(T5)討論MvcConventions(命名規則)。討論AdoNet、BusinessLayer的Update  
CourseGUID: 8503b39c-5887-4634-8291-facfb3117924  
=======================================================================  
(T5)討論MvcConventions(命名規則)。討論AdoNet、BusinessLayer的Update  
=======================================================================  
0. Summary

-----------

1. OnlineGame DB

1.1. TSQL

1.2. Security login

-----------

2. BusinessLayer

2.1. BusinessLayer/IGamer.cs

2.2. BusinessLayer/Gamer.cs

2.3. BusinessLayer/GamerBusinessLayer.cs

-----------

3. OnlineGame.Web

3.1. OnlineGame.Web/Controllers/GamerController.cs

3.2. OnlineGame.Web/Views/Gamer/Edit.cshtml  
=======================================================================

0. Summary

In this tutorial, we will discuss

\* MvcConventions

\* AdoDotNet

\* BusinessLayer

\* UpdateData

\* UnintendedUpdate

This is continuous with the previous tutorial.

Please ensure you finish the previous tutorial before you continue this tutorial.

1. OnlineGame DB

1.1. TSQL

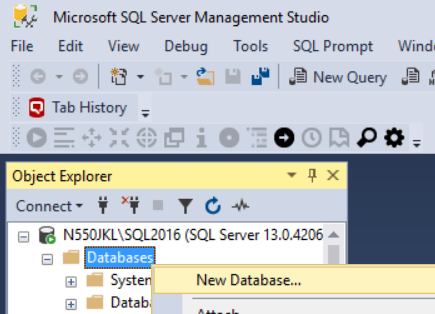
In SQL server Management Studio (SSMS)

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

In General Tab -->

Name: **OnlineGame**

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



Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

--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

--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 ,

      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' );

INSERT  Gamer

VALUES  ( N'Name01 ABB', N'Male', N'City01', '1979/4/28', 1 );

INSERT  Gamer

VALUES  ( N'Name02 CDDE', N'Female', N'City03', '1981/7/24', 2 );

INSERT  Gamer

VALUES  ( N'Name03 FIJK', N'Female', N'City01', '1984/12/5', 3 );

INSERT  Gamer

VALUES  ( N'Name04 LMOPPQ', N'Male', N'City02', '1983/5/29', 1 );

INSERT  Gamer

VALUES  ( N'Name05 QRSTT', N'Male', N'City01', '1979/6/20', 3 );

INSERT  Gamer

VALUES  ( N'Name06 TUVVX', N'Female', N'City03', '1984/5/15', 3 );

INSERT  Gamer

VALUES  ( N'Name07 XYZZXX', N'Female', N'City01', '1986/4/29', 2 );

INSERT  Gamer

VALUES  ( N'Name08 ABBCDE', N'Male', N'City02', '1985/7/28', 1 );

INSERT  Gamer

VALUES  ( N'Name09 QRSTTUVXX', N'Male', N'City02', '1983/4/16', 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 ,

      @TeamId INT

    )

AS

    BEGIN

        INSERT  INTO Gamer

        VALUES  ( @Name, @Gender, @City, @DateOfBirth, @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 ,

      @TeamId INT

    )

AS

    BEGIN

        UPDATE  dbo.Gamer

        SET     Name = @Name ,

                Gender = @Gender ,

                City = @City ,

                DateOfBirth = @DateOfBirth ,

                TeamId = @TeamId

        WHERE   Id = @Id;

    END;

GO -- Run the previous command and begins new batch

--EXEC spGetGamers

--GO -- Run the previous command and begins new batch

1.2. Security login

In SQL server

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

-->

General Tab

Login Name :

**Tester**

Password:

**1234**

Default Database:

**OnlineGame**

-->

Server Roles Tab

Select

**sysadmin**

-->

User Mapping Tab

Select **OnlineGame**

Select every single role.





