

(T9)討論 EfDbFirst 。討論 ModelAttributes 、 DisplayTemplates 、 EditorTemplates 。討論 DateTime 、 JQueryUiDatepicker
CourseGUID: 8503b39c-5887-4634-8291-facfb3117924

(T9)討論 EfDbFirst 。討論 ModelAttributes 、 DisplayTemplates 、 EditorTemplates 。討論 DateTime 、 JQueryUiDatepicker

(T9-1)討論 EfDbFirst (1. to 5.3.)

(T9-2)討論 ModelAttributes 、 DisplayTemplates 、 EditorTemplates (5.4. to 5.13)

(T9-3)討論 DateTime 、 JQueryUiDatepicker (5.14 to 5.19)

0. Summary

1. MVC conventions

2. OnlineGame DB

2.1. TSQL

2.2. Security login

3. New Project - OnlineGame

3.1. New Project - OnlineGame.Web

3.1.1. Global.asax.cs

3.1.1. Global.asax.cs

3.2. ADO.Net Entity Data Model - Entity Framework

4. OnlineGame.Web

4.1. Controllers/GamersController.cs

5. OnlineGame.Web

5.1. web.config

5.2. Controllers/GamersController.cs

5.3. Views/Gamer/Index.cshtml

5.4. Models/Gamer/Gamer.cs

5.5. Models/Gamer/GamerMetaData.cs

5.6. Models/Gamer/BoardGame.cs

5.7. Models/Gamer/GamerA.cs

5.8. Views/Gamer/Details.cshtml

5.9. Views/Gamer/DetailsTwo.cshtml (Display Complex Type Sample)

5.10. Views/Shared/DisplayTemplates/UrlToNewWindow.cshtml (UIHint Sample)

5.11. Views/Gamer/DetailsThree.cshtml ([HiddenInput(DisplayValue = false)], [ReadOnly(true)])

5.12. Views/Gamer/DetailsFour.cshtml (@Html.Display("GamerData"))

5.13. Views/Gamer/EditTwo.cshtml ([HiddenInput(DisplayValue = false)], [ReadOnly(true)])

5.14. Views/Gamer/EditThree.cshtml

5.15. Views/Shared/EditorTemplates/DateTime.cshtml

5.15. Views/Shared/EditorTemplates/DateTime.cshtml

5.17. Add JQuery UI

5.18. Views/Gamer/Create.cshtml

5.19. Views/Gamer/Delete.cshtml

0. Summary

In this tutorial, we will discuss

- * MvcConventions
- * AdoDotNetEntityDataModel
- * EntityFramework
- * AutoGenerate Delete, Update, Insert, Read
- * DisplayName attribute
 - * [DisplayName("Full Name")]
- * DisplayAttribute attribute
 - * [DisplayAttribute(Name="Full Name")]
- * Display attribute
 - * [Display(Name = "Full Name")]
- * DisplayFormat attribute
 - * [DisplayFormat(NullDisplayText = "Gender not specified")]
 - * [DisplayFormat(DataFormatString = "{0:d}")]
 - * Display only the date part. E.g. 29/04/1986
 - * [DisplayFormat(DataFormatString = "{0:dd/MM/yyyy HH:mm:ss}")]
 - * Display in 24 hour notation. E.g. 29/04/1986 13:00:00
 - * [DisplayFormat(DataFormatString = "{0:dd/MM/yyyy hh:mm:ss tt}")]
 - * Display in 12 hour notation. E.g. 29/04/1986 1:00:00 PM
- * DisplayFormatAttribute attribute
 - * [DisplayFormatAttribute(DataFormatString="{0:d}")]
 - * Display only the date part. E.g. 29/04/1986
- * DataType attribute
 - * [DataType(DataType.Date)]
 - * Display only date part
 - * Please be aware, it actually covert DateTime to Date,

so Views/Shared/EditorTemplates/DateTime.cshtml will not Work.

- * [DataType(DataType.Time)]
 - * Display only 12 hour notation Time part
- * [DataType(DataType.EmailAddress)]
 - * Display mailto hyperlink
- * [DataType(DataType.Url)]
 - * display a hyperlink.
- * [DataType(DataType.Currency)]
 - * Display the currency symbol by globalization culture in web.config file.
 - * In System.Web tag of web.config file,

<system.web> <globalization culture = "en-au" /> ...

Display au \$ symbol.

- * scaffoldcolumn attribute
 - * [ScaffoldColumn(false)]
 - * it will not display the column when using @Html.DisplayForModel() helper.
 - * HiddenInput attribute
 - * [HiddenInput(DisplayValue = false)]
 - * It will become a hidden input when using @Html.DisplayForModel() helper or @Html.EditorForModel()
 - * ReadOnly attribute
 - * [ReadOnly(true)] or you may delete Setter.
 - * It will make this property un-editable.
 - * You may set a breakpoint to see the [HttpPost] action model for this property is null.
 - * DisplayColumn attribute
 - * [DisplayColumn("Name")]
 - * @Html.DisplayTextFor(model => model.GameHolder)
- model.GameHolder will return a Gamer object.
- * The Gamer class has [DisplayColumn("Name")] attribute,

so it will display Gamer Name property value which is the full name of that gamer.

* UIHint attribute

* [UIHint("UrlToNewWindow")]

* UIHint specify the name of view DisplayTemplate

* 秒殺 Entity Framework 新增更新移除，征服 Model Attributes，Display Templates，Editor Templates，Jquery Ui datepicker

* 秒殺 Entity Framework 新增更新移除。

* 完整攻略 Model Attributes，Display Templates，Editor Templates。

* 初步使用 Jquery Ui datepicker。

=====

1. MVC conventions

In MVC conventions,

1. Controllers must have the word "Controller" as the suffix and must extend "IController" interface.
2. A view must remain under "Views" folder.
3. If the view is for GamerController, then the view must remain under "Views/Gamer" folder.
4. In the "HomeController", when "Index" action "return View()", it will search the following files in order.
 - 4.1. ~/Views/Home/Index.aspx
 - 4.2. ~/Views/Home/Index.ascx
 - 4.3. ~/Views/Shared/Index.aspx
 - 4.4. ~/Views/Shared/Index.ascx
 - 4.5. ~/Views/Home/Index.cshtml
 - 4.6. ~/Views/Home/Index.vbhtml
 - 4.7. ~/Views/Shared/Index.cshtml
 - 4.8. ~/Views/Shared/Index.vbhtml
5. By MVC convention, MVC will look for the view in the following locations
 - 5.1. Views/ControllerName
 - 5.2. Views/Shared
6. The extension name of view can be cshtml, vbhtml, aspx, or ascx.
7. Models can be anywhere, even can be in another project. However, it is better to put it in "Models" folder.
8. You may put Models in another project as business layer.
9. Shared folder stores shared views.

E.g. Master for aspx and Layout pages for cshtml

10. EditorTemplates and DisplayTemplates by MVC convention

10.1.

DisplayTemplates

10.1.1.

Views\Shared\DisplayTemplates\UrlToNewWindow.cshtml

Views\Gamer\DisplayTemplates\UrlToNewWindow.cshtml

UrlToNewWindow.cshtml is the DisplayTemplate which must under "DisplayTemplates" folder.

Views\Shared\DisplayTemplates\UrlToNewWindow.cshtml means the template is available for all the views.

Views\Gamer\DisplayTemplates\UrlToNewWindow.cshtml means the template is available for only the views of Gamer controller.

10.1.2.

Using DisplayTemplates

10.1.2.1.

In the Models/Gamer/GamerMetaData.cs

```
//[DataType(DataType.Url)]
```

```
//[UIHint("UrlToNewWindow")]
```

```
//public string ProfileUrl { get; set; }
```

[DataType(DataType.Url)] attribute will display a hyperlink.

[UIHint("UrlToNewWindow")] attribute specify the name of view DisplayTemplate to display the property data.

In this case, it will look for "DisplayTemplates/UrlToNewWindow.cshtml" under "Shared" folder or "Gamer" folder.

Use that view template to display the data of this property.

10.1.2.2.

```
//<a href="@ViewData.Model" target="_blank">@ViewData.Model</a>
```

In the Shared/DisplayTemplates/UrlToNewWindow.cshtml,

@ViewData.Model will take the Model data from the parent view.

In this case, it will return a profile url.

10.2.

EditorTemplates

10.2.1.

Views\Shared\EditorTemplates\DateTime.cshtml

Views\Gamer\EditorTemplates\DateTime.cshtml

DateTime.cshtml is the EditorTemplate which must under "EditorTemplates" folder.

Views\Shared\EditorTemplates\DateTime.cshtml means

the template is available for all the views.

Views\Gamer\EditorTemplates\DateTime.cshtml means

the template is available for only the views of Gamer controller.

10.2.2.

Using EditorTemplates

The EditorTemplate Name must match View Model property Type Name.

E.g. DateTime.ascx or DateTime.cshtml

10.2.2.1.

In the Models/Gamer/GamerMetaData.cs

```
////[DataType(DataType.Date)] //Views/Shared/EditorTemplates/DateTime.cshtml will not Work.
```

```
////[DisplayFormat(DataFormatString = "{0:dd/MM/yyyy hh:mm:ss tt}")]
```

```
//[DisplayFormat(DataFormatString = "{0:d}")]
```

```
//public Nullable<System.DateTime> DateOfBirth { get; set; }
```

The type is DateTime, so it will look for the EditorTemplate from

Views\Shared\EditorTemplates\DateTime.cshtml or

Views\Gamer\EditorTemplates\DateTime.cshtml

In this case, Views\Shared\EditorTemplates\DateTime.cshtml will be the EditorTemplate.

The View Model Property in Edit mode will use the EditorTemplate to display.

In this case,

```
//@model DateTime?
```

```
//@Html.TextBox("", (Model.HasValue ? Model.Value.ToString("yyyy/MM/dd") : string.Empty), new { @class = "date" })
```

So it will add the class="date" to the textbox input.

10.2.2.2.

In the Edit.cshtml

```
//<link href="~/Content/themes/base/jquery-ui.min.css" rel="stylesheet" />
```

```
//<link href="~/Content/bootstrap.css" rel="stylesheet" />
```

```
//<script src="~/Scripts/jquery-1.12.4.min.js"></script>
```

```
//<script src="~/Scripts/jquery-ui-1.12.1.min.js"></script>
```

```
//<script src="~/Scripts/bootstrap.min.js"></script>
```

```
...
```

```
//<script type="text/javascript">
```

```
// $(function () {
```

```
//     $("input:text.date").datepicker(
```

```
//     {
```

```
//         dateFormat: "yy/mm/dd"
```

```
//     });
```

```
// });
```

```
//</script>
```

11.

There are 2 categories of built-in templated helpers.

11.1.

Display Templates

11.1.1.

```
//@Html.DisplayFor(model => model.Name)
```

The view must have strongly typed view Model.

It can work with the complex type Model property.

It is similar to @Html.DisplayTextFor(model => model.GameHolder)

```
//@Html.DisplayTextFor(model => model.GameHolder)
```

model.GameHolder will return a Gamer object.

The Gamer class has [DisplayColumn("Name")] attribute,

so it will display Gamer Name property value

which is the full name of that gamer.

11.1.2.

```
//@Html.DisplayForModel()
```

The view must have strongly typed view Model.

It will display every property in view model

except the properties with [ScaffoldColumn(false)] attribute.

11.1.3.

@Html.Display helper does not need strongly typed view Mode.

```
//ViewData["GamerData"] = gamer;
```

```
//return View();
```

In the controller, we put the gamer object into ViewData["GamerData"]

"GamerData" in this case is the key of ViewData.

ViewData["GamerData"] contains that gamer object data,

so we don't have to use a view model.

```
//@Html.Display("GamerData")
```

In the view, we use @Html.Display("GamerData")

to retrieve the Gamer data from ViewData["GamerData"].

It will display everything

except the properties with [ScaffoldColumn(false)] attribute.

11.2.

Editor Templates

11.2.1.

```
//@Html.EditorFor(model => model.Name)
```

The view must have strongly typed view Model.
It will create a textbox for the property value input.

11.2.2.

```
//@Html.EditorForModel()
```

The view must have strongly typed view Model.
It will create textbox input for every property in view model
except the properties with [ScaffoldColumn(false)] attribute.

11.2.3.

@Html.Editor helper does not need strongly typed view Mode.

```
//ViewData["GamerData"] = gamer;
```

```
//return View();
```

In the controller, we put the gamer object into ViewData["GamerData"]

"GamerData" in this case is the key of ViewData.

ViewData["GamerData"] contains that gamer object data,
so we don't have to use a view model.

```
//@Html.Editor("GamerData")
```

In the view, we use @Html.Editor("GamerData")

to retrieve the Gamer data from ViewData["GamerData"].

It will create textbox input for every properties in ViewData["GamerData"]
except the properties with [ScaffoldColumn(false)] attribute.

However, we pressed submit button and call the [HttpPost] action

```
//public async Task<ActionResult>
```

```
EditThree(int id, string name, string gender, string city, DateTime? dateOfBirth, string emailAddress, int? score,  
string profileUrl, int? gameMoney, int? teamId)
```

OR

```
//public async Task<ActionResult>
```

```
EditThree(Gamer gamer)
```

Both ways can not retrieve the data because it is not strongly typed.

I don't suggest to use @Html.Editor helper

2. OnlineGame DB

2.1. TSQL

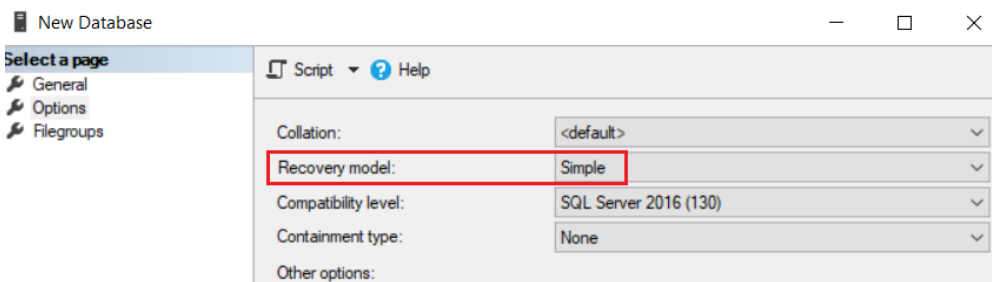
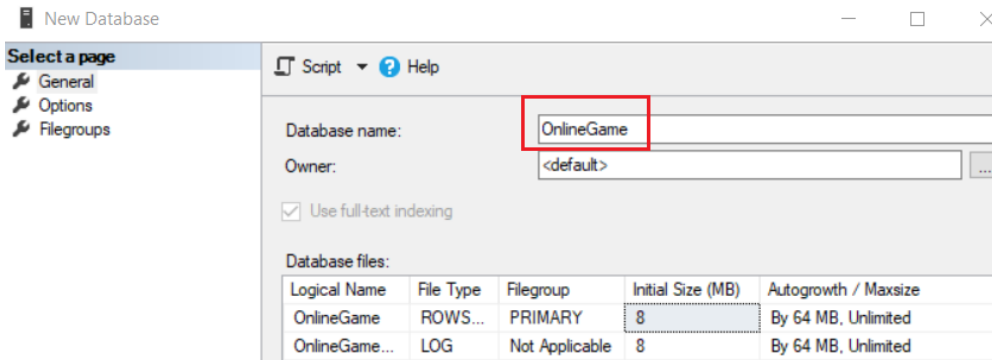
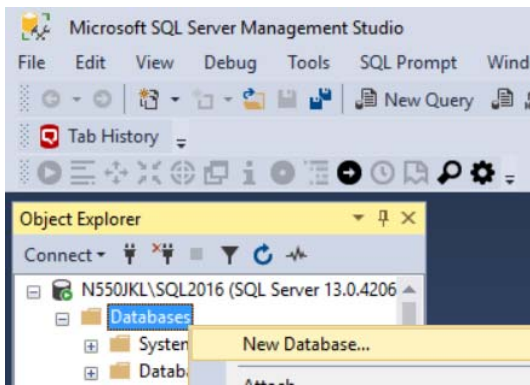
In SQL server Management Studio (SSMS)

