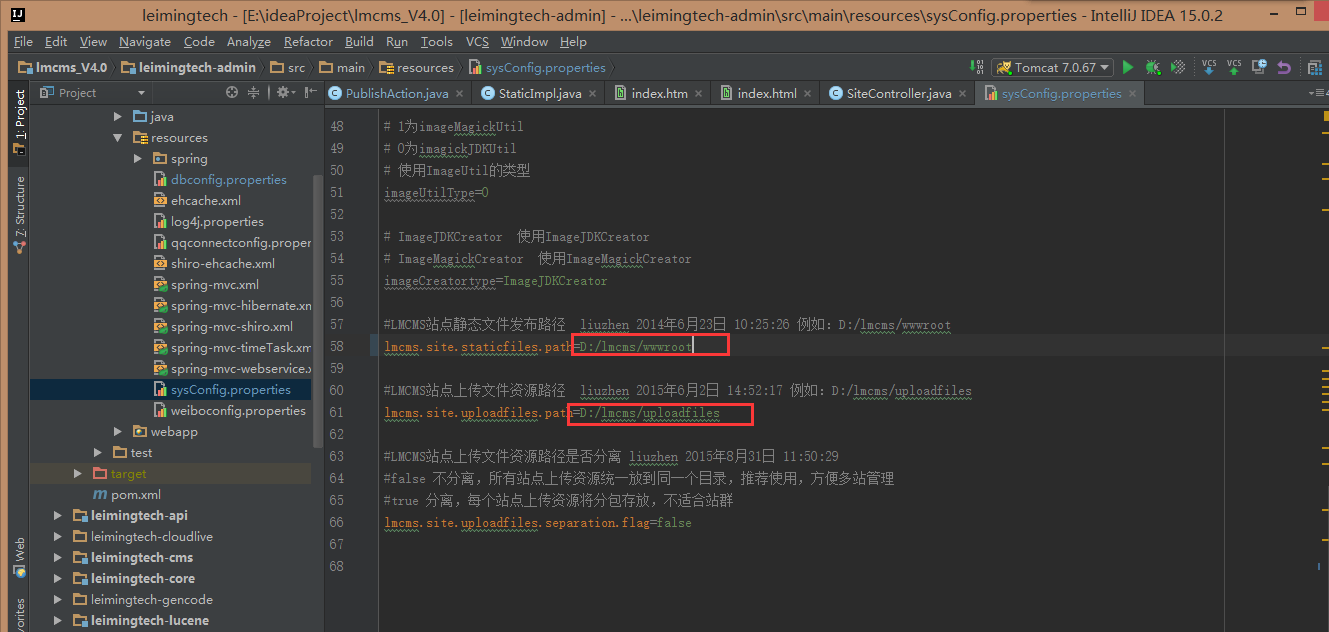
# LMCMS站点流程

# 1.站点建立步骤

## 设置静态页面路径

在sysConfig.properties 文件中配置生成静态页面路径



其中lmcms.site.staticfiles.path 为生成的静态页面地址,如果不填写会默认生成在服务器所运行的wwwroot下方.

例如

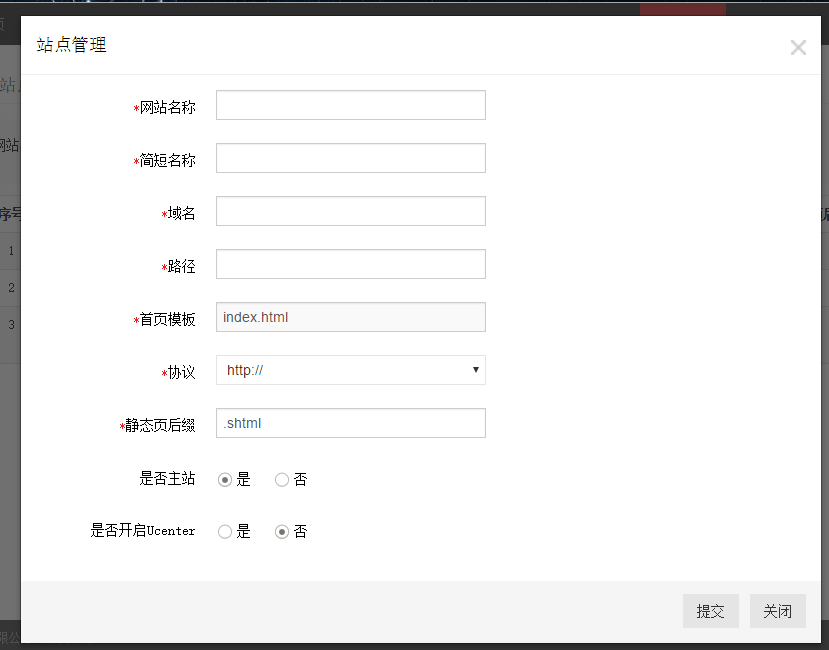
lmcms.site.staticfiles.path=D:/lmcms/wwwroot

