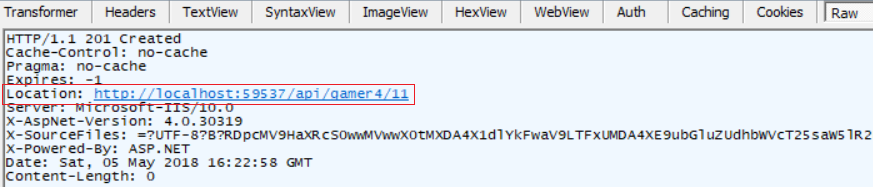
{"Name":"Name11","Gender":"Male","Score":3000,"TeamId":1}

-->



-->





4.5.3.5. Post Request - public async Task<IHttpActionResult> AddGamer4(Gamer gamer) - Fix Bug

// GET: api/Gamer4/1

[ResponseType(typeof(Gamer))]

[Route("{id:int}", Name = "GetGamerById")]

public async Task<IHttpActionResult> GetGamerById(int id)

{

    Gamer gamer = await \_db.Gamers.FindAsync(id);

    if (gamer == null) return NotFound();   //404

    return Ok(gamer);

}

// POST: api/Gamer4/AddGamer4

[HttpPost]

[Route("AddGamer4")]

public async Task<IHttpActionResult> AddGamer4(Gamer gamer)

{

    if (!ModelState.IsValid) return BadRequest(ModelState); //400

    \_db.Gamers.Add(gamer);

    await \_db.SaveChangesAsync();

    //Return Created/201.

    return CreatedAtRoute("GetGamerById", new { id = gamer.Id }, gamer);    //Created/201

}

-->

Post: api/Gamer4/AddGamer4

<http://localhost:59537/api/Gamer4/AddGamer4>

Request Header:

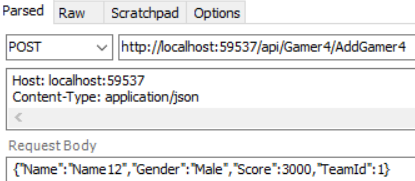
Host: localhost:59537

Content-Type: application/json

Request Body:

{"Name":"Name12","Gender":"Male","Score":3000,"TeamId":1}

-->



-->



Graphical user interface, text, application, Word

Description automatically generated

4.6. OnlineGame.WebApi/Controllers/Api/GamerFiveController.cs - Route names

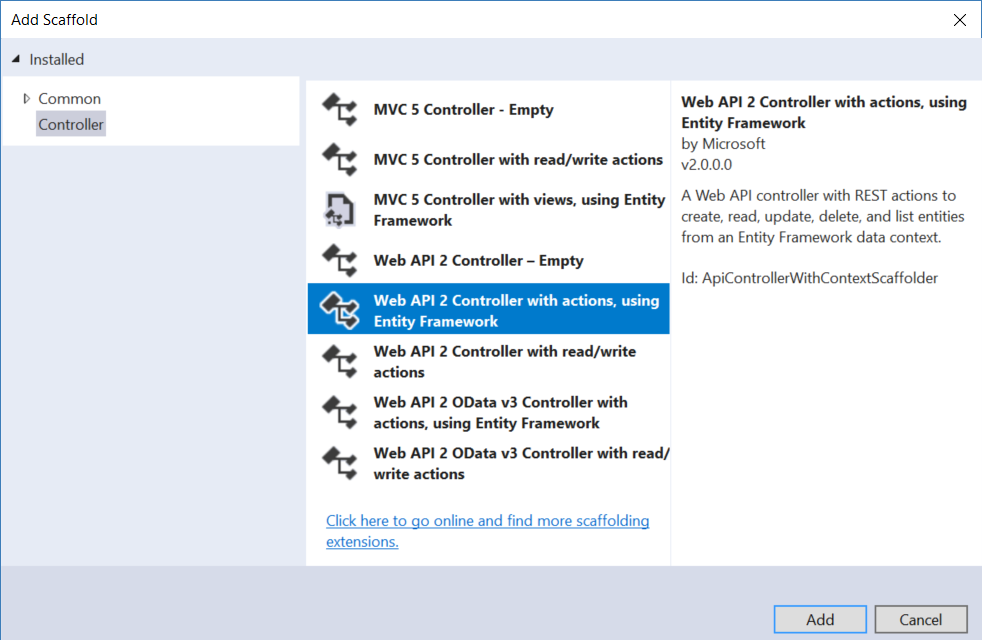
4.6.1. OnlineGame.WebApi/Controllers/Api/GamerFiveController.cs

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

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

--> **GamerFiveController**

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



Graphical user interface, text, application, email

Description automatically generated

4.6.2. OnlineGame.WebApi/Controllers/Api/GamerFiveController.cs

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.WebApi.Models;

namespace OnlineGame.WebApi.Controllers.Api

{

    [RoutePrefix("api/gamer5")]

    public class GamerFiveController : ApiController

    {

        private OnlineGameContext \_db = new OnlineGameContext();

        // GET: api/gamer5

        [Route("")]

        public IQueryable<Gamer> GetGamers()

        {

            return \_db.Gamers;

        }

        // GET: api/gamer5/GetGamers2

        [Route("GetGamers2")]

        public IHttpActionResult GetGamers2()

        {

            return Ok(\_db.Gamers);

        }

        // GET: api/gamer5/GetGamers3

        [Route("GetGamers3")]

        public HttpResponseMessage GetGamers3()

        {

            return Request.CreateResponse(\_db.Gamers);

        }

        // GET: api/gamer5/GetGamer/1

        [Route("GetGamer/{id:int}")]

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> GetGamer(int id)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(id);

            if (gamer == null) return NotFound();   //404

            return Ok(gamer);

        }

        // GET: api/gamer5/GetGamer2/1

        [Route("GetGamer2/{id:int}")]

        [ResponseType(typeof(Gamer))]

        public async Task<HttpResponseMessage> GetGamer2(int id)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(id);

            if (gamer == null)

                return Request.CreateErrorResponse(HttpStatusCode.NotFound,

                "Gamer not found"); //404

            return Request.CreateResponse(gamer);

        }

        // GET: api/gamer5/GetGamer3/1

        [Route("GetGamer3/{id:int}")]

        [ResponseType(typeof(Gamer))]

        public async Task<IHttpActionResult> GetGamer3(int id)

        {

            Gamer gamer = await \_db.Gamers.FindAsync(id);

            if (gamer == null)

                return Content(HttpStatusCode.NotFound, "Gamer not found"); //404

            return Ok(gamer);

        }

        protected override void Dispose(bool disposing)

        {

            if (disposing) \_db.Dispose();

            base.Dispose(disposing);

        }

        private bool GamerExists(int id)

        {

            return \_db.Gamers.Count(e => e.Id == id) > 0;

        }

    }

}

/\*

10.

IHttpActionResult vs HttpResponseMessage

10.1.

IHttpActionResult

10.1.1.

HttpResponseMessage is from Web API 1

IHttpActionResult is from Web API 2

10.1.2.

IHttpActionResult make code cleaner.

10.1.3.

The following type implements IHttpActionResult interface.

Unauthorized()

BadRequest()

NotFound()

Created()

OK()

InternalServerError()

\*/