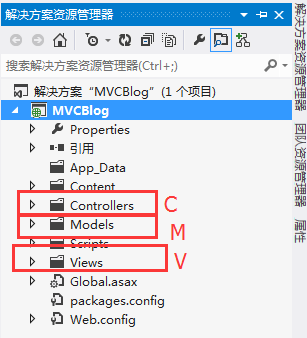
[MVC+LINQ+EF实战（.NET平台）](http://doc.okbase.net/u010929604/archive/124443.html)

u010929604 2015-1-19 8:18:48 阅读(3959) [评论](http://doc.okbase.net/u010929604/archive/124443.html#comments)(0)

    学习软件开发的重点是动手实践，只是站在河边学习动作要领和游泳理论，是永远学不会游泳的，你一定要下水试试。这次实例练习主要涉及了三个核心知识，一个是MVC框架，一个是EntityFramework，还有一个就是Linq。

    第一步：新建项目，创建一个基于MVC的应用程序，如下图所示：

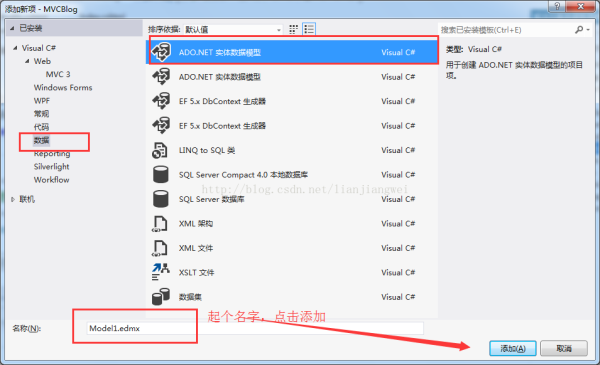


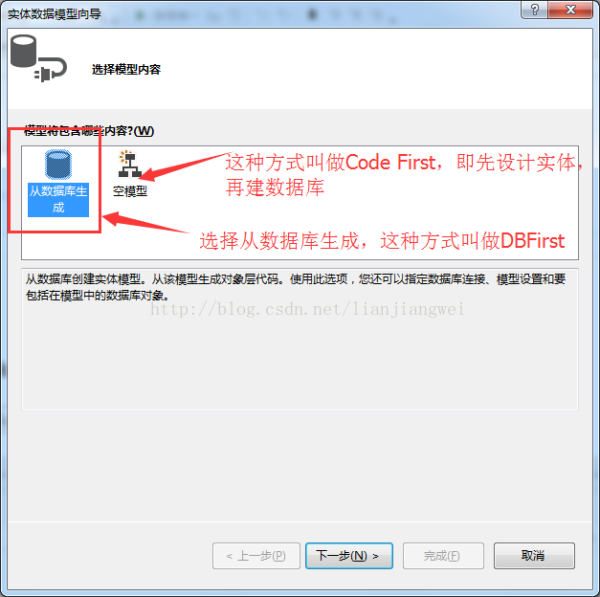
    第二步：在Model层创建edmx文件及其附属类，这里要用到EF。简单来介绍一下EF吧，EF就是EntityFrameWork，即ORM（ObjectRelationalMapping）对象关系数据库映射框架，根据实体对象操作数据表中数据的一种面向对象的操作框架。下面我们来看如何用实体对象映射数据库表的数据：

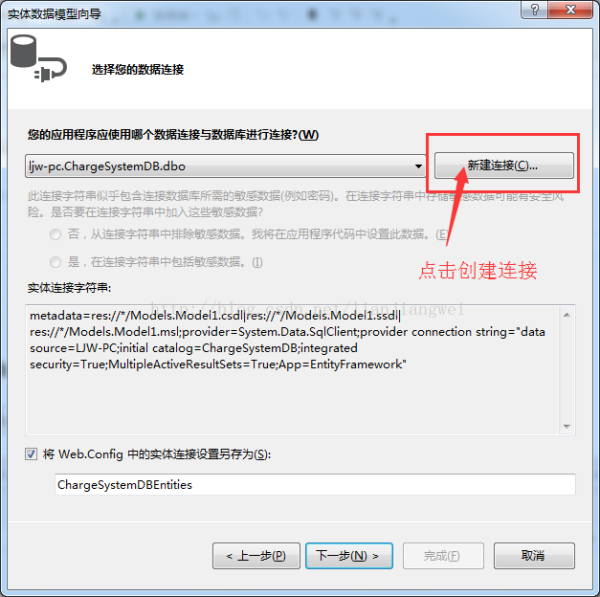
    在我的SQL Server中有事先建好的几张表如下图所示，我们主要对news表进行操作