lmcms.site.uploadfiles.path 为上传文件的地址

lmcms.site.uploadfiles.path=D: /uploadfiles

## 2建立站点

在站点管理点击新增建立一个站点



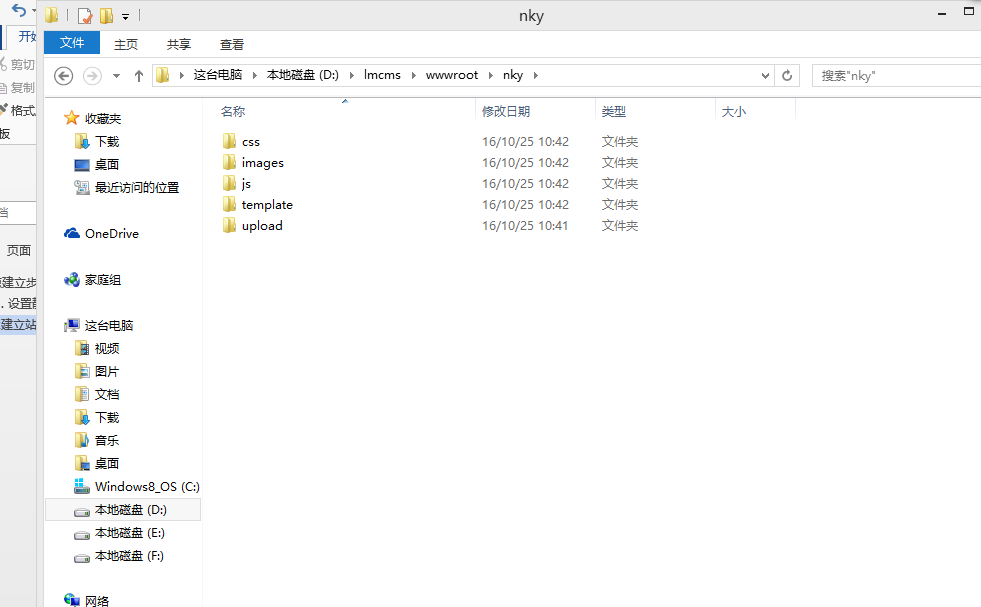
站点名称或简短名称可以为前台页面title的值

域名为你所访问的地址

路径为你创建的站点的地址

创建站点成功之后,你会发现在你所配置的lmcms.site.staticfiles.path 这个文件夹的下面会多出来一个文件夹,这个文件夹的名称为你站点的路径,

例如:我建立了一个农科院的站点,路径为nky



其中template文件夹为你所放模板的位置,

## 配置nginx

上述操作完成后,配置nginx

在nginx中的conf中找到nginx.conf文件夹进入

创建一个站点

upstream cms\_server {

server 127.0.0.1:8080;

} //匹配的服务器访问地址 ,

#农科院 //说明站点名称

server {

listen 84; //端口号

server\_name localhost; //访问的域名 //站点管理的域名要和它俩一致,为了进行预览访问时

#charset koi8-r;

#access\_log logs/host.access.log main;

location /leimingtech-admin/ #所有jsp的页面均交由tomcat处理

{

index Login.jsp;

proxy\_pass http://cms\_server/leimingtech-admin/; #转向tomcat处理 //后台访问地址

}

location /static #所有jsp的页面均交由tomcat处理

{

alias D:\lmcms\wwwroot\static; //它的位置为在lmcms.site.staticfiles.path 所配置的路径, static 这个文件在每个站点中都需要进行配置

}

location / {

root D:/lmcms/wwwroot/nky; //访问站点的路径,路径和上面一致,其中nky 为生成的站点的文件夹 ,该路径指向到站点的文件夹

index index.shtml index.html index.htm;

}

location /upload {

alias D:\fileServers\upload; //访问图片的路径,它的位置在lmcms.site.uploadfiles.path 所配置的路径

}

#error\_page 404 /404.html;

# redirect server error pages to the static page /50x.html

#

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

}

以上配置完成之后,启动nginx.

需要在角色管理找到该角色,分配该角色的站点,之后重新登录,

然后,建立栏目,,建立内容,最后进行发布

用在nginx配置的域名进行访问

例如

localhost:84

如果是在本地建立站点，需要服务器上面建立的站点，复制一份到自己的工程目录下的wwwroot下方，下次tomcat重新引入工程时，可以检索到

## 栏目的建立

点击PC栏目管理菜单后，选择左边栏目，栏目下面的相关的内容在列表中显示。



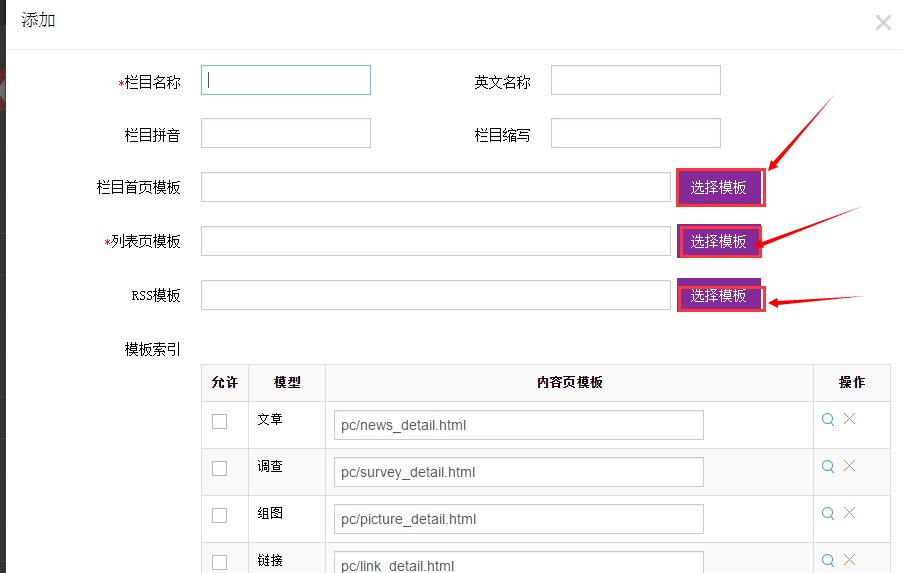
点击每个栏目时，列表显示所点栏目下面的栏目信息，当点击三级栏目时，显示当前栏目信息列表。列表数据与左边所选栏目下面的数据一致。



