Mục tiêu:

- Áp dụng ADO.NET để thực hiện các thao tác Select, Insert, Update, Delete trong ASP.NET MVC.
- Hiểu và Vận dụng được mô hình sinh code khi tạo view có Model trong ASP.NET MVC.

1 CHUẨN BỊ:

1.1 Database:

Tạo database `WebShopHocTap`

```
--Tạo bảng Product:
Create database WebShopHocTap
Go
USE [WebShopHocTap]
GO

/****** Object:  Table [dbo].[Product]    Script Date: 03/14/2018 22:01:04 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Product](
	[ID] [int] NOT NULL identity(1,1),
	[Name] [nvarchar](50) NULL,
	[Alias] [nvarchar](50) NULL,
	[CategoryID] [int] NULL,
	[Images] [nvarchar](50) NULL,
	[CreateDate] [datetime] NULL,
```

```
	[Price] [decimal](18, 0) NULL,
	[Detail] [ntext] NULL,
	[Status] [bit] NULL,--1: đang sử dụng; 0: đã xóa
 CONSTRAINT [PK_Product] PRIMARY KEY CLUSTERED
(
	[ID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
--Tạo Bảng Category
USE [WebShopHocTap]
GO

/****** Object:  Table [dbo].[Category]    Script Date: 03/14/2018 22:02:32 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Category](
	[ID] [INT] NOT NULL identity(1,1),
	[Name] [NVARCHAR](50) NULL,
	[Alias] [NVARCHAR](50) NULL,
	[ParentID] [INT] NULL,
	[CreateDate] [DATETIME] NULL,
	[Order] [INT] NULL,--Thứ tự
	[Status] [BIT] NULL,--trạng thái delete; 1: đang sử dụng; 0: đã xóa
 CONSTRAINT [PK_Category] PRIMARY KEY CLUSTERED
(
	[ID] ASC
```

```
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY
= OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]


GO
SET ANSI_PADDING OFF
GO
```

1.2 Thủ tục SQL cho 4 tác vụ: Select, insert, update, Delete

1. Thủ tục lấy danh sách category:

```
USE [WebShopHocTap]
GO
CREATE PROC [dbo].[PSP_Category_GetCategories]
AS
SELECT ID, Name, Alias, ParentID, CreateDate, [Order], [STATUS]
FROM [dbo].[Category]
ORDER BY [Order] DESC
```

2. Thủ tục insert – Update category

```
USE [WebShopHocTap]
GO
CREATE PROC [dbo].[PSP_Category_InsertUpdateCategory]
    @ID INT,
    @Name NVARCHAR(50) ,
    @Alias NVARCHAR(50) ,
    @ParentID INT ,
    @CreateDate DATE ,
    @Order INT ,
    @Status BIT
AS
IF EXISTS (SELECT 1 FROM dbo.Category where ID=@ID)
BEGIN
    UPDATE dbo.Category
    SET  Name=@Name, Alias=@Alias, ParentID=@ParentID,
CreateDate=@CreateDate, [ORDER]=@Order, [Status]=@Status
    WHERE ID=@ID
END
ELSE
BEGIN
     INSERT  INTO Category( Name, Alias, ParentID,
CreateDate, [Order], [Status])
    VALUES( @Name, @Alias, @ParentID, @CreateDate,
@Order, @Status)
END
```

3. Thủ tục xóa category (cập nhật status=0 trong bảng category)

```
USE [WebShopHocTap]
GO
CREATE PROC [dbo].[PSP_Category_DeleteByID]
@ID int
AS
IF EXISTS (SELECT 1 FROM dbo.Category where id=@ID)
BEGIN
    UPDATE dbo.Category
    SET [status]=0
    where id=@ID
END
```

4. Thủ tục Select Category theo ID

```sql
USE [WebShopHocTap]
GO
CREATE PROC [dbo].[PSP_Category_GetCategories]
AS
SELECT ID, Name, Alias, ParentID, CreateDate, [Order],
[STATUS]
FROM [dbo].[Category]
ORDER BY [Order] DESC
```

2 THỰC HIỆN NHỮNG THAO TÁC TRÊN ĐỐI TƯỢNG CATEGORY

Tạo project ASP.NET Web Application. (Empty / MVC).

