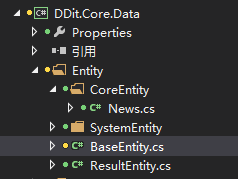
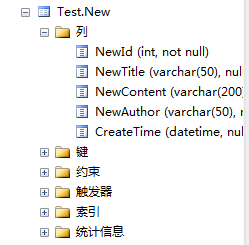
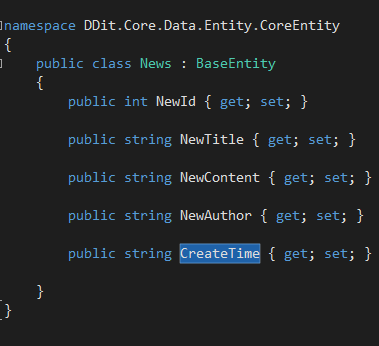
# DDit快速开发框架应用文档

从第一个页面展示开始一步一步完成我们自己的模块。一切接对象，所有模块的完成都是无数个对象开始的，那么就添加我们的实体类。

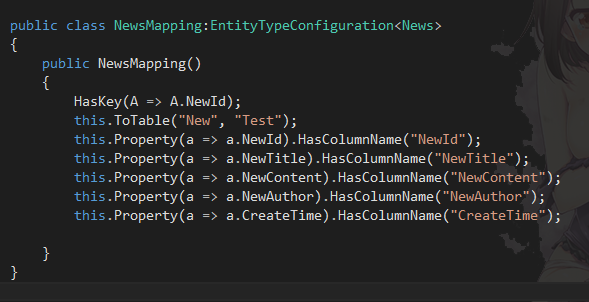


一．如图coreEntity 这个文件夹里面建立我的实体类 News.cs文件 就是我们测试用的类，SystemEntity就是我们框架系统配置的实体类，News.cs要继承BaseEntity.cs这个类，BaseEntity是一个公共基类里面放置了分页、排序等公共属性。同时数据库创建New表。

这里我不太喜欢用entity framework 生成数据库，因为会生成很多奇怪的东西，对于新手来说我建议你们手动来生成，注意我给New表设置了Test架构，这样做可以很容易的找到我们的表，使我们的数据库结构更加清晰，系统配置的表我都用的BASE.

注：这里还有一点SystemEntity下面有一个DoEntity文件夹，这里说明一下这个文件夹的作用。当你的业务逻辑要关联表的时候，不要把两个表所有的字段都一次返给前台页面，这样影响客户的浏览速度，所以这里面就是创建前台页面展示的属性类。同事这个里面的类也能解决前台排序的问题，具体可以看我的代码例子。

二．接下来我们要做的是数据库映射，其实就是把数据库的表字段跟我们的实体类关联起来，DDit.Core.Data.Repository -> Mapping -> CoreMapping -> 创建 NewsMapping.cs

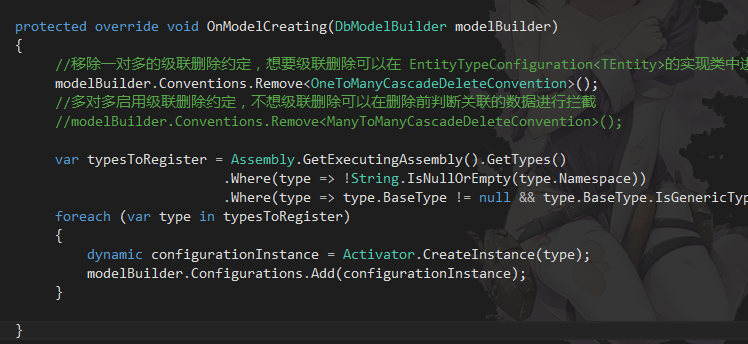


这的配置都是entity framework 基本的映射配置，如果不明白的同学赶紧去看看基础吧，这里我推荐一个人的博客。

[http://www.cnblogs.com/lsxqw2004/p/4701979.html#!comments](http://www.cnblogs.com/lsxqw2004/p/4701979.html%23!comments)

注：mapping 映射最重要的 就是HasKey（）这个方法是告诉我们的程序 你的表主键是那一个，增删改查每一步操作都是通过主键完成。 This.Totable（）这里我设置了test架构如果你的表架构是默认的dbo 可以不写。

接下来是理论上还应该把你的映射告诉数据源上下文（CoreDbContext.cs），但是这里本人用了反射 在编译的时候自动加载我们的Mappinged，所以你无须在写代码。

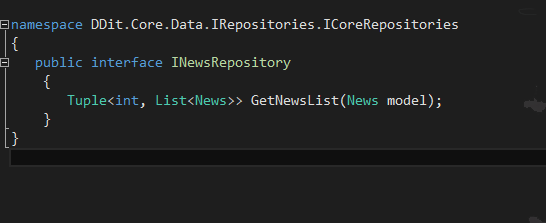


关于数据库的基本操作就到这里完成了，其实里面还有很多细节，会在下面的流程中讲解。

三．接下来就是逻辑操作了，建立仓储接口DDit.Core.Data –> IRepositories –> ICoreRepositories –> INewsRepository.cs 接口。这里的名字是有硬性要求的必须是“I”开头”Repository”作为结尾，因为我在Global.asax文件进行了动态依赖注入配置，用的第三方IOC容器 autofac 。关于autofac和依赖注入不明白的可以去看博客。