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

In General Tab -->

Name: **OnlineGame**

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



```
--1. Drop if it exists
--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
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
--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 = 'spGetGamers' ) )

BEGIN
    DROP PROCEDURE spGetGamers;
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 = 'spAddGamer' ) )

BEGIN
    DROP PROCEDURE spAddGamer;
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 = 'spSaveGamer' ) )

BEGIN
    DROP PROCEDURE spSaveGamer;
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 = 'spDeleteGamer' ) )

BEGIN
    DROP PROCEDURE spDeleteGamer;
END;

GO -- Run the previous command and begins new batch
--2. Create Table
CREATE TABLE Team
(
    Id INT PRIMARY KEY
        IDENTITY(1, 1)
        NOT NULL ,
    [Name] NVARCHAR(100) NULL
);

GO -- Run the previous command and begins new batch
CREATE TABLE Gamer
(
    Id INT PRIMARY KEY
        IDENTITY(1, 1)
        NOT NULL ,
    [Name] NVARCHAR(100) NULL ,
    Gender NVARCHAR(10) NULL ,
    City NVARCHAR(50) NULL ,

```



```

    DateOfBirth DATETIME NULL ,
    EmailAddress NVARCHAR(100) ,
    Score INT ,
    ProfileUrl NVARCHAR(100) ,
    GameMoney INT,
    TeamId INT FOREIGN KEY REFERENCES Team ( Id )
);

GO -- Run the previous command and begins new batch
--3. Insert Data
INSERT Team
VALUES ( N'Team1' );
INSERT Team
VALUES ( N'Team2' );
INSERT Team
VALUES ( N'Team3' );
GO -- Run the previous command and begins new batch
INSERT Gamer
VALUES ( N'Name01 ABB', N'Male', N'City01', '1979/4/28', '1@AAA.com', 3500,
        'https://ithandyguytutorial.blogspot.com.au/', 1000, 1 );
INSERT Gamer
VALUES ( N'Name02 CDDE', N'Female', N'City03', '1981/7/24', '2@BBB.com', 3500,
        'https://ithandyguytutorial.blogspot.com.au/', 1500, 2 );
INSERT Gamer
VALUES ( N'Name03 FIJK', N'Female', N'City01', '1984/12/5', '3@CCCC.com', 3500,
        'https://ithandyguytutorial.blogspot.com.au/', 4000, 3 );
INSERT Gamer
VALUES ( N'Name04 LMOPPQ', N'Male', N'City02', '1983/5/29', '4@DD.com', 3500,
        'https://ithandyguytutorial.blogspot.com.au/', 2500, 1 );
INSERT Gamer
VALUES ( N'Name05 QRSTT', N'Male', N'City01', '1979/6/20', '5@EEE.com', 3500,
        'https://ithandyguytutorial.blogspot.com.au/', 3500, 3 );
INSERT Gamer
VALUES ( N'Name06 TUVVX', N'Female', N'City03', '1984/5/15', '6@FF.com',
        3500, 'https://ithandyguytutorial.blogspot.com.au/', 2500, 3 );
INSERT Gamer
VALUES ( N'Name07 XYZZXX', N'Female', N'City01', '1986/4/29', '7@GGGG.com',
        3500, 'https://ithandyguytutorial.blogspot.com.au/', 4550, 2 );
INSERT Gamer
VALUES ( N'Name08 ABBCDE', N'Male', N'City02', '1985/7/28', '8@HH.com', 3500,
        'https://ithandyguytutorial.blogspot.com.au/', 3550, 1 );
INSERT Gamer
VALUES ( N'Name09 QRSTTUVXX', N'Male', N'City02', '1983/4/16', '9@IIII.com',
        3500, 'https://ithandyguytutorial.blogspot.com.au/', 2510, 1 );
GO -- Run the previous command and begins new batch
--4. SP
CREATE PROCEDURE spGetGamers
AS
BEGIN
    SELECT *
    FROM Gamer;
END;
GO -- Run the previous command and begins new batch
CREATE PROCEDURE spAddGamer
(
    @Name NVARCHAR(50) ,
    @Gender NVARCHAR(10) ,

```

```

        @City NVARCHAR(50) ,
        @DateOfBirth DateTime ,
        @EmailAddress NVARCHAR(100) ,
        @Score INT ,
        @ProfileUrl NVARCHAR(100) ,
        @GameMoney INT ,
        @TeamId INT
    )
AS
BEGIN
    INSERT INTO Gamer
    VALUES ( @Name, @Gender, @City, @DateOfBirth, @EmailAddress, @Score, @ProfileUrl, @GameMoney, @TeamId
);
END;
GO -- Run the previous command and begins new batch
CREATE PROCEDURE spSaveGamer
(
    @Id INT ,
    @Name NVARCHAR(50) ,
    @Gender NVARCHAR(10) ,
    @City NVARCHAR(50) ,
    @DateOfBirth DateTime ,
    @EmailAddress NVARCHAR(100) ,
    @Score INT ,
    @ProfileUrl NVARCHAR(100) ,
    @GameMoney INT ,
    @TeamId INT
)
AS
BEGIN
    UPDATE dbo.Gamer
    SET     Name = @Name ,
           Gender = @Gender ,
           City = @City ,
           DateOfBirth = @DateOfBirth ,
           EmailAddress = @EmailAddress ,
           Score = @Score ,
           ProfileUrl = @ProfileUrl ,
           GameMoney = @GameMoney ,
           TeamId = @TeamId
    WHERE   Id = @Id;
END;
GO -- Run the previous command and begins new batch
CREATE PROCEDURE spDeleteGamer ( @Id int )
AS
BEGIN
    DELETE FROM Gamer
    WHERE   Id = @Id;
END;
GO -- Run the previous command and begins new batch
--EXEC spGetGamers
--GO -- Run the previous command and begins new batch

```

2.2. Security login

In SQL server

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

-->

General Tab

Login Name :

Tester

Password:

1234

Default Database:

OnlineGame

-->

Server Roles Tab

Select

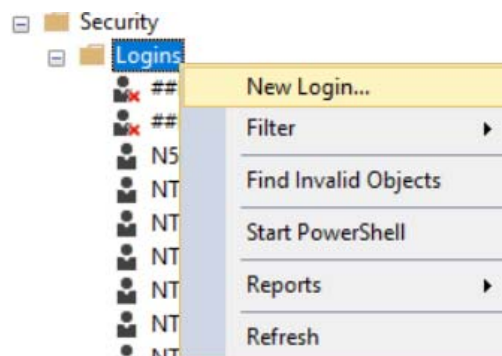
sysadmin

-->

User Mapping Tab

Select **OnlineGame**

Select every single role.



Login - New

Select a page

- General
- Server Roles
- User Mapping
- Securables
- Status

Connection

Server: N550JKL\SQL2016

Connection: N550JKL\pmp1

[View connection properties](#)

Progress

Ready

Script Help

Login name: Search...

☐ Windows authentication

☒ SQL Server authentication

Password:

Confirm password:

☐ Specify old password

Old password:

☒ Enforce password policy

☒ Enforce password expiration

☒ User must change password at next login

☐ Mapped to certificate

☐ Mapped to asymmetric key

☐ Map to Credential

Mapped Credentials

Credential	Provider
------------	----------

Add

Remove

Default database:

Default language:

OK Cancel

Login Properties - Tester

Select a page

- General
- Server Roles
- User Mapping
- Securables
- Status

Connection

Server: N550JKL\SQL2016

Connection: N550JKL\pmp1

[View connection properties](#)

Progress

Ready

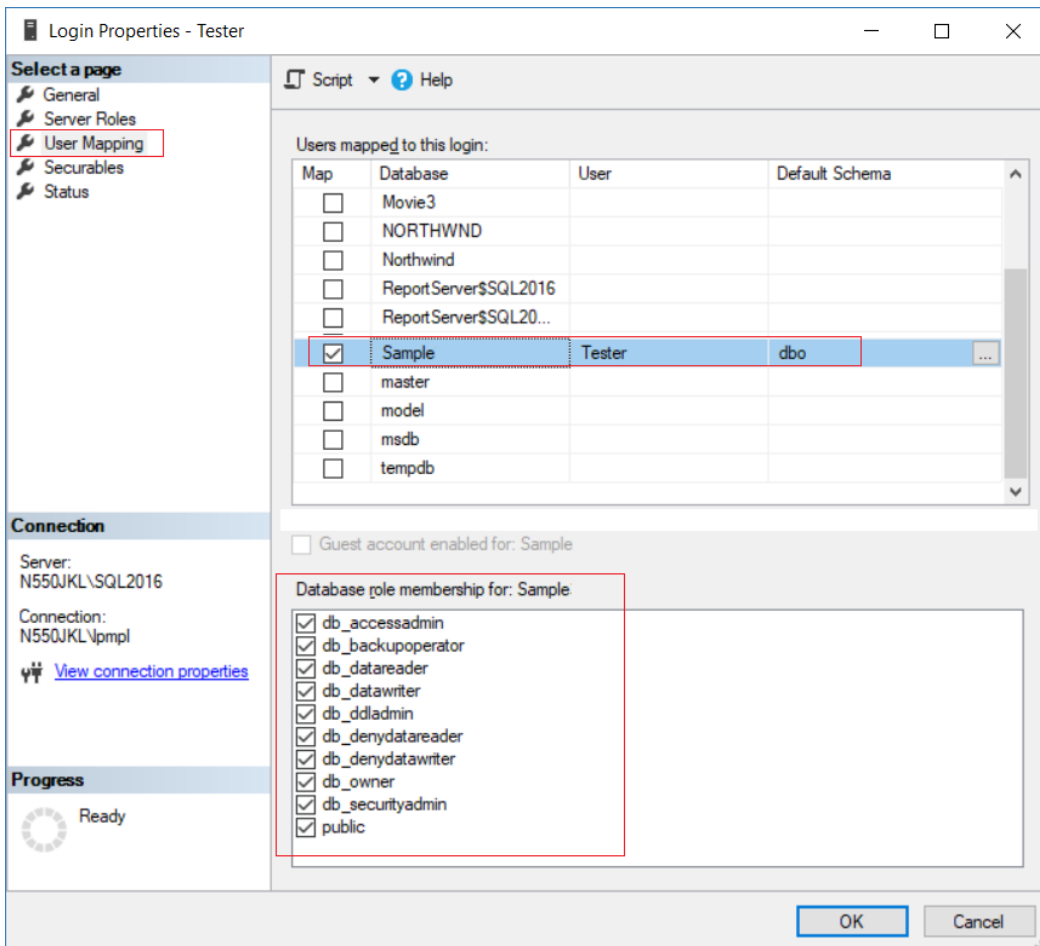
Script Help

Server role is used to grant server-wide security privileges to a user.

Server roles:

- ☐ bulkadmin
- ☐ dbcreator
- ☐ diskadmin
- ☐ processadmin
- ☒ public
- ☐ securityadmin
- ☐ serveradmin
- ☐ setupadmin
- ☒ sysadmin

OK Cancel



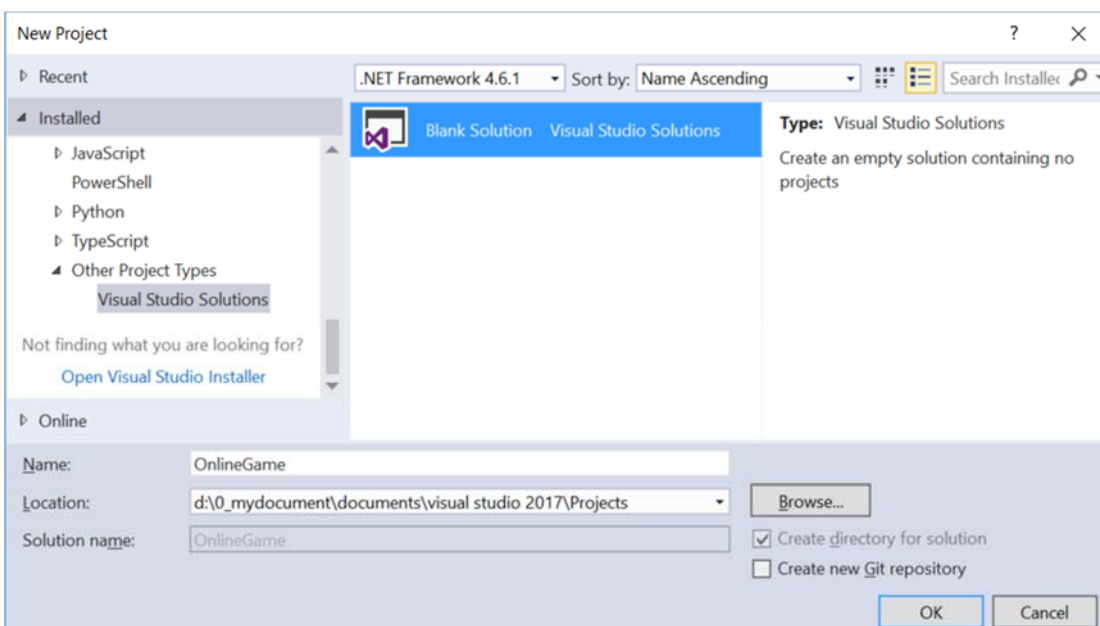
3. New Project - OnlineGame

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

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

-->

Name: **OnlineGame**



3.1. New Project - OnlineGame.Web

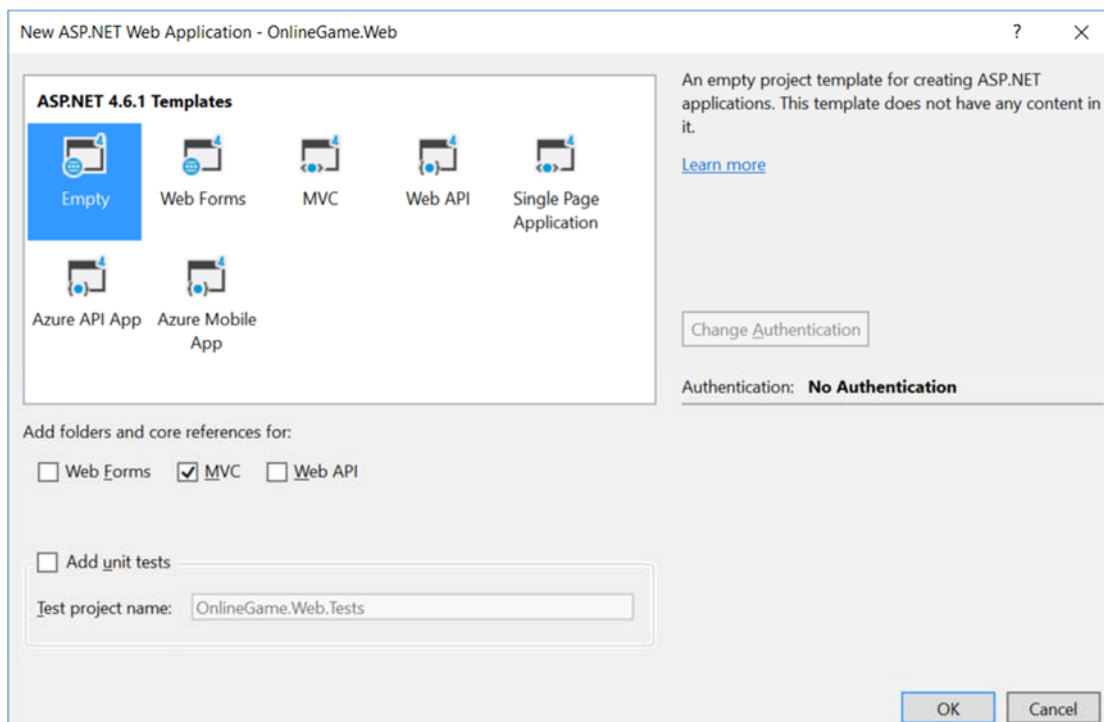
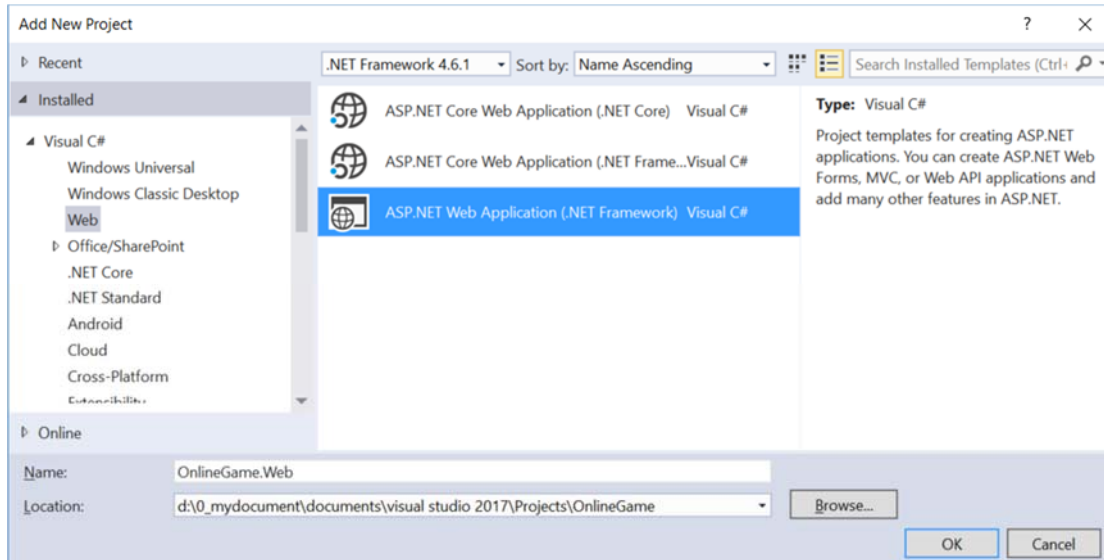
Solutions Name --> Add --> New Project -->