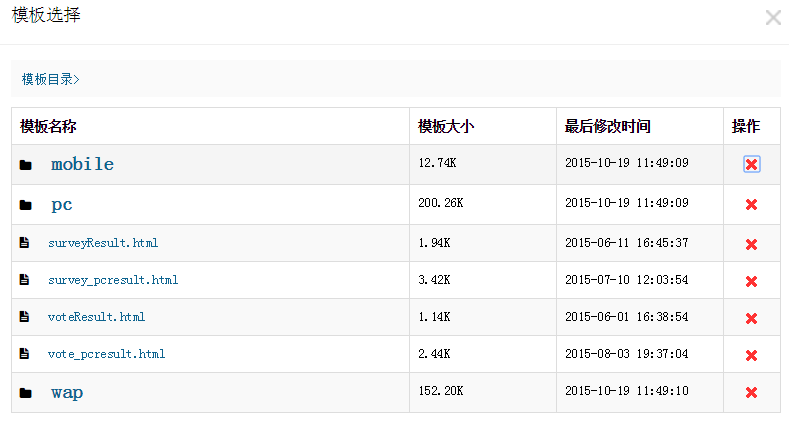
点击第三极栏目时，可直接对该栏目进行操作，包括添加子节点，批量添加修，修改，删除等。

添加子节点时，直接点击按钮，弹出一个子节点的添加页面。

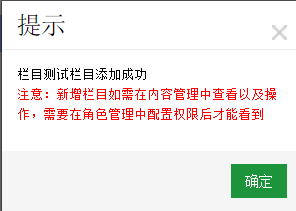
如下图所示。



点击添加页面上面的按钮，弹出列表页面，直接点击对应模板名称，如下图所示。点击选择后，会自动回到添加页面，信息录入完成后，直接点击“提交”按钮。完成添加子节点操作。



提交后，提示信息如图所示，说明添栏目成功。

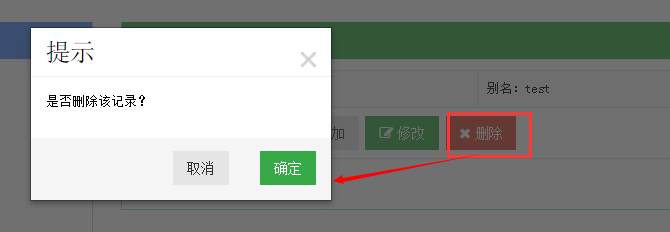


批量添加栏目，按钮，弹出“批量添加栏目”页面，操作和添加子节点类似。录入信息，选择模板列表后，直接点击提交即可。后面再细说。

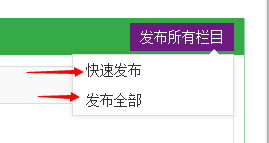


修改和添加一样，但是是在原来的栏目信息上做修改操作，也不再多说。

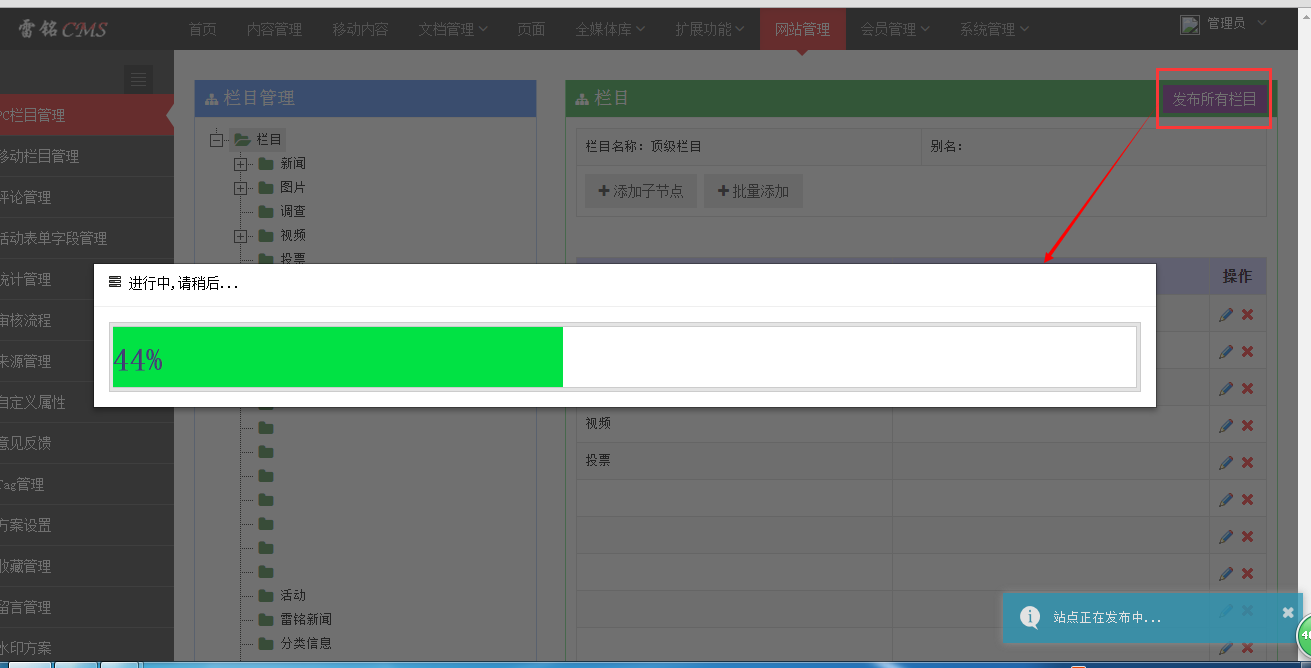
删除针对唯一的栏目列表，直接点击删除按钮，出现删除提示时，直接点击“确定”完成删除操作。如下图所示。如果不是最后一级栏目时，可点击刘表后面操作栏中的按钮，电梯弹出框上的确定按钮。完成删除操作。