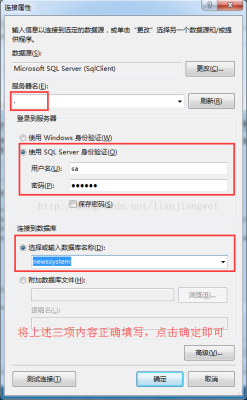


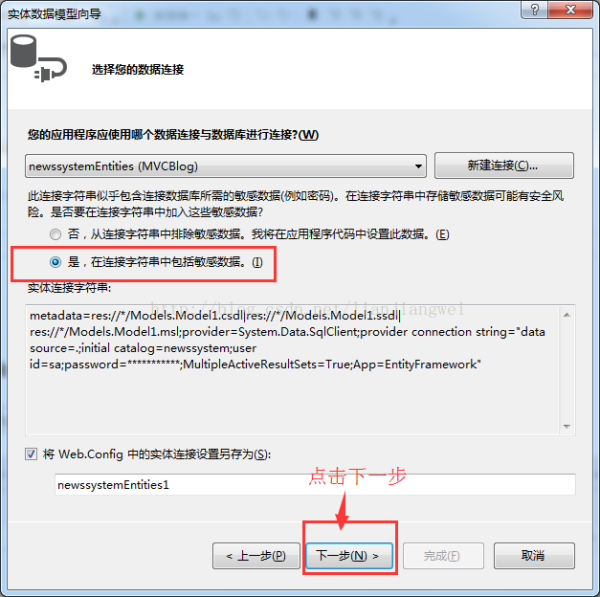
    在Models文件夹右击添加新建项，在对话框的左边列表中选择数据，之后在右边的列表中选择ADO.NET实体数据模型，如下图所示：

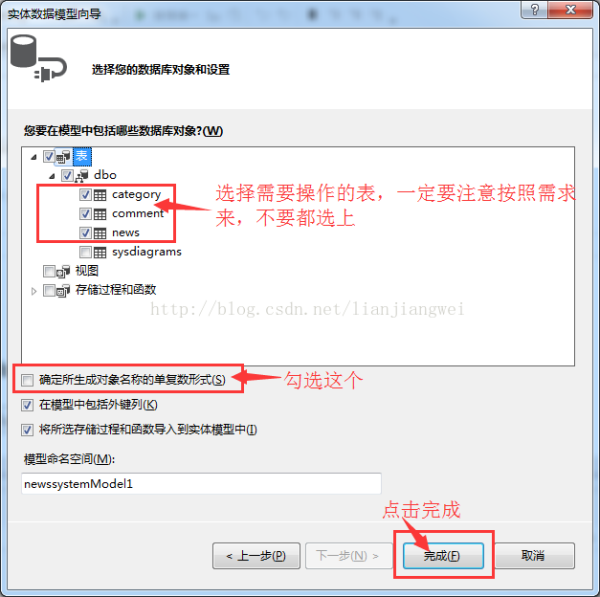




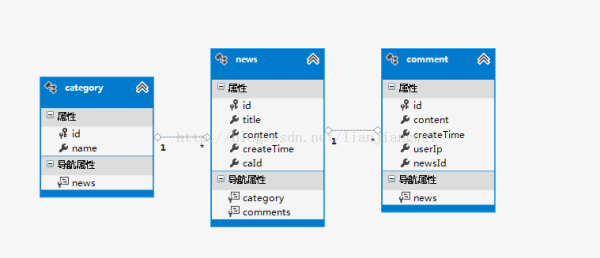


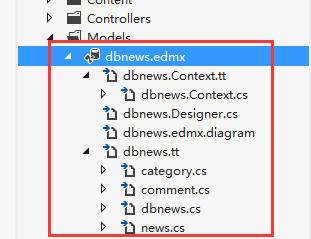






    经过上述所示的一系列操作，我们会看到系统为我们创建的edmx文件的图形化显示以及文件组织结构如下图





    在解决方案资源管理器中的Models文件下可以看到这个文件及其子文件的组织结构，具体每个文件的作用和原理在这里不再说明，大家感兴趣可以自己研究。

    第三步：进行代码编写，主要是对Controller进行设计和编码，需要写代码的主要有三个文件，分别是HomeController.cs、Index.cshtml和Modify.cshtml，分别给出如下：

