(T2)討論EntityFramework  
CourseGUID: 8503b39c-5887-4634-8291-facfb3117924  
=======================================================================  
(T2)討論EntityFramework  
=======================================================================  
0. Summary

-----------

1. Create DB: OnlineGame

1.1. Create DB: OnlineGame

1.2. Create Gamer Table

1.3. Create Database level security login

-----------

2. Entity Framework

2.1. Install Entity Framework

2.2. Web.config : Add Connection String

2.3. Data/OnlineGameContext.cs

2.4. Models/Gamer.cs

2.5. Controllers/GamerController.cs

2.6. Views/Gamer/Details.cshtml

2.7. URL  
=======================================================================

0. Summary

\* 完全手寫Entity Framework連接資料庫

    \* 身為一個專業的工程師，你不能只會使用自動生成而不求甚解，該部分不依靠任何工具，讓你完全手寫出Database First的Entity Framework程式碼。這和市面上那些拉來拉去自動生成的課程是完全不一樣的。你只有透過完全手寫，才能完善你的理解。本課程就是要讓你完美K.O.資料庫連結觀念。

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

This is the continuous of the previous torrential.

In this tutorial, we will discuss

\* Entity Framework

=========================================

由於MVC課程 的範例都是由Entity Framework所寫成，

所以必須要先有一點點的Entity Framework的基礎才可以繼續學MVC下去。

由於Entity Framework課程 的使用者介面的範例都是搭配MVC所寫成，

所以必須要先有一點點的MVC的基礎才可以繼續學Entity Framework下去。

原則上我的學程必須要先學MVC才能學Entity Framework

但是我擔心，有人沒學過我的MVC課程就跑去先學Entity Framework課程，

這樣肯定學Entity Framework的時候會GG。

因此

在設計課程的時候，我逼不得已，只好讓

MVC課程裡面的Tutorial 1 和 Tutorial 2 和

Entity Framework課程裡面的Tutorial 1 和 Tutorial 2 是完全一樣的。

這樣的課程設計，

一方面是讓已經學過MVC課程的學生在學Entity Framework課程的時候，

重新快速複習MVC觀念。

另一方面是讓沒學過MVC課程的學生在學Entity Framework課程的時候，

可以有稍微一點點MVC基礎觀念。

如果你事先讀過MVC的Tutorial 1 和 Tutorial 2，

那麼在學Entity Framework課程的時候，

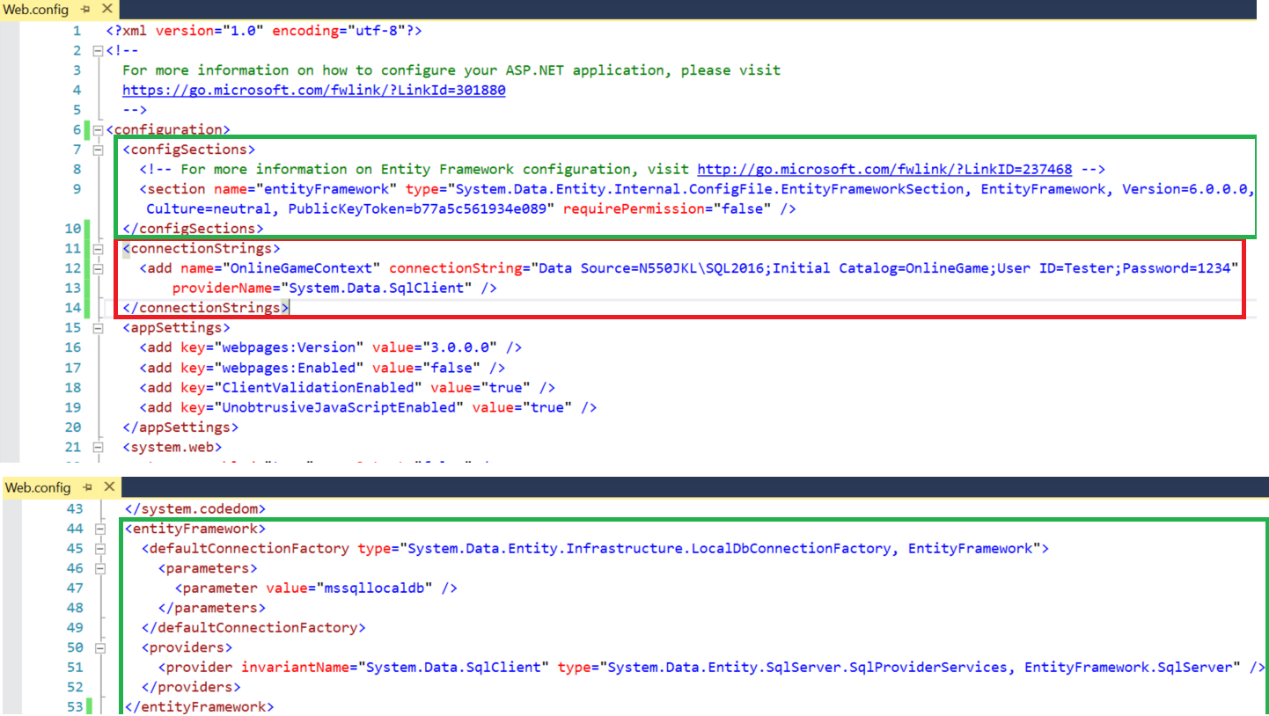
覺得完全不需要複習MVC的基礎觀念的話，