Visual C# --> Web --> [ASP.NET](#) Web Application (.NET Framework)

-->

Name: **OnlineGame.Web**

Empty --> Select "MVC" --> OK



3.1.1. Global.asax.cs

```
using System.Web.Mvc;
using System.Web.Routing;
namespace OnlineGame.Web
{
    public class MvcApplication : System.Web.HttpApplication
```

```

{
    //Application_Start() is the magic start point of this application
    protected void Application_Start()
    {
        AreaRegistration.RegisterAllAreas();
        //1.
        //Register Route Configure in RouteConfig.cs
        //If you want to see route configuration,
        //you may find it in RouteConfig.cs
        //2.
        //System.Web.Routing.RouteCollection Routes { get; }
        //Gets a collection of objects that derive from the System.Web.Routing.RouteBase class.
        RouteConfig.RegisterRoutes(RouteTable.Routes);
    }
}

```

3.1.2. App_Start/RouteConfig.cs

```

using System.Web.Mvc;
using System.Web.Routing;
namespace OnlineGame.Web
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            //Handle the Route of the axd request file.
            //E.g. ASP.Net Tracing
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
            //Handle the Route called "Default".
            //The mapping URL is "{controller}/{action}/{id}"
            //Set the default value of Controller, action, and id.
            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Gamer", action = "Index", id = UrlParameter.Optional }
            );
        }
    }
}
/*
1.
//routes.MapRoute(
//    name: "Default",
//    url: "{controller}/{action}/{id}",
//    defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
//);
1.1.
When a request comes in,
it's trying to do a pattern match based on
all the templates it sees in these mapped routes.
A route is some instructions for
how to take a URI coming into a request
and map it to some code,
normally a controller.
In this case,
look at defaults parameter,
when user request http://localhost:PortNumber/

```

IIS Express will run
HomeController Index action.
It will map to Controllers/HomeController.cs
and map to Index Method
1.2.

By convention in MVC.
All controllers will have Controller suffix.
This suffix is not required in the URL.
So, if you want to invoke Home controller,
you specify /Home and not /HomeController.

2.
//routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
2.1.

Reference:

<https://stackoverflow.com/questions/9016650/what-is-routes-ignoreroresource-axd-pathinfo>

This line can handle the axd file request route,
E.g. trace.axd
.axd files don't exist physically.

ASP.NET uses URLs with .axd extensions
(ScriptResource.axd and WebResource.axd) internally,
and they are handled by an HttpHandler.
Therefore, you should keep this rule,
to prevent ASP.NET MVC from trying to handle the request
instead of letting the dedicated HttpHandler do it.

2.2.
trace.axd

Reference:

<https://msdn.microsoft.com/en-us/library/wwh16c6c.aspx>

trace.axd trace details for a specific request.
If you want to enable trace.axd,
then you have to go to Web.config
Add <trace enabled="true" pageOutput="false"/> under <system.web>
Then run the project, type the following URL
<http://localhost/OnlineGame.Web/trace.axd>

This will return ASP.NET trace, trace.axd.

If you do not have
// routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
then you can not enable the trace.axd.
*/

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

-->

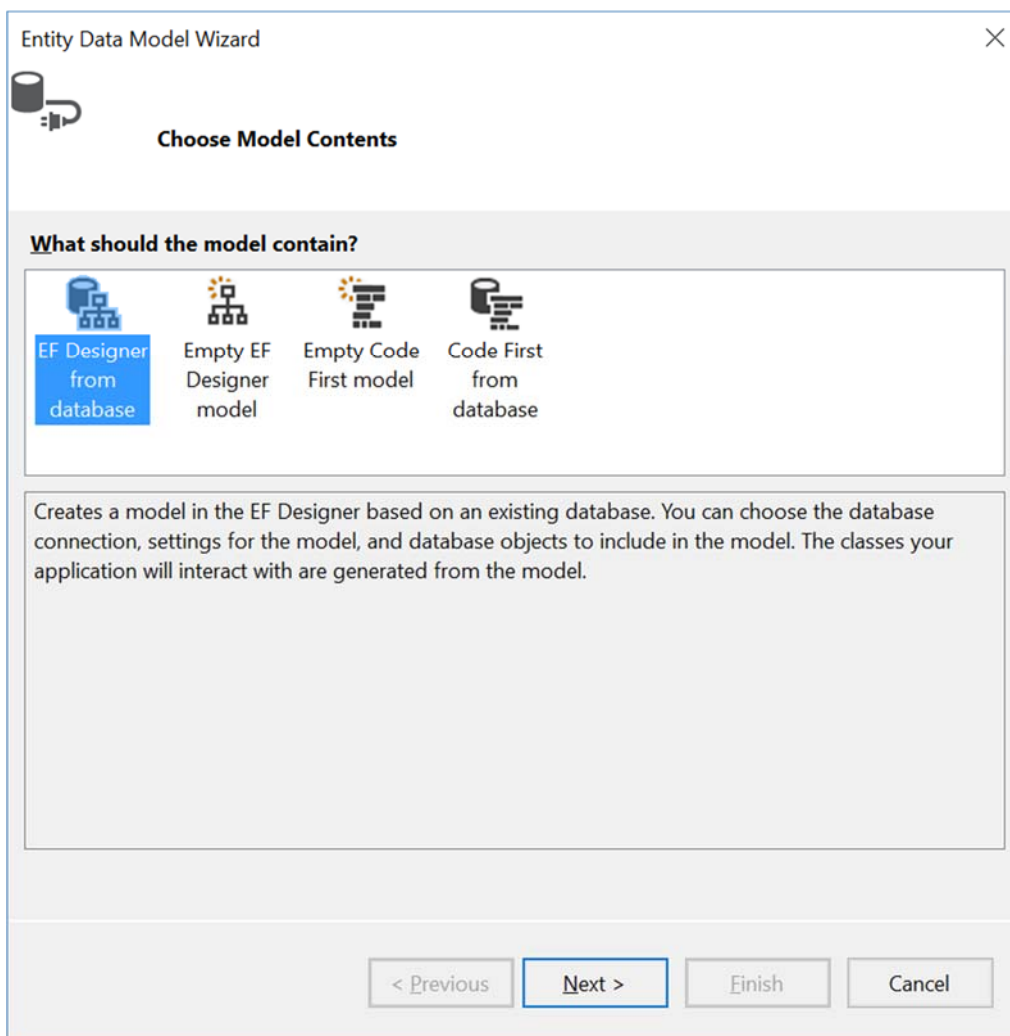
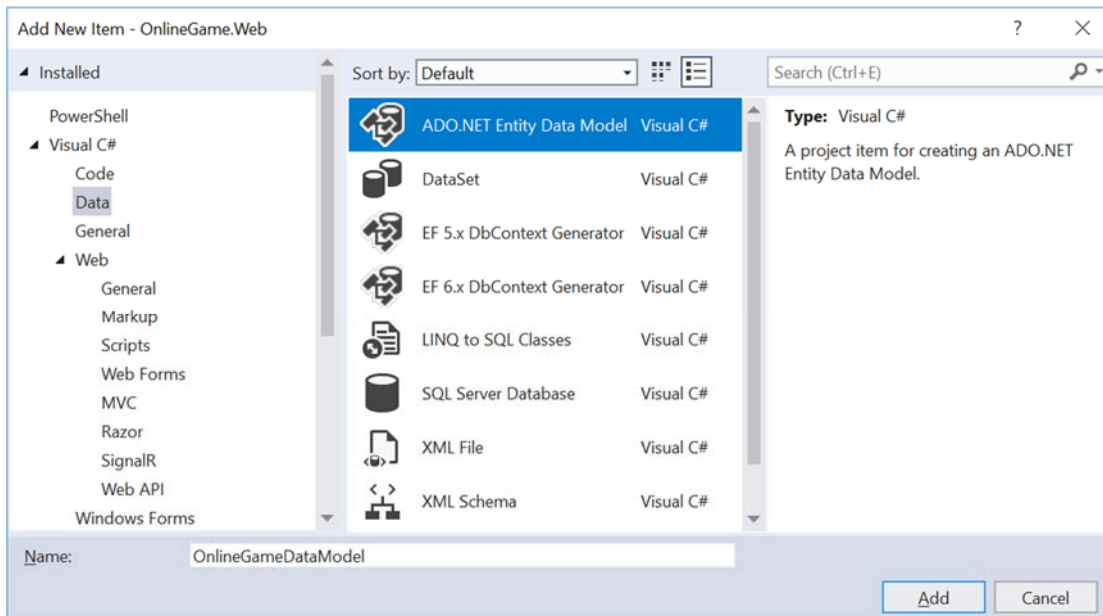
EF Designer from database

....

-->

Save Connection settings in Web.Config as:

OnlineGameContext



**Choose Your Data Connection**

Which data connection should your application use to connect to the database?

New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

- ☐ No, exclude sensitive data from the connection string. I will set it in my application code.
- ☐ Yes, include the sensitive data in the connection string.

Connection string:

☒ Save connection settings in Web.Config as:

< Previous

Next >

Finish

Cancel

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:

Microsoft SQL Server (SqlClient)

Change...

Server name:

N550JKL\SQL2016

Refresh

Log on to the server

Authentication: SQL Server Authentication

User name: Tester

Password: ●●●●

☒ Save my password

Microsoft Visual Studio



Test connection succeeded.

OK

Connect to a database

☒ Select or enter a database name:

OnlineGame

☐ Attach a database file:

Browse...

Advanced...

Test Connection

OK

Cancel

**Choose Your Data Connection****Which data connection should your application use to connect to the database?**

n550jkl\sql2016.OnlineGame.dbo



New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

- ☐ No, exclude sensitive data from the connection string. I will set it in my application code.
- ☒ Yes, include the sensitive data in the connection string.

Connection string:

```
metadata=res://*/Models.OnlineGameDataModel.csdl|
res://*/Models.OnlineGameDataModel.ssdl|
res://*/Models.OnlineGameDataModel.msl;provider=System.Data.SqlClient;provider connection
string="data source=N550JKL\SQL2016;initial catalog=OnlineGame;persist security info=True;user
id=Tester;password=*****;MultipleActiveResultSets=True;App=EntityFramework"
```

☒ Save connection settings in Web.Config as:

OnlineGameContext

< Previous

Next >

Finish

Cancel

**Choose Your Version****Which version of Entity Framework do you want to use?**

- ☒ Entity Framework 6.x
☐ Entity Framework 5.0

i It is also possible to install and use other versions of Entity Framework.
[Learn more about this](#)


< Previous

Next >

Finish


Cancel


Entity Data Model Wizard





Choose Your Database Objects and Settings


Which database objects do you want to include in your model?


☒  Tables


☒  dbo


☒  Gamer


☒  Team


☐  sysdiagrams


☐  Views


☒  Stored Procedures and Functions


☒  dbo


☐  fn_diagramobjects

☒  spAddGamer

☒  spDeleteGamer

☒  spGetGamers

☒  spSaveGamer

☐  sp_alterdiagram

☒ Pluralize or singularize generated object names☒ Include foreign key columns in the model☒ Import selected stored procedures and functions into the entity model

Model Namespace:

OnlineGameModel

< Previous

Next >

Finish

Cancel

Security Warning

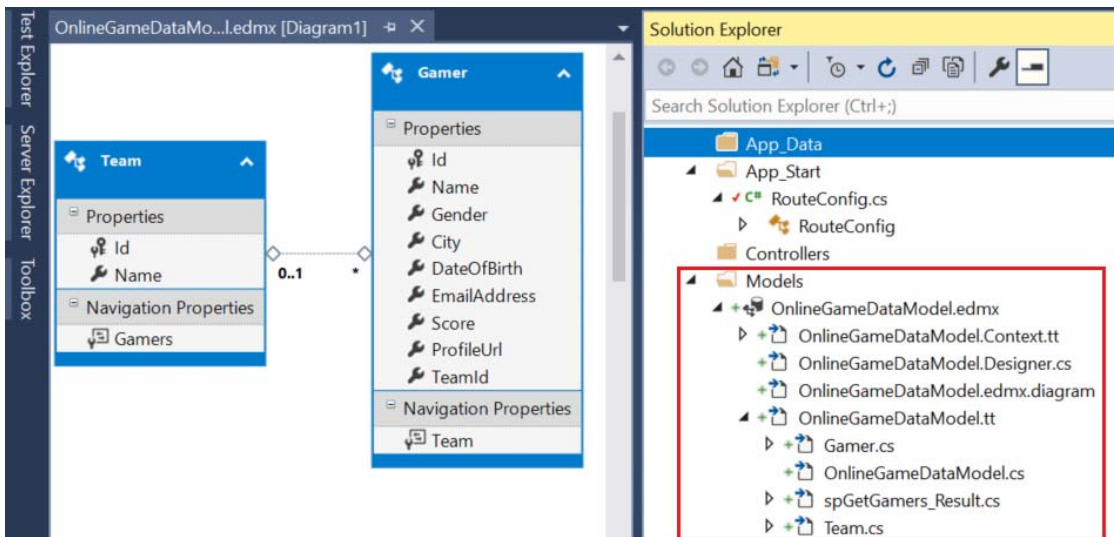
Running this text template can potentially harm your computer. Do not run it if you obtained it from an untrusted source.

Click OK to run the template.
Click Cancel to stop the process.