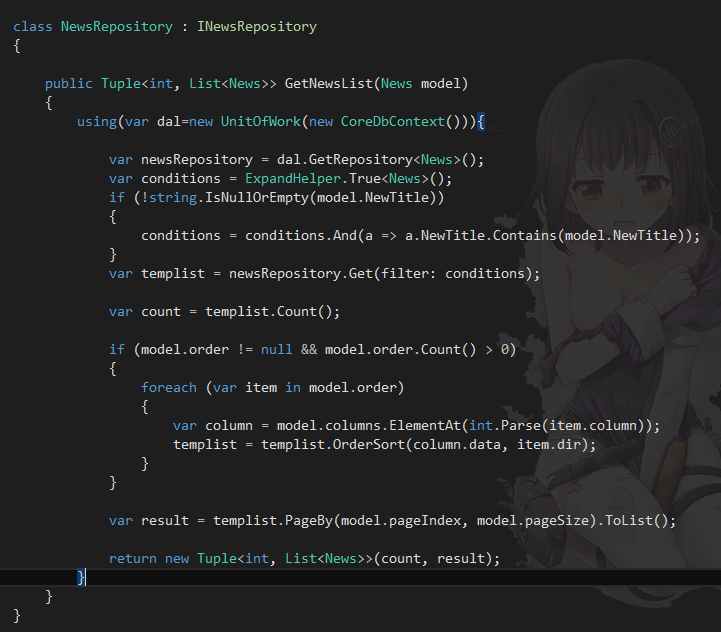
楼主推荐博客：<http://www.cnblogs.com/showjan/p/3950989.html> IOC讲解。

<http://www.cnblogs.com/WeiGe/p/3871451.html> autofac 应用。



这里本人添加了一个得到所有 数据的接口 返回的类型是 Tuple<int,List<News>> 因为方法要返回两个数据 一个是 集合数 count 一个是列表集合 都是为前台分页。

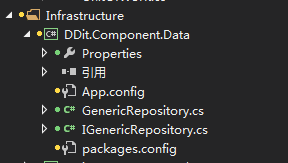
接下来创建仓储实现类 DDit.Core.Data.Repository –> Repositories –> CoreRepositories –> NewsRepository.cs 类 同样这个类命名也是有硬性要求跟接口一样只是去掉了前面的“I”.并且这个类继承上面的接口。



下面对代码进行讲解

using(var dal=new UnitOfWork(new CoreDbContext())){

这段是创建工作单元 unitofwork 参数代入了数据库上下文。Unitofwork的作用是给每一个表创建了增删改查基本的操作，楼主在entity framework的基础上又进行了二次封装使方法更加丰富。具体可以看GenericRepository.cs



var conditions = ExpandHelper.True<News>();

if (!string.IsNullOrEmpty(model.NewTitle))

{

conditions = conditions.And(a => a.NewTitle.Contains(model.NewTitle));

}

var templist = newsRepository.Get(filter: conditions);

这段代码是对条件的筛选 conditions 是创建一个条件链接类。最后把筛选条件放到我的过滤中 也就是最后一句 Get(filter:conditions).

注：Get()方法有三个参数 第一个过滤条件，第二个关联其他表，第三个字段排序，框架中有例子。

var count = templist.Count();

得到集合总数，为了分页

if (model.order != null && model.order.Count() > 0)

{

foreach (var item in model.order)

{

var column = model.columns.ElementAt(int.Parse(item.column));

templist = templist.OrderSort(column.data, item.dir);

} 这段代码是前台字段的排序 基本不用改动

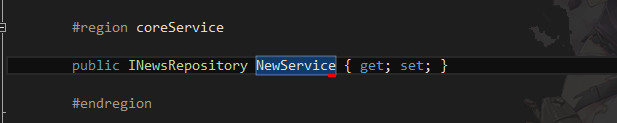
Varresult=templist.PageBy(model.pageIndex, model.pageSize).ToList();

分页并返回集合

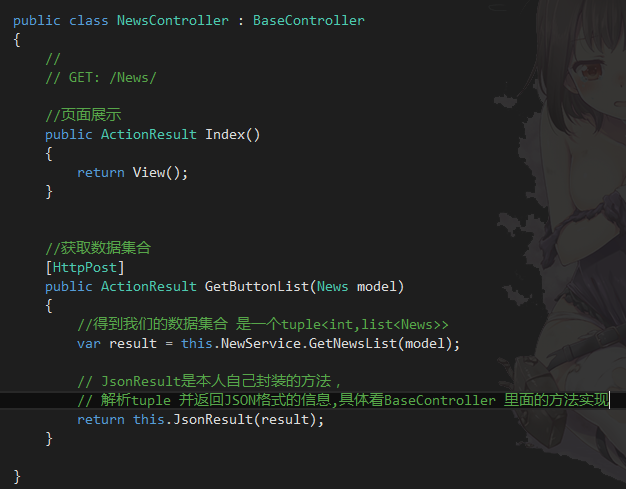
四． 接下来就是我们控制器Controllers 登场了 Controllers –>创建 NewsController.cs控制器