发布栏目：将鼠标放在页面右上方的按钮，是，会有连个选择，快速发布或者打不全部，直接根据操作需要点击即可。



点击后，会显示发布进度，发布成功后会有发布成功提示信息，如下图所示。



如果需要栏目需要跳转链接,那么选择这个按钮,在链接地址里面写入要添加到的链接地址,进行发布栏目,前端的栏目就会变为访问链接地址



## 建立内容

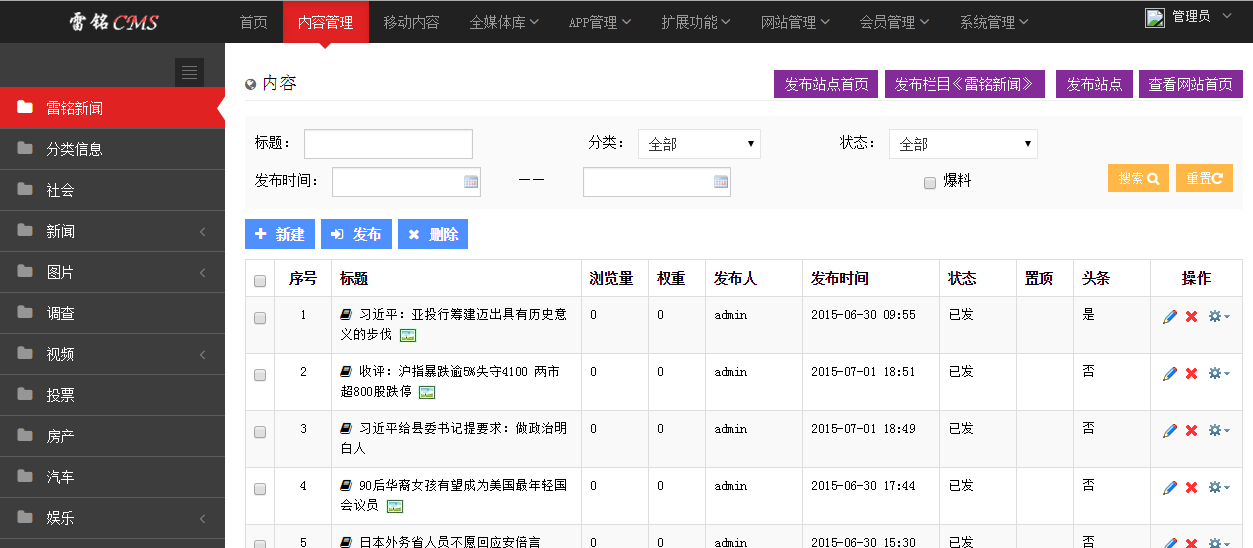
### 4.1.文章

作为CMS最核心的功能，不仅提供了高效快捷的内容录入和排版，还提供了发布后的内容推送、组织和传播。

* 一键排版，轻松调整稿件格式，提升编辑工作效率
* 可自定义属性和扩展字段，可通过属性进行查询和调用，满足个性化需求。
* 可推送至页面、专题、移动端，实现内容的精准推送和组织

允许编辑人员编辑并发布文章，查看并编辑处于已发、待发、已撤、待审、退稿、草稿、回收站等多种状态的文章内容。文章系统内置简洁而强大的编辑器，可以方便的对文章内容进行常规的排版操作，同时允许文章插入分页、特殊符号、标签和关键字链接、多媒体、百度地图、图片等内容。