☐ Do not show this message again

OK

Cancel



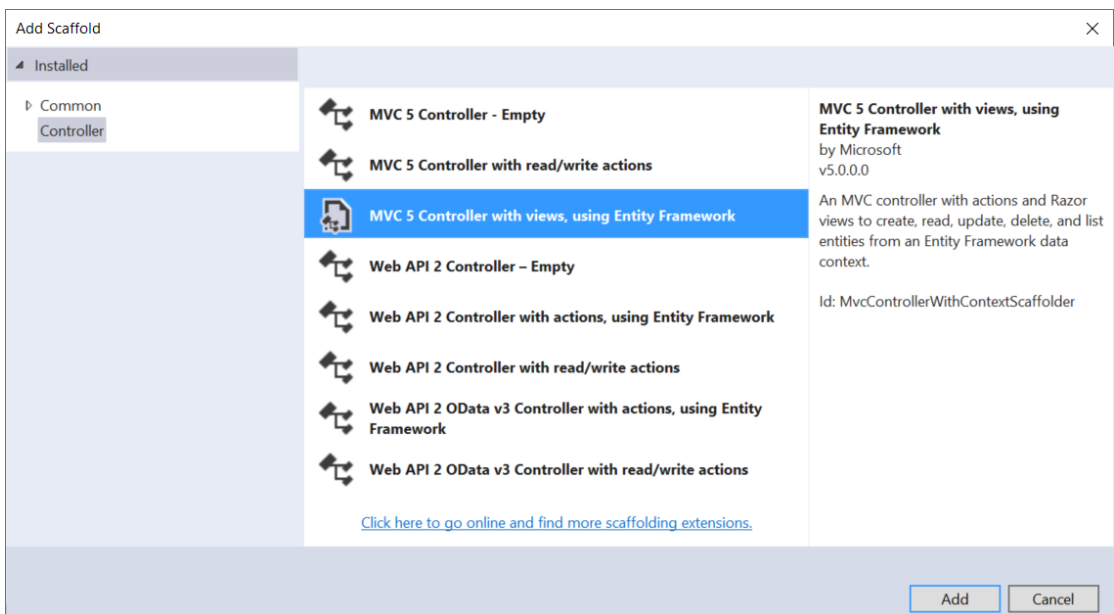
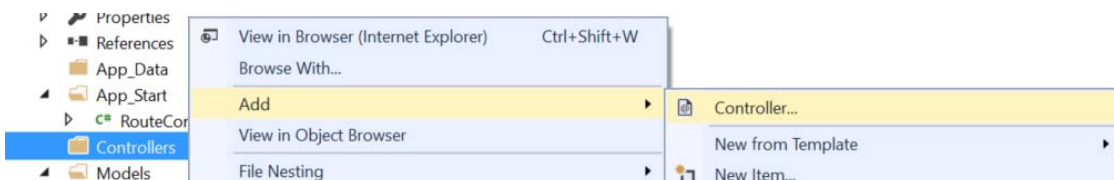
4. OnlineGame.Web

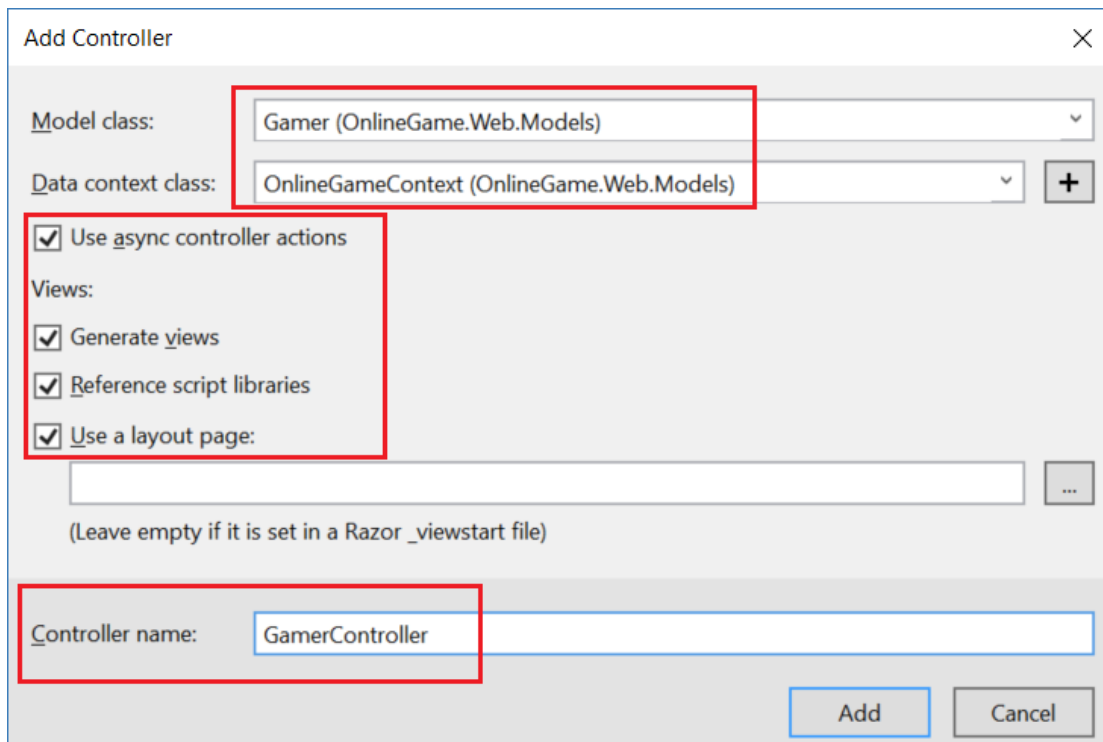
4.1. Controllers/GamersController.cs

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

-->

MVC 5 Controller with views, using Entity Framework





The 'Add Controller' dialog box in Visual Studio. It features a title bar with a close button. The 'Model class' dropdown is set to 'Gamer (OnlineGame.Web.Models)'. The 'Data context class' dropdown is set to 'OnlineGameContext (OnlineGame.Web.Models)' with a plus button to its right. A red box highlights the 'Views' section, which contains four checked options: 'Use async controller actions', 'Generate views', 'Reference script libraries', and 'Use a layout page:'. Below these is an empty text field with a browse button ('...'). A note below the field says '(Leave empty if it is set in a Razor _viewstart file)'. At the bottom, the 'Controller name' text box contains 'GamerController'. There are 'Add' and 'Cancel' buttons at the bottom right.

Add Controller

Model class: Gamer (OnlineGame.Web.Models)

Data context class: OnlineGameContext (OnlineGame.Web.Models) +

☒ Use async controller actions

Views:

☒ Generate views

☒ Reference script libraries

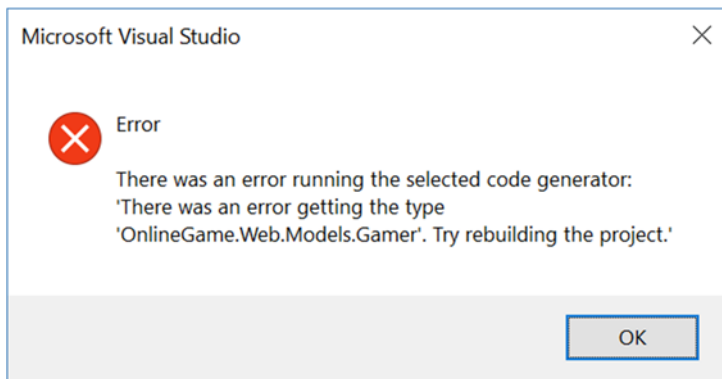
☒ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Controller name: GamerController

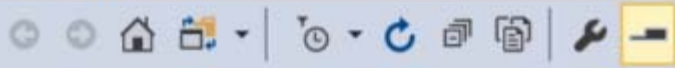
Add Cancel

If you see the following error message, then you have to re-build solution before you create the controller.



It will automatically generate the controller, views, and several javascript and css files.

Solution Explorer



Search Solution Explorer (Ctrl+;)

Solution 'OnlineGame' (1 project)

OnlineGame.Web

Connected Services

Properties

References

App_Data

App_Start

RouteConfig.cs

Content

bootstrap.css

bootstrap.min.css

Site.css

Controllers

GamersController.cs

fonts

Models

GamerDataModel.edmx

GamerDataModel.Context.tt

GamerDataModel.Designer.cs

GamerDataModel.edmx.diagram

GamerDataModel.tt

Gamer.cs

GamerDataModel.cs

spGetGamers_Result.cs

Team.cs

- Scripts
 - bootstrap.js
 - bootstrap.min.js
 - jquery-1.10.2.intellisense.js
 - jquery-1.10.2.js
 - jquery-1.10.2.min.js
 - jquery-1.10.2.min.map
 - jquery.validate-vsdoc.js
 - jquery.validate.js
 - jquery.validate.min.js
 - jquery.validate.unobtrusive.js
 - jquery.validate.unobtrusive.min.js
 - modernizr-2.6.2.js

- Views
 - Gamer
 - ✓[@] Create.cshtml
 - [@] Delete.cshtml
 - [@] Details.cshtml
 - ✓[@] Edit.cshtml
 - [@] Index.cshtml
 - Shared
 - [@] _Layout.cshtml
 - _ViewStart.cshtml
 - web.config
- Global.asax
 - Global.asax.cs
- packages.config
- Web.config

Properties Solution Explorer Team Explorer

Index

[Create Now](#)

Name	Gender	City	DateOfBirth	EmailAddress	Score	ProfileUrl	GameMoney	Name
Name01 ABB	Male	City01	28/04/1979 12:00:00 AM	1@AAA.com	3500	https://ithandyguytutorial.blogspot.com.au/	1000	Team1 Edit Details Delete
Name02 CDDE	Female	City03	24/07/1981 12:00:00 AM	2@BBB.com	3500	https://ithandyguytutorial.blogspot.com.au/	1500	Team2 Edit Details Delete
Name03 FIJK	Female	City01	5/12/1984 12:00:00 AM	3@CCCC.com	3500	https://ithandyguytutorial.blogspot.com.au/	4000	Team3 Edit Details Delete
Name04 LMOPPQ	Male	City02	29/05/1983 12:00:00 AM	4@DD.com	3500	https://ithandyguytutorial.blogspot.com.au/	2500	Team1 Edit Details Delete
Name05 QRSTT	Male	City01	20/06/1979 12:00:00 AM	5@EEE.com	3500	https://ithandyguytutorial.blogspot.com.au/	3500	Team3 Edit Details Delete
Name06 TUVVX	Female	City03	15/05/1984 12:00:00 AM	6@FF.com	3500	https://ithandyguytutorial.blogspot.com.au/	2500	Team3 Edit Details Delete
Name07 XYZZXX	Female	City01	29/04/1986 12:00:00 AM	7@GGGG.com	3500	https://ithandyguytutorial.blogspot.com.au/	4550	Team2 Edit Details Delete
Name08 ABBCDE	Male	City02	28/07/1985 12:00:00 AM	8@HH.com	3500	https://ithandyguytutorial.blogspot.com.au/	3550	Team1 Edit Details Delete
Name09 QRSTTUVVXX	Male	City02	16/04/1983 12:00:00 AM	9@III.com	3500	https://ithandyguytutorial.blogspot.com.au/	2510	Team1 Edit Details Delete

5. OnlineGame.Web

5.1. web.config

```
Web.config x Gamer.cs
4  https://go.microsoft.com/fwlink/?LinkId=301880
5  -->
6  <configuration>
7  <configSections>
8  <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/
   fwlink/?LinkId=237468 -->
9  <section name="entityFramework"
   type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework,
   Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />
10 </configSections>
11 <appSettings>
12 <add key="webpages:Version" value="3.0.0.0" />
13 <add key="webpages:Enabled" value="false" />
14 <add key="ClientValidationEnabled" value="true" />
15 <add key="UnobtrusiveJavaScriptEnabled" value="true" />
16 </appSettings>
17 <system.web>
18 <globalization culture="en-au"/>
19 <compilation debug="true" targetFramework="4.6.1" />
20 <httpRuntime targetFramework="4.6.1" />
21 </system.web>
22 <runtime>
```

```
<system.web>
  <globalization culture="en-au"/>
```

5.2. Controllers/GamersController.cs

```

using System;
using System.Data.Entity;
using System.Linq;
using System.Threading.Tasks;
using System.Net;
using System.Web.Mvc;
using OnlineGame.Web.Models;
namespace OnlineGame.Web.Controllers
{
    public class GamerController : Controller
    {
        private OnlineGameContext _db = new OnlineGameContext();
        // GET: Gamer
        [HttpGet]
        public async Task<ActionResult> Index()
        {
            IQueryable<Gamer> gamers = _db.Gamers.Include(g => g.Team);
            return View(await gamers.ToListAsync());
        }
        // 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/DetailsTwo
        [HttpGet]
        public ActionResult DetailsTwo()
        {
            BoardGame boardGame = new BoardGame();
            return View(boardGame);
        }
        // GET: Gamer/DetailsThree/5
        [HttpGet]
        public async Task<ActionResult> DetailsThree(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Gamer gamer = await _db.Gamers.FindAsync(id);
            if (gamer == null)
            {
                return HttpNotFound();
            }
            GamerA gamerA = GamerToGamerA(gamer);
            return View(gamerA);
        }
    }
}

```

```

    }
    [HttpGet]
    public async Task<ActionResult> DetailsFour(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        Gamer gamer = await _db.Gamers.FindAsync(id);
        if (gamer == null)
        {
            return HttpNotFound();
        }
        ViewData["GamerData"] = gamer;
        return View();
    }
    // GET: Gamer/Create
    [HttpGet]
    public ActionResult Create()
    {
        ViewBag.TeamId = new SelectList(_db.Teams, "Id", "Name");
        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,City,DateOfBirth,EmailAddress,Score,ProfileUrl,GameMoney,TeamId")] Gamer gamer)
    {
        if (ModelState.IsValid)
        {
            _db.Gamers.Add(gamer);
            await _db.SaveChangesAsync();
            return RedirectToAction("Index");
        }
        ViewBag.TeamId = new SelectList(_db.Teams, "Id", "Name", gamer.TeamId);
        return View(gamer);
    }
    // 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();
        }
        ViewBag.TeamId = new SelectList(_db.Teams, "Id", "Name", gamer.TeamId);
        return View(gamer);
    }
    // GET: Gamer/Edit/5
    [HttpGet]

```

```

public async Task<ActionResult> EditTwo(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Gamer gamer = await _db.Gamers.FindAsync(id);
    if (gamer == null)
    {
        return HttpNotFound();
    }
    GamerA gamerA = GamerToGamerA(gamer);
    ViewBag.TeamId = new SelectList(_db.Teams, "Id", "Name", gamerA.TeamId);
    return View(gamerA);
}
// GET: Gamer/Edit/5
[HttpGet]
public async Task<ActionResult> EditThree(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Gamer gamer = await _db.Gamers.FindAsync(id);
    if (gamer == null)
    {
        return HttpNotFound();
    }
    ViewData["GamerData"] = gamer;
    return View();
    //ViewBag.TeamId = new SelectList(_db.Teams, "Id", "Name", gamer.TeamId);
    //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,City,DateOfBirth,EmailAddress,Score,ProfileUrl,GameMoney,TeamId")] Gamer gamer)
{
    if (ModelState.IsValid)
    {
        _db.Entry(gamer).State = EntityState.Modified;
        await _db.SaveChangesAsync();
        return RedirectToAction("Index");
    }
    ViewBag.TeamId = new SelectList(_db.Teams, "Id", "Name", gamer.TeamId);
    return View(gamer);
}
[HttpPost]
public async Task<ActionResult> EditTwo(GamerA gamerA)
{
    if (ModelState.IsValid)
    {
        Gamer gamer = GamerAToGamer(gamerA);
        //Retrieve data from DB
        Gamer gamerFromDb = await _db.Gamers.SingleAsync(g => g.Id == gamerA.Id);

```

```

        //Update all properties except Email and Score
        gamerFromDb.Name = gamer.Name;
        gamerFromDb.Gender = gamer.Gender;
        gamerFromDb.City = gamer.City;
        gamerFromDb.DateOfBirth = gamer.DateOfBirth;
        //gamerFromDb.EmailAddress = gamer.EmailAddress;
        //gamerFromDb.Score = gamer.Score;
        gamerFromDb.ProfileUrl = gamer.ProfileUrl;
        gamerFromDb.GameMoney = gamer.GameMoney;
        gamerFromDb.TeamId = gamer.TeamId;
        _db.Entry(gamerFromDb).State = EntityState.Modified;
        await _db.SaveChangesAsync();
        //return RedirectToAction("Index");
        return RedirectToAction("DetailsThree", new { id = gamerA.Id });
    }
    ViewBag.TeamId = new SelectList(_db.Teams, "Id", "Name", gamerA.TeamId);
    return View(gamerA);
}
[HttpPost]
public async Task<ActionResult> EditThree(int id, string name, string gender, string city, DateTime?
dateOfBirth, string emailAddress, int? score, string profileUrl, int? gameMoney, int? teamId)
//public async Task<ActionResult> EditThree(Gamer gamer)
{
    var gamerData = ViewData["GamerData"];
    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);
    _db.Gamers.Remove(gamer);
    await _db.SaveChangesAsync();
    return RedirectToAction("Index");
}
private static GamerA GamerToGamerA(Gamer gamer)
{
    GamerA gamerA = new GamerA
    {
        Id = gamer.Id,
        Name = gamer.Name,
        Gender = gamer.Gender,
        City = gamer.City,
        DateOfBirth = gamer.DateOfBirth,

```

```

        EmailAddress = gamer.EmailAddress,
        Score = gamer.Score,
        ProfileUrl = gamer.ProfileUrl,
        GameMoney = gamer.GameMoney,
        TeamId = gamer.TeamId
    };
    return gamerA;
}
private static Gamer GamerAToGamer(GamerA gamerA)
{
    Gamer gamer = new Gamer
    {
        Id = gamerA.Id,
        Name = gamerA.Name,
        Gender = gamerA.Gender,
        City = gamerA.City,
        DateOfBirth = gamerA.DateOfBirth,
        EmailAddress = gamerA.EmailAddress,
        Score = gamerA.Score,
        ProfileUrl = gamerA.ProfileUrl,
        GameMoney = gamerA.GameMoney,
        TeamId = gamerA.TeamId
    };
    return gamer;
}
protected override void Dispose(bool disposing)
{
    if (disposing)
    {
        _db.Dispose();
    }
    base.Dispose(disposing);
}
}
}