2.1 Thực hiện class Database – Sử dụng ADO.NET để kết nối data SqlServer.

Bước 01: Trong thư mục models của project tạo class Category.cs, CategoryDb.cs, Database.cs

- Tạo Class Category: Class Models, chứa những thuộc tính của table Category trong database Sql server.
- Tạo class Database: Class chịu tránh nhiệm kết nối và thực hiện các phương thức thực thi thủ tục cho 4 tác vụ select, Insert, Update, Delete.
- File Web.Config thêm chuỗi kết nối:

```
<connectionStrings>
    <add
      name="connectionString"
      connectionString="server=MINHPHUC\SQLSERVER2014;database=WebShopHocTap;
integrated security=true"/>
  </connectionStrings>
```

Trong đó:

+ name: tên của chuỗi kết nối

+ connectionString: Chứa chuỗi kết nối (theo giá trị của thuộc tính chuỗi kết nối)

- Tạo Class CategoryDb: Class chịu trách nhiệm liên kết database với Controller. Viết những phương thức thực thi thành phần của class database.cs để trả về dữ liệu cho Controller.

2.2 Xây dựng lớp Database.cs trong thư mục models.

1. Khai báo namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
```

2. Khai báo biến (field) của đối tượng ADO.NET

```
field - biến thành viên
private SqlConnection connect;Kết nối đến SQL Server
private SqlCommand command;thực thi thủ tục
```

3. Khai báo hàm tạo (Constructor) cho class Database

```
public Database()
{
    khởi tạo đối tượng kết nối
    connect = new SqlConnection() { ConnectionString =
      ConfigurationManager.ConnectionStrings
      ["connectionString"].ConnectionString };
}
```

4. Viết phương thức (method) dùng chung để lấy về 1 danh sách

MyExcuteReader	Phương thức thực hiện thực thi thủ tục (store Procedure) trong sql với tác vụ SELECT
err	Tham số tham chiếu lưu lỗi
commandText	thuộc tính commandtext
commandType	Thuộc tính commandType
param	Danh sách parameter của thủ tục sql
SqlDataReader	SqlDataReader

```
2 references | 0 authors | 0 changes
public SqlDataReader MyExcuteReader(ref string err,string commandText,CommandType commandType,params
  SqlParameter[] param)
{
    SqlDataReader datareader = null;
    try
    {   Mở kết nối
        if (connect.State == ConnectionState.Open)
            connect.Close();
        connect.Open();
        khởi tạo đối tượng SqlCommand
        command = new SqlCommand() { Connection = connect, CommandText = commandText, CommandType =
          commandType, CommandTimeout = 600 };
        if (param != null)Kiểm tra Add tham số cho đối tượng SqlCommand.
        {
            foreach (SqlParameter item in param)
            {
                command.Parameters.Add(item);
            }
        }
        datareader = command.ExecuteReader();Phương thức của đối tượng SqlCommand trả về 1 danh
          sách kiểu SqlDataReader.
    }
    catch (Exception ex)
    {
        err = ex.Message;
    }
    return datareader;
}
```

5. Viết Phương thức (method) dùng chung cho các tác vụ Insert, Update, Delete

MyExcuteNonQuery	Phương thức dùng để thực thi thủ tục (store procedure) với các tác vụ: Insert, Update, Delete
err	tham số tham chiếu lưu lỗi
rows	tham số tham chiếu lưu số dòng thực hiện được của SqlCommand
commandText	thuộc tính command text
commandType	Thuộc tính command type
param	danh sách tham số (SqlParameter)
bool	thành công true; lỗi: false