Graphical user interface, text, application

Description automatically generated



2. BusinessLayer

2.1. BusinessLayer/IGamer.cs

using System;

namespace BusinessLayer

{

    public interface IGamer

    {

        int Id { get; set; }

        string Gender { get; set; }

        string City { get; set; }

        DateTime? DateOfBirth { get; set; }

        int TeamId { get; set; }

    }

}

2.2. BusinessLayer/Gamer.cs

using System;

using[System.ComponentModel.DataAnnotations](http://system.componentmodel.dataannotations/);

namespace BusinessLayer

{

    public class Gamer : IGamer

    {

        public int Id { get; set; }

        //[Required]

        public string Name { get; set; }

        [Required]

        public string Gender { get; set; }

        [Required]

        public string City { get; set; }

        [Required]

        public DateTime? DateOfBirth { get; set; }

        [Required]

        public int TeamId { get; set; }

    }

}

2.3. BusinessLayer/GamerBusinessLayer.cs

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

namespace BusinessLayer

{

    public class GamerBusinessLayer

    {

        public IEnumerable<Gamer> Gamers

        {

            get

            {

                string connectionString =

                    ConfigurationManager.ConnectionStrings["OnlineGameContext"].ConnectionString;

                List<Gamer> gamers = new List<Gamer>();

                using (SqlConnection con = new SqlConnection(connectionString))

                {

                    SqlCommand cmd = new SqlCommand("spGetGamers", con);

[cmd.CommandType](http://cmd.commandtype/) = CommandType.StoredProcedure;

                    con.Open();

                    SqlDataReader rdr = cmd.ExecuteReader();

                    while (rdr.Read())

                    {

                        Gamer gamer = new Gamer();

                        gamer.Id = Convert.ToInt32(rdr["Id"]);

                        gamer.Name = rdr["Name"].ToString();

                        gamer.Gender = rdr["Gender"].ToString();

                        gamer.City = rdr["City"].ToString();

                        gamer.DateOfBirth = Convert.ToDateTime(rdr["DateOfBirth"]);

                        gamer.TeamId = Convert.ToInt32(rdr["TeamId"]);

                        gamers.Add(gamer);

                    }

                }

                return gamers;

            }

        }

        public void AddGamer(Gamer gamer)

        {

            string connectionString =

            ConfigurationManager.ConnectionStrings["OnlineGameContext"].ConnectionString;

            using (SqlConnection con = new SqlConnection(connectionString))

            {

                SqlCommand cmd = new SqlCommand("spAddGamer", con)

                {

                    CommandType = CommandType.StoredProcedure

                };

                SqlParameter sqlParamName = new SqlParameter

                {

                    ParameterName = "@Name",

                    Value = gamer.Name

                };

                cmd.Parameters.Add(sqlParamName);

                SqlParameter sqlParamGender = new SqlParameter

                {

                    ParameterName = "@Gender",

                    Value = gamer.Gender

                };

                cmd.Parameters.Add(sqlParamGender);

                SqlParameter sqlParamCity = new SqlParameter

                {

                    ParameterName = "@City",

                    Value = gamer.City

                };

                cmd.Parameters.Add(sqlParamCity);

                SqlParameter sqlParamDateOfBirth = new SqlParameter

                {

                    ParameterName = "@DateOfBirth",

                    Value = gamer.DateOfBirth

                };

                cmd.Parameters.Add(sqlParamDateOfBirth);

                SqlParameter sqlParamTeamId = new SqlParameter

                {

                    ParameterName = "@TeamId",

                    Value = gamer.TeamId

                };

                cmd.Parameters.Add(sqlParamTeamId);

                con.Open();

                cmd.ExecuteNonQuery();

            }

        }

       public void SaveGamer(Gamer gamer)

        {

            string connectionString =

                    ConfigurationManager.ConnectionStrings["OnlineGameContext"].ConnectionString;

            using (SqlConnection con = new SqlConnection(connectionString))

            {

                SqlCommand cmd = new SqlCommand("spSaveGamer", con)

                {

                    CommandType = CommandType.StoredProcedure

                };

                SqlParameter sqlParamId = new SqlParameter

                {

                    ParameterName = "@Id",

                    Value = gamer.Id

                };

                cmd.Parameters.Add(sqlParamId);

                SqlParameter sqlParamName = new SqlParameter

                {

                    ParameterName = "@Name",

                    Value = gamer.Name

                };

                cmd.Parameters.Add(sqlParamName);

                SqlParameter sqlParamGender = new SqlParameter

                {

                    ParameterName = "@Gender",

                    Value = gamer.Gender

                };

                cmd.Parameters.Add(sqlParamGender);