```

5.3. Views/Gamer/Index.cshtml

```

@model IEnumerable<OnlineGame.Web.Models.Gamer>
@{
    ViewBag.Title = "Index";
}
<h2>Index</h2>
<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Gender)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.City)
        </th>
    </tr>

```



```

        <th>
            @Html.DisplayNameFor(model => model.DateOfBirth)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.EmailAddress)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Score)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.ProfileUrl)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.GameMoney)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Team.Name)
        </th>
    <th></th>
</tr>
@foreach (var item in Model)
{
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Name)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Gender)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.City)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.DateOfBirth)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.EmailAddress)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Score)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.ProfileUrl)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.GameMoney)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Team.Name)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id = item.Id }) |
            @Html.ActionLink("EditTwo", "EditTwo", new { id = item.Id }) |

```

```

@Html.ActionLink("EditThree", "EditThree", new { id = item.Id }) |
@Html.ActionLink("Details", "Details", new { id = item.Id }) |
@Html.ActionLink("DetailsTwo", "DetailsTwo", new { id = item.Id }) |
@Html.ActionLink("DetailsThree", "DetailsThree", new { id = item.Id }) |
@Html.ActionLink("DetailsFour", "DetailsFour", new { id = item.Id }) |
@Html.ActionLink("Delete", "Delete", new { id = item.Id })

</td>
</tr>
}
</table>

```

Index

[Create New](#)

Full Name	Gender	City	DateOfBirth	EmailAddress	Score	ProfileUrl	GameMoney	Name	
Name01 ABB	Male	City01ss	30/04/1979	1@AAA.com	3500	https://ithandyguytutorial.blogspot.com.au/	\$1,000.00	Team1	Edit EditTwo EditThree Details DetailsTwo DetailsThree DetailsFour Delete
Name02 CDDE	Female	City03	30/09/1981	2@BBB.com	3500	https://ithandyguytutorial.blogspot.com.au/	\$1,500.00	Team2	Edit EditTwo EditThree Details DetailsTwo DetailsThree DetailsFour Delete
Name03 FIJK	Female	City01	05/12/1984	3@CCCC.com	3500	https://ithandyguytutorial.blogspot.com.au/	\$4,000.00	Team3	Edit EditTwo EditThree Details DetailsTwo DetailsThree DetailsFour Delete

5.4. Models/Gamer/Gamer.cs

```

using System.ComponentModel.DataAnnotations;
namespace OnlineGame.Web.Models
{
    [MetadataType(typeof(GamerMetaData))]
    //[DisplayColumn("Id")]
    [DisplayColumn("Name")]
    public partial class Gamer
    {
    }
}

```

5.5. Models/Gamer/GamerMetaData.cs

```

using System;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
namespace OnlineGame.Web.Models
{
    public class GamerMetaData
    {
        //[HiddenInput(DisplayValue = false)]
        public int Id { get; set; }

        //1.
        ///[DisplayAttribute(Name="Full Name")]
        ///[Display(Name = "Full Name")]
        ///[DisplayName("Full Name")]
    }
}

```

```

//Display Name as "Full Name"
[DisplayName("Full Name")]
public string Name { get; set; }
//2.
//If gender is NULL, then display "Gender not specified".
[DisplayFormat(NullDisplayText = "Gender not specified")]
public string Gender { get; set; }
public string City { get; set; }
//3.
//DateTime
//3.1.
////[DisplayFormat(DataFormatString = ".....")]
//3.1.1.
/////[DisplayFormat(DataFormatString = "{0:d}")]
/////[DisplayFormatAttribute(DataFormatString="{0:d}")]
////public Nullable<System.DateTime> DateOfBirth { get; set; }
//Display only the date part. E.g. 29/04/1986
//3.1.2.
////[DisplayFormat(DataFormatString = "{0:dd/MM/yyyy HH:mm:ss}")]
////public Nullable<System.DateTime> DateOfBirth { get; set; }
//Display in 24 hour notation. E.g. 29/04/1986 13:00:00
//3.1.3.
////[DisplayFormat(DataFormatString = "{0:dd/MM/yyyy hh:mm:ss tt}")]
////public Nullable<System.DateTime> DateOfBirth { get; set; }
//Display in 12 hour notation. E.g. 29/04/1986 1:00:00 PM
//3.2.
////[DataType(DataType....)]
//3.2.1.
////[DataType(DataType.Date)]
////public Nullable<System.DateTime> DateOfBirth { get; set; }
//Display only date part
//Please be aware, it actually covert DateTime to Date.
//so Views/Shared/EditorTemplates/DateTime.cshtml will not Work.
//3.2.2.
////[DataType(DataType.Time)]
////public Nullable<System.DateTime> DateOfBirth { get; set; }
//Display only 12 hour notation Time part
//Please be aware, it actually covert DateTime to Date.
//so Views/Shared/EditorTemplates/DateTime.cshtml will not Work.
//[DataType(DataType.Date)] //Views/Shared/EditorTemplates/DateTime.cshtml will not Work.
//[DisplayFormat(DataFormatString = "{0:dd/MM/yyyy hh:mm:ss tt}")]
[DisplayFormat(DataFormatString = "{0:dd/MM/yyyy}", ApplyFormatInEditMode = true)]
public Nullable<System.DateTime> DateOfBirth { get; set; }
//5.
// Display mailto hyperlink
[DataType(DataType.EmailAddress)]
//[ReadOnly(true)]
public string EmailAddress { get; set; }
//4.
////[ScaffoldColumn(false)]
//[ScaffoldColumn(false)] attribute means it will not display the column
//when using @Html.DisplayForModel() helper.
[ScaffoldColumn(false)]

```

```

public Nullable<int> Score { get; set; }
//8.
//8.1.
//In the Models/Gamer/GamerMetaData.cs
////[DataType(DataType.Url)]
////[UIHint("UrlToNewWindow")]
////public string ProfileUrl { get; set; }
//[DataType(DataType.Url)] attribute will display a hyperlink.
//[UIHint("UrlToNewWindow")] attribute specify the name of view DisplayTemplate
//to display the property data.
//In this case, it will look for "DisplayTemplates/UrlToNewWindow.cshtml"
//under "Shared" folder or "Gamer" folder.
//Use that view template to display the data of this property.
//8.2.
////<a href="@ViewData.Model" target="_blank">@ViewData.Model</a>
//In the Shared/DisplayTemplates/UrlToNewWindow.cshtml,
//@ViewData.Model will take the Model data from the parent view.
//In this case, it will return a profile url.
[DataType(DataType.Url)]
[UIHint("UrlToNewWindow")]
public string ProfileUrl { get; set; }
//7.
////[DataType(DataType.Currency)]
////public Nullable<int> GameMoney { get; set; }
//Display the currency symbol by globalization culture in web.config file.
//7.1.
//In System.Web tag of web.config file,
//7.1.1.
//<system.web> <globalization culture = "en-au" /> ...
//Display au $ symbol.
//7.1.2.
//<system.web> <globalization culture = "en-uk" /> ...
//Display UK pound symbol.
[DataType(DataType.Currency)]
public Nullable<int> GameMoney { get; set; }
public Nullable<int> TeamId { get; set; }
}
}
/*
-----
1.
////[DisplayAttribute(Name="Full Name")]
////[Display(Name = "Full Name")]
//[DisplayName("Full Name")]
//public string Name { get; set; }
Display Name as "Full Name"
-----
2.
//[DisplayFormat(NullDisplayText = "Gender not specified")]
//public string Gender { get; set; }
If gender is NULL, then display "Gender not specified".
-----
3.
DateTime
-----
3.1.
//[DisplayFormat(DataFormatString = ".....")]
3.1.1.

```

```

////[DisplayFormat(DataFormatString = "{0:d}")]
//[DisplayFormatAttribute(DataFormatString="{0:d}")]
//public Nullable<System.DateTime> DateOfBirth { get; set; }
Display only the date part. E.g. 29/04/1986
3.1.2.
//[DisplayFormat(DataFormatString = "{0:dd/MM/yyyy HH:mm:ss}")]
//public Nullable<System.DateTime> DateOfBirth { get; set; }
Display in 24 hour notation. E.g. 29/04/1986 13:00:00
3.1.3.
//[DisplayFormat(DataFormatString = "{0:dd/MM/yyyy hh:mm:ss tt}")]
//public Nullable<System.DateTime> DateOfBirth { get; set; }
Display in 12 hour notation. E.g. 29/04/1986 1:00:00 PM
-----
3.2.
//[DataType(DataType....)]
3.2.1.
//[DataType(DataType.Date)]
//public Nullable<System.DateTime> DateOfBirth { get; set; }
Display only date part
Please be aware, it actually covert DateTime to Date.
So Views/Shared/EditorTemplates/DateTime.cshtml will not Work.
3.2.2.
//[DataType(DataType.Time)]
//public Nullable<System.DateTime> DateOfBirth { get; set; }
Display only 12 hour notation Time part
Please be aware, it actually covert DateTime to Date.
So Views/Shared/EditorTemplates/DateTime.cshtml will not Work.
-----
4.
//[ScaffoldColumn(false)]
public Nullable<int> Score { get; set; }
[ScaffoldColumn(false)] attribute means it will not display the column
when using @Html.DisplayForModel() helper
-----
5.
//[DataType(DataType.EmailAddress)]
//public string EmailAddress { get; set; }
Display mailto hyperlink
-----
6.
//[DataType(DataType.Url)]
//public string ProfileUrl { get; set; }
Display hyperlink
-----
7.
//[DataType(DataType.Currency)]
//public Nullable<int> GameMoney { get; set; }
Display the currency symbol by globalization culture in web.config file.
7.1.
In System.Web tag of web.config file,
7.1.1.
<system.web> <globalization culture="en-au"/> ...
Display au $ symbol.
7.1.2.
<system.web> <globalization culture="en-uk"/> ...
Display UK pound symbol.
-----
8.
8.1.
In the Models/Gamer/GamerMetaData.cs
//[DataType(DataType.Url)]
//[UIHint("UrlToNewWindow")]
//public string ProfileUrl { get; set; }
[DataType(DataType.Url)] attribute will display a hyperlink.
[UIHint("UrlToNewWindow")] attribute specify the name of view DisplayTemplate
to display the property data.
In this case, it will look for "DisplayTemplates/UrlToNewWindow.cshtml"

```

under "Shared" folder or "Gamer" folder.

Use that view template to display the data of this property.

8.2.

```
//<a href="@ViewData.Model" target="_blank">@ViewData.Model</a>
```

In the Shared/DisplayTemplates/UrlToNewWindow.cshtml,

@ViewData.Model will take the Model data from the parent view.

In this case, it will return a profile url.

9.

```
//[HiddenInput(DisplayValue = false)]
```

```
//public int Id { get; set; }
```

```
[HiddenInput(DisplayValue = false)] attribute
```

means it will become a hidden input

When using @Html.DisplayForModel() helper or @Html.EditorForModel()

10.

```
//[DataType(DataType.EmailAddress)]
```

```
//[ReadOnly(true)]
```

```
//public string EmailAddress { get; set; }
```

```
[DataType(DataType.EmailAddress)] attribute display mailto hyperlink.
```

```
[ReadOnly(true)] or you may delete Setter.
```

It will make this property un-editable.

You may set a breakpoint to see the [HttpPost] action model for this property is null.

```
*/
```

5.6. Models/Gamer/BoardGame.cs

```
using System.Data.Entity;
```

```
using System.Linq;
```

```
namespace OnlineGame.Web.Models
```

```
{
    public class BoardGame
    {
        public Gamer GameHolder
        {
            get
            {
                using (OnlineGameContext dbContext = new OnlineGameContext())
                {
                    return dbContext.Gamers.SingleAsync(x => x.Id == 1).Result;
                }
            }
        }
    }
}
```

5.7. Models/Gamer/GamerA.cs

```
using System;
```

```
using System.ComponentModel;
```

```
using System.ComponentModel.DataAnnotations;
```

```
using System.Web.Mvc;
```

```
namespace OnlineGame.Web.Models
```

```
{
    public class GamerA
    {
```

```

//9.
////[HiddenInput(DisplayValue = false)]
////public int Id { get; set; }
//[HiddenInput(DisplayValue = false)] attribute
//means it will become a hidden input
//When using @Html.DisplayForModel() helper or @Html.EditorForModel()
[HiddenInput(DisplayValue = false)]
public int Id { get; set; }
//1.
////[DisplayAttribute(Name="Full Name")]
////[Display(Name = "Full Name")]
////[DisplayName("Full Name")]
//Display Name as "Full Name"
[DisplayName("Full Name")]
public string Name { get; set; }
//2.
//If gender is NULL, then display "Gender not specified".
[DisplayFormat(NullDisplayText = "Gender not specified")]
public string Gender { get; set; }
public string City { get; set; }
//3.
//DateTime
//3.1.
////[DisplayFormat(DataFormatString = ".....")]
//3.1.1.
/////[DisplayFormat(DataFormatString = "{0:d}")]
////[DisplayFormatAttribute(DataFormatString="{0:d}")]
////public Nullable<System.DateTime> DateOfBirth { get; set; }
//Display only the date part. E.g. 29/04/1986
//3.1.2.
////[DisplayFormat(DataFormatString = "{0:dd/MM/yyyy HH:mm:ss}")]
////public Nullable<System.DateTime> DateOfBirth { get; set; }
//Display in 24 hour notation. E.g. 29/04/1986 13:00:00
//3.1.3.
////[DisplayFormat(DataFormatString = "{0:dd/MM/yyyy hh:mm:ss tt}")]
////public Nullable<System.DateTime> DateOfBirth { get; set; }
//Display in 12 hour notation. E.g. 29/04/1986 1:00:00 PM
//3.2.
////[DataType(DataType....)]
//3.2.1.
////[DataType(DataType.Date)]
////public Nullable<System.DateTime> DateOfBirth { get; set; }
//Display only date part
//Please be aware, it actually covert DateTime to Date.
//So Views/Shared/EditorTemplates/DateTime.cshtml will not Work.
//3.2.2.
////[DataType(DataType.Time)]
////public Nullable<System.DateTime> DateOfBirth { get; set; }
//Display only 12 hour notation Time part
//Please be aware, it actually covert DateTime to Date.
//So Views/Shared/EditorTemplates/DateTime.cshtml will not Work.
//[DataType(DataType.Date)] //Views/Shared/EditorTemplates/DateTime.cshtml will not Work.
//[DisplayFormat(DataFormatString = "{0:dd/MM/yyyy hh:mm:ss tt}")]

```

```

[DisplayFormat(DataFormatString = "{0:d}")]
public DateTime? DateOfBirth { get; set; }
//10.
////[DataType(DataType.EmailAddress)]
////[ReadOnly(true)]
////public string EmailAddress { get; set; }
//[DataType(DataType.EmailAddress)] attribute display mailto hyperlink.
//[ReadOnly(true)] or you may delete Setter.
//It will make this property un-editable.
//You may set a breakpoint to see the [HttpPost] action model for this property is null.
[DataType(DataType.EmailAddress)]
[ReadOnly(true)]
public string EmailAddress { get; set; }
//4.
////[ScaffoldColumn(false)]
//[ScaffoldColumn(false)] attribute means it will not display the column
//when using @Html.DisplayForModel() helper.
[ScaffoldColumn(false)]
public int? Score { get; set; }
//8.
//8.1.
//In the Models/Gamer/GamerMetaData.cs
////[DataType(DataType.Url)]
////[UIHint("UrlToNewWindow")]
////public string ProfileUrl { get; set; }
//[DataType(DataType.Url)] attribute will display a hyperlink.
//[UIHint("UrlToNewWindow")] attribute specify the name of view DisplayTemplate
//to display the property data.
//In this case, it will look for "DisplayTemplates/UrlToNewWindow.cshtml"
//under "Shared" folder or "Gamer" folder.
//Use that view template to display the data of this property.
//8.2.
//<a href="@ViewData.Model" target="_blank">@ViewData.Model</a>
//In the Shared/DisplayTemplates/UrlToNewWindow.cshtml,
//@ViewData.Model will take the Model data from the parent view.
//In this case, it will return a profile url.
[DataType(DataType.Url)]
[UIHint("UrlToNewWindow")]
public string ProfileUrl { get; set; }
//7.
////[DataType(DataType.Currency)]
////public Nullable<int> GameMoney { get; set; }
//Display the currency symbol by globalization culture in web.config file.
//7.1.
//In System.Web tag of web.config file,
//7.1.1.
//<system.web> <globalization culture = "en-au" /> ...
//Display au $ symbol.
//7.1.2.
//<system.web> <globalization culture = "en-uk" /> ...
//Display UK pound symbol.
[DataType(DataType.Currency)]
public int? GameMoney { get; set; }
public int? TeamId { get; set; }

