بسم الله الرحمن الرحیم

آموزش

ارتباط با بانک اطلاعاتی در

ASP.NET MVC

تهیه کننده :

ابوالفضل عقیلی کوهستانی

فهرست

در این فایل آموزشی قصد داریم تا پس از آشنایی اندک با تکنولوژی جدید MVC شرکت مایکروسافت یک پروژه بصورت عملی انجام دهیم تا بتوانیم تا حدودی به ماهیت MVC پی ببریم . قبل از هر چیز جا دارد که از استاد عزیزم جناب آقـــای ستـایش تشکر نمایم که همواره من را در زمینه های مختلف بهره مند می سازند .

MVCچیست ؟

MvC مخفف سه کلمه Model View Controller هست .در واقع MVCبر روی معماری های چند لایه ای جهت جداسازی قسمت های مختلف برنامه و به طور دقیق تر جدا کردن بخش ها منطقی برنامه اعم از دیتا ، permission ها ، چک کردن صحت داده ها و .... از لایه Presentation layer یا در واقع همان لایه ای که مستقیما با کاربر نهایی (End user) در ارتباط است ،قرار میگیرد. پس بر اساس توضیحات فوق میتوانیم هر یک از بخش های معماری MVC یعنی Modelو View و controller را به شکل زیر تعریف کنیم.

۱ –Model :

در واقع بار اصلی معماری MVC بر عهده این بخش است . این بخش میتواند با داده ها در ارتباط باشد .الزاما منظور از داده حتما ارتباط با پایگاه های داده همچون MSSQL و Access و ... نیست ، حتی منبع داده ها در بخش

Model میتواند یک آرایه از اعداد و یا هر چیز دیگری باشد . همچنین Model وظیفه چک کردن داده ها جهت صحت درستی داده ها را هم در بر عهده دارد (در این زمینه همکاری بیشتری با بخش Controller دارد) و همینطور وظایف دیگری که در مثال ها ی عملی که در آینده خواهم زد بیشتر آشنا خواهید شد.

۲ –View :

این بخش که در واقع همان بخش Presentation Layer در معماری ۳ لایه میباشد وظیفه بر قراری ارتباط با کاربر نهایی و گرفتن داده از کاربر و نمایش داده های اماده با کاربراز طریق برقراری ارتباط با دو بخش دیگر یعنی Modelو controller است .در واقع نکته مهمی که در بخش View باید ان را مد نظر داشت این است که این لایه مسئول کنترل صحت داده های وارد شده از طریق کاربر و همچنین مسئول صحت داده های نشان داده شده به کاربر نیست . در واقع این بخش یا داده های خام کار میکند . به عنوان یک مثال ساده خیلی از برنامه نویسان موقعی که در فرم Login برنامه ،کاربر کلمه عبور خود را وارد میکند ، در همان فرم Login اقدام به چک کردن پسورد مبنی بر صحت آن و ... می کنند . که این عمل در معماری MVC قابل قبول نیست. در واقع برای حل مسئله فوق در معماری MVC در فرم Login هنگامی که کاربر کلمه عبور را وارد کرد و دکمه Login یا ورود را زد ، کلمه عبور داده شده بدون هیچ گونه اعمالی اعم از Encrypt کردن و ... به بخش های دیگر فرستاده میشود و فقط یک نتیجه ساده مبنی بر این که کاربر اجازه ورود دارد یا خیر را از بخش های دیگر دریافت میکند که بر اساس ان اجازه ورود کاربر به برنامه داده میشود.

۳ – Controller :

این بخش همانطور که از اسم ان مشخص است یه بخش کنترل کننده می باشد ، و در واقع واسطی بین دو بخش Model و View میباشد.حال ببینیم روند اجرای برنامه در معماری MVC به چه نحوی خواهد بود.

در معماری MVC روند کلی برنامه (جزئیات را در ادامه خواهید دید) به این شکل است که کاربر تقاضای خود را از طریق واسط های برنامه نویسی (نظیر Form ها و User Control ها و ..) از برنامه (از بخش View) درخواست می کند . بخش Viewدر خواست ها را به بخش Controller فرستاده و این بخش با برقراری ارتباط با بخش Model در خواست های کاربر را پردازش کرده و پس از پایان پردازش زمانی که خروجی درخواست داده شده آماده گردید بخش Controller بخش Viewرا آگاه میسازد تا خود را بر اسا س تغییرات جدید که اصطلاحا در معماری MVC به آن حال Model می گویند ، به روز سازد . در واقع چیزی که باعث میشود تا بخش Controllerبه بخش View اطلاع دهد که باید حالت جدید model را دریافت کند و خود را Update کند این است که بخش Viewباید قبلا خودش را در بخش Model اصطلاحا Registerکرده باشد که البته عمل Register کردن توسط بخش Controllerانجاام میگیرد . نحوه register کردن بخش View به معماری آن محیط و همچنین زبانی که توسط آن برنامه را گسترش میدهید و همچنین قابلیت های آن زبان بستگی دارد

نکته اول : نرم افزار مورد استفاده در این برنامه visual studio می باشد .