继承BaseController 类，BC类 里面 本人把业务逻辑仓储都注入到里面了，也就是说我们的控制器只要继承了BC 就可以调用逻辑仓储了。

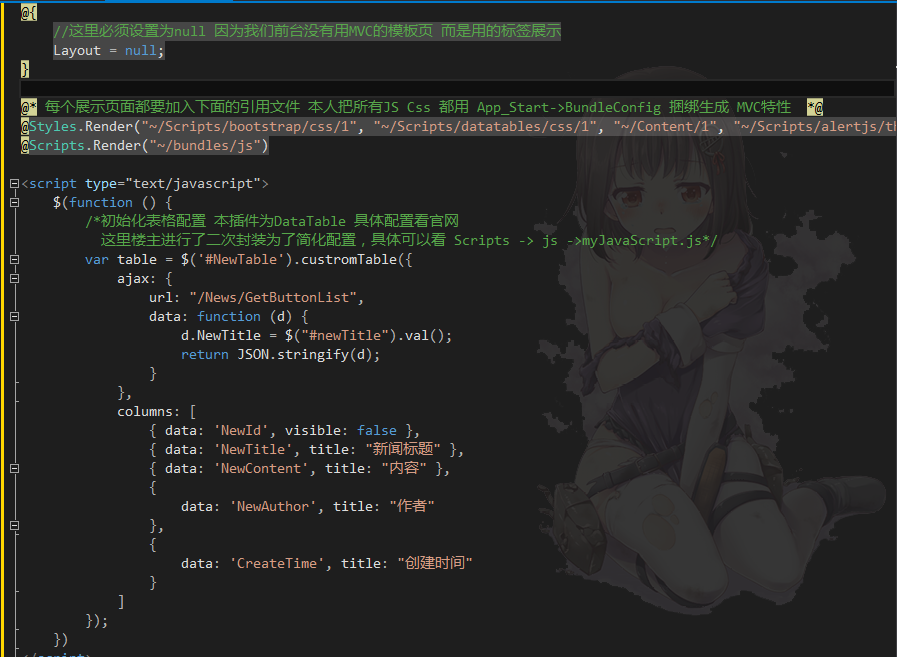
首先我们要完成bc 类里面News实体的注入如下图



很简单 就是 一个属性 是不是很神奇，接下来我们要在NewsController控制器中创建两个方法 一个是 视图展示，一个是获取News列表。



下面我们对View进行编写



图片中本人已经进行了注释解释就不在做说明了。



Html代码 就比较简单了 关于

@Html.Action("CreateButtonByMuen", "Button", new { menuId = 4, mark = "user" })