                SqlParameter sqlParamCity = new SqlParameter

                {

                    ParameterName = "@City",

                    Value = gamer.City

                };

                cmd.Parameters.Add(sqlParamCity);

                SqlParameter sqlParamDateOfBirth = new SqlParameter

                {

                    ParameterName = "@DateOfBirth",

                    Value = gamer.DateOfBirth

                };

                cmd.Parameters.Add(sqlParamDateOfBirth);

                SqlParameter sqlParamTeamId = new SqlParameter

                {

                    ParameterName = "@TeamId",

                    Value = gamer.TeamId

                };

                cmd.Parameters.Add(sqlParamTeamId);

                con.Open();

                cmd.ExecuteNonQuery();

            }

        }

    }

}

3. OnlineGame.Web

3.1. OnlineGame.Web/Controllers/GamerController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web.Mvc;

using BusinessLayer;

using OnlineGame.Web.Data;

using Gamer = OnlineGame.Web.Models.Gamer;

namespace OnlineGame.Web.Controllers

{

    public class GamerController : Controller

    {

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

        //public ActionResult Details()

        //{

        //    var gamer = new Gamer

        //    {

        //        Id = 1,

        //        Name = "Name1",

        //        Gender = "Male",

        //        City = "City1"

        //    };

        //    return View(gamer);

        //}

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

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

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

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

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

        public ActionResult Details(int id = 0)

        {

            var onlineGameContext = new OnlineGameContext();

            Gamer gamer;

            if (id == 0)

            {

                gamer = new Gamer

                {

                    Id = 0,

                    Name = "Name0",

                    Gender = "NULL",

                    City = "NULL"

                };

                // or you may throw exception here.

            }

            else

            {

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

                //Throws exception if can not find the single entity

            }

            return View(gamer);

        }

        //Entity Framework

        public ActionResult Index(int teamId)

        {

            //Entity Framework

            OnlineGameContext context = new OnlineGameContext();

            List<Gamer> gamers = context.Gamers.Where(gamer => gamer.TeamId == teamId).ToList();

            return View(gamers);

        }

        //Ado.Net

        public ActionResult Index2()

        {

            //[Ado.Net](http://ado.net/)

            GamerBusinessLayer gamerBusinessLayer = new GamerBusinessLayer();

            List<BusinessLayer.Gamer> gamers = gamerBusinessLayer.Gamers.ToList();

            return View(gamers);

        }

        //Ado.Net

        //[HttpGet] attribute means it only respond to the "GET" request.

        [HttpGet]

        public ActionResult Create()

        {

            return View();

        }

        //Ado.Net

        // 1. Retrieve form data using FormCollection

        [HttpPost]

        public ActionResult Create(FormCollection formCollection)

        {

            ////FormCollection implement C# indexer.

            ////See each key and value of formCollection.

            //foreach (string key in formCollection.AllKeys)

            //{

            //    Response.Write($"key=={key}, {formCollection[key]}, <br/>");

            //}

            int teamId;

            BusinessLayer.Gamer gamer = new BusinessLayer.Gamer

            {

                Name = formCollection["Name"],

                Gender = formCollection["Gender"],

                City = formCollection["City"],

                DateOfBirth = Convert.ToDateTime(formCollection["DateOfBirth"]),

                TeamId = int.TryParse(formCollection["TeamId"], out teamId) ? teamId : 0

            };

            GamerBusinessLayer gamerBusinessLayer =

                new GamerBusinessLayer();

            gamerBusinessLayer.AddGamer(gamer);

            return RedirectToAction("Index2");

        }

        //[Ado.Net](http://ado.net/)

        // 2. Retrieve form data using name attribute of input tag from cshtml

        [HttpPost]

        public ActionResult Create2(string name, string gender, string city, DateTime dateOfBirth, int teamId)

        {

            BusinessLayer.Gamer gamer = new BusinessLayer.Gamer

            {

                Name = name,

                Gender = gender,

                City = city,

                DateOfBirth = dateOfBirth,

                TeamId = teamId

            };

            GamerBusinessLayer gamerBusinessLayer =

                new GamerBusinessLayer();

            gamerBusinessLayer.AddGamer(gamer);

            return RedirectToAction("Index2");

        }

       //Ado.Net

        // 3. Retrieve form data using model binding

        [HttpPost]

        public ActionResult Create3(BusinessLayer.Gamer gamer)

        {

            //if any of input is not valid.

            if (!ModelState.IsValid)

            {

                return View("Create");

                //Go to Create.cshtml,

                //so users can correct their input value.