         HomeController.cs的代码如下：

<span style="font-size:18px;">namespace MVCBlog.Controllers

{

public class HomeController : Controller

{

/// <summary>

/// 数据上下文对象

/// </summary>

newssystemEntities db = new newssystemEntities();

#region 查询新闻列表+ActionResult Index()

/// <summary>

/// 查询新闻列表

/// </summary>

/// <returns></returns>

public ActionResult Index()

{

//1.查询数据库里的新闻数据（通过EF执行）

//1.1第一种方法：使用SQO（标准查询运算符）,查询所有新闻

//db.news.Where(d =>d.id !=0 );

//List<Models.news> list = db.news.Where(d => d.id != 0).ToList ();

//1.2第二种方法：使用linq语句，查询所有新闻标题

//Linq仅仅是给程序员使用的语法糖，.net编译器会在编译时将linq转化为sqo

List <Models.news> list = (from d in db.news where d.id != 0 select d).ToList ();

//2.将数据集合传给视图

ViewData["datalist"] = list;

//3.加载视图

return View();

}

#endregion

#region 执行删除操作（根据id）+ActionResult Del(int id)

/// <summary>

/// 执行删除操作（根据id）

/// </summary>

/// <param name="id">要删除的新闻id</param>

/// <returns></returns>

public ActionResult Del(int id)

{

try

{

//1.创建要删除的对象

news modelDel = new news() { id = id };

//2.将对象添加到EF管理容器中

db.news.Attach(modelDel);

//3.将对象包装类的状态标识为删除状态

db.news.Remove(modelDel);

//4.更新到数据库

db.SaveChanges();

//5.更新成功，则令浏览器跳转到list方法

return RedirectToAction("Index","Home");

}

catch (Exception ex)

{

return Content("删除失败！"+ex .Message );

}

}

#endregion

#region 显示要修改的数据（根据id）+ActionResult Modify(int id)

[HttpGet ]

/// <summary>

/// 执行修改操作（根据id）

/// </summary>

/// <param name="id">要修改的新闻id</param>

/// <returns></returns>

public ActionResult Modify(int id)

{

try

{

//根据id查询数据库，返回集合中，拿到第一个实体对象

news n = (from a in db.news where a.id == id select a).FirstOrDefault();

//生成分类下拉框列表集合List<SelectListItem> list

IEnumerable <SelectListItem> listitem = (from c in db.categories select c ).ToList (). Select (c=> new SelectListItem {Value =c.id .ToString (),Text =c.name });

ViewBag.CateList = listitem;

//将n传递给视图显示 viewbag 或者viewdata

//加载视图，使用view的构造函数 将数据传给视图上的名为model的属性

return View(n);

}

catch (Exception ex)

{

return Content("修改失败！" + ex.Message);

}

}

#endregion

#region 执行修改操作+ActionResult Modify(news model)

[HttpPost ]

/// <summary>

/// 执行修改操作

/// </summary>

/// <param name="model"></param>

/// <returns></returns>

public ActionResult Modify(news model)

{

try

{

//将实体对象加入到EF对象容器中，并获取伪包装类对象

DbEntityEntry<news> entry = db.Entry<news>(model);

//将包装类对象的状态设置为unchanged

entry.State = System.Data.EntityState.Unchanged;

//设置需要提交的实体属性

entry.Property(a => a.title).IsModified = true;

entry.Property(a => a.content).IsModified = true;

//提交到数据库 完成修改

db.SaveChanges();

//5.更新成功，则令浏览器跳转到list方法

return RedirectToAction("Index", "Home");

}

catch (Exception ex)

{

return Content("修改失败！" + ex.Message);

}

}

#endregion

}

}</span>

         Index.cshtml的代码如下：

<span style="font-size:18px;">@using MVCBlog.Models

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<title>Index</title>

<style type ="text/css" >

#tblist

{

border :1px solid #0ff;

width :800px;

margin :10px auto ;

border-collapse :collapse ;

}

#tblist th,td

{

border :1px solid #0ff;

padding :10px;

}

</style>

<script type ="text/javascript" >

function Del(id) {

alert("运行到了这里");

if (confirm("您确定要删除吗？亲~~")) {

window.location="/Home/Del/" + id;