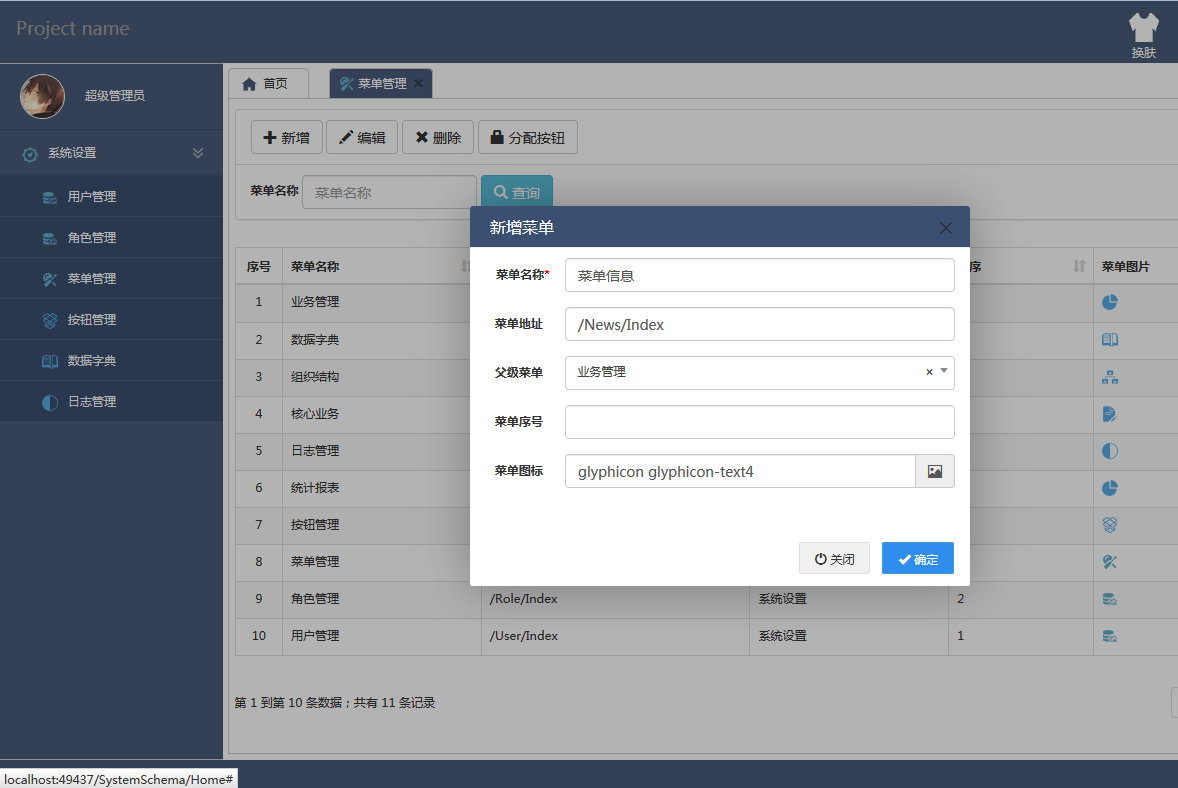
这句话就是按钮配置，会在后面按钮权限中讲解。

要想显示我们的新闻列表 要在菜单中新建菜单，首先我们建立一个一级菜单



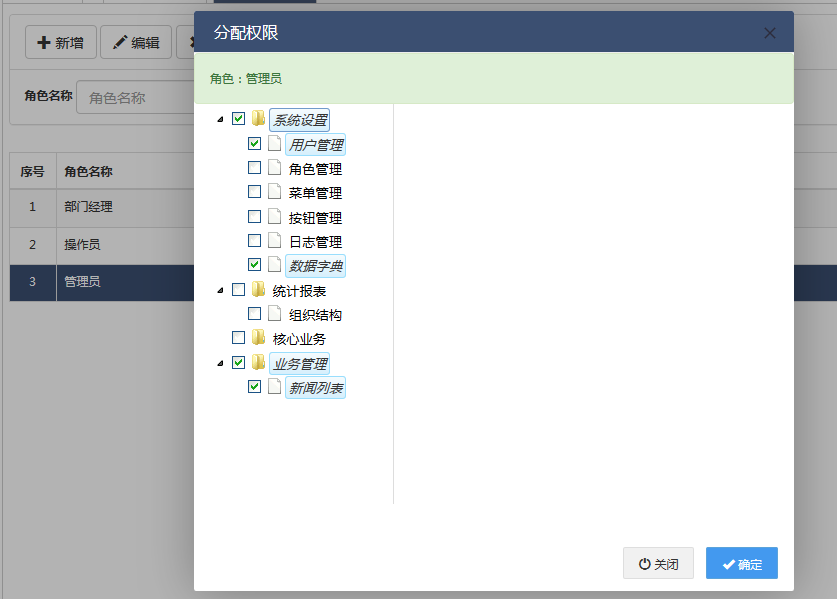
一级菜单 不需要菜单地址，也不需要父级菜单的选项序号也可以不填自动生成。

接下来就是建立我们的新闻信息菜单了。