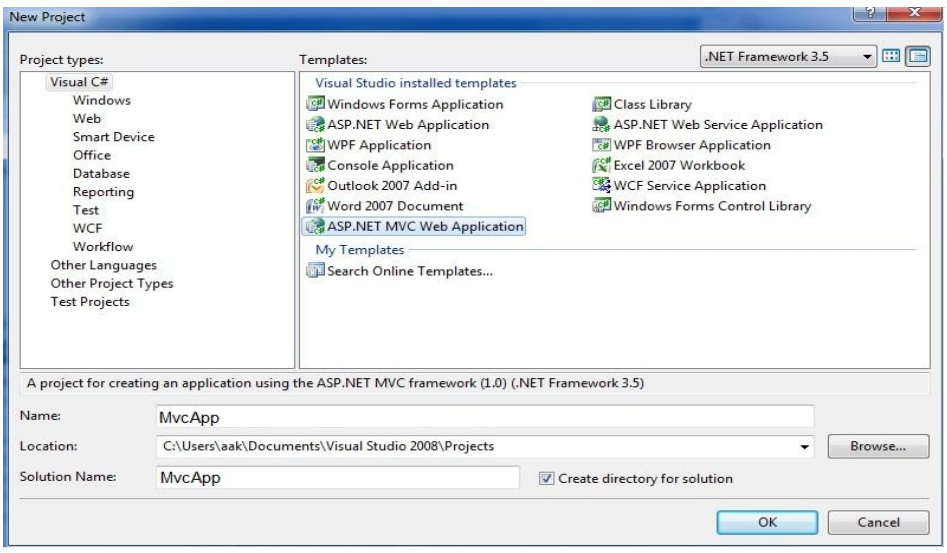
نکته دوم : بهتر است برای ایجاد این صفحات با asp.net و LINQ و یا ADO.net آشنایی داشته باشید .

قدم اول– دریافت asp.net mvc و نصب آن :

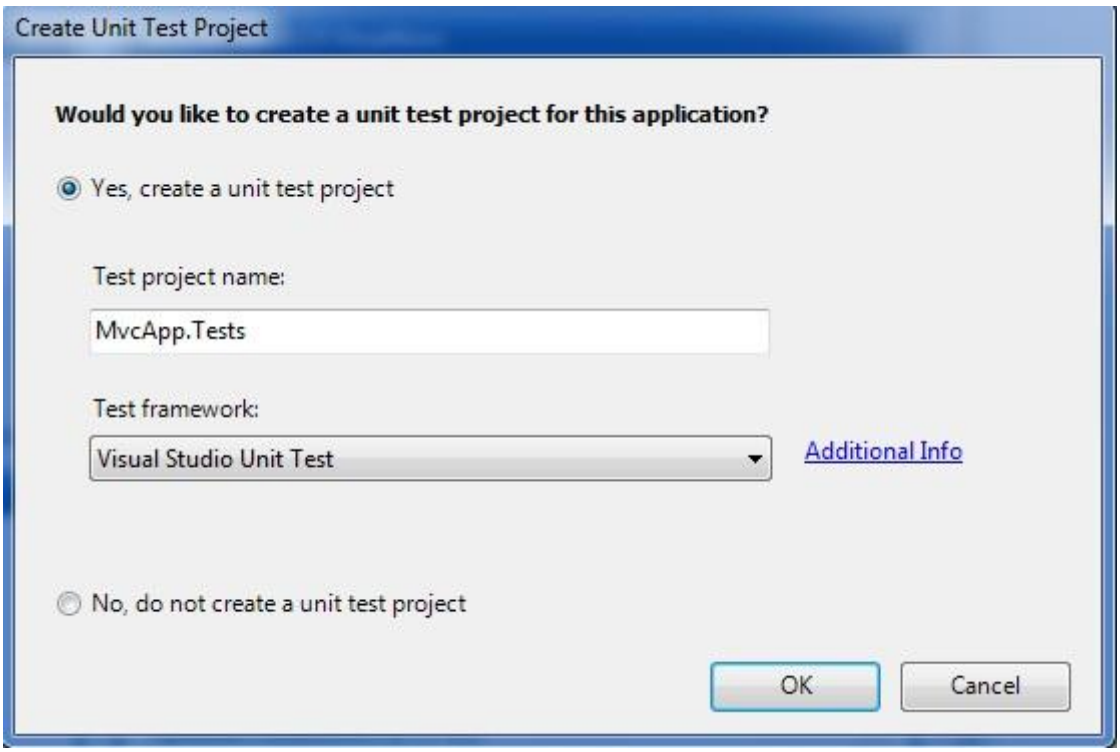
ابتدا پلت فرم asp.net mvc را از اینجا دریافت نمایید و آنرا درون سیستم خود نصب نمایید .(در Visual Studio 2010 بصورت پیش فرض قرار دارد و نیاز به نصب آن نیست) .