            }

            GamerBusinessLayer gamerBusinessLayer =

                new GamerBusinessLayer();

            gamerBusinessLayer.AddGamer(gamer);

            return RedirectToAction("Index2");

        }

        //Ado.Net

        // 4. Retrieve form data using model binding by UpdateModel() or TryUpdateModel()

        [HttpPost]

        [ActionName("Create4")]

        public ActionResult Create\_Post()

        {

            //if any of input is not valid.

            if (!ModelState.IsValid)

            {

                return View("Create");

                //Go to Create.cshtml,

                //so users can correct their input value.

            }

            GamerBusinessLayer gamerBusinessLayer =

                new GamerBusinessLayer();

            BusinessLayer.Gamer gamer = new BusinessLayer.Gamer();

            //UpdateModel<BusinessLayer.Gamer>(gamer);

            //UpdateModel(gamer);

            TryUpdateModel(gamer);

            //1.

            // UpdateModel() and TryUpdateModel() inspects all the HttpRequest inputs

            // such as posted Form data, QueryString,

            // Cookies and Server variables and populate the gamer object.

            gamerBusinessLayer.AddGamer(gamer);

            return RedirectToAction("Index2");

        }

        //Ado.Net

        //[HttpGet] attribute means it only respond to the "GET" request.

        [HttpGet]

        public ActionResult Edit(int id)

        {

            GamerBusinessLayer gamerBusinessLayer = new GamerBusinessLayer();

            BusinessLayer.Gamer gamer = gamerBusinessLayer.Gamers.Single(g => g.Id == id);

            return View(gamer);

        }

       //Ado.Net

        //1.

        //Edit by Model binding will open the back door for unintended update.

        [HttpPost]

        public ActionResult Edit(BusinessLayer.Gamer gamer)

        {

            if (!ModelState.IsValid)

            {

                return View(gamer);

            }

            GamerBusinessLayer gamerBusinessLayer =

                new GamerBusinessLayer();

            gamerBusinessLayer.SaveGamer(gamer);

            return RedirectToAction("Index2");

        }

        //Ado.Net

        //2.

        //Solved the unintended update.

        //Edit by UpdateModel() and TryUpdateModel()

        [HttpPost]

        [ActionName("Edit2")]

        public ActionResult Edit\_Post(int id)

        {

            GamerBusinessLayer gamerBusinessLayer =

                new GamerBusinessLayer();

            BusinessLayer.Gamer gamer = gamerBusinessLayer.Gamers.Single(g => g.Id == id);

            //1.

            ////UpdateModel(gamer, new[] { "Id", "Gender", "City", "DateOfBirth", "TeamId" });

            //The second parameter of UpdateModel() and TeyUpdateModel() is included properties.

            //In this case, it will only update the following properties into the model.

            //"Id", "Gender", "City", "DateOfBirth", "TeamId"

            //The Name property is not included so it will not be updated.

            //2.

            ////UpdateModel(gamer, null, null, new[] { "Name" });

            //update all properties except Name property

            UpdateModel(gamer, null, null, new[] { "Name" });

            if (!ModelState.IsValid)

            {

                return View("Edit", gamer);

            }

            gamerBusinessLayer.SaveGamer(gamer);

            return RedirectToAction("Index2");

        }

        //Ado.Net

        //3.

        //Solved the unintended update.

        //Edit by Model binding with Bind include attribute

        [HttpPost]

        public ActionResult Edit3([Bind(Include = "Id, Gender, City, DateOfBirth, TeamId")]BusinessLayer.Gamer gamer)

        {

            GamerBusinessLayer gamerBusinessLayer =

                new GamerBusinessLayer();

            gamer.Name = gamerBusinessLayer.Gamers.Single(g => g.Id == gamer.Id).Name;

            if (!ModelState.IsValid)

            {

                return View("Edit", gamer);

            }

            gamerBusinessLayer.SaveGamer(gamer);

            return RedirectToAction("Index2");

        }

        //Ado.Net

        //4.