```
3 references | 0 authors | 0 changes
public bool MyExcuteNonQuery(ref string err,ref int rows,string commandText,CommandType commandType,params SqlParameter[] param)
{
    bool result = false;
    try
    {   Mở kết nối
        if (connect.State == ConnectionState.Open)
            connect.Close();
        connect.Open();
        khởi tạo đối tượng SqlCommand
        command = new SqlCommand() { Connection = connect, CommandText = commandText, CommandType = commandType, CommandTimeout = 600 };
        if (param != null)
        {
            foreach (SqlParameter item in param)
            {
                command.Parameters.Add(item);
            }
        }
        rows = command.ExecuteNonQuery();Phuong thức thực thi không trả về truy vấn. Kết quả trả về số lượng dòng được thực hiện thành
           công.
        result = true;
    }
    catch (Exception ex)
    {
        err = ex.Message;
    }
    finally
    {
        connect.Close();//Đóng kết nối
    }
    return result;
}
```

2.3 Tạo class Category.cs

Xây dựng Class Category

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.ComponentModel.DataAnnotations;
5  using System.Linq;
6  using System.Web;
7
8  namespace Project_20210322_SIUD_ADO.Models
9  {
       13 references | 0 authors | 0 changes
10     public class Category
11     {
12         ID, Name, Alias, ParentID, CreateDate, Order, Status
13         [Display(Name="Mã Loại SP")]Thuộc hiển thị tên của thuộc tính category.
               6 references | 0 authors | 0 changes
14     1   public int ID { get; set; }ID tăng tự động trong sql
15         [DisplayName("Tên loại SP")]
               4 references | 0 authors | 0 changes
16     1   public string Name { get; set; }tên của category
               4 references | 0 authors | 0 changes
17     1   public string Alias { get; set; }bí danh
               4 references | 0 authors | 0 changes
18     1   public int ParentID { get; set; }category cha
               4 references | 0 authors | 0 changes
19     1   public DateTime CreateDate { get; set; }
               4 references | 0 authors | 0 changes
20     1   public int Order { get; set; } begin 1;
               4 references | 0 authors | 0 changes
21     1   public bool Status { get; set; }1. đang sử dụng; 0: đã xóa
22     }
23 }
```

2.4 Xây dựng Class CategoryDb.cs

Lớp CategoryDb là phương thức dùng để khởi tạo đối tượng database, và viết những phương thức thực thi dữ liệu cho từng model cụ thể. Class này được xây dựng như sau:

Xây dựng lớp BasicDb

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Project_20210322_SIUD_ADO.Models
{
    3 references | 0 authors | 0 changes
    public class BasicDb
    {
        Khai báo một đối tượng database
        public Database data;
        1 reference | 0 authors | 0 changes
        public BasicDb()
        {
            data = new Database();
        }
        Class này dùng để:
        -1. Khai báo đối tượng thuộc class Database (đã được làm từ bước trước)
        -2. Khởi tạo đối tượng data trong hàm tạo.
        -3. Dùng cho lớp __Db khác kế thừa lại.
    }
}
```

Xây dựng lớp CategoryDb.cs

```
8 references | 0 authors | 0 changes
public class CategoryDb:BasicDb
{
    7 references | 0 authors | 0 changes
    public CategoryDb():base()
    { }
```

Thực hiện kế thừa từ lớp basicDb để khởi tạo Database.

Phương thức Lấy danh sách Category [ref]

GetCategories	Phương thức Lấy danh sách Category từ database Sql Server
err	tham số tham chiếu lưu lỗi
List<Category>	Danh sách Category được trả về