就麻你跳過

Thanks

=========================================

Clarify澄清:



在我的Video

我說

"connectionString 可以放在 <configuration>裡面的任何地方

但是一定要放在EF底下"

有些人誤會成一定要放在44-53行底下

其實

這句話的意思是

一定要放在  第7到10行 的 <configurations> 底下

你也看到 <configurations>裡面 有一個 < section name="entityFramework" ...>

所以我在講課的時候，就講成一定要放在EF底下

造成有些學生的誤會。

所以我的課程講義裡面

放了兩張圖

代表你connectionString最高可以放到這個位置 (11~14行)

最低可以放到53行之後

本來以為已經講得很清楚了

還是被誤會了

好吧，是我講的不夠"精確"

這邊澄清一下

希望對你學習有幫助

=======================================================

1.

MVC其實是Model View Controller.

1.1.

Entity Framework Model:

EF Model其實是你產生EF的時候所使用到的Class

然後放進DBContext的DBSet。

這個EF Model與MVC的Model無關

1.2.

Model:

1.2.1.

MVC底下的Model其實就是所謂的ViewModel

ViewModel可以是任一Class

你可以使用EF Model當作你的ViewModel

我在MVC課程中並沒有把EF Model和View Model分開變成兩個Class

但是要分開也可以。

E.g.

Gamer.cs  是EF Model

GamerViewModel.cs  是ViewModel

然後你可能需要AutoMapper去將裡面的Properties做互相對應的動作。

1.2.2.

ViewModel裡面的Properties

就是用來暫時儲存資料的地方。這些Data稍後可以顯示在View上面。

1.3.

View:

MVC的View的副檔名是CSHTML，也就是所謂的Razor View

Razor View是用來顯示給使用者看。

也就是裡面其實是HTML

如果想在Razor view裡面打入C#

必須要使用@

E.g.1.

@Html.ActionLink("link text", "someaction", "somecontroller", new { id = "123" }, null)

E.g.2.

@{

int a=1;

int b = a+1;

}

1.4.

Controller其實就像是Web Form年代的code-behind

1.4.1.

你可以在Controller把data塞進ViewModel object

然後把ViewMode object傳給View

然後View就可以顯示這些Data

1.4.2.

如果你的View有表單

當你按下Submit button的時候

它會把表單內的Data存進ViewModel object

然後ViewModel object會傳進Controller裡面的Post action

接下來在Post action可以做一些修改

然後Data就可以丟進Database。

=======================================================

1. Create DB: OnlineGame

1.1. Create DB: OnlineGame

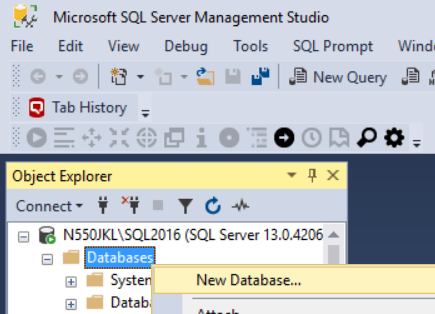
Open SQL server Management Studio (SSMS) and connect SQL server 2016