```



```

        public virtual Team Team { get; set; }
    }
}
/*
-----
1.
////[DisplayAttribute(Name="Full Name")]
////[Display(Name = "Full Name")]
//[DisplayName("Full Name")]
//public string Name { get; set; }
Display Name as "Full Name"
-----
2.
//[DisplayFormat(NullDisplayText = "Gender not specified")]
//public string Gender { get; set; }
If gender is NULL, then display "Gender not specified".
-----
3.
DateTime
-----
3.1.
//[DisplayFormat(DataFormatString = ".....")]
3.1.1.
////[DisplayFormat(DataFormatString = "{0:d}")]
//[DisplayFormatAttribute(DataFormatString="{0:d}")]
//public Nullable<System.DateTime> DateOfBirth { get; set; }
Display only the date part. E.g. 29/04/1986
3.1.2.
//[DisplayFormat(DataFormatString = "{0:dd/MM/yyyy HH:mm:ss}")]
//public Nullable<System.DateTime> DateOfBirth { get; set; }
Display in 24 hour notation. E.g. 29/04/1986 13:00:00
3.1.3.
//[DisplayFormat(DataFormatString = "{0:dd/MM/yyyy hh:mm:ss tt}")]
//public Nullable<System.DateTime> DateOfBirth { get; set; }
Display in 12 hour notation. E.g. 29/04/1986 1:00:00 PM
-----
3.2.
//[DataType(DataType....)]
3.2.1.
//[DataType(DataType.Date)]
//public Nullable<System.DateTime> DateOfBirth { get; set; }
Display only date part
Please be aware, it actually covert DateTime to Date.
So Views/Shared/EditorTemplates/DateTime.cshtml will not Work.
3.2.2.
//[DataType(DataType.Time)]
//public Nullable<System.DateTime> DateOfBirth { get; set; }
Display only 12 hour notation Time part
Please be aware, it actually covert DateTime to Date.
So Views/Shared/EditorTemplates/DateTime.cshtml will not Work.
-----
4.
//[ScaffoldColumn(false)]
public Nullable<int> Score { get; set; }
[ScaffoldColumn(false)] attribute means it will not display the column
when using @Html.DisplayForModel() helper
-----
5.
//[DataType(DataType.EmailAddress)]
//public string EmailAddress { get; set; }
Display mailto hyperlink
-----
6.
//[DataType(DataType.Url)]
//public string ProfileUrl { get; set; }
Display hyperlink
-----

```

```

7.
//[DataType(DataType.Currency)]
//public Nullable<int> GameMoney { get; set; }
Display the currency symbol by globalization culture in web.config file.
7.1.
In System.Web tag of web.config file,
7.1.1.
<system.web> <globalization culture="en-au"/> ...
Display au $ symbol.
7.1.2.
<system.web> <globalization culture="en-uk"/> ...
Display UK pound symbol.
-----
8.
8.1.
In the Models/Gamer/GamerMetaData.cs
//[DataType(DataType.Url)]
//[UIHint("UrlToNewWindow")]
//public string ProfileUrl { get; set; }
[DataType(DataType.Url)] attribute will display a hyperlink.
[UIHint("UrlToNewWindow")] attribute specify the name of view DisplayTemplate
to display the property data.
In this case, it will look for "DisplayTemplates/UrlToNewWindow.cshtml"
under "Shared" folder or "Gamer" folder.
Use that view template to display the data of this property.
8.2.
//<a href="@ViewData.Model" target="_blank">@ViewData.Model</a>
In the Shared/DisplayTemplates/UrlToNewWindow.cshtml,
@ViewData.Model will take the Model data from the parent view.
In this case, it will return a profile url.
-----
9.
//[HiddenInput(DisplayValue = false)]
//public int Id { get; set; }
[HiddenInput(DisplayValue = false)] attribute
means it will become a hidden input
When using @Html.DisplayForModel() helper or @Html.EditorForModel()
-----
10.
//[DataType(DataType.EmailAddress)]
//[ReadOnly(true)]
//public string EmailAddress { get; set; }
[DataType(DataType.EmailAddress)] attribute display mailto hyperlink.
[ReadOnly(true)] or you may delete Setter.
It will make this property un-editable.
You may set a breakpoint to see the [HttpPost] action model for this property is null.
*/

```

5.8. Views/Gamer/Details.cshtml

```

@model OnlineGame.Web.Models.Gamer
@{
    ViewBag.Title = "Details";
}
<h2>Details (Model is Gamer)</h2>
<div>
    <h4>Gamer</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Name)

```

```

</dt>
<dd>
    @Html.DisplayFor(model => model.Name)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.Gender)
</dt>
<dd>
    @Html.DisplayFor(model => model.Gender)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.City)
</dt>
<dd>
    @Html.DisplayFor(model => model.City)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.DateOfBirth)
</dt>
<dd>
    @Html.DisplayFor(model => model.DateOfBirth)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.EmailAddress)
</dt>
<dd>
    @Html.DisplayFor(model => model.EmailAddress)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.Score)
</dt>
<dd>
    @Html.DisplayFor(model => model.Score)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.ProfileUrl)
</dt>
<dd>
    @Html.DisplayFor(model => model.ProfileUrl)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.GameMoney)
</dt>
<dd>
    @Html.DisplayFor(model => model.GameMoney)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.Team.Name)
</dt>
<dd>
    @Html.DisplayFor(model => model.Team.Name)
</dd>
</dl>

```

```

</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.Id }) |
    @Html.ActionLink("Back to List", "Index")
</p>
<hr />
<h2>Html.DisplayForModel()</h2>
@Html.DisplayForModel()
@*
1.
//@Html.DisplayNameFor(model => model.Name)
It will display "Full Name",
because Name property in Gamer has [DisplayName("Full Name")] attribute.
//@Html.DisplayFor(model => model.Name)
It will display "Name02 CDDE"
-----
2.
//@Html.DisplayForModel()
It will display everything
except the properties with [ScaffoldColumn(false)] attribute.
-----
3.
//@Html.ActionLink("Back to List", "Index")
It will create
//<a href="/Gamer/Index">Back to List</a>
-----
4.
There are 2 categories of built-in templated helpers.
-----
4.1.
Display Templates
-----
4.1.1.
//@Html.DisplayFor(model => model.Name)
The view must have strongly typed view Model.
It can work with the complex type Model property.
It is similar to @Html.DisplayTextFor(model => model.GameHolder)
//@Html.DisplayTextFor(model => model.GameHolder)
model.GameHolder will return a Gamer object.
The Gamer class has [DisplayColumn("Name")] attribute,
so it will display Gamer Name property value
which is the full name of that gamer.
-----
4.1.2.
//@Html.DisplayForModel()
The view must have strongly typed view Model.
It will display every property in view model
except the properties with [ScaffoldColumn(false)] attribute.
-----
4.1.3.
@Html.Display helper does not need strongly typed view Mode.
//ViewData["GamerData"] = gamer;
//return View();
In the controller, we put the gamer object into ViewData["GamerData"]
"GamerData" in this case is the key of ViewData.
ViewData["GamerData"] contains that gamer object data,
so we don't have to use a view model.
//@Html.Display("GamerData")
In the view, we use @Html.Display("GamerData")
to retrieve the Gamer data from ViewData["GamerData"].
It will display everything
except the properties with [ScaffoldColumn(false)] attribute.
-----
4.2.
Display Templates
-----

```

4.2.1.

```
//@Html.EditorFor(model => model.Name)
```

The view must have strongly typed view Model.

It will create a textbox for the property value input.

4.2.2.

```
//@Html.EditorForModel()
```

The view must have strongly typed view Model.

It will create textbox input for every properties in view model

except the properties with [ScaffoldColumn(false)] attribute.

4.2.3.

@Html.Editor helper does not need strongly typed view Mode.

```
//ViewData["GamerData"] = gamer;
```

```
//return View();
```

In the controller, we put the gamer object into ViewData["GamerData"]

"GamerData" in this case is the key of ViewData.

ViewData["GamerData"] contains that gamer object data,

so we don't have to use a view model.

```
//@Html.Editor("GamerData")
```

In the view, we use @Html.Editor("GamerData")

to retrieve the Gamer data from ViewData["GamerData"].

It will create textbox input for every properties in ViewData["GamerData"]

except the properties with [ScaffoldColumn(false)] attribute.

However, we pressed submit button and call the [HttpPost] action

```
//public async Task<ActionResult> EditThree(int id, string name, string gender, string city, DateTime? dateOfBirth, string emailAddress, int? score, string profileUrl, int? gameMoney, int? teamId)
```

OR

```
//public async Task<ActionResult> EditThree(Gamer gamer)
```

Both ways can not retrieve the data because it is not strongly typed.

I don't suggest to use @Html.Editor helper

*@

Details (Model is Gamer)

Gamer

Full Name	Name01 ABB
Gender	Male
City	City01ss
DateOfBirth	30/04/1979
EmailAddress	1@AAA.com
Score	3500
ProfileUrl	https://ithandyguytutorial.blogspot.com.au/
GameMoney	\$1,000.00
Name	Team1

[Edit](#) | [Back to List](#)

Html.DisplayForModel()

Id
1
Full Name
Name01 ABB
Gender
Male
City
City01ss
DateOfBirth
30/04/1979
EmailAddress
1@AAA.com
ProfileUrl
<https://ithandyguytutorial.blogspot.com.au/>
GameMoney
\$1,000.00
TeamId

5.9. Views/Gamer/DetailsTwo.cshtml (Display Complex Type Sample)

Add View

View name:

Template:

Model class:

Data context class:

Options:

☐ Create as a partial view

☒ Reference script libraries

☒ Use a layout page:

...

(Leave empty if it is set in a Razor _viewstart file)

Add Cancel

```
@using OnlineGame.Web.Models
@model BoardGame
@{
    ViewBag.Title = "DetailsTwo";
}
```

```
<h2>Display Complex Type Sample</h2>
@Html.DisplayTextFor(model => model.GameHolder)
@*
@Html.DisplayTextFor(model => model.GameHolder)
model.GameHolder will return a Gamer object.
The Gamer class has [DisplayColumn("Name")] attribute,
so it will display Gamer Name property value
which is the full name of that gamer.
*@
```

Display Complex Type Sample

Name01 ABB

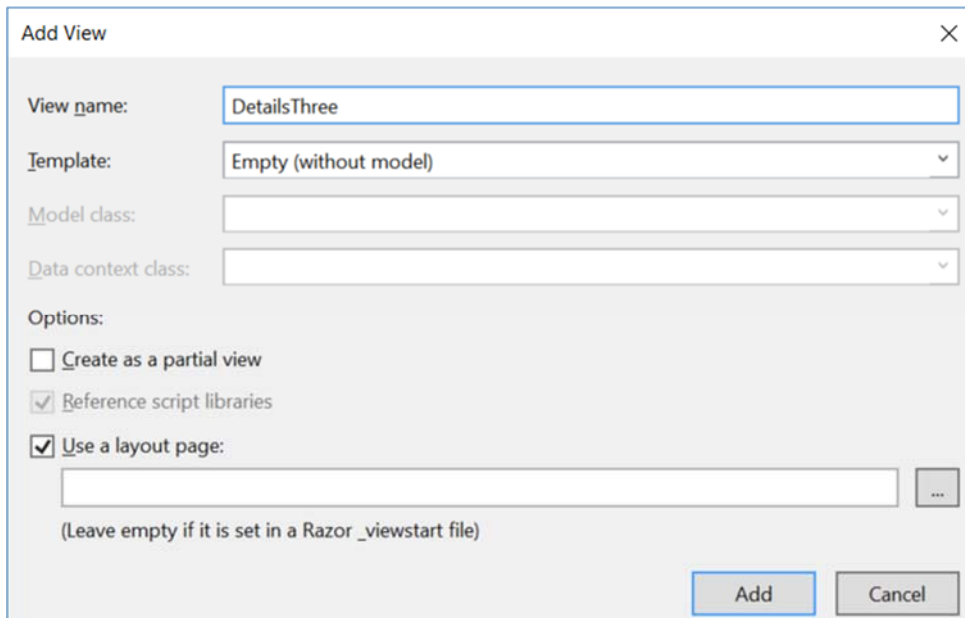
5.10. Views/Shared/DisplayTemplates/UrlToNewWindow.cshtml (UIHint Sample)

```
<a href="@ViewData.Model" target="_blank">@ViewData.Model</a>
@*
1.
//target="_blank"
measn open the link to new windows.
8.
8.1.
In the Models/Gamer/GamerMetaData.cs
//[DataType(DataType.Url)]
//[UIHint("UrlToNewWindow")]
//public string ProfileUrl { get; set; }
[DataType(DataType.Url)] attribute will display a hyperlink.
[UIHint("UrlToNewWindow")] attribute specify the name of view DisplayTemplate
to display the property data.
In this case, it will look for "DisplayTemplates/UrlToNewWindow.cshtml"
under "Shared" folder or "Gamer" folder.
Use that view template to dispaly the data of this property.
8.2.
//<a href="@ViewData.Model" target="_blank">@ViewData.Model</a>
In the Shared/DisplayTemplates/UrlToNewWindow.cshtml,
@ViewData.Model will take the Model data from the parent view.
```

In this case, it will return a profile url.

*@

5.11. Views/Gamer/DetailsThree.cshtml ([HiddenInput(DisplayValue = false)], [ReadOnly(true)])



Add View

View name: DetailsThree

Template: Empty (without model)

Model class:

Data context class:

Options:

☐ Create as a partial view

☒ Reference script libraries

☒ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Add Cancel

@model OnlineGame.Web.Models.GamerA

@{

ViewBag.Title = "DetailsThree";

}

<h2>DetailsThree</h2>

<h2>Details (Model is GamerA)</h2>

<div>

<h4>Gamer</h4>

<hr />

<dl class="dl-horizontal">

<dt>

@Html.DisplayNameFor(model => model.Name)

</dt>

<dd>

@Html.DisplayFor(model => model.Name)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Gender)

</dt>

<dd>

@Html.DisplayFor(model => model.Gender)

</dd>

<dt>

@Html.DisplayNameFor(model => model.City)

</dt>

<dd>

@Html.DisplayFor(model => model.City)

</dd>


```

<dt>
    @Html.DisplayNameFor(model => model.DateOfBirth)
</dt>
<dd>
    @Html.DisplayFor(model => model.DateOfBirth)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.EmailAddress)
</dt>
<dd>
    @Html.DisplayFor(model => model.EmailAddress)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.Score)
</dt>
<dd>
    @Html.DisplayFor(model => model.Score)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.ProfileUrl)
</dt>
<dd>
    @Html.DisplayFor(model => model.ProfileUrl)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.GameMoney)
</dt>
<dd>
    @Html.DisplayFor(model => model.GameMoney)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.Team.Name)
</dt>
<dd>
    @Html.DisplayFor(model => model.Team.Name)
</dd>
</dl>
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.Id }) |
    @Html.ActionLink("Back to List", "Index")
</p>
<hr />
<h2>@Html.DisplayForModel()</h2>
@Html.DisplayForModel()
@*
4.

```

There are 2 categories of built-in templated helpers.

4.1. Display Templates

4.1.1.
 //@Html.DisplayFor(model => model.Name)
 The view must have strongly typed view Model.
 It can work with the complex type Model property.

It is similar to `@Html.DisplayTextFor(model => model.GameHolder)`
`//@Html.DisplayTextFor(model => model.GameHolder)`
model.GameHolder will return a Gamer object.
The Gamer class has `[DisplayColumn("Name")]` attribute,
so it will display Gamer Name property value
which is the full name of that gamer.

4.1.2.

`//@Html.DisplayForModel()`
The view must have strongly typed view Model.
It will display every property in view model
except the properties with `[ScaffoldColumn(false)]` attribute.

4.1.3.

`@Html.Display` helper does not need strongly typed view Mode.
`//ViewData["GamerData"] = gamer;`
`//return View();`
In the controller, we put the gamer object into `ViewData["GamerData"]`
"GamerData" in this case is the key of `ViewData`.
`ViewData["GamerData"]` contains that gamer object data,
so we don't have to use a view model.
`//@Html.Display("GamerData")`
In the view, we use `@Html.Display("GamerData")`
to retrieve the Gamer data from `ViewData["GamerData"]`.
It will display everything
except the properties with `[ScaffoldColumn(false)]` attribute.