قدم دوم– ساخت اولین برنامه با mvc :

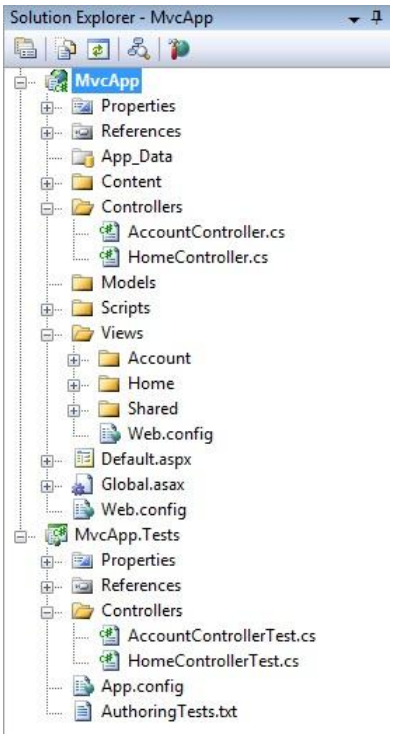
۱–پس از نصب این پلت فرم وارد محیط Visual Studio شده و گزینه new -> project ... را انتخاب می نماییم واز پنجره باز شده asp.net mvc web application را انتخاب می کنیم و نام آنرا MvcApp می گذاریم و مسیر مورد نظر را داده و بر روی دکمه ok کلیک می کنیم .



۲- پس از کلیک بر روی دکمه ok پنجره create unit test project باز می شود . این پنجره از ما می پرسد که آیا مایلید که محلی برای تست برنامه تان درون پروژه داشته باشید . چون ما به این قسمت نیاز داریم همان تنظیمات پیش فرض را انتخاب کرده و گزینه ok را می زنیم



پس از کلیک بر روی دکمه ok در قسمت solusion Explorer فایل ها و پوشه هایی بطور پیش فرض قرار می گیرد. این فایل ها در دو دسته قرار می گیرند: ۱- mvcApp: که فایل های پروژه درونش قرار می گیرد . ۲- MvcApp.Tests: که مربوط به تست صفحات پروژه می باشد.



۳- حالا باید تعدادی از این فایل ها و پوشه هایی که بطور پیش فرض قرار گرفته را پاک کنیم ۱- ابتدا به قسمت Controlles مربوط به MvcApp رفته و فایل HomeController.cs را حذف می نماییم. ۲- به قسمت Views مربوط به MvcApp رفته و پوشه

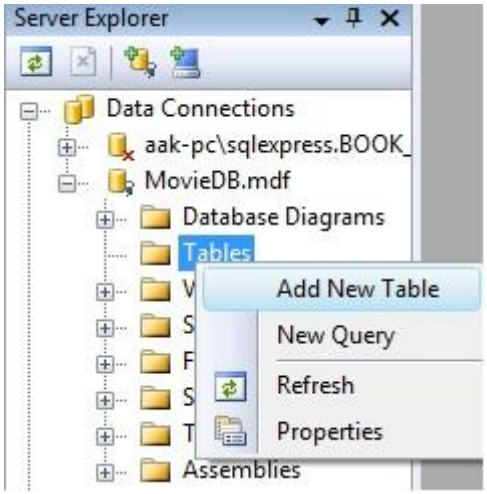
Home را پاک می کنیم . ۳- در انتها هم به قسمت Controlles مربوط به MvcApp.Tests رفته و فایل HomeController را حذف می نماییم.

اکنون می توانیم تنظیمات مورد نظر خود را انجام دهیم.ما می خواهیم در این برنامه با یک بانک اطلاعاتی (sql) ارتباط برقرار کرده و عملیاتی نظیر درج، حذف و ... را انجام هیم.

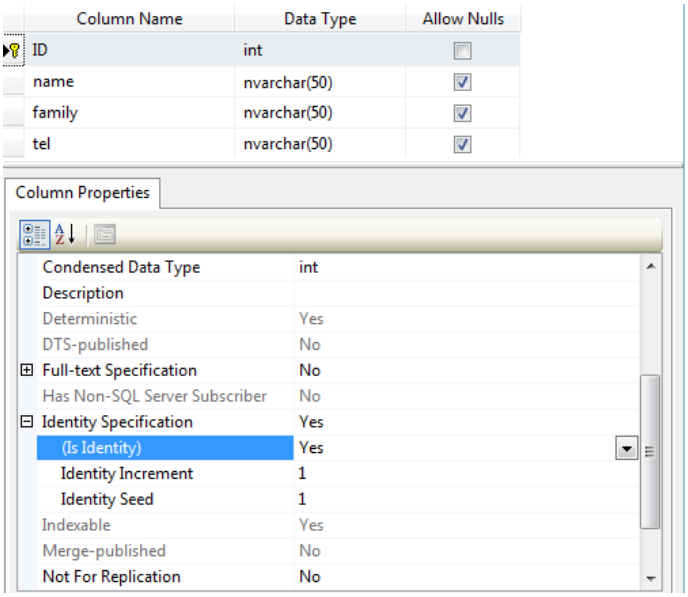
قدم سوم – ایجاد یک بانک برای پروژه :

بر روی MvcApp در قسمت solusion Explorer راست کلیک کرده و گزینه add -> new Item... را انتخاب کرده و از منوی باز شده گزینه sql server database را انتخاب می کنیم و نام آنرا MovieDB می گذاریم و دکمه ok را می زنیم . با اینکار یک بانک اطلاعاتی برای پروژه ما ایجاد می شود.

بر روی MovieDB.mdf دابل کلیک می کنیم تا اطلاعات آن در قسمت server Explorer ظاهر شود.بر روی پوشه table راست کلیک کرده و گزینه add new table را انتخاب می کنیم .



در پنجره باز شده فیلد های زیر را وارد می کنیم .