依次点击<顶部主导航区>的[内容管理]→<左侧主菜单区>的栏目，打开<内容管理界面>。如图：



内容管理界面

鼠标移至蓝色【新建】按钮，选择下拉表单的“发文章”命令。如下图：



进入文章发布界面，如下图：



录入完成后页面展示如下：



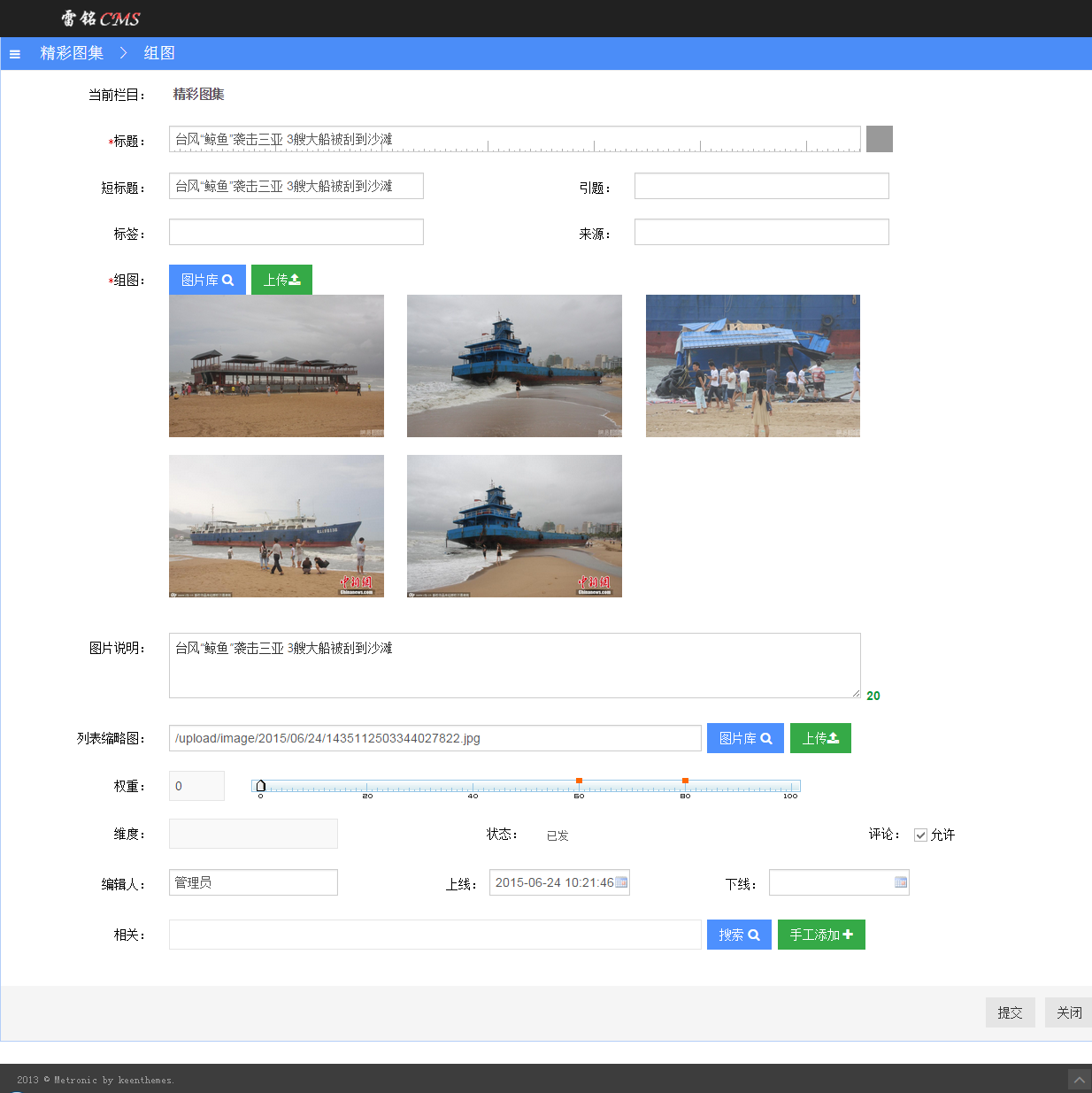
### 4.2.组图

提供高清幻灯组图功能，很适合大型活动以及以图片为主的新闻报道。

