[**解析ASP.NET Mvc开发之删除修改数据**](http://blog.csdn.net/yisuowushinian/article/details/17854955)

**http://blog.csdn.net/yisuowushinian/article/details/17854955**

2014-01-04 23:41 12441人阅读 [评论](http://blog.csdn.net/yisuowushinian/article/details/17854955#comments)(3) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/yisuowushinian/article/details/17854955#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

ASP.NET（14） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

**目录：**

[**从明源动力到创新工场这一路走来**](http://blog.csdn.net/yisuowushinian/article/details/17649631)

[**解析ASP.NET WebForm和Mvc开发的区别**](http://blog.csdn.net/yisuowushinian/article/details/17646121)

[**解析ASP.NET 和Mvc开发之查询数据实例**](http://blog.csdn.net/yisuowushinian/article/details/17743407)

[**解析ASP.NET Mvc开发之EF延迟加载**](http://blog.csdn.net/yisuowushinian/article/details/17808123)

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

上边博客介绍了ASP[**.NET**](http://lib.csdn.net/base/dotnet) MVC使用EF来查询数据和EF中DbQuery<T>泛型对象对数据的延迟加载。今天我们就来看看我们怎么使用EF来删除数据。

其实现在的Web开发通用的模式就是前端使用Js和**[jQuery](http://lib.csdn.net/base/jquery" \o "jQuery知识库" \t "_blank)**来和后端进行数据交互。那么我们就在前端来加两个删除和更新的脚本。代码如下：

**[html]** [view plain](http://blog.csdn.net/yisuowushinian/article/details/17854955) [copy](http://blog.csdn.net/yisuowushinian/article/details/17854955)

[在CODE上查看代码片](https://code.csdn.net/snippets/142587)

1. <!--遍历 Action方法 设置给 ViewData 的集合数据，生成HTML代码-->
2. @foreach (BlogArticle a in ViewData["DataList"] as List**<BlogArticle>**)
3. {
4. **<tr>**
5. **<td>**@a.AId**</td>**
6. **<td>**@a.ATitle**</td>**
7. **<td>**@a.BlogArticleCate.Name**</td>**
8. **<td>**@a.Enumeration.e\_cname**</td>**
9. **<td>**@a.AAddtime**</td>**
10. <!---------为文章列表添加删除按钮--------->**>**
11. **<td><a** href="javascript:del(@a.AId)"**>**删除**</a></td>**
13. **</tr>**
14. }

然后为该按钮编写Js脚本函数，代码如下：

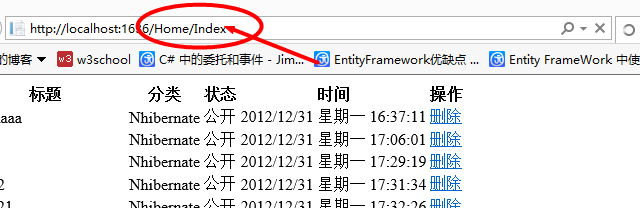
**[javascript]** [view plain](http://blog.csdn.net/yisuowushinian/article/details/17854955) [copy](http://blog.csdn.net/yisuowushinian/article/details/17854955)

[在CODE上查看代码片](https://code.csdn.net/snippets/142587)

1. <script type="text/javascript">
2. **function** del(id) {
3. **if** (confirm("确定要删除么？")) {
5. <!--这里配置当用户确定删除时，js让页面跳转到的的url地址-->
6. window.location="/home/del/"+ id;
7. }
8. }
9. </script>

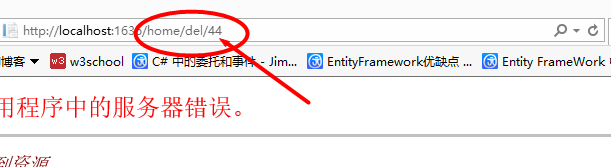
上面的Js代码的意思是，当用户点击删除按钮时，url跳转的地址是"/home/del"+id

点击前的效果如下图，注意url地址是



它代表的意思是，浏览器现根据路由，请求了控制器（Controller），控制器根据路由的配置， 返回视图方法，然后试图再把Html，Js等回传给浏览器。

点击删除按钮并确定，注意url地址是



说明我们配置的路由信息是“home/del”

所以我们在控制器的HomeController中添加一个del方法，来完成删除操作，代码如下：

首先我们先来看看MVC默认的路由表配置，在App\_Start文件夹下的RouteConfig.cs，代码如下：

**[csharp]** [view plain](http://blog.csdn.net/yisuowushinian/article/details/17854955) [copy](http://blog.csdn.net/yisuowushinian/article/details/17854955)

[在CODE上查看代码片](https://code.csdn.net/snippets/142587)

1. //路由表配置
2. **public** **class** RouteConfig
3. {
5. **public** **static** **void** RegisterRoutes(RouteCollection routes)
6. {
7. routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");
9. routes.MapRoute(
10. name: "Default",
11. url: "{controller}/{action}/{id}",
12. defaults: **new** { controller = "Home", action = "Index", id = UrlParameter.Optional }
13. );
14. }
15. }

我们可以发现ur的默认配置模式是"{controller}/{action}/{id}"

默认的配置信息是defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

特别注意的是id = UrlParameter.Optional是可选的。

所以我们应该在homeController.cs中创建一个del类型的Action方法，代码如下：

**[csharp]** [view plain](http://blog.csdn.net/yisuowushinian/article/details/17854955) [copy](http://blog.csdn.net/yisuowushinian/article/details/17854955)

[在CODE上查看代码片](https://code.csdn.net/snippets/142587)