بر روی فیلد ID راست کلیک کرده و set primary key را انتخاب می کنیم .همچنین مقدار Identity این فیلد را yes می کنیم . با اینکار فیلد id بعنوان کلید اصلی جدول انتخاب می شود و هر گاه درجی در جدول صورت گیرد سیستم بطور خودکار به آن یک کد غیر تکراری می دهد .

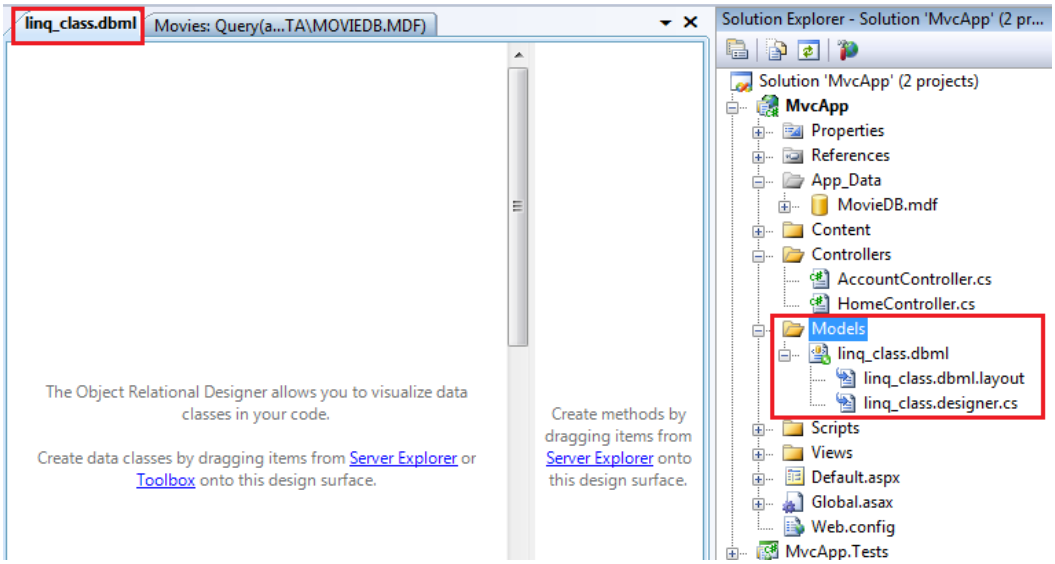
دکمه های CTRL + S را فشار می دهیم .پنجره ای باز می شود که نام جدول را از ما می خواهد .نام این جدول را Movies گذاشته و دکمه ok را کلیک می کنیم . حالا برای تست جدول مقادیری را در آن وارد می کنیم. برای اینکار بر روی جدول ایجاد شده راست کلیک کرده و Show Table Data را انتخاب می کنیم و مقادیر خود را وارد می کنیم . برای مثال ما مقادیر زیر را وارد کردیم .

ID	name	family	tel
1	abolfazl	aghili	091581235
2	mgms	sarviha	091581275
3	ali	kashani	091586852

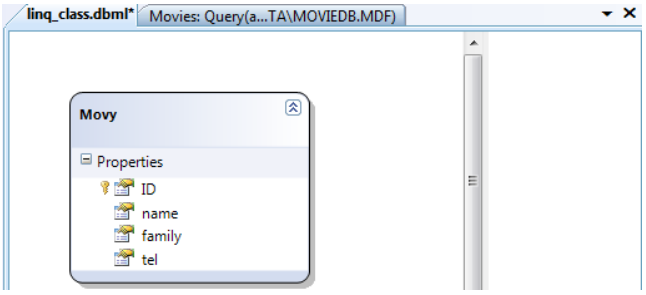
فعلاً کار با بانک اطلاعاتی تمام شد و حالا باید بین بانک و صفحات وب ارتباط برقرار کنیم.

قدم چهارم – برقراری ارتباط بین بانک و پروژه :

بر روی پوشه Model راست کلیک کرده و گزینه add -> new Item... را انتخاب می کنیم و در پنجره باز شده گزینه LINQ to SQL Classes را انتخاب می کنیم و نام آنرا linq_class می گذاریم .



پنجره ای به شکل بالا باز خواهد شد .در سمت راست در پوشه Models فایل های مربوط به Llinq_class را مشاهده می کنیم. بر روی فایل MovieDB.mdf رفته و بر روی آن دابل کلیک کرده تا در پنجره server Explorer ظاهر شود . حالا جدولی را که که ساخته بودیم با Drag & Drop به صفحه linq_class.dbml می کشیم و رها می کنیم و صفحه را ذخیره می کنیم .



باید شکلی همانند شکل بالا ایجاد شود . (برای خلوت شدن صفحه تمام صفحات باز را می بندیم .)