        //Solved the unintended update.

        //Edit by Model binding with Bind exclude attribute

        [HttpPost]

        public ActionResult Edit4([Bind(Exclude = "Name")]BusinessLayer.Gamer gamer)

        {

            GamerBusinessLayer gamerBusinessLayer =

                new GamerBusinessLayer();

            gamer.Name = gamerBusinessLayer.Gamers.Single(g => g.Id == gamer.Id).Name;

            if (!ModelState.IsValid)

            {

                return View("Edit", gamer);

            }

            gamerBusinessLayer.SaveGamer(gamer);

            return RedirectToAction("Index2");

        }

       //Ado.Net

        //5.

        //Solved the unintended update.

        //Edit by UpdateModel() and TryUpdateModel() with Interface

        [HttpPost]

        public ActionResult Edit5(int id)

        {

            GamerBusinessLayer gamerBusinessLayer =

                new GamerBusinessLayer();

            BusinessLayer.Gamer gamer = gamerBusinessLayer.Gamers.Single(g => g.Id == id);

            //1.

            ////TryUpdateModel(gamer, new[] { "Id", "Gender", "City", "DateOfBirth", "TeamId" });

            ////UpdateModel(gamer, new[] { "Id", "Gender", "City", "DateOfBirth", "TeamId" });

            //The second parameter of UpdateModel() and TeyUpdateModel() is included properties.

            //In this case, it will only update the following properties into model.

            //"Id", "Gender", "City", "DateOfBirth", "TeamId"

            //The Name property is not included so it will not be updated.

            //2.

            ////TryUpdateModel(gamer, null, null, new[] { "Name" });

            ////UpdateModel(gamer, null, null, new[] { "Name" });

            //update all properties except Name property

            //3.

            ////TryUpdateModel<IGamer>(gamer);

            ////UpdateModel<IGamer>(gamer);

            //The UpdateModel() function will update only the properties

            //that are present in the interface.

            UpdateModel<IGamer>(gamer);

            if (!ModelState.IsValid)

            {

                return View("Edit", gamer);

            }

            gamerBusinessLayer.SaveGamer(gamer);

            return RedirectToAction("Index2");

        }

    }

}

/\*

1.

//var onlineGameContext = new OnlineGameContext();

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

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

and sotre its data into a temp place called DBSet.

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

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

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

Then store gamer entity data into Gamer Model object.

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

Then we pass the Gamer Model object as the ViewModel,

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

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

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

2.

//[HttpGet]

//public ActionResult Create()

The GET request will direct to Views/Gamer/Create.cshtml.

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

3.

//[HttpPost]

//Create

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

3.1.

//[HttpPost]

//public ActionResult Create(FormCollection formCollection)

Retrieve form data using FormCollection.

The key is the name attribute of input or select tag from cshtml.

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

3.2.

//[HttpPost]

//public ActionResult Create2(string name, string gender, string city, DateTime dateOfBirth, int teamId)

Retrieve form data using name attribute of input tag from cshtml.

3.2.1.

string name is from

//<input class="form-control text-box single-line" id="Name" name="Name" type="text" value="">

3.2.2.

string gender is from

//<select id="Gender" name="Gender">...</select>

3.2.3.

string city is from

<input class="form-control text-box single-line" id="City" name="City" type="text" value="">

3.2.4.

DateTime dateOfBirth is from

//<input class="form-control text-box single-line" data-val="true" data-val-date="The field DateOfBirth must be a date." data-val-required="The DateOfBirth field is required." id="DateOfBirth" name="DateOfBirth" type="datetime" value="">

3.2.5.

int teamId is from

//<input class="form-control text-box single-line" data-val="true" data-val-number="The field TeamId must be a number." data-val-required="The TeamId field is required." id="TeamId" name="TeamId" type="number" value="">

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

3.3.

//[HttpPost]

//public ActionResult Create3(BusinessLayer.Gamer gamer)

If the view has a lot of input,

then the previous two ways is not a good idea.

It is always better to retrieve form data using model binding.

The model of the cshtml is BusinessLayer.Gamer,

so we can pass the model object into HttpPost action.