4.2.

Display Templates

4.2.1.

`//@Html.EditorFor(model => model.Name)`
The view must have strongly typed view Model.
It will create a textbox for the property value input.

4.2.2.

`//@Html.EditorForModel()`
The view must have strongly typed view Model.
It will create textbox input for every properties in view model
except the properties with `[ScaffoldColumn(false)]` attribute.

4.2.3.

`@Html.Editor` helper does not need strongly typed view Mode.
`//ViewData["GamerData"] = gamer;`
`//return View();`
In the controller, we put the gamer object into `ViewData["GamerData"]`
"GamerData" in this case is the key of `ViewData`.
`ViewData["GamerData"]` contains that gamer object data,
so we don't have to use a view model.
`//@Html.Editor("GamerData")`
In the view, we use `@Html.Editor("GamerData")`
to retrieve the Gamer data from `ViewData["GamerData"]`.
It will create textbox input for every properties in `ViewData["GamerData"]`
except the properties with `[ScaffoldColumn(false)]` attribute.
However, we pressed submit button and call the `[HttpPost]` action
`//public async Task<ActionResult> EditThree(int id, string name, string gender, string city, DateTime? dateOfBirth, string emailAddress, int? score, string profileUrl, int? gameMoney, int? teamId)`
OR
`//public async Task<ActionResult> EditThree(Gamer gamer)`
Both ways can not retrieve the data because it is not strongly typed.
I don't suggest to use `@Html.Editor` helper

*@

DetailsThree

Details (Model is GamerA)

Gamer

Full Name	Name01 ABB
Gender	Male
City	City01ss
DateOfBirth	30/04/1979
EmailAddress	1@AAA.com
Score	3500
ProfileUrl	https://rthandyguytutorial.blogspot.com.au/
GameMoney	\$1,000.00
Name	

[Edit](#) | [Back to List](#)

Html.DisplayForModel()

Full Name
Name01 ABB
Gender
Male
City
City01ss
DateOfBirth
30/04/1979
EmailAddress
1@AAA.com
ProfileUrl
<https://rthandyguytutorial.blogspot.com.au/>
GameMoney
\$1,000.00
TeamId
1

5.12. Views/Gamer/DetailsFour.cshtml (@Html.Display("GamerData"))

Add View

View name:

Template:

Model class:

Data context class:

Options:

☐ Create as a partial view

☒ Reference script libraries

☒ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Add Cancel

```
@{
    ViewBag.Title = "DetailsFour";
}
<h2>Html.Display("GamerData")</h2>
@Html.Display("GamerData")
@*
```

4.

There are 2 categories of built-in templated helpers.

4.1.

Display Templates

4.1.1.

```
//@Html.DisplayFor(model => model.Name)
```

The view must have strongly typed view Model.
It can work with the complex type Model property.
It is similar to @Html.DisplayTextFor(model => model.GameHolder)
//@Html.DisplayTextFor(model => model.GameHolder)
model.GameHolder will return a Gamer object.
The Gamer class has [DisplayColumn("Name")] attribute,
so it will display Gamer Name property value
which is the full name of that gamer.

4.1.2.

```
//@Html.DisplayForModel()
```

The view must have strongly typed view Model.
It will display every property in view model
except the properties with [ScaffoldColumn(false)] attribute.

4.1.3.

@Html.Display helper does not need strongly typed view Mode.
//ViewData["GamerData"] = gamer;
//return View();
In the controller, we put the gamer object into ViewData["GamerData"]
"GamerData" in this case is the key of ViewData.
ViewData["GamerData"] contains that gamer object data,
so we don't have to use a view model.
//@Html.Display("GamerData")
In the view, we use @Html.Display("GamerData")
to retrieve the Gamer data from ViewData["GamerData"].
It will display everything
except the properties with [ScaffoldColumn(false)] attribute.

```

-----
4.2.
Display Templates
-----
4.2.1.
//@Html.EditorFor(model => model.Name)
The view must have strongly typed view Model.
It will create a textbox for the property value input.
-----
4.2.2.
//@Html.EditorForModel()
The view must have strongly typed view Model.
It will create textbox input for every properties in view model
except the properties with [ScaffoldColumn(false)] attribute.
-----
4.2.3.
@Html.Editor helper does not need strongly typed view Mode.
//ViewData["GamerData"] = gamer;
//return View();
In the controller, we put the gamer object into ViewData["GamerData"]
"GamerData" in this case is the key of ViewData.
ViewData["GamerData"] contains that gamer object data,
so we don't have to use a view model.
//@Html.Editor("GamerData")
In the view, we use @Html.Editor("GamerData")
to retrieve the Gamer data from ViewData["GamerData"].
It will create textbox input for every properties in ViewData["GamerData"]
except the properties with [ScaffoldColumn(false)] attribute.
However, we pressed submit button and call the [HttpPost] action
//public async Task<ActionResult> EditThree(int id, string name, string gender, string city, DateTime?
dateOfBirth, string emailAddress, int? score, string profileUrl, int? gameMoney, int? teamId)
OR
//public async Task<ActionResult> EditThree(Gamer gamer)
Both ways can not retrieve the data because it is not strongly typed.
I don't suggest to use @Html.Editor helper
*@

```

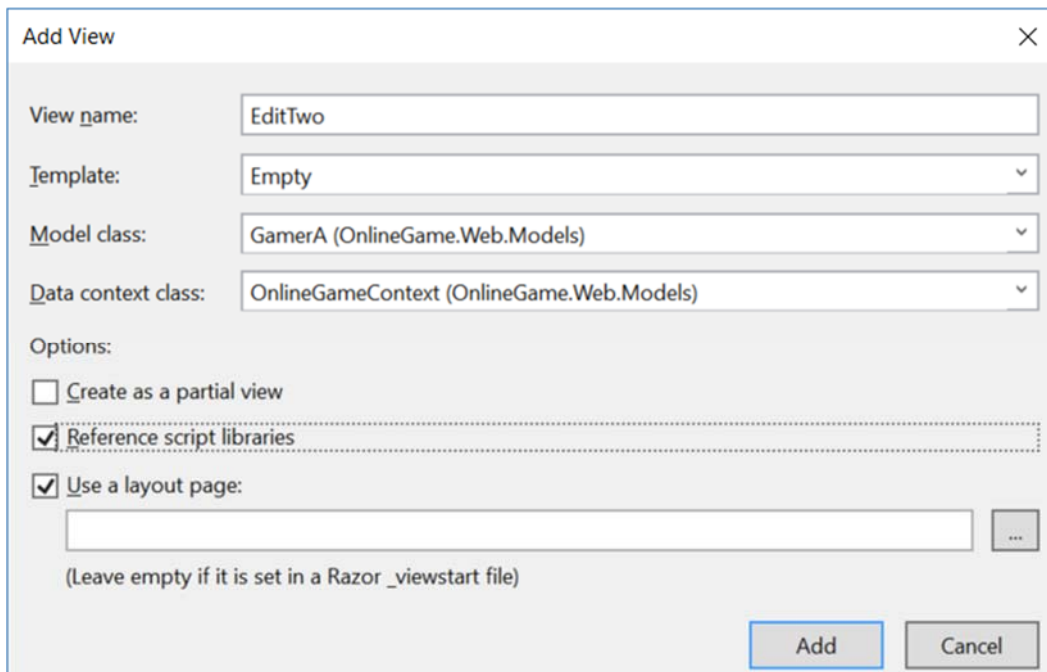
Html.Display("GamerData")

```

Id
1
Full Name
Name01 ABB
Gender
Male
City
City01ss
DateOfBirth
30/04/1979
EmailAddress
1@AAA.com
ProfileUrl
https://ithandyguytutorial.blogspot.com.au/
GameMoney
$1,000.00
TeamId
1

```

5.13. Views/Gamer/EditTwo.cshtml ([HiddenInput(DisplayValue = false)], [ReadOnly(true)])



Add View

View name: EditTwo

Template: Empty

Model class: GamerA (OnlineGame.Web.Models)

Data context class: OnlineGameContext (OnlineGame.Web.Models)

Options:

☐ Create as a partial view

☒ Reference script libraries

☒ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Add Cancel

```
@model OnlineGame.Web.Models.GamerA
@{
    ViewBag.Title = "EditTwo";
}
<h2>EditTwo</h2>
@using (Html.BeginForm("EditTwo", "Gamer"))
{
    @Html.EditorForModel()
    <br />
    <br />
    <input type="submit" value="Save" />
}
@*
```

4.
There are 2 categories of built-in templated helpers.

4.1.
Display Templates

4.1.1.
//@Html.DisplayFor(model => model.Name)
The view must have strongly typed view Model.
It can work with the complex type Model property.
It is similar to @Html.DisplayTextFor(model => model.GameHolder)
//@Html.DisplayTextFor(model => model.GameHolder)
model.GameHolder will return a Gamer object.
The Gamer class has [DisplayColumn("Name")] attribute,
so it will display Gamer Name property value
which is the full name of that gamer.

4.1.2.
//@Html.DisplayForModel()
The view must have strongly typed view Model.
It will display every property in view model
except the properties with [ScaffoldColumn(false)] attribute.

4.1.3.

@Html.Display helper does not need strongly typed view Model.

```
//ViewData["GamerData"] = gamer;
```

```
//return View();
```

In the controller, we put the gamer object into ViewData["GamerData"]

"GamerData" in this case is the key of ViewData.

ViewData["GamerData"] contains that gamer object data,

so we don't have to use a view model.

```
//@Html.Display("GamerData")
```

In the view, we use @Html.Display("GamerData")

to retrieve the Gamer data from ViewData["GamerData"].

It will display everything

except the properties with [ScaffoldColumn(false)] attribute.

4.2.

Display Templates

4.2.1.

```
//@Html.EditorFor(model => model.Name)
```

The view must have strongly typed view Model.

It will create a textbox for the property value input.

4.2.2.

```
//@Html.EditorForModel()
```

The view must have strongly typed view Model.

It will create textbox input for every properties in view model

except the properties with [ScaffoldColumn(false)] attribute.

4.2.3.

@Html.Editor helper does not need strongly typed view Model.

```
//ViewData["GamerData"] = gamer;
```

```
//return View();
```

In the controller, we put the gamer object into ViewData["GamerData"]

"GamerData" in this case is the key of ViewData.

ViewData["GamerData"] contains that gamer object data,

so we don't have to use a view model.

```
//@Html.Editor("GamerData")
```

In the view, we use @Html.Editor("GamerData")

to retrieve the Gamer data from ViewData["GamerData"].

It will create textbox input for every properties in ViewData["GamerData"]

except the properties with [ScaffoldColumn(false)] attribute.

However, we pressed submit button and call the [HttpPost] action

```
//public async Task<ActionResult>
```

```
EditThree(int id, string name, string gender, string city, DateTime? dateOfBirth, string emailAddress,
int? score, string profileUrl, int? gameMoney, int? teamId)
```

OR

```
//public async Task<ActionResult>
```

```
EditThree(Gamer gamer)
```

Both ways can not retrieve the data because it is not strongly typed.

I don't suggest to use @Html.Editor helper

*@

EditTwo

Full Name

Name01 ABB

Gender

Male

City

City01ss

DateOfBirth

30/04/1979 12:00:00 AM OnlineGame.Web.Models.GamerA {0:d}

EmailAddress

1@AAA.com

ProfileUrl

https://ithandyguytutorial.

GameMoney

1000

TeamId

1

Save

5.14. Views/Gamer/EditThree.cshtml

Add View

View name:

EditThree

Template:

Empty (without model)

Model class:

Data context class:

Options:

☐ Create as a partial view

☐ Reference script libraries

☒ Use a layout page:

...

(Leave empty if it is set in a Razor _viewstart file)

Add

Cancel


```

        ViewBag.Title = "EditThree";
    }
    <h2>Html.Editor("GamerData")</h2>
    @using (Html.BeginForm("EditThree", "Gamer"))
    {
        @Html.Editor("GamerData")
        <br />
        <br />
        <input type="submit" value="Save" />
    }
    @*

```

4.
There are 2 categories of built-in templated helpers.

4.1.
Display Templates

4.1.1.
//@Html.DisplayFor(model => model.Name)
The view must have strongly typed view Model.
It can work with the complex type Model property.
It is similar to @Html.DisplayTextFor(model => model.GameHolder)
//@Html.DisplayTextFor(model => model.GameHolder)
model.GameHolder will return a Gamer object.
The Gamer class has [DisplayColumn("Name")] attribute,
so it will display Gamer Name property value
which is the full name of that gamer.

4.1.2.
//@Html.DisplayForModel()
The view must have strongly typed view Model.
It will display every property in view model
except the properties with [ScaffoldColumn(false)] attribute.

4.1.3.
@Html.Display helper does not need strongly typed view Mode.
//ViewData["GamerData"] = gamer;
//return View();
In the controller, we put the gamer object into ViewData["GamerData"]
"GamerData" in this case is the key of ViewData.
ViewData["GamerData"] contains that gamer object data,
so we don't have to use a view model.
//@Html.Display("GamerData")
In the view, we use @Html.Display("GamerData")
to retrieve the Gamer data from ViewData["GamerData"].
It will display everything
except the properties with [ScaffoldColumn(false)] attribute.

4.2.
Display Templates

4.2.1.
//@Html.EditorFor(model => model.Name)
The view must have strongly typed view Model.
It will create a textbox for the property value input.

4.2.2.
//@Html.EditorForModel()
The view must have strongly typed view Model.
It will create textbox input for every properties in view model
except the properties with [ScaffoldColumn(false)] attribute.

4.2.3.
@Html.Editor helper does not need strongly typed view Mode.
//ViewData["GamerData"] = gamer;

```
//return View();
In the controller, we put the gamer object into ViewData["GamerData"]
"GamerData" in this case is the key of ViewData.
ViewData["GamerData"] contains that gamer object data,
so we don't have to use a view model.
//@Html.Editor("GamerData")
In the view, we use @Html.Editor("GamerData")
to retrieve the Gamer data from ViewData["GamerData"].
It will create textbox input for every properties in ViewData["GamerData"]
except the properties with [ScaffoldColumn(false)] attribute.
However, we pressed submit button and call the [HttpPost] action
//public async Task<ActionResult>
EditThree(int id, string name, string gender, string city, DateTime? dateOfBirth, string emailAddress,
int? score, string profileUrl, int? gameMoney, int? teamId)
OR
//public async Task<ActionResult>
EditThree(Gamer gamer)
Both ways can not retrieve the data because it is not strongly typed.
I don't suggest to use @Html.Editor helper
*@
```

Html.Editor("GamerData")

Id
1

Full Name
Name01 ABB

Gender
Male

City
City01ss

DateOfBirth
30/04/1979 System.Data.Entity.DynamicProxies.Gamer_B8DD630DA012D23F069296887D11B7AD47814FD0A6804BB375C5F941D3AA4925 {0:dd/MM/yyyy}

EmailAddress
1@AAA.com

ProfileUrl
https://ithandyguytutorial.

GameMoney
1000

TeamId
1

Save

5.15. Views/Shared/EditorTemplates/DateTime.cshtml

@model DateTime?

```
<link href="~/Content/themes/base/jquery-ui.min.css" rel="stylesheet" />
<link href="~/Content/bootstrap.css" rel="stylesheet" />
<script src="~/Scripts/jquery-1.12.4.min.js"></script>
<script src="~/Scripts/jquery-ui-1.12.1.min.js"></script>
<script src="~/Scripts/bootstrap.min.js"></script>
<script src="~/Scripts/jquery.validate.min.js"></script>
<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>
```

```
@*@Html.TextBox("", (Model.HasValue ? Model.Value.ToString("yyyy/MM/dd") : string.Empty), new { @class = "date" })*)*
```

```
@{
    object formattedModelValue = ViewData.TemplateInfo.FormattedModelValue;
    ViewBag.Title = "Edit";
}
```

```

@Html.TextBox("", (Model.HasValue ?
@ViewData.TemplateInfo.FormattedModelValue : string.Empty), new { @class = "date" })
@ViewData.ModelMetadata.ContainerType.ToString()
@*OnlineGame.Web.Models.Gamer*@
@ViewData.ModelMetadata.DisplayFormatString
@*{0:dd/MM/yyyy}*@
<script type="text/javascript">
    $(function () {
        $("input:text.date").datepicker(
            {
                //dateFormat: "yy/mm/dd"
                dateFormat: "dd/mm/yy"
            });
    });
</script>
@*

```

10. EditorTemplates and DisplayTemplates by MVC convention

10.1.