}

}

function Modify(id) {

window.location = "/Home/Modify/" + id;

}

</script>

</head>

<body>

<table id ="tblist" >

<tr>

<th >id</th>

<th >标题</th>

<th >发布时间</th>

<th >新闻分类</th>

<th >操作</th>

</tr>

<!--遍历Action方法 设置给viewdata的数据集合，生成html代码-->

@foreach (news n in ViewData["datalist"] as List<news>)

{

<tr>

<td>@n.id </td>

<td >@n.title </td>

<td>@n.createTime </td>

<td >@n.category.name </td>

<td >

<a href ="javascript:Del(@n.id) ">删除</a>

<a href ="javascript:Modify(@n.id )">修改</a>

</td>

</tr>

}

</table>

</body>

</html></span>

         Modify.cshtml的代码如下:

<span style="font-size:18px;">@model MVCBlog .Models .news

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<title>修改</title>

<style type ="text/css" >

#tblist

{

border :1px solid #0ff;

width :600px;

margin :10px auto ;

border-collapse :collapse ;

}

#tblist th,td

{

border :1px solid #0ff;

padding :10px;

}

</style>

</head>

<body>

@using (Html.BeginForm("Modify", "Home", FormMethod.Post))

{

<table id="tblist">

<tr>

<td colspan ="2">修改 @Html.HiddenFor(a =>a.id)</td>

</tr>

<tr>

<td >标题：</td>

@\*<td >@Html.TextBox("txtName",(object )Model.title)</td>\*@

<!--使用htmlhelper的强类型方法直接从model中根据title属性生成文本框-->

<td>@Html.TextBoxFor(a=>a.title )</td>

</tr>

<tr>

<td>分类：</td>

<td >@Html.DropDownListFor(a=>a .category ,ViewBag.CateList as IEnumerable <SelectListItem >)</td>

</tr>

<tr>

<td >内容：</td>

<td>@Html.TextAreaFor(a=>a.content,10,60,null )</td>

</tr>

<tr>

<td colspan="2"><input type ="submit" value ="确定修改" />@Html.ActionLink("返回","Index","Home")</td>

</tr>

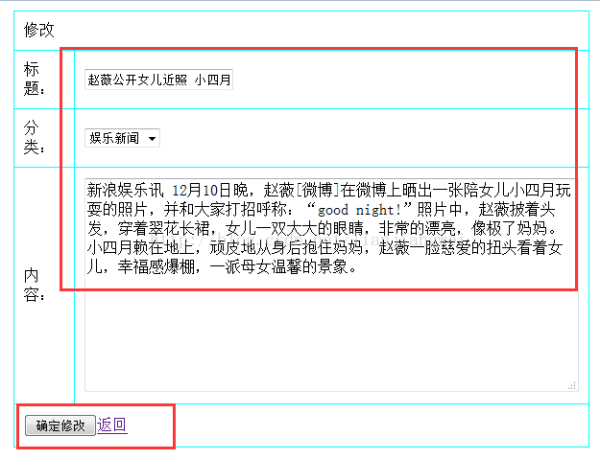
</table>

}

</body>

</html></span>

    第四步，运行程序，我们要实现的是对news表的查询、删除和修改，通过主视图Index来显示news表中的所有新闻，然后通过链接进行新闻的删除和修改，运行的截图如下



    最后提一下这个Linq，没听说过的小伙伴们可能会有困惑。所谓Linq就是语言集成查询， 是 Visual Studio 2008 和 .NET Framework 3.5版中引入的一项创新功能，它在对象领域和数据领域之间架起了一座桥梁。Linq仅仅是给程序员使用的语法糖，.net编译器会在编译时将linq转化为SQO，而这个SQO是定义在System.Linq.Enumerable类中的50多个为IEnumerable<T>准备的扩展方法,这些方法用来对它操作的集合进行查询筛选，所以说我们本质上是在使用SQO进行查询。

    总结一下这个实例的要点，首先要熟悉MVC框架，这个一般没什么问题，其次要会创建实体框架，这个熟练了也没什么问题，在本例中我觉得最核心的东西是Controller的编写，那里要用到很多的知识，比如如何创建对象，将操作对象添加到EF管理容器中等。此外在分布视图中还要用到HtmlHelper对象的强类型方法直接根据model的属性生成html控件，当然这个强类型方法的很多参数还不太熟练，有待进一步的学习。