```
2 references | 0 authors | 0 changes
public List<Category> GetCategories(ref string err)
{
    Khai báo biến list dạng danh sách kiểu Category.
    List<Category> list = new List<Category>();
    Khai báo đối tượng SqlDataReader để nhận giá trị trả về từ phương thức thực thi thủ tục
      được viết trong class Database ở bước trước.
    var dataReader = data.MyExcuteReader(ref err, "PSP_Category_GetCategories",
      CommandType.StoredProcedure, null);
    //Đọc giá trị sau khi đã nhận được kết quả
    if (dataReader != null){
        while (dataReader.Read()){
            list.Add(
                new Category(){
                    ID = Convert.ToInt32(dataReader["ID"].ToString()),
                    Name = dataReader["Name"].ToString(),
                    Alias = dataReader["Alias"].ToString(),
                    ParentID = Convert.ToInt32(dataReader["ParentID"].ToString()),
                    CreateDate = Convert.ToDateTime(dataReader["CreateDate"].ToString()),
                    Order = Convert.ToInt32(dataReader["Order"].ToString()),
                    Status = Convert.ToBoolean(dataReader["Status"].ToString())
                });
        }
    }
    return list;
}
```

Phương thức lấy Category theo ID [ref]

GetCategoryByID	Phương thức lấy Category theo ID
err	Tham số tham chiếu lưu lỗi
ID	mã số của Category muốn lấy

2 references | 0 authors | 0 changes

```
public Category GetCategoryByID(ref string err, int ID)
{
    Category category = null; Khởi tạo đối tượng category,
    var dataReader = data.MyExcuteReader(ref err, "PSP_Category_GetCategoryByID",
      CommandType.StoredProcedure, new SqlParameter("@ID",ID));Gọi thủ tục lấy 1 category theo
      ID từ SQL.
    if (dataReader != null)Kiểm tra giá trị trả về
    {
        while (dataReader.Read())
        {Gán giá trị cho những thuộc tính của Category
            category = new Category();
            category.ID = Convert.ToInt32(dataReader["ID"].ToString());
            category.Name = dataReader["Name"].ToString();
            category.Alias = dataReader["Alias"].ToString();
            category.ParentID = Convert.ToInt32(dataReader["ParentID"].ToString());
            category.CreateDate = Convert.ToDateTime(dataReader["CreateDate"].ToString());
            category.Order = Convert.ToInt32(dataReader["Order"].ToString());
            category.Status = Convert.ToBoolean(dataReader["Status"].ToString());
        }
    }
    return category;Trả về đối tượng Category tìm được
}
```

Phương thức Insert Category [ref]

InsertCategory	Phương thức thực hiên thao tác insert Category vào trong database
err	Tham số tham chiếu lưu lỗi
rows	Tham số tham chiếu lưu số dòng thực hiện thành công
category	Đối tượng Category
bool	Thực hiện thành công trả về : true; ngược lại trả về false

1 reference | 0 authors | 0 changes

```
public bool InsertCategory(ref string err,ref int rows,Category category)
{
    return data.MyExcuteNonQuery(ref err, ref rows, "PSP_Category_InsertUpdateCategory",
      CommandType.StoredProcedure,
           new SqlParameter("@ID", category.ID),
           new SqlParameter("@Name", category.Name),
          new SqlParameter("@Alias", category.Alias),
           new SqlParameter("@ParentID", category.ParentID),
           new SqlParameter("@CreateDate", category.CreateDate),
            new SqlParameter("@Order", category.Order),
             new SqlParameter("@Status", category.Status));
}
```

Phương thức update category [ref]

UpdateCategory	Phương thức thực hiên thao tác Update Category vào trong database
err	Tham số tham chiếu lưu lỗi
rows	Tham số tham chiếu lưu số dòng thực hiện thành công
category	Đối tượng Category
bool	Thực hiện thành công trả về : true; ngược lại trả về false

1 reference | 0 authors | 0 changes