قدم پنجم – ایجاد صفحات وب سایت :

مرحله آخر کار رسیده است . در ادامه بر روی پوشه Controllers مربوط به MvcApp راست کلیک کرده و گزینه add -> controller را انتخاب می کنیم و در پنجره باز شده در قسمت نام کنترلر homeController را وارد کرده و تیک پایین را برای انجام عملیاتی نظیر درج، حذف ، بروزرسانی و ... می گذاریم و دکمه add را می زنیم . با اینکار یک فایل از نوع cs با نام homeController ایجاد می شود که حاوی کدهای زیر است:

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

namespace MvcApp.Controllers
{
    public class homeController : Controller
    {
        //
        // GET: /home/

        public ActionResult Index()
        {
            return View();
        }

        //
        // GET: /home/Details/5

        public ActionResult Details(int id)
        {
            return View();
        }

        //
        // GET: /home/Create

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

        //
        // POST: /home/Create

        [AcceptVerbs(HttpVerbs.Post)]
        public ActionResult Create(FormCollection collection)
        {
            try
            {
                // TODO: Add insert logic here

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

```
        //
        // GET: /home/Edit/5

        public ActionResult Edit(int id)
        {
            return View();
        }

        //
        // POST: /home/Edit/5

        [AcceptVerbs(HttpVerbs.Post)]
        public ActionResult Edit(int id, FormCollection collection)
        {
            try
            {
                // TODO: Add update logic here

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

این صفحه اصلی ترین صفحه در mvc است و در واقع کنترل کننده و ارتباط دهنده بین قسمت های view و model می باشد .در ادامه با ایجاد ساخت صفحات توسط این صفحه و انجام عملیات های لازم برای صفحات در این کنترلر بحث خواهیم کرد.

۱ – ایجاد صفحه برای نمایش اطلاعات بانک (index) :

برای ادامه کار کد زیر را در قسمت using ها(فضاهای نام) اضافه کنید.

```
using MvcApp.Models;
```

پس از اضافه کردن فضای نام مورد نظر کد زیر را درون کلاس `public class homeController : Controller` قرار دهید

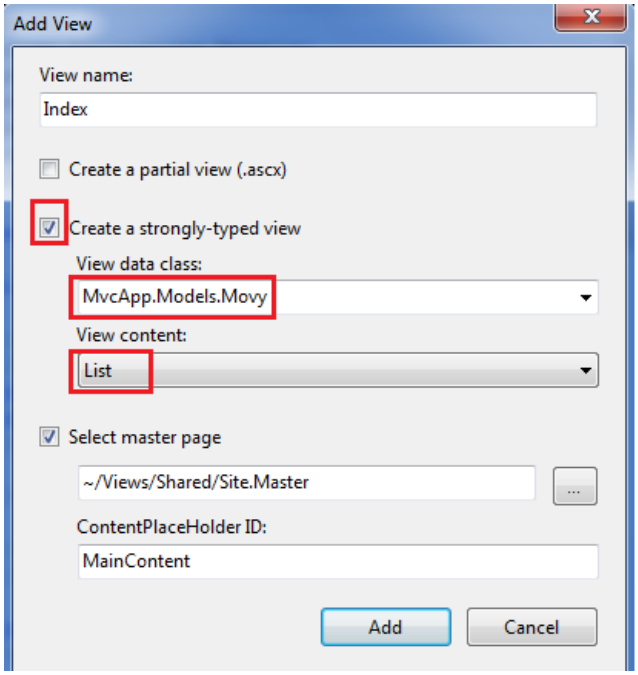
```
linq_classDataContext test = new linq_classDataContext();
```

با این کار یک کلاس از نوع `linq_classDataContext` با نام `test` در صفحه ایجاد می شود و جدولی را که درون فایل linq_class قرار دارد را برای استفاده های بعدی درون این برنامه مهیا می کند.

با استفاده از منوی Build و گزینه build solusion یک بار برنامه را چک می کنیم . حالا کد درون تابع Index() را که بصورت زیر در می آوریم:

```
return View(test.Movies.ToList());
```

test نام کلاس `linq_classDataContext` است و `Movies` نام جدولی است که از بانک درون linq_class قرار دادیم و تابع `ToList()` هم اطلاعات درون این جدول را بصورت لیست به کاربر نمایش می دهد. حالا نام تابع Index() را انتخاب می کنیم و راست کلیک کرده و گزینه Add View را انتخاب می کنیم صفحه ای همانند صفحه زیر باز می شود . اطلاعات صفحه را همانند شکل زیر تغییر می دهیم .



با زدن دکمه add بطور خودکار یک صفحه برای شما با نام Index.aspx ایجاد می شود که درون آن کد های زیر قرار دارد.

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/Shared/Site.Master"
Inherits="System.Web.Mvc.ViewPage<IEnumerable<MvcApp.Models.Movy>>" %>

<asp:Content ID="Content1" ContentPlaceHolderID="TitleContent" runat="server">
      Index
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">

    <h2>Index</h2>

    <table>
        <tr>
            <th></th>
            <th>
                ID
            </th>
            <th>
                name
            </th>
            <th>
                family
            </th>
            <th>
                tel
            </th>
        </tr>

    <% foreach (var item in Model) { %>

        <tr>
            <td>
                <%= Html.ActionLink("Edit", "Edit", new { id=item.ID }) %> |
                <%= Html.ActionLink("Details", "Details", new { id=item.ID })%>
            </td>
            <td>
                <%= Html.Encode(item.ID) %>
            </td>
            <td>
                <%= Html.Encode(item.name) %>
            </td>
            <td>
```

```
                <%= Html.Encode(item.family) %>
            </td>
            <td>
                <%= Html.Encode(item.tel) %>
            </td>
        </tr>

    <% } %>

    </table>

    <p>
        <%= Html.ActionLink("Create New", "Create") %>
    </p>

</asp:Content>
```

یک بار برنامه را اجرا کنید .ملاحظه می کنید اطلاعات درون جدول شما نمایش داده شده است. به همین سادگی یک صفحه با کمترین کد نویسی توسط برنامه نویس ایجاد می شود . حالا تنظیم کردن و مطابق میل خود در آوردن این صفحه دست برنامه نویس یا طراح سایت می باشدکه می تواند تنظیمات خود را در صفحه index.aspx انجام دهد .