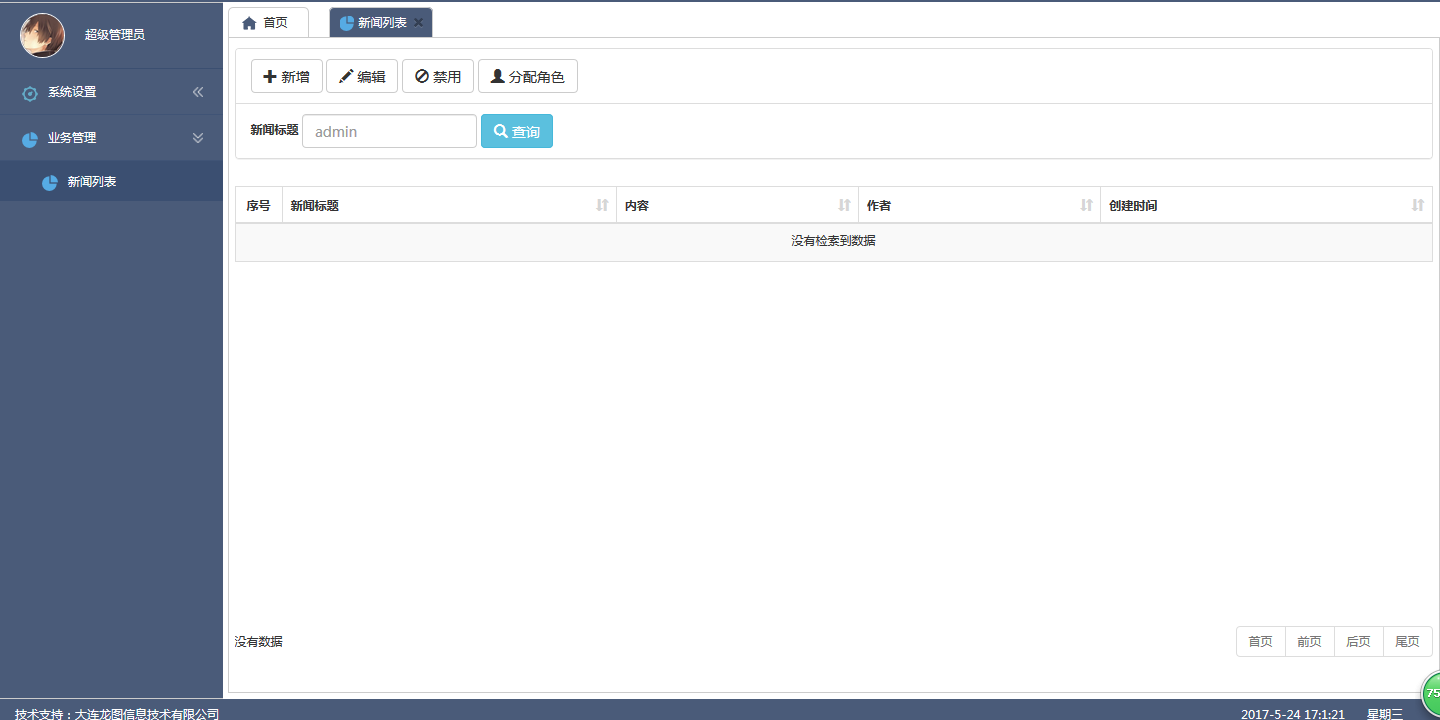


注：菜单地址就是咱们Controller 路径，父菜单选择我们刚才建立的业务管理，关于菜单框架最多支持三级菜单。

这个时候大家刷新后还没有发现我们新建的菜单不要着急，因为我们还没有给我们的角色分配菜单。下面我们给“管理员”这个角色分配权限



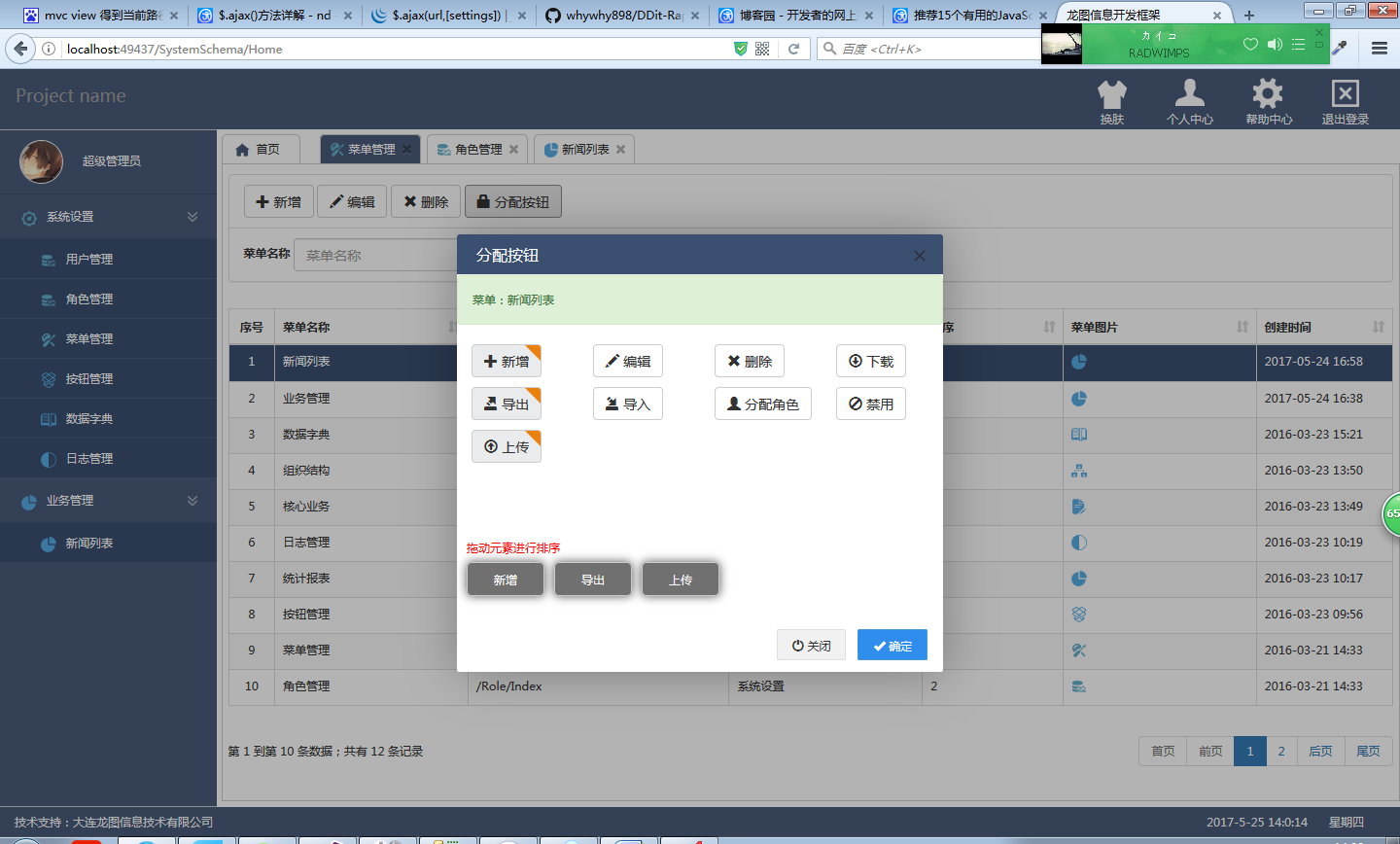