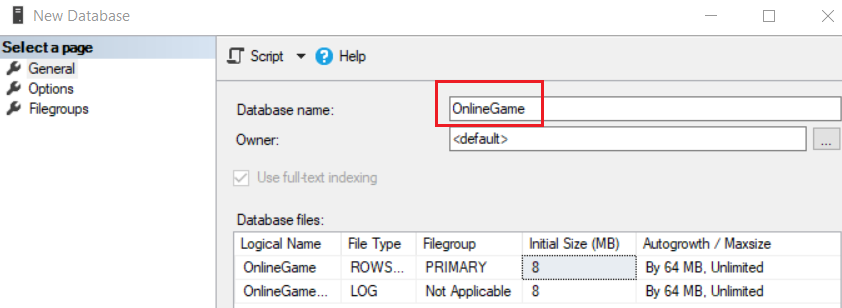
Database --> Right Click --> New Database -->

In General Tab -->

Name: **OnlineGame**

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





Graphical user interface, text, application

Description automatically generated

1.2. Create Gamer Table

--Run the following query to create Gamer Table

--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

CREATE TABLE Gamer

    (

      Id INT PRIMARY KEY

             IDENTITY(1, 1)

             NOT NULL ,

      [Name] NVARCHAR(100) NULL ,

      Gender NVARCHAR(10) NULL ,

      City NVARCHAR(50) NULL,

    );

GO -- Run the previous command and begins new batch

--3. Insert Data

INSERT  Gamer

VALUES  ( N'Name01', N'Male', N'City01' );

INSERT  Gamer

VALUES  ( N'Name02', N'Female', N'City03' );

INSERT  Gamer

VALUES  ( N'Name03', N'Female', N'City01' );

INSERT  Gamer

VALUES  ( N'Name04', N'Male', N'City02' );

1.3. Create Database level security login

In developing process, you may use Windows Authentication to access SQL server.

However, In the real situation, you are going to host your web application somewhere online.

Thus, you will not be able to use Windows Authentication when you publish your website to Web Hoster such as GoDaddy.

Thus, I am going to create a SQL Server Authentication Role, **Tester**.

I will set User Name: **Tester**, and the password: **1234**

This username and password are only for this video.

In the real situation, please use other username and password.

In SQL server

Object Explorer --> Security --> Logins --> New Logins

-->

General Tab

Login Name :

**Tester**

Password:

**1234**

Default Database:

**OnlineGame**

-->

Server Roles Tab

Select

**sysadmin**

-->

User Mapping Tab

Select **OnlineGame**

Select every single role.