1. //删除文章
2. **public** ActionResult Del(**int** id)
3. {
4. //操作数据库使用try...catch来捕捉异常
5. **try**
6. {
7. //创建删除对象
8. BlogArticle artTicle = **new** BlogArticle();
9. artTicle.AId = id;
10. //将删除对象添加到EF 对象管理容器
11. db.BlogArticles.Attach(artTicle);
12. //将对象包装类的状态标识为删除状态
13. db.BlogArticles.Remove(artTicle);
14. //更新到数据库
15. db.SaveChanges();
16. //更新成功后，页面跳转到Index页面
17. **return** RedirectToAction("Index", "Home");
18. }
19. **catch** (Exception ex)
20. {
21. **return** RedirectToAction("友好页面");
22. }
23. //return View();
24. }

这样就完成了数据的删除操作。

接下来我们就来完成修改的代码，我们先展示视图（View）代码如下

**[html]** [view plain](http://blog.csdn.net/yisuowushinian/article/details/17854955) [copy](http://blog.csdn.net/yisuowushinian/article/details/17854955)

[在CODE上查看代码片](https://code.csdn.net/snippets/142587)

1. <!--生成一个表单，并且指明表单提交方法，和路由-->
2. @using (Html.BeginForm("Modify", "Home", FormMethod.Post))
3. {
4. **<table** id="tbList"**>**
5. **<tr>**
6. **<td** colspan="2"**>**修改 @Html.HiddenFor(a=**>**a.AId) **</td>**
7. **</tr>**
8. **<tr>**
9. **<td>**标题：**</td>**
10. @\***<td>**@Html.TextBox("txtName",(object)Model.ATitle)**</td>**\*@
12. <!--使用HtmlHelper的强类型方法 直接 从 Model 中 根据 ATitle 属性生成文本框-->
13. **<td>**@Html.TextBoxFor(a=**>**a.ATitle)**</td>**
14. **</tr>**
15. **<tr>**
16. **<td>**分类：**</td>**
17. <!--使用强类型方法生成下拉框，并自动根据 model属性里的ACate值 设置 下拉框的默认选中项-->
18. **<td>**@Html.DropDownListFor(a=**>**a.ACate,ViewBag.CateList as IEnumerable**<SelectListItem>**)**</td>**
19. **</tr>**
20. **<tr>**
21. **<td>**内容：**</td>**
22. <!--使用HtmlHelper的强类型方法 直接 从 Model 中 根据 AContent 属性生成文本域-->
23. **<td>**@Html.TextAreaFor(a =**>** a.AContent, 10, 60, null)**</td>**
24. **</tr>**
25. **<tr>**
26. **<td** colspan="2"**><input** type="submit" value="确定修改" **/>** @Html.ActionLink("返回","Index","Home")**</td>**
27. **</tr>**
28. **</table>**
29. }

我们的后端代码改如何编写呢？

**[csharp]** [view plain](http://blog.csdn.net/yisuowushinian/article/details/17854955) [copy](http://blog.csdn.net/yisuowushinian/article/details/17854955)

[在CODE上查看代码片](https://code.csdn.net/snippets/142587)

1. //因为我们这个Action是在表达以post方式提交时执行的，所以加上标识
2. [HttpGet]
3. /// <summary>
4. /// 加载需要修改文章
5. /// </summary>
6. /// <param name="id">需要修改文章的Id</param>
7. /// <returns></returns>
8. **public** ActionResult EditArticle(**int** id)
9. {
10. //获取需要编辑文章，并且返回该实体对象的第一个元素
11. BlogArticle art = (from c **in** db.BlogArticles where c.AId == id select c).FirstOrDefault();
13. //我们把文章的分类做成一个下拉列表，并且给DropList的<option>赋值
14. IEnumerable<SelectListItem> seleListItem = (from a **in** db.BlogArticleCates where a.IsDel == **false** select a).ToList().Select(a => **new** SelectListItem { Value = a.Id.ToString(), Text = a.Name });
16. //返回List对象
17. ViewBag.CateList = seleListItem;
19. **return** View();
20. }

接下来就是执行修改的代码：

**[csharp]** [view plain](http://blog.csdn.net/yisuowushinian/article/details/17854955) [copy](http://blog.csdn.net/yisuowushinian/article/details/17854955)

[在CODE上查看代码片](https://code.csdn.net/snippets/142587)

1. [HttpPost]
2. /// <summary>
3. /// 执行修改的代码
4. /// </summary>
5. /// <param name="model"></param>
6. /// <returns></returns>
7. **public** ActionResult Modify(BlogArticle model)
8. {
9. **try**
10. {
11. //1.将实体对象 a.加入 EF 对象容器中，并 b.获取 伪包装类对象
12. DbEntityEntry<BlogArticle> entry = db.Entry<BlogArticle>(model);
13. //2.将包装类对象的状态设置为 unchanged
14. entry.State = System.Data.EntityState.Unchanged;
15. //3.设置 被改变的属性
16. entry.Property(a => a.ATitle).IsModified = **true**;
17. entry.Property(a => a.AContent).IsModified = **true**;
18. entry.Property(a => a.ACate).IsModified = **true**;
20. //4.提交到数据库 完成修改
21. db.SaveChanges();
22. //5.更新成功，则命令浏览器 重定向 到 /Home/List 方法
23. **return** RedirectToAction("Index", "Home");
24. }
25. **catch** (Exception ex)
26. {
27. **return** Content("修改失败~~~" + ex.Message);
28. }
29. }

到此我们就完成了使用MVC建立一个小站点的任务，而且也完成了对数据的增删改查的操作。

欢迎大家对本系列的文章提出宝贵意见。