۲ – ایجاد صفحه ای برای ثبت اطلاعات در جدول (Create):

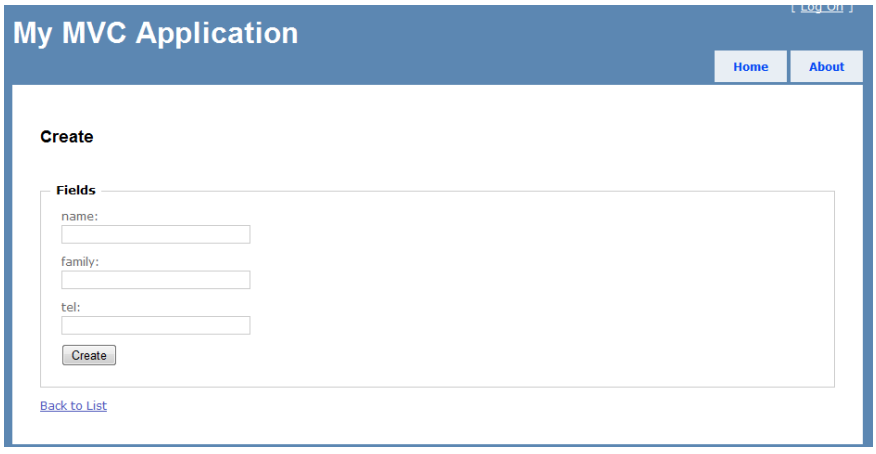
برای ثبت در اطلاعات جدول دو تابع درون فایل homeController قرار دارد که نام هر دو create() می باشد .تابع اول که فاقد مقدار ورودی می باشد مربوط به ظاهر صفحه ثبت اطلاعات می باشد و تابع دوم مربوط به فعل و انفعالاتی است که موجب ثبت اطلاعات می شود.

بر روی نام تابع اول راست کلیک کرده و گزینه add view را انتخاب کنید . تنظیمات این صفحه همانند صفحه index می باشد ولی یک تفاوت عمده دارد. در قسمت view cintent باید بجای list از create استفاده کنیم. چون این صفحه وظیفه اش ثبت اطلاعات است.با زدن دکمه Add صفحه create ایجاد می شود.کار با تابع اول تمام شد و صفحه create ایجاد شد و کد های درون ان بصورت زیرمی باشد:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/Shared/Site.Master"
Inherits="System.Web.Mvc.ViewPage<MvcApp.Models.Movy>" %>
```

```
<asp:Content ID="Content1" ContentPlaceHolderID="TitleContent" runat="server">
      Create
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">

    <h2>Create</h2>

    <%= Html.ValidationSummary("Create was unsuccessful. Please correct the errors and
try again.") %>

    <% using (Html.BeginForm()) {%>

        <fieldset>
            <legend>Fields</legend>
            <p>
                <label for="ID">ID:</label>
                <%= Html.TextBox("ID") %>
                <%= Html.ValidationMessage("ID", "*") %>
            </p>
            <p>
                <label for="name">name:</label>
                <%= Html.TextBox("name") %>
                <%= Html.ValidationMessage("name", "*") %>
            </p>
            <p>
                <label for="family">family:</label>
                <%= Html.TextBox("family") %>
                <%= Html.ValidationMessage("family", "*") %>
            </p>
            <p>
                <label for="tel">tel:</label>
                <%= Html.TextBox("tel") %>
                <%= Html.ValidationMessage("tel", "*") %>
            </p>
            <p>
                <input type="submit" value="Create" />
            </p>
        </fieldset>

    <% } %>

    <div>
        <%=Html.ActionLink("Back to List", "Index") %>
    </div>

</asp:Content>
```

مقداری که در این کدها با پس زمینه زرد قرار دارد را ازصفحه حذف نمایید . چون به این کدها نیازی نداریم.با اجرای برنامه می توانیم صفحه create را ملاحظه کنیم که به شکل زیر است :



دکمه create را در این صفحه ملاحظه می کنید ولی این دکمه هیچ عملی را انجام نمی دهد. برای اینکه این دکمه عملیات ثبت اطلاعات را انجام دهد باید تابع دوم create را کمی دستکاری کرد. به صفحه homeController برگردید و کد های تابع دوم را به شکل زیر تغییر دهید.