Graphical user interface, application

Description automatically generated with medium confidence







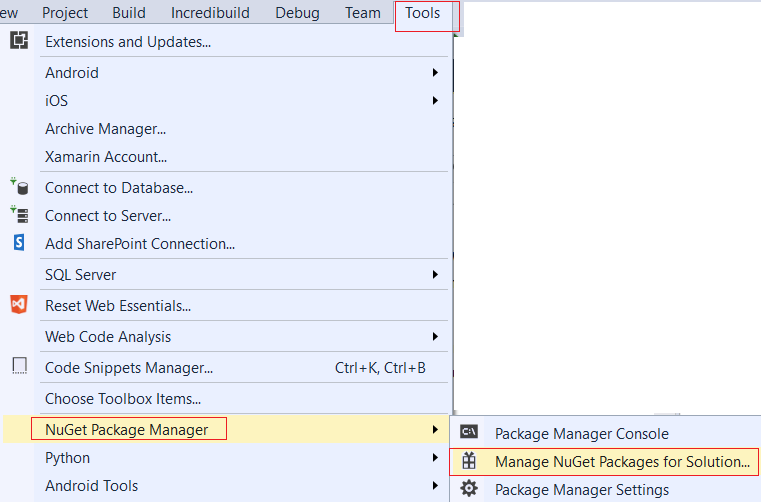
2. Entity Framework

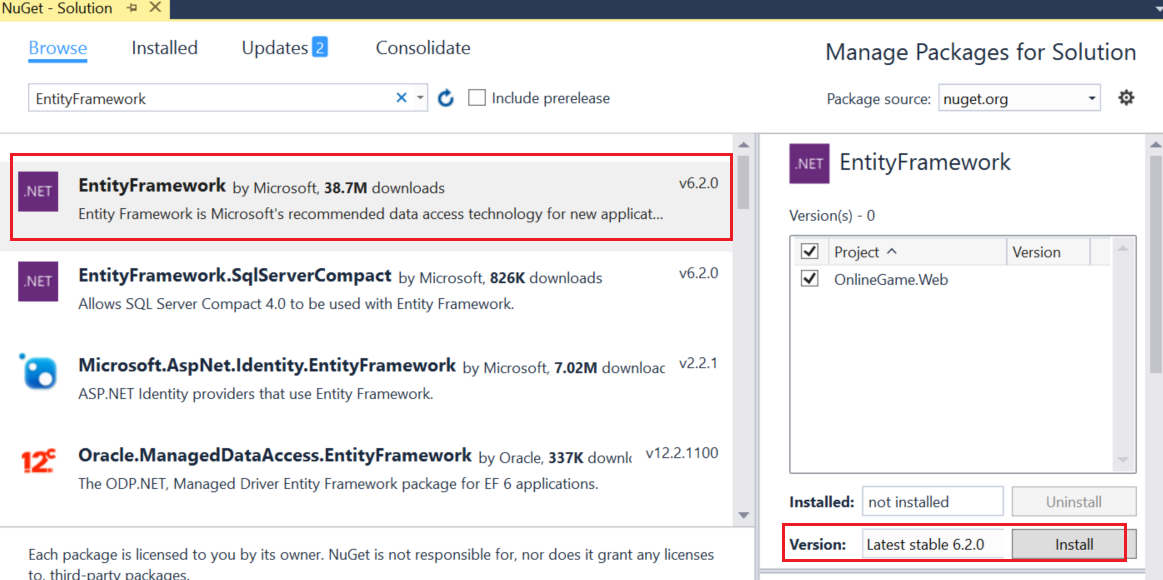
2.1. Install Entity Framework

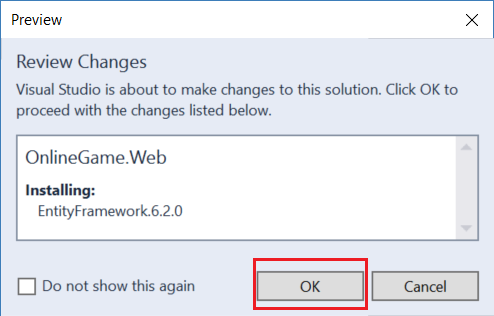
Tools --> NuGet Package Manager --> Manage NuGet Packages for Solutions...

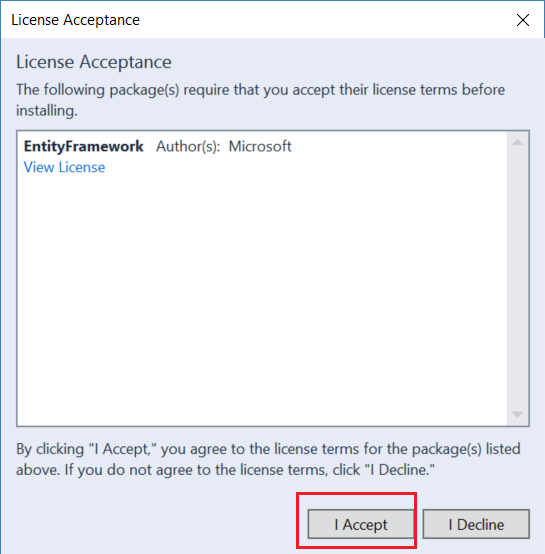
--> Browse tab --> Search  :  **EntityFramework**

--> Install it

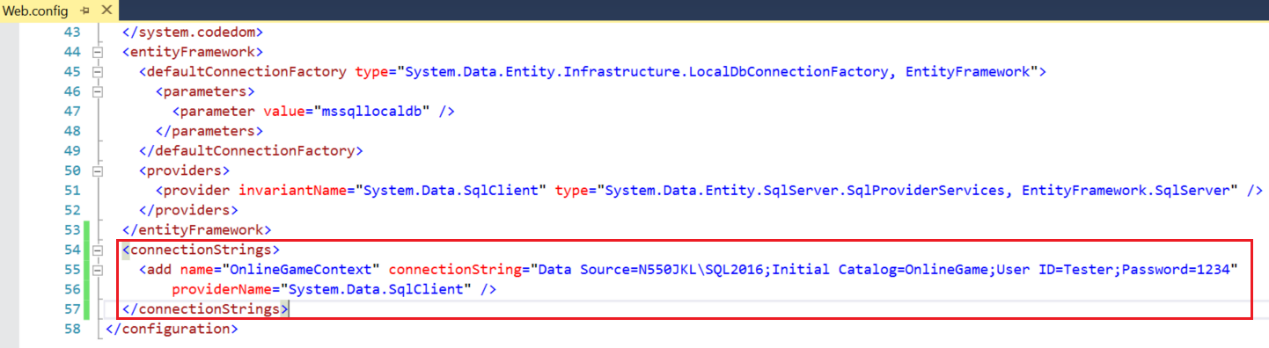








2.2. Web.config : Add Connection String





<connectionStrings>

    <add name="OnlineGameContext" connectionString="Data Source=N550JKL\SQL2016;Initial Catalog=OnlineGame;User ID=Tester;Password=1234"