The property value of model object will contain the value

from input or select tag from cshtml based on name attribute.

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

3.4.

//[HttpPost]

//[ActionName("Create4")]

//public ActionResult Create\_Post()

Retrieve form data using model binding by UpdateModel() or TryUpdateModel()

...

//BusinessLayer.Gamer gamer = new BusinessLayer.Gamer();

//UpdateModel<BusinessLayer.Gamer>(gamer);

//UpdateModel(gamer);

//TryUpdateModel(gamer);

---------

3.4.1.

UpdateModel() and TryUpdateModel() inspects all the HttpRequest inputs

such as posted Form data, QueryString,

Cookies and Server variables and populate the gamer object.

---------

3.4.2.

UpdateModel() throws an exception if validation fails.

TryUpdateModel() will never throw an exception and

return false if validation fails.

---------

3.4.3.

//UpdateModel(gamer, new[] { "Id", "Gender", "City", "DateOfBirth", "TeamId" });

//TryUpdateModel(gamer, new[] { "Id", "Gender", "City", "DateOfBirth", "TeamId" });

The second parameter of UpdateModel() and TeyUpdateModel() is included properties.

In this case, it will only update the following properties into model.

"Id", "Gender", "City", "DateOfBirth", "TeamId"

The Name property is not included so it will not be updated.

---------

3.4.4.

//TryUpdateModel(gamer, null, null, new[] { "Name" });

//UpdateModel(gamer, null, null, new[] { "Name" });

update all properties except Name property

---------

3.4.5.

//TryUpdateModel<IGamer>(gamer);

//UpdateModel<IGamer>(gamer);

The UpdateModel() function will update only the properties

that are present in the interface.

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

4.

//[HttpPost]

//Edit

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

4.1.

//[HttpPost]

//public ActionResult Edit(BusinessLayer.Gamer gamer)

Edit by Model binding will open the back door for unintended update.

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

4.2.

//[HttpPost]

//[ActionName("Edit2")]

//public ActionResult Edit\_Post(int id)

Solved the unintended update.

Edit by UpdateModel() and TryUpdateModel()

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

4.3.

//[HttpPost]

//public ActionResult Edit3([Bind(Include = "Id, Gender, City, DateOfBirth, TeamId")]BusinessLayer.Gamer gamer)

Solved the unintended update.

Edit by Model binding with Bind include attribute

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

4.4.

//[HttpPost]

//public ActionResult Edit4([Bind(Exclude = "Name")]BusinessLayer.Gamer gamer)

Solved the unintended update.

Edit by Model binding with Bind exclude attribute

\*/

3.2. OnlineGame.Web/Views/Gamer/Edit.cshtml

Graphical user interface, text, application, email

Description automatically generated

@model BusinessLayer.Gamer

@{

    Layout = null;

}

<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>

<div class="row">

    <h2>Update Gamer</h2>

    @\*@using (Html.BeginForm("Edit", "Gamer"))\*@

    @\*@using (Html.BeginForm("Edit2", "Gamer"))\*@

    @\*@using (Html.BeginForm("Edit3", "Gamer"))\*@

    @\*@using (Html.BeginForm("Edit4", "Gamer"))\*@

    @using (Html.BeginForm("Edit5", "Gamer"))

    {

        <div class="form-horizontal">

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

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

            <div class="form-group">

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

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

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

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

                    @Html.DisplayFor(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, new { @class = "control-label col-md-2" })

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

                    @\*@Html.EditorFor(model => model.Gender, new { htmlAttributes = new { @class = "form-control" } })\*@

                    @\*@Html.DropDownList("Gender",new List<SelectListItem>

                        {

                            new SelectListItem{Text = "Male", Value = "Male"},

                            new SelectListItem{Text = "Female", Value = "Female"}

                        })\*@

                    @Html.DropDownList("Gender", new List<SelectListItem>

                    {

                        new SelectListItem{Text = "Male", Value = "Male"},

                        new SelectListItem{Text = "Female", Value = "Female"}

                    }, "Select Gender")

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

                </div>

            </div>

            <div class="form-group">

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

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

                    @Html.EditorFor(model => model.TeamId, new { 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", "Index2")

    </div>

</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">

\*@