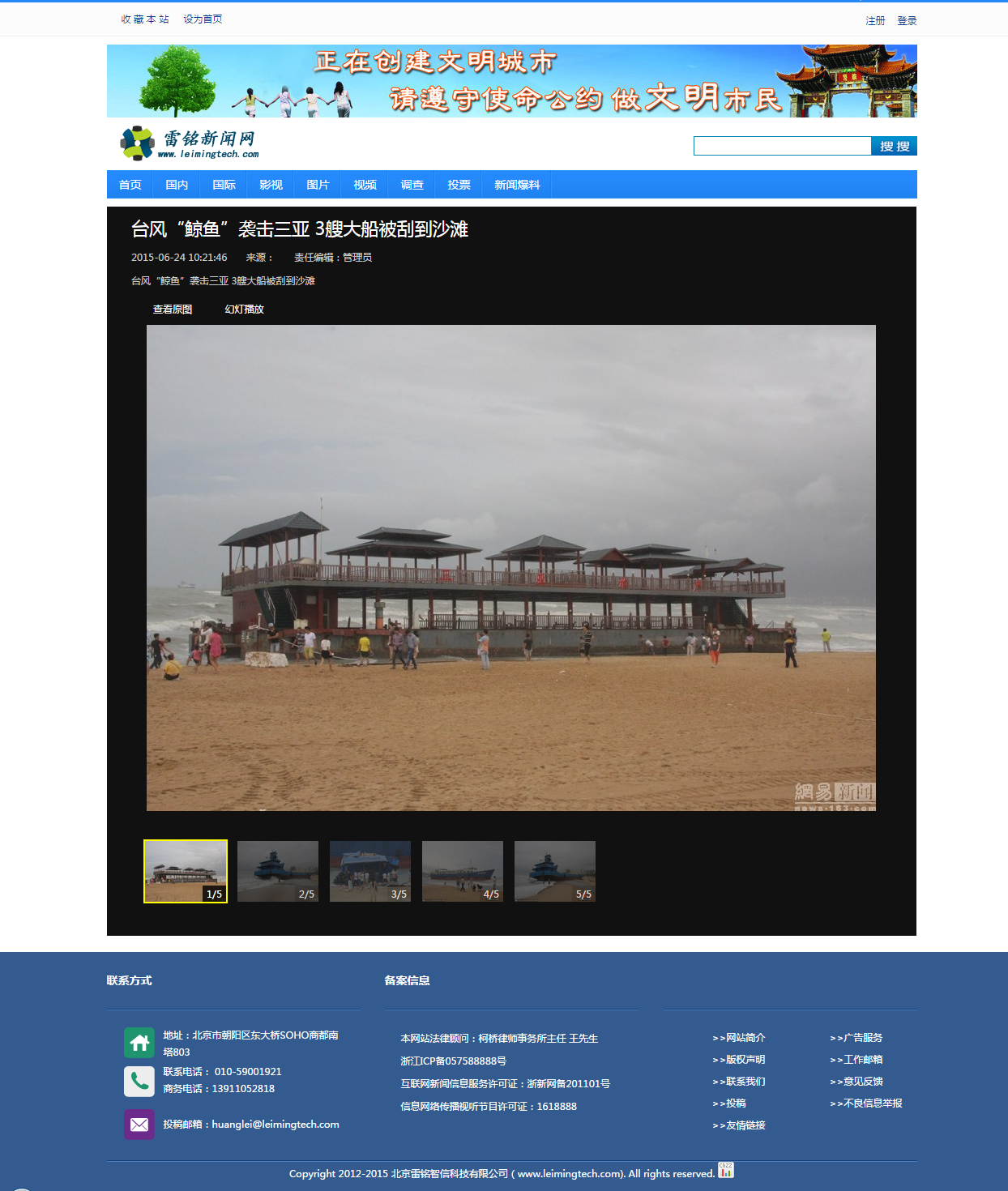
* 支持批量上传图片
* 支持类似门户网站高清幻灯组图前台无刷新展示效果
* 支持尾图播放完后推荐相关组图

允许编辑人员编辑并发布图片集，查看并编辑处于已发、待发、已撤、待审、退稿、草稿、回收站等多种状态的组图内容。系统内置多种图片上传方式，包括本地批量上传、图库。编辑人员可按需选择合适的上传方式。