        providerName="System.Data.SqlClient" />

  </connectionStrings>

Go to Web.config

Add the connection sting on the bottom of Web.config

Please ensure the connection string name "**OnlineGameContext**" must match the DBContext Name "**OnlineGameContext**" in next section.

OnlineGameContext class extends DbContext.

When we create an instance object of OnlineGameContext,

it will search the connection string with the name "OnlineGameContext"  within web config.

2.3. Data/OnlineGameContext.cs

Add New File

Mads Kristensen

3.5.129

Visual Studio 2013, 2015, 2017

The fastest and easiest way to add new files to any project - including files that start with a dot

<https://marketplace.visualstudio.com/items?itemName=MadsKristensen.AddNewFile>

Because **Add New File** **(extension and update)**

press **Shift+F2**

**Data/OnlineGameContext.cs**

Text

Description automatically generated

Please ensure the connection string name "**OnlineGameContext**" must match the DBContext Name "**OnlineGameContext**"

Type the following code.

using System.Data.Entity;

using OnlineGame.Web.Models;

namespace OnlineGame.Web.Data

{

    public class OnlineGameContext : DbContext

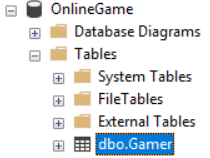
    {

        public DbSet<Gamer> Gamers { get; set; }

    }

}

2.4. Models/Gamer.cs



Reference:

<http://www.entityframeworktutorial.net/code-first/table-dataannotations-attribute-in-code-first.aspx>

<http://www.entityframeworktutorial.net/code-first/column-dataannotations-attribute-in-code-first.aspx>

In order to map the **Gamer**Table entity into the **Model, Models/Gamer.cs**.

You need to add the **[Table("Gamer")] attribute**  in the class level.

Then the EntityFramework will automatically map the **Gamer table fields** into **Gamer Model properties**.

**Gamer table field, id**will automatically map to **Gamer Model properties, id**,

because it has the same name.

If you want to map 2 different name, then you need [Column("Name")] **attribute.**

[Column("Name")]

public string Name2 { get; set; }

This [Column("Name")] **attribute**

will map Table Column, **Name**, to Model Property, **Name2.**

using System.ComponentModel.DataAnnotations.Schema;

namespace OnlineGame.Web.Models

{

    //[Table("Gamer"]

    [Table("Gamer", Schema = "dbo")]

    public class Gamer

    {

        public int Id { get; set; }

        [Column("Name")]

        public string Name { get; set; }

        public string Gender { get; set; }

        public string City { get; set; }

    }

}

/\*

1.

////[Table("Gamer"]

//[Table("Gamer", Schema = "dbo")]

//...

//[Column("Name")]

//public string Name { get; set; }

Reference:

<http://www.entityframeworktutorial.net/code-first/table-dataannotations-attribute-in-code-first.aspx>

<http://www.entityframeworktutorial.net/code-first/column-dataannotations-attribute-in-code-first.aspx>

In order to map the Gamer Table entity into the Model, Models/Gamer.cs.

You need to add the [Table("Gamer")] attribute  in the class level.

Then the EntityFramework will automatically map the Gamer table fields into Gamer Model properties.

Gamer table field, id will automatically map to Gamer Model properties, id,

because it has the same name.

If you want to map 2 different name, then you need [Column("Name")] attribute.

//[Column("Name")]

//public string Name2 { get; set; }

This [Column("Name")] attribute

will map Table Column, Name, to Model Property, Name2.