```
public ActionResult Create([Bind (Exclude="id")]Movy mtc)
{
    try
    {
        // TODO: Add insert logic here
        test.Movies.InsertOnSubmit(mtc);
        test.SubmitChanges();
        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}
```

دوباره برنامه را اجرا کنید و اطلاعاتی را وارد کرده و دکمه create را کلیک کنید . مشاهده می کنید که اطلاعات ورودی در بانک ثبت شده است.

۳ – حذف اطلاعات درون جدول (Delete) :

می خواهیم لینکی را در صفحه index قرار دهیم که با کلیک بر روی آن اطلاعات سطر مورد نظر از جدول را پاک نماییم.به صفحه index.aspx در پوشه home بروید . وکد زیر را به آن اضافه کنید

```
<%= Html.ActionLink("Delete", "Delete", new { id= item.id }) %> |
```

با اضافه کردن این کد یک لینک بنام Delete به صفحه index اضافه می شود. کد های زیر کدهای اصلاح شده صفحه index می باشد

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/Shared/Site.Master"
Inherits="System.Web.Mvc.ViewPage<IEnumerable<movieapp.Models.Table1>>" %>

<asp:Content ID="Content1" ContentPlaceHolderID="TitleContent" runat="server">
      Index
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">

    <h2>Index</h2>

    <table>
        <tr>
            <th></th>
            <th>
                کد
            </th>
            <th>
                نام
            </th>
            <th>
                خانوادگی نام
            </th>
            <th>
                تلفن شماره
            </th>
```

```
        </tr>

    <% foreach (var item in Model) { %>

        <tr>
            <td>
                <%= Html.ActionLink("Edit", "Edit", new { id= item.id }) %> |
                <%= Html.ActionLink("Delete", "Delete", new { id= item.id }) %> |
            </td>
            <td>
                <%= Html.Encode(item.id) %>
            </td>
            <td>
                <%= Html.Encode(item.name) %>
            </td>
            <td>
                <%= Html.Encode(item.family) %>
            </td>
            <td>
                <%= Html.Encode(item.tel) %>
            </td>
        </tr>

    <% } %>

    </table>

    <p>
        <%= Html.ActionLink("Create New", "Create") %>
    </p>

</asp:Content>
```

خوب صفحه index.aspx دیگر تکمیل شده است .آن را save کرده و می بندیم.حال به صفحه اصلی در mvc یعنی homecontroller.cs می رویم . تابع زیر(تابع Delete) را به انتهای کدها اضافه می کنیم :

```
        public ActionResult Delete(int id)
        {
            var movieToDelete = test.Movies.First(m => m.ID == id);

            // Delete
            test.Movies.DeleteOnSubmit(movieToDelete);
            test.SubmitChanges();

            // Show Index view
            return RedirectToAction("Index");
        }
```

کاربر با کلیک بر روی دکمه delete مقدار id آن سطر را به تابع delete ارسال می کند و تابع delete هم با توجه به آن id را حذف می نماید.مهم ترین خط در کد های بالا کد زیر می باشد.

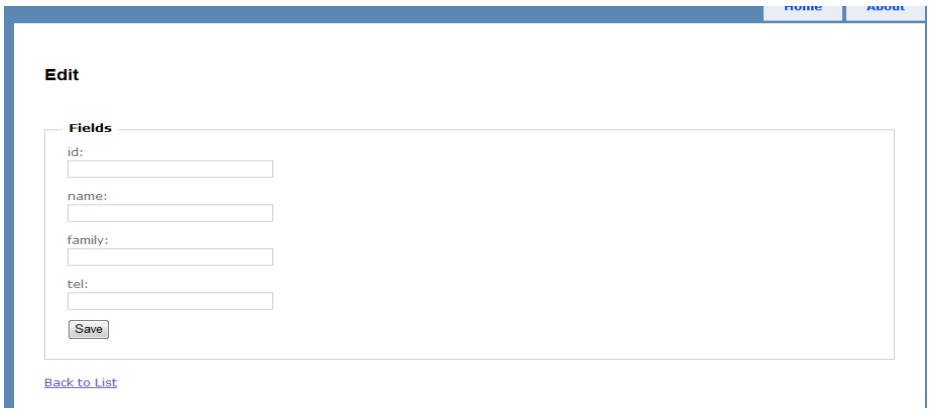
```
var movieToDelete = test.Table1s.First(m => m.id == id);
```

وظیفه این خط کد گرفتن id سطری که بر روی لینک delete آن کلیک شده است می باشد که این مقدار را درون متغیری ذخیره کرده و در خط بعد با توجه به آن مقدار سطری را حذف می نماید. در خط بعد هم تغییرات در بانک انجام می شود .در آخرین خط این کدها کاربر دوباره به صفحه index.aspx هدایت می شود. به همین سادگی فرمان حذف نیز در صفحه ایجاد شد.

۴ – ایجاد صفحه ای برای بروز رسانی (Edit) :

آخرین صفحه هم مربوط به تغییرات اطلاعات بانک می باشد . فایل homecontroller را باز می کنیم . درون این فایل دو تابع به نام Edit دیده می شود . همانطور که قبلاً گفتیم تابع اول مربوط به ظاهر صفحه edit می باشد. بر روی نام تابع (Edit) راست کلیک کرده و گزینه Add View... را انتخاب می کنیم . در صفحه Add view تنها مقداری را که تغییر می دهیم ، مقدار view cintentمی باشد .مقدار آنرا به Edit تغییر می هیم و دکمه Add را می فشاریم . کدهای این صفحه به شکل زیر است که بهتر است مقداری را که با زمینه زرد نشان می دهیم را حذف نماییم چرا که کاربردی ندارد .