刷新网站就可以看到我们刚才生成的菜单了



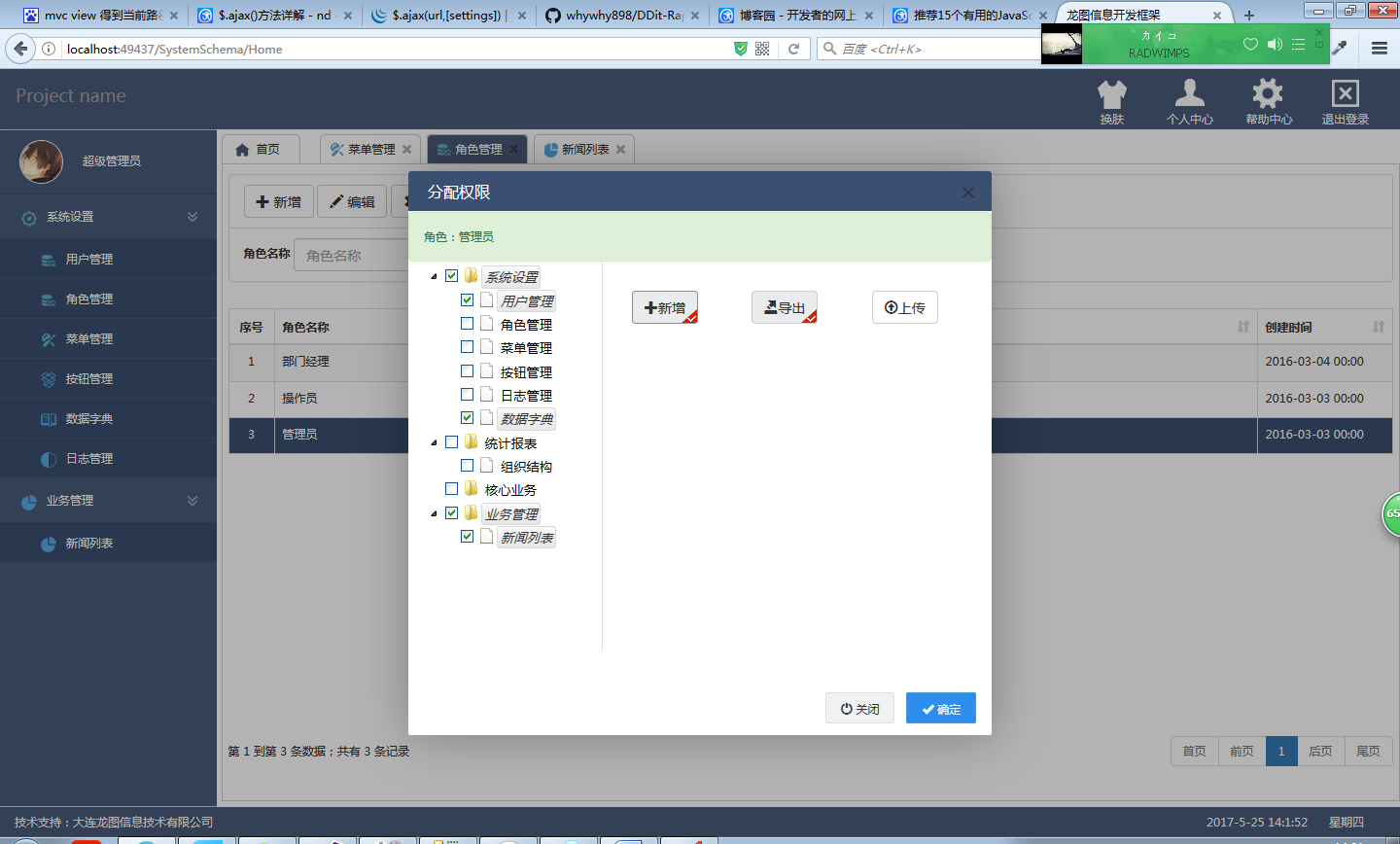
这样就生成了我的新闻列表页面，暂时列表里面还没有数据

五．角色分配按钮。

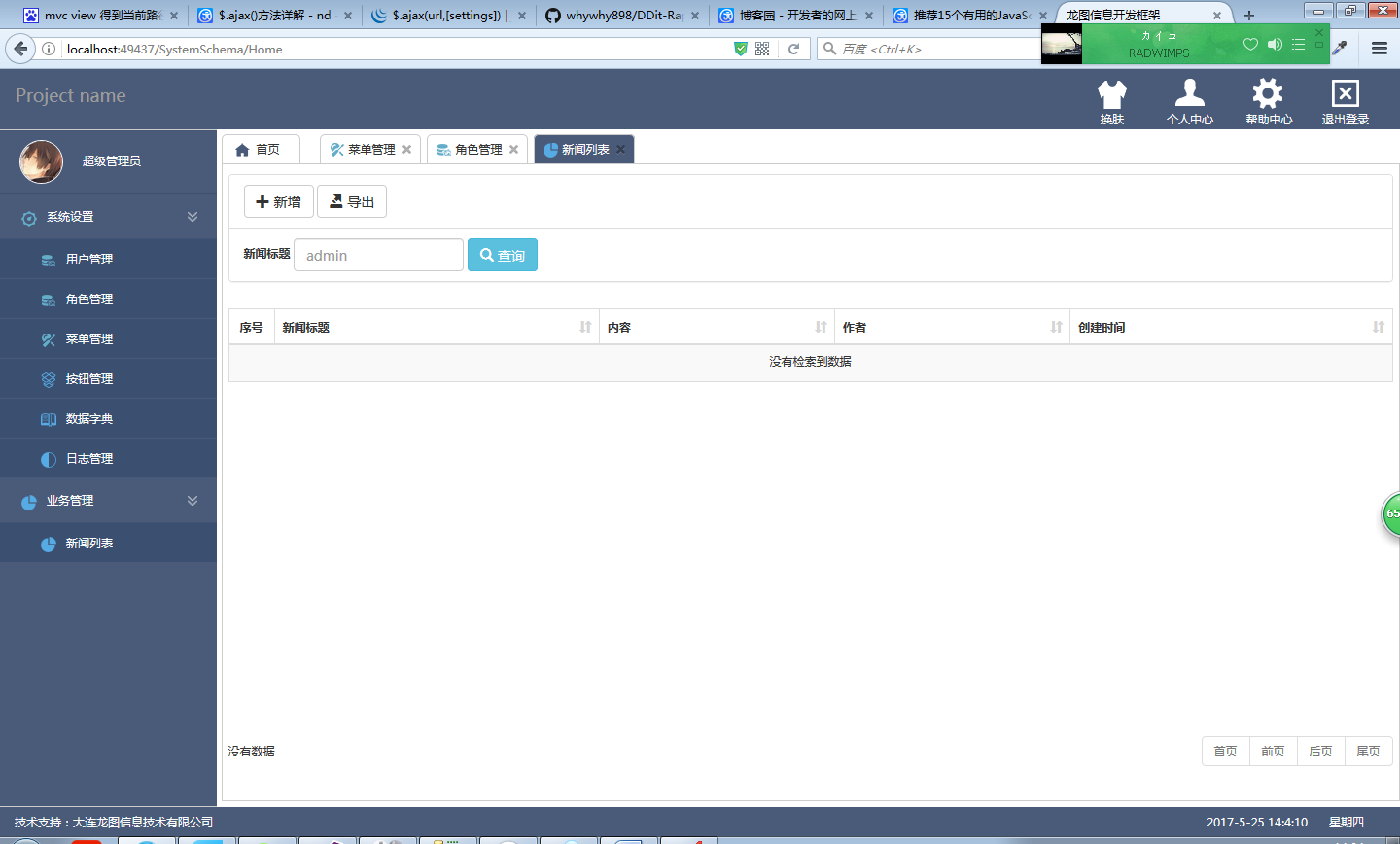
首先确定菜单中有那些按钮，我们给菜单先分配按钮



如图我们看到 给菜单分配了三个按钮 “新增”“导出”“上传”，接下来在角色页面具体指定按钮权限。



我们来到角色管理页面， 给 ‘管理员’的 “新闻列表” 菜单分配了新增和导出按钮，这时候刷新我们的页面会看到只有两个按钮显示。



接下来给大家讲解一下按钮分配的原理

@Html.Action("CreateButtonByMuen", "Button", new { menuId =46, mark = "news" })