DisplayTemplates

10.1.1.

Views\Shared\DisplayTemplates\UrlToNewWindow.cshtml

Views\Gamer\DisplayTemplates\UrlToNewWindow.cshtml

UrlToNewWindow.cshtml is the DisplayTemplate which must under "DisplayTemplates" folder.

Views\Shared\DisplayTemplates\UrlToNewWindow.cshtml means

the template is available for all the views.

Views\Gamer\DisplayTemplates\UrlToNewWindow.cshtml means

the template is available for only the views of Gamer controller.

10.1.2.

Using DisplayTemplates

10.1.2.1.

In the Models/Gamer/GamerMetaData.cs

```
//[DataType(DataType.Url)]
```

```
//[UIHint("UrlToNewWindow")]
```

```
//public string ProfileUrl { get; set; }
```

[DataType(DataType.Url)] attribute will display a hyperlink.

[UIHint("UrlToNewWindow")] attribute specify the name of view DisplayTemplate to display the property data.

In this case, it will look for "DisplayTemplates/UrlToNewWindow.cshtml"

under "Shared" folder or "Gamer" folder.

Use that view template to display the data of this property.

10.1.2.2.

```
//<a href="@ViewData.Model" target="_blank">@ViewData.Model</a>
```

In the Shared/DisplayTemplates/UrlToNewWindow.cshtml,

@ViewData.Model will take the Model data from the parent view.

In this case, it will return a profile url.

10.2.

EditorTemplates

10.2.1.

Views\Shared\EditorTemplates\DateTime.cshtml

Views\Gamer\EditorTemplates\DateTime.cshtml

DateTime.cshtml is the EditorTemplate which must under "EditorTemplates" folder.

Views\Shared\EditorTemplates\DateTime.cshtml means

the template is available for all the views.

Views\Gamer\EditorTemplates\DateTime.cshtml means

the template is available for only the views of Gamer controller.

10.2.2.

Using EditorTemplates

The EditorTemplate Name must match View Model property Type Name.

E.g. DateTime.ascx or DateTime.cshtml

10.2.2.1.

In the Models/Gamer/GamerMetaData.cs

```
////[DataType(DataType.Date)] //Views/Shared/EditorTemplates/DateTime.cshtml will not Work.
```

```
////[DisplayFormat(DataFormatString = "{0:dd/MM/yyyy hh:mm:ss tt}")]
```

```
//[DisplayFormat(DataFormatString = "{0:d}")]
```

```
//public Nullable<System.DateTime> DateOfBirth { get; set; }
```

The type is DateTime, so it will look for the EditorTemplate from

Views\Shared\EditorTemplates\DateTime.cshtml or

Views\Gamer\EditorTemplates\DateTime.cshtml

In this case, Views\Shared\EditorTemplates\DateTime.cshtml will be the EditorTemplate.

The View Model Property in Edit mode will use the EditorTemplate to display.

In this case,

```
//@model DateTime?
```

```
//@Html.TextBox("", (Model.HasValue ? Model.Value.ToString("yyyy/MM/dd") : string.Empty), new { @class = "date" })
```

So it will add the class="date" to the textbox input.

10.2.2.2.

In the Edit.cshtml

```
//<link href="~/Content/themes/base/jquery-ui.min.css" rel="stylesheet" />
```

```
//<link href="~/Content/bootstrap.css" rel="stylesheet" />
```

```
//<script src="~/Scripts/jquery-1.12.4.min.js"></script>
```

```
//<script src="~/Scripts/jquery-ui-1.12.1.min.js"></script>
```

```
//<script src="~/Scripts/bootstrap.min.js"></script>
```

```
...
//<script type="text/javascript">
```

```
//    $(function () {
```

```
//        $("input:text.date").datepicker(
```

```
//            {
```

```
//                dateFormat: "yy/mm/dd"
```

```
//            });
```

```
//    });
```

```
//</script>
```

```
*@
```

5.16. Views/Gamer/Edit.cshtml

```
@model OnlineGame.Web.Models.Gamer
```

```
@{
```

```
    ViewBag.Title = "Edit";
```

```
}
```

```
<link href="~/Content/themes/base/jquery-ui.min.css" rel="stylesheet" />
```

```
<link href="~/Content/bootstrap.css" rel="stylesheet" />
```

```
<script src="~/Scripts/jquery-1.12.4.min.js"></script>
```

```
<script src="~/Scripts/jquery-ui-1.12.1.min.js"></script>
```

```
<script src="~/Scripts/bootstrap.min.js"></script>
```

```
<script src="~/Scripts/jquery.validate.min.js"></script>
```

```
<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>
```

```
<h2>Edit</h2>
```

```
@using (Html.BeginForm())
```

```
{
```

```
    @Html.AntiForgeryToken()
```

```
    <div class="form-horizontal">
```

```
        <h4>Gamer</h4>
```

```
        <hr />
```

```
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
```

```
        @Html.HiddenFor(model => model.Id)
```

```
        <div class="form-group">
```

```
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label col-md-2" })
```

```
            <div class="col-md-10">
```

```
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-control" } })
```

```
                @Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })
```

```
            </div>
```

```
        </div>
```

```

<div class="form-group">
    @Html.LabelFor(model => model.Gender, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Gender, new { htmlAttributes = new { @class = "form-
control" } })
        @Html.ValidationMessageFor(model => model.Gender, "", new { @class = "text-danger" })
    </div>
</div>
<div class="form-group">
    @Html.LabelFor(model => model.City, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.City, new { htmlAttributes = new { @class = "form-
control" } })
        @Html.ValidationMessageFor(model => model.City, "", new { @class = "text-danger" })
    </div>
</div>
<div class="form-group">
    @Html.LabelFor(model => model.DateOfBirth, htmlAttributes: new { @class = "control-label col-
md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.DateOfBirth, new { htmlAttributes = new { @class = "form-
control" } })
        @Html.ValidationMessageFor(model => model.DateOfBirth, "", new { @class = "text-danger" })
    </div>
</div>
<div class="form-group">
    @Html.LabelFor(model => model.EmailAddress, htmlAttributes: new { @class = "control-label col-
md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.EmailAddress, new { htmlAttributes = new { @class = "form-
control" } })
        @Html.ValidationMessageFor(model => model.EmailAddress, "", new { @class = "text-danger" })
    </div>
</div>
<div class="form-group">
    @Html.LabelFor(model => model.Score, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Score, new { htmlAttributes = new { @class = "form-
control" } })
        @Html.ValidationMessageFor(model => model.Score, "", new { @class = "text-danger" })
    </div>
</div>
<div class="form-group">
    @Html.LabelFor(model => model.ProfileUrl, htmlAttributes: new { @class = "control-label col-md-
2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.ProfileUrl, new { htmlAttributes = new { @class = "form-
control" } })
        @Html.ValidationMessageFor(model => model.ProfileUrl, "", new { @class = "text-danger" })
    </div>
</div>
<div class="form-group">

```

```

    @Html.LabelFor(model => model.GameMoney, htmlAttributes: new { @class = "control-label col-md-2" })

    <div class="col-md-10">
        @Html.EditorFor(model => model.GameMoney, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.GameMoney, "", new { @class = "text-danger" })
    </div>
</div>
<div class="form-group">
    @Html.LabelFor(model => model.TeamId, "TeamId", htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.DropDownList("TeamId", null, htmlAttributes: new { @class = "form-control" })
        @Html.ValidationMessageFor(model => model.TeamId, "", new { @class = "text-danger" })
    </div>
</div>
<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <input type="submit" value="Save" class="btn btn-default" />
    </div>
</div>
</div>
}
<div>
    @Html.ActionLink("Back to List", "Index")
</div>
@*
1.
1.1.
//@Html.HiddenFor(model => model.Name, new { htmlAttributes = new { @class = "form-control" } })
It will create the following.
//<input data-val="true" data-val-required="The Name field is required." htmlattributes="{ class = form-control }" id="Name" name="Name" type="hidden" value="Name01 ABB">
1.2.
//@Html.DisplayFor(model => model.Name, new { htmlAttributes = new { @class = "form-control" } })
It will create the following.
//Name01 ABB
1.3.
//@Html.EditorFor(model => model.Name, new {htmlAttributes = new {@class = "form-control"}})
It will create the following.
//<input class="form-control text-box single-line valid" id="Name" name="Name" type="text" value="Name01 ABB">
-----
4.
There are 2 categories of built-in templated helpers.
-----
4.1.
Display Templates
-----
4.1.1.
//@Html.DisplayFor(model => model.Name)
The view must have strongly typed view Model.
It can work with the complex type Model property.
It is similar to @Html.DisplayTextFor(model => model.GameHolder)
//@Html.DisplayTextFor(model => model.GameHolder)
model.GameHolder will return a Gamer object.
The Gamer class has [DisplayColumn("Name")] attribute,
so it will display Gamer Name property value
which is the full name of that gamer.
-----
4.1.2.

```

```
//@Html.DisplayForModel()
The view must have strongly typed view Model.
It will display every property in view model
except the properties with [ScaffoldColumn(false)] attribute.
-----
```

4.1.3.

```
@Html.Display helper does not need strongly typed view Mode.
//ViewData["GamerData"] = gamer;
//return View();
In the controller, we put the gamer object into ViewData["GamerData"]
"GamerData" in this case is the key of ViewData.
ViewData["GamerData"] contains that gamer object data,
so we don't have to use a view model.
//@Html.Display("GamerData")
In the view, we use @Html.Display("GamerData")
to retrieve the Gamer data from ViewData["GamerData"].
It will display everything
except the properties with [ScaffoldColumn(false)] attribute.
-----
```

4.2.

Display Templates

4.2.1.

```
//@Html.EditorFor(model => model.Name)
The view must have strongly typed view Model.
It will create a textbox for the property value input.
-----
```

4.2.2.

```
//@Html.EditorForModel()
The view must have strongly typed view Model.
It will create textbox input for every properties in view model
except the properties with [ScaffoldColumn(false)] attribute.
-----
```

4.2.3.

```
@Html.Editor helper does not need strongly typed view Mode.
//ViewData["GamerData"] = gamer;
//return View();
In the controller, we put the gamer object into ViewData["GamerData"]
"GamerData" in this case is the key of ViewData.
ViewData["GamerData"] contains that gamer object data,
so we don't have to use a view model.
//@Html.Editor("GamerData")
In the view, we use @Html.Editor("GamerData")
to retrieve the Gamer data from ViewData["GamerData"].
It will create textbox input for every properties in ViewData["GamerData"]
except the properties with [ScaffoldColumn(false)] attribute.
However, we pressed submit button and call the [HttpPost] action
//public async Task<ActionResult>
EditThree(int id, string name, string gender, string city, DateTime? dateOfBirth, string emailAddress,
int? score, string profileUrl, int? gameMoney, int? teamId)
OR
//public async Task<ActionResult>
EditThree(Gamer gamer)
Both ways can not retrieve the data because it is not strongly typed.
I don't suggest to use @Html.Editor helper
```

*@

Edit

Gamer

Full Name	<input type="text" value="Name01 ABB"/>
Gender	<input type="text" value="Male"/>
City	<input type="text" value="City01ss"/>
DateOfBirth	<input type="text" value="30/04/1979"/> x OnlineGame.Web.Models.Gamer {0:dd/MM/yyyy}
EmailAddress	<input type="text"/>
Score	<input type="text"/>
ProfileUrl	<input type="text" value="m.at"/>
GameMoney	<input type="text"/>
TeamId	<input type="text" value="Team1"/> <input type="button" value="v"/>
<input type="button" value="Save"/>	

[Back to List](#)

5.17. Add Jquery UI

Install Jquery UI from NuGet Package

[Browse](#)

Installed

Updates **7**

NuGet

x



Include prerelease



jQuery.UI.Combined by jQuery UI Team, **15.7M** downloads

v1.12.1



The full jQuery UI library as a single combined file. Includes the base theme.

5.18. Views/Gamer/Create.cshtml

```
@model OnlineGame.Web.Models.Gamer
```

```
@{
```

```
    ViewBag.Title = "Create";
```

```
}
```

```
<h2>Create</h2>
```

```
@using (Html.BeginForm())
```



```

{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Gamer</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.Gender, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Gender, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.Gender, "", new { @class = "text-danger" })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.City, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.City, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.City, "", new { @class = "text-danger" })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.DateOfBirth, htmlAttributes: new { @class = "control-label col-
md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.DateOfBirth, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.DateOfBirth, "", new { @class = "text-danger" })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.EmailAddress, htmlAttributes: new { @class = "control-label col-
md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.EmailAddress, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.EmailAddress, "", new { @class = "text-danger" })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.Score, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">

```

```

        @Html.EditorFor(model => model.Score, new { htmlAttributes = new { @class = "form-
control" } })
        @Html.ValidationMessageFor(model => model.Score, "", new { @class = "text-danger" })
    </div>
</div>
<div class="form-group">
    @Html.LabelFor(model => model.ProfileUrl, htmlAttributes: new { @class = "control-label col-md-
2" })

    <div class="col-md-10">
        @Html.EditorFor(model => model.ProfileUrl, new { htmlAttributes = new { @class = "form-
control" } })
        @Html.ValidationMessageFor(model => model.ProfileUrl, "", new { @class = "text-danger" })
    </div>
</div>
<div class="form-group">
    @Html.LabelFor(model => model.GameMoney, htmlAttributes: new { @class = "control-label col-md-
2" })

    <div class="col-md-10">
        @Html.EditorFor(model => model.GameMoney, new { htmlAttributes = new { @class = "form-
control" } })
        @Html.ValidationMessageFor(model => model.GameMoney, "", new { @class = "text-danger" })
    </div>
</div>
<div class="form-group">
    @Html.LabelFor(model => model.TeamId, "TeamId", htmlAttributes: new { @class = "control-label
col-md-2" })

    <div class="col-md-10">
        @Html.DropDownList("TeamId", null, htmlAttributes: new { @class = "form-control" })
        @Html.ValidationMessageFor(model => model.TeamId, "", new { @class = "text-danger" })
    </div>
</div>
<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <input type="submit" value="Create" class="btn btn-default" />
    </div>
</div>
</div>
}
<div>
    @Html.ActionLink("Back to List", "Index")
</div>
<script src="~/Scripts/jquery-1.10.2.min.js"></script>
<script src="~/Scripts/jquery.validate.min.js"></script>
<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>

```

Create

Gamer

Full Name	<input type="text"/>
Gender	<input type="text"/>
City	<input type="text"/>
DateOfBirth	<input type="text"/> OnlineGame.Web.Models.Gamer {0:dd/MM/yyyy}
EmailAddress	<input type="text"/>
Score	<input type="text"/>
ProfileUrl	<input type="text"/>
GameMoney	<input type="text"/>
TeamId	<input type="text" value="Team1"/> ▼
<input type="button" value="Create"/>	

[Back to List](#)

5.19. Views/Gamer/Delete.cshtml

```
@model OnlineGame.Web.Models.Gamer
@{
    ViewBag.Title = "Delete";
}
<h2>Delete</h2>
<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Gamer</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Name)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Name)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.Gender)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Gender)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.City)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.City)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.DateOfBirth)
        </dt>
```

```

<dd>
    @Html.DisplayFor(model => model.DateOfBirth)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.EmailAddress)
</dt>
<dd>
    @Html.DisplayFor(model => model.EmailAddress)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.Score)
</dt>
<dd>
    @Html.DisplayFor(model => model.Score)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.ProfileUrl)
</dt>
<dd>
    @Html.DisplayFor(model => model.ProfileUrl)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.GameMoney)
</dt>
<dd>
    @Html.DisplayFor(model => model.GameMoney)
</dd>
<dt>
    @Html.DisplayNameFor(model => model.Team.Name)
</dt>
<dd>
    @Html.DisplayFor(model => model.Team.Name)
</dd>
</dl>
@using (Html.BeginForm()) {
    @Html.AntiForgeryToken()
    <div class="form-actions no-color">
        <input type="submit" value="Delete" class="btn btn-default" /> |
        @Html.ActionLink("Back to List", "Index")
    </div>
}
</div>

```

Delete

Are you sure you want to delete this?

Gamer

Full Name	Name02 CDDE
Gender	Female
City	City03
DateOfBirth	30/09/1981
EmailAddress	2@BBB.com
Score	3500
ProfileUrl	https://ithandyguytutorial.blogspot.com.au/
GameMoney	\$1,500.00
Name	Team2

Delete | [Back to List](#)