```
public bool UpdateCategory(ref string err, ref int rows, Category category)
{
    SqlParameter[] param = new SqlParameter[]{
        new SqlParameter("@ID",category.ID),
         new SqlParameter("@Name",category.Name),
         new SqlParameter("@Alias", category.Alias),
          new SqlParameter("@ParentID", category.ParentID),
          new SqlParameter("@CreateDate", category.CreateDate),
           new SqlParameter("@Order", category.Order),
            new SqlParameter("@Status", category.Status)
    };
    return data.MyExcuteNonQuery(ref err, ref rows, "PSP_Category_InsertUpdateCategory",
      CommandType.StoredProcedure, param);
}
```

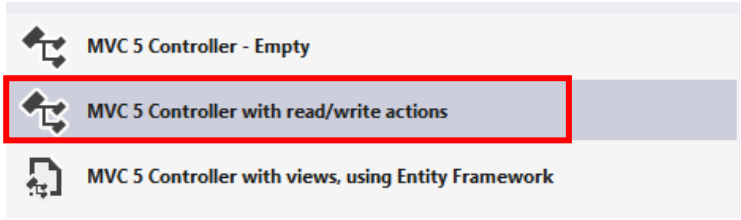
Phương thức Xóa Category [ref]

DeleteCategoryByID	PHương thức xóa Category
err	Tham số tham chiếu lưu lỗi
rows	Tham số tham chiếu lưu số dòng thực hiện thành công
category	Đối tượng Category

```
1 reference | 0 authors | 0 changes
public bool DeleteCategoryByID(ref string err, ref int rows, Category category)
{
    SqlParameter[] param = new SqlParameter[]{
        new SqlParameter("@ID",category.ID)
    };
    return data.MyExcuteNonQuery(ref err, ref rows, "PSP_Category_DeleteByID",
      CommandType.StoredProcedure, param);
}
```

2.5 Tạo Controller cho project

- HomeController.cs (trang chủ tạo theo template Empty) – trong trang chủ tạo 1 menu để gọi category. Chỉ có một action Index để hiển thị trang chủ của website.
- CategoryController.cs (trang quản lý Category) –url : /Category/Index; Tạo Controller Category sử dụng template: with read/write action. Sau khi tạo xong trong controller này chứa đầy đủ nhưng Action của các tác vụ select, insert, update, delete.



Trong CategoryController hãy thực hiện lần lượt những Action theo thứ tự sau.

(vì sau khi sử dụng template Controller đã có sẵn các Action, nên chỉ thực hiện code trên các Action đó)

Action index (hiển thị danh sách category)

```
public ActionResult Index()
{
    string err=string.Empty;
    var categories = new CategoryDb().GetCategories(ref err);
    return View(categories);
}
```

Action Create (hiển thị view cho phép thêm (Insert) database)

```
public ActionResult Create()
{
    return View();
}

POST: Category/Create
[HttpPost]
0 references | 0 authors | 0 changes
public ActionResult Create(Category collection)
{
    try
    {
        TODO: Add insert logic here
        string err = string.Empty;
        int rows = 0;
        var result = new CategoryDb().InsertCategory(ref err, ref rows,
          collection);

        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}
```

Với tác vụ create có 2 action :

- Một Action Create theo [**HttpGet**] dùng để hiển thị view chứa các form trống để nhập liệu.
- Một Action Create theo [**HttpPost**] dùng để lấy data nhập từ form truyền về controller theo Model. Sau đó Action này sẽ truyền data vào cho phương thức được viết trong CategoryDb để insert data vào trong SqlServer.

Action Edit (hiển thị view cho phép update database)

```
public ActionResult Edit(int id)
{

    var category = new CategoryDb().GetCategoryByID(ref err, id);
    return View(category);
}

POST: Category/Edit/5
[HttpPost]
0 references | 0 authors | 0 changes
public ActionResult Edit(int id, Category collection)
{
    try
    {
        TODO: Add update logic here

        var result = new CategoryDb().UpdateCategory(ref err, ref rows,
          collection);

        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}
```

Action detail (hiển thị view detail của category)