\*/

2.5. Controllers/GamerController.cs

using System.Linq;

using System.Web.Mvc;

using OnlineGame.Web.Data;

using OnlineGame.Web.Models;

namespace OnlineGame.Web.Controllers

{

    public class GamerController : Controller

    {

        // <http://localhost/OnlineGame.Web/Gamer/Details>

        //public ActionResult Details()

        //{

        //    var gamer = new Gamer

        //    {

        //        Id = 1,

        //        Name = "Name1",

        //        Gender = "Male",

        //        City = "City1"

        //    };

        //    return View(gamer);

        //}

        // <http://localhost/OnlineGame.Web/Gamer/Details>

        // <http://localhost/OnlineGame.Web/Gamer/Details/1>

        // <http://localhost/OnlineGame.Web/Gamer/Details/2>

        // <http://localhost/OnlineGame.Web/Gamer/Details/3>

        // <http://localhost/OnlineGame.Web/Gamer/Details/4>

        public ActionResult Details(int id = 0)

        {

            var onlineGameContext = new OnlineGameContext();

            Gamer gamer;

            if (id == 0)

            {

                gamer = new Gamer

                {

                    Id = 0,

                    Name = "Name0",

                    Gender = "NULL",

                    City = "NULL"

                };

                // or you may throw exception here.

            }

            else

            {

                gamer = onlineGameContext.Gamers.Single(p => p.Id == id);

                //Throws exception if can not find the single entity

            }

            return View(gamer);

        }

    }

}

/\*

//var onlineGameContext = new OnlineGameContext();

//Gamer gamer = onlineGameContext.Gamers.Single(p => p.Id == id);

When user request, EntityFramework will request the data from the database

and sotre its data into a temp place called DBSet.

onlineGameContext.Gamers is a DBSet which is kind of temp place to store the Gamer Table Data.

We use LINQ to map the Gamer Table Column id to Gamer Model property, id.

Thus, we can get the gamer entity from Gamer Table by its id.

Then store gamer entity data into Gamer Model object.

Thus, each Gamer Model object is a temp place to store each Gamer Table entity from the database.

Then we pass the Gamer Model object as the ViewModel,

Thus, the Details.cshtml view can use the values from Gamer Model object

which is actually the temp place to store Gamer Table entity data.

\*/

2.6. Views/Gamer/Details.cshtml

@model OnlineGame.Web.Models.Gamer

@{

    ViewBag.Title = "Gamer Details";

}

<h2>Gamer Details</h2>

<table>

    <tr>

        <td>

            Gamer ID:

        </td>

        <td>

            @Model.Id

        </td>

    </tr>

    <tr>

        <td>

            Name:

        </td>

        <td>

            @Model.Name

        </td>

    </tr>

    <tr>

        <td>

            Gender:

        </td>

        <td>

            @Model.Gender

        </td>

    </tr>

    <tr>

        <td>

            City:

        </td>

        <td>

            @Model.City

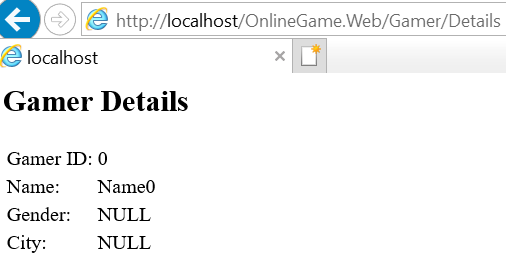
        </td>

    </tr>

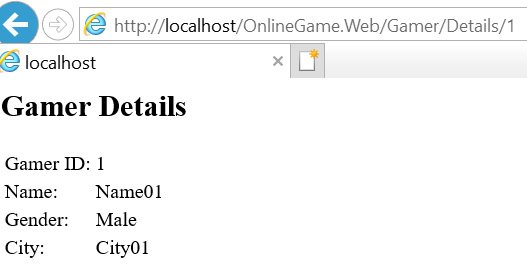
</table>

2.7. URL

<http://localhost/OnlineGame.Web/Gamer/Details>



<http://localhost/OnlineGame.Web/Gamer/Details/1>



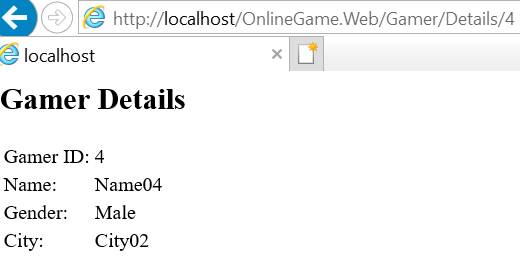
<http://localhost/OnlineGame.Web/Gamer/Details/2>



<http://localhost/OnlineGame.Web/Gamer/Details/3>



<http://localhost/OnlineGame.Web/Gamer/Details/4>



<http://localhost/OnlineGame.Web/Gamer/Details/4>