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/Shared/Site.Master"
Inherits="System.Web.Mvc.ViewPage<movieapp.Models.Table1>" %>
<asp:Content ID="Content1" ContentPlaceHolderID="TitleContent" runat="server">Edit
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
    <h2>Edit</h2>

    <%= Html.ValidationSummary("Edit was unsuccessful. Please correct the errors and
try again.") %>

    <% using (Html.BeginForm()) {%>

        <fieldset> <legend>Fields</legend>
            <p>
                <label for="id">id:</label>
                <%= Html.TextBox("id", Model.id) %>
                <%= Html.ValidationMessage("id", "*") %>
            </p>
            <p>
                <label for="name">name:</label>
                <%= Html.TextBox("name", Model.name) %>
                <%= Html.ValidationMessage("name", "*") %>
            </p>
            <p>
                <label for="family">family:</label>
                <%= Html.TextBox("family", Model.family) %>
                <%= Html.ValidationMessage("family", "*") %>
            </p>
            <p>
                <label for="tel">tel:</label>
                <%= Html.TextBox("tel", Model.tel) %>
                <%= Html.ValidationMessage("tel", "*") %>
            </p>
            <p>
                <input type="submit" value="Save" />
            </p> </fieldset> <% } %><div>
        <%=Html.ActionLink("Back to List", "Index") %>
    </div></asp:Content>
```



ظاهر صفحه edit هم براحتی و با فشار چند کلیک ساخته شد. دوباره به صفحه معروف homecontroller می رویم . تابع edit اول باید به شکل زیر باشد:

```
public ActionResult Edit(int id)
{
    var movieToDelete = test.Movies.First(m => m.id == id);
    ViewData.Model = movieToDelete;
    return View();
}
```

متغیر movieToDelete در این کدها حاوی id سطری خواهد شد که بر روی دکمه edit آن سطر کلیک می کنیم . پس از ان اطلاعات سطر متناظر با id انتخاب شده را درون فیلدهای صفحه edit.aspx قرار می دهد و سپس کاربررا به صفحه edit.aspx هدایت می کند.

حالا باید کدهایی را وارد نماییم تا دکمه save را در صفحه edit.aspx عملیاتی کند . در صفحه homecontroller تابع دوم Edit را به صورت زیر درمی آوریم:

```
public ActionResult Edit(FormCollection form)
{
    try
    {
        var id= Int32.Parse(form["id"]);
        var movieToUpdate = test .Movies.First(m => m.id == id);

TryUpdateModel(movieToUpdate,new string[]{"name","family","TEL"},form.ToValueProvider());

        test.SubmitChanges();
        return RedirectToAction("Index");
        return View(movieToUpdate);
    }
    catch
    {
        return View();
    }
}
```

صفحه edit هم ساخته شد . دیدید که با تکنولوزی mvc می توان صفحاتی ساخت که برنامه نویس نیاز به نوشتن کد های غیر ضروری ندارد.

با انجام کارهای فوق کدهای صفحه homecontroller بصورت زیر خواهد شد :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Mvc.Ajax;
using MvcApp.Models;

namespace MvcApp.Controllers
{
    public class homeController : Controller
    {
        //
        // GET: /home/
        private  linq_classDataContext test = new linq_classDataContext();
        public ActionResult Index()
        {
            return View(test.Movies.ToList());
```

```
        }

        //
        // GET: /home/Details/5

        public ActionResult Details(int id)
        {
            return View();
        }

        //
        // GET: /home/Create

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

        //
        // POST: /home/Create

        [AcceptVerbs(HttpVerbs.Post)]
        public ActionResult Create([Bind (Exclude="id")]Movy mtc)
        {
            try
            {
                // TODO: Add insert logic here
                test.Movies.InsertOnSubmit(mtc);
                test.SubmitChanges();
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /home/Edit/5

        public ActionResult Edit(int id)
        {
            var mtu = test.Movies.First(m => m.ID == id);
            ViewData.Model = mtu;
            return View();
        }

        //
        // POST: /home/Edit/5

        [AcceptVerbs(HttpVerbs.Post)]
        public ActionResult Edit(FormCollection form)
        {
            try
            {
                // TODO: Add update logic here
                var id = Int32.Parse(form["id"]);
                var movieToUpdate = test.Movies.First(m => m.ID == id);

                // Deserialize (Include white list!)

                TryUpdateModel(movieToUpdate, new string[] { "name", "family", "TEL" },
form.ToValueProvider());
                test.SubmitChanges();
                return RedirectToAction("Index");

                // Otherwise, reshow form
```

```
                return View(movieToUpdate);
            }
            catch
            {
                return View();
            }
        }
        public ActionResult Delete(int id)
        {
            var movieToDelete = test.Movies.First(m => m.ID == id);

            // Delete
            test.Movies.DeleteOnSubmit(movieToDelete);
            test.SubmitChanges();

            // Show Index view
            return RedirectToAction("Index");
        }
    }
}
```

پایان

منابع : www.asp.net ، www. softprojects.org

جهت دریافت فایل های برنامه و همچنین فیلم آموزشی این فایل به آدرس http://itn88.blogfa.com/post-258.aspx مراجعه نمایید .

لطفاً نواقص و ایرادات این کتاب را با آدرس aghili65@gmail.com و یا http://itn88.blogfa.com/post-258.aspx مکاتبه نمایید .

با تشکر از شما

لطفاً اگر مطالب این کتاب به شما کمکی کرده است من و پدر مرحومم را از دعای خیرتان محروم نفرمایید .

جهت سلامتی و فرج امام زمان صلوات

ابوالفضل عقیلی کوهستانی – *مشهد مقدس*

بهار ۱۳۸۹