```
public ActionResult Details(int id)
{
    string err=string.Empty;
     var category = new CategoryDb().GetCategoryByID(ref err, id);
    var categories = new CategoryDb().GetCategories(ref err);
    var category = categories.SingleOrDefault(x => x.ID == id);
    if (category != null)
        return View(category);
    else
    {
        if(!string.IsNullOrEmpty(err))
        ViewBag.err =string.Format("Lỗi: {0}", err);
        else
        {
            ViewBag.err = "Không có giá trị";
        }
        return View();
    }
}
```

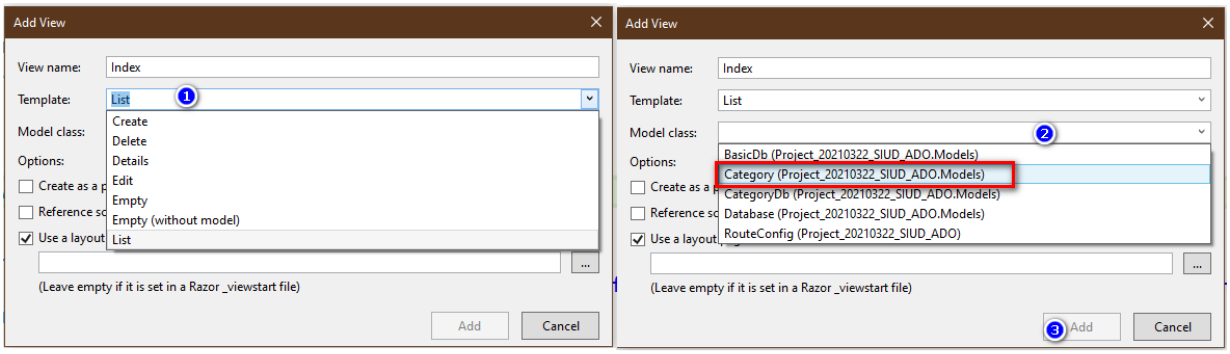
Action Delete (hiển thị view delete database)

```
public ActionResult Delete(int id)
{
    var category = new CategoryDb().GetCategoryByID(ref err, id);
    return View();
}

 POST: Category/Delete/5
[HttpPost]
0 references | 0 authors | 0 changes
public ActionResult Delete(int id, Category collection)
{
    try
    {
        TODO: Add delete logic here
        var result = new CategoryDb().DeleteCategoryByID(ref err, ref rows,
          collection);
        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}
```

2.6 Tạo View cho từng Action trong Controller.

Từ mỗi Action theo từng tác vụ :Select, Insert, Update, Delete: tạo view theo những template tương ứng.



- List: hiện thị danh sách (index)
- Create: Form thêm mới một record;
- Delete: View xóa 1 record.
- Details: hiển thị chi tiết 1 record;
- Edit: Form hiển thị cho phép chỉnh sửa.

- Chọn class Model cho template

Chú ý: chọn class xây dựng chứa thuộc tính của đối tượng. trong trường hợp này là **Category**.

Sau khi tạo xong view cho Action tương ứng code sẽ tự động sinh ra theo đúng thuộc tính trong class model. Khi này có thể chỉnh sửa view theo yêu cầu nếu muốn.

Những khai báo cần chú ý trong view.

Khai báo Model nay đầu view.

```
@using Project_20210322_SIUD_ADO.Models;
@model IEnumerable<Category>
```

Khi sử dụng cần chú ý

```
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Name)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Alias)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.ParentID)
        </td>
        <td>
            @item.CreateDate.ToShortDateString();
            @*@Html.DisplayFor(modelItem => item.CreateDate)*@
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Order)
        </td>
        <td>
            @(item.Status?"Đang sử dụng":"đã xóa")
            @*@Html.DisplayFor(modelItem => item.Status)*@
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.ID })
            @Html.ActionLink("Details", "Details", new { id=item.ID })
            @Html.ActionLink("Delete", "Delete", new { id=item.ID })
        </td>
    </tr>
}
```

1. Sử dụng vòng lặp Foreach để duyệt trên Model.
2. Lấy ID của dòng được chọn từ danh sách.
3. Có thể dùng trực tiếp từ @Item. để hiển thị dữ liệu

Với view có dùng form (edit, Create)


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

    <div class="form-horizontal">
        <h4>Category</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.Alias, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Alias, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Alias, "", 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>
```