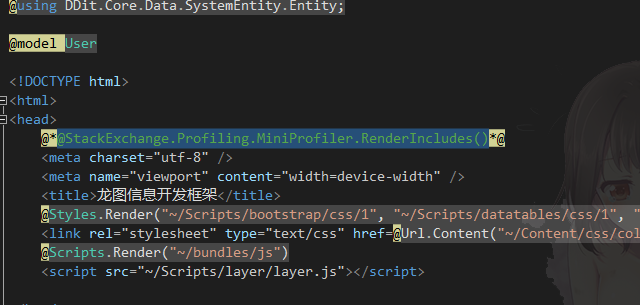
这句话关键的地方是后面的两个参数 menuId、mark 前者是菜单的ID 我们可以在数据库中查看，后者是按钮ID的前缀，用来添加事件。

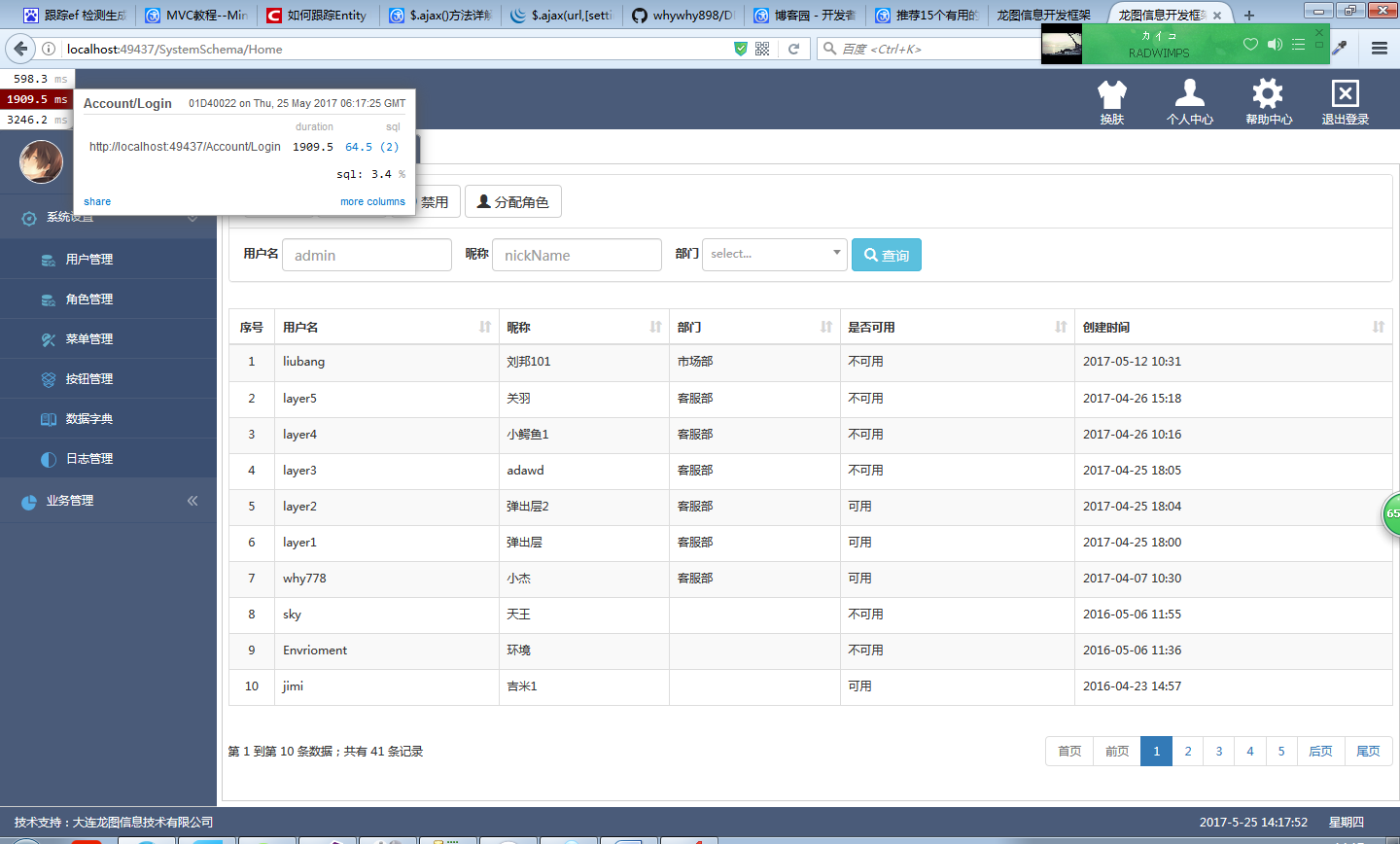


比如我们的新增按钮ID 就是 newsCreate,方法具体实现可以看/Button/CreateButtonByMuen方法。

六．件事我们生成的sql语句以便检测错误和提高性能。

**作者在框架中集成了[MiniProfiler.EF](http://www.cnblogs.com/tianboblog/p/4774810.html) 组件用来监控sql生成。我们只需要在Views –>** **Shared - > \_Layout.cshtml 页面** @StackExchange.Profiling.MiniProfiler.RenderIncludes() 这行代码接触注释就可以了





如图左上角，这个是开发调试的利器啊。