组图录入界面如下：



页面展示如下：

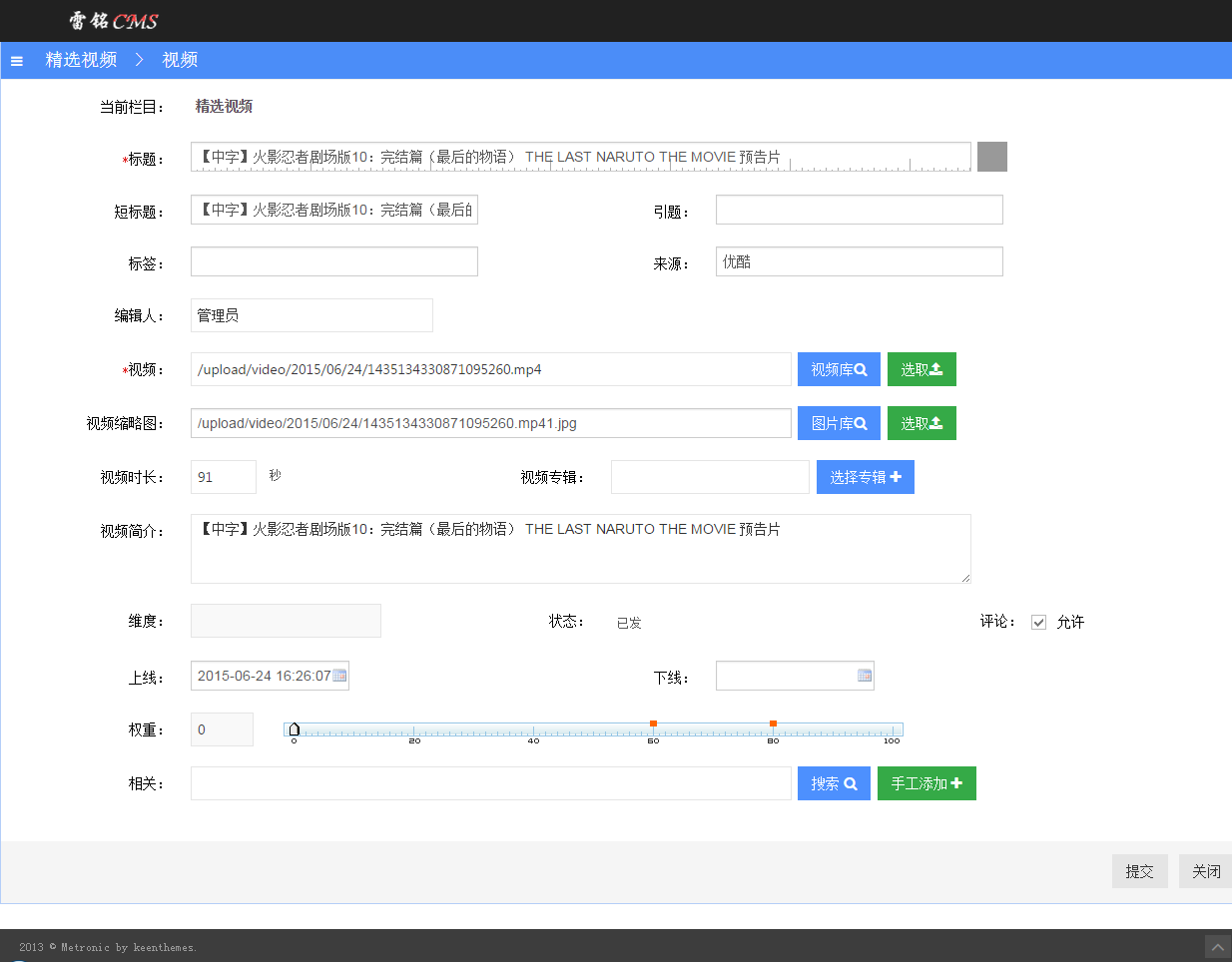


### 4.3.视频

支持视频点播和直播，帮助客户低成本解决视频技术需求。

* 支持多种视频格式在线转码
* 支持iOS和Android移动终端播放
* 支持自动提取视频缩略图、时长和码率
* 支持视频专辑和连播，提供开放性API

录入界面入下：



页面展示如下：

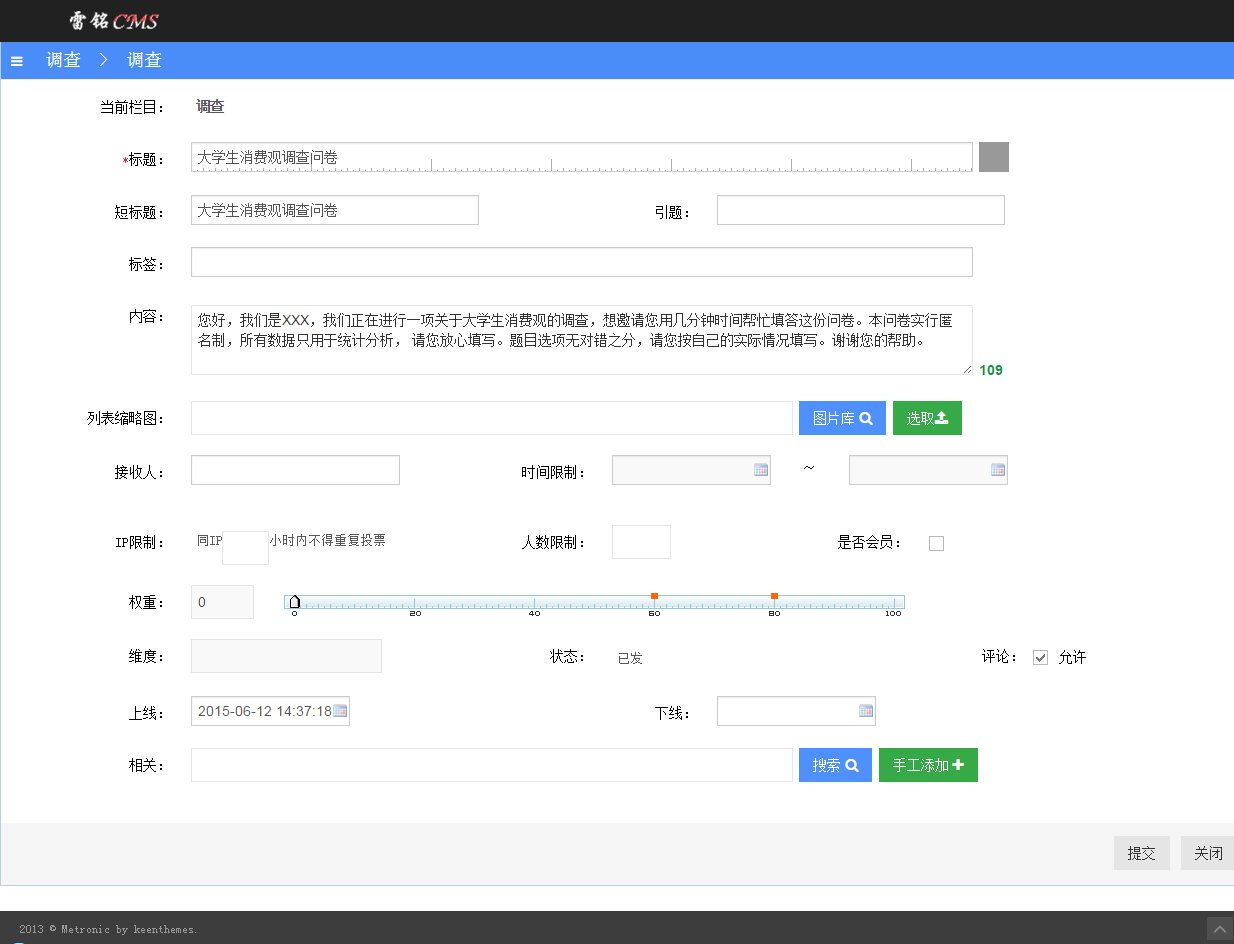


### 4.4.调查

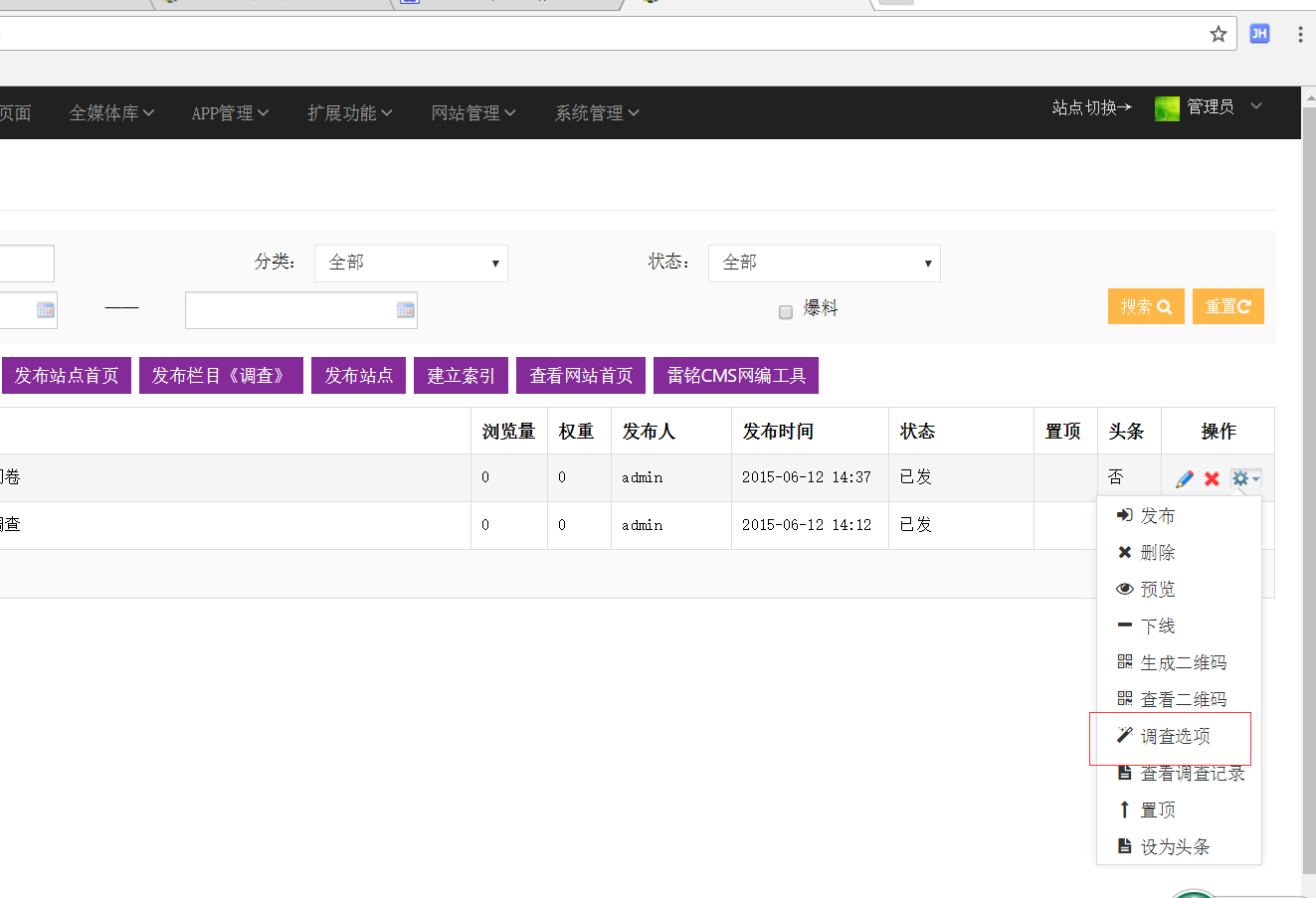
可自由设计调查表单，进行在线问卷调查和结果统计分析，可以做市场调查。

* 支持单选、多选、单行文本和多行文本等表单类型。支持问题自由排序
* 支持生成调查结果综合报表和分析报表及图形显示
* 支持后台上传调查页面背景图

录入界面如下：



点击内容管理处的，进入调查选项界面





页面展示效果如下：



### 4.5投票

可广泛应用于在线投票调查和网络评选活动。

* 支持网络评选模式，选项可添加图片和链接
* 支持防刷限制，支持单选和多选投票
* 支持普通、高级验证码，以及IP防刷策略
* 支持服务器端IP防刷攻击和防刷过滤
* 支持投票地域限制

录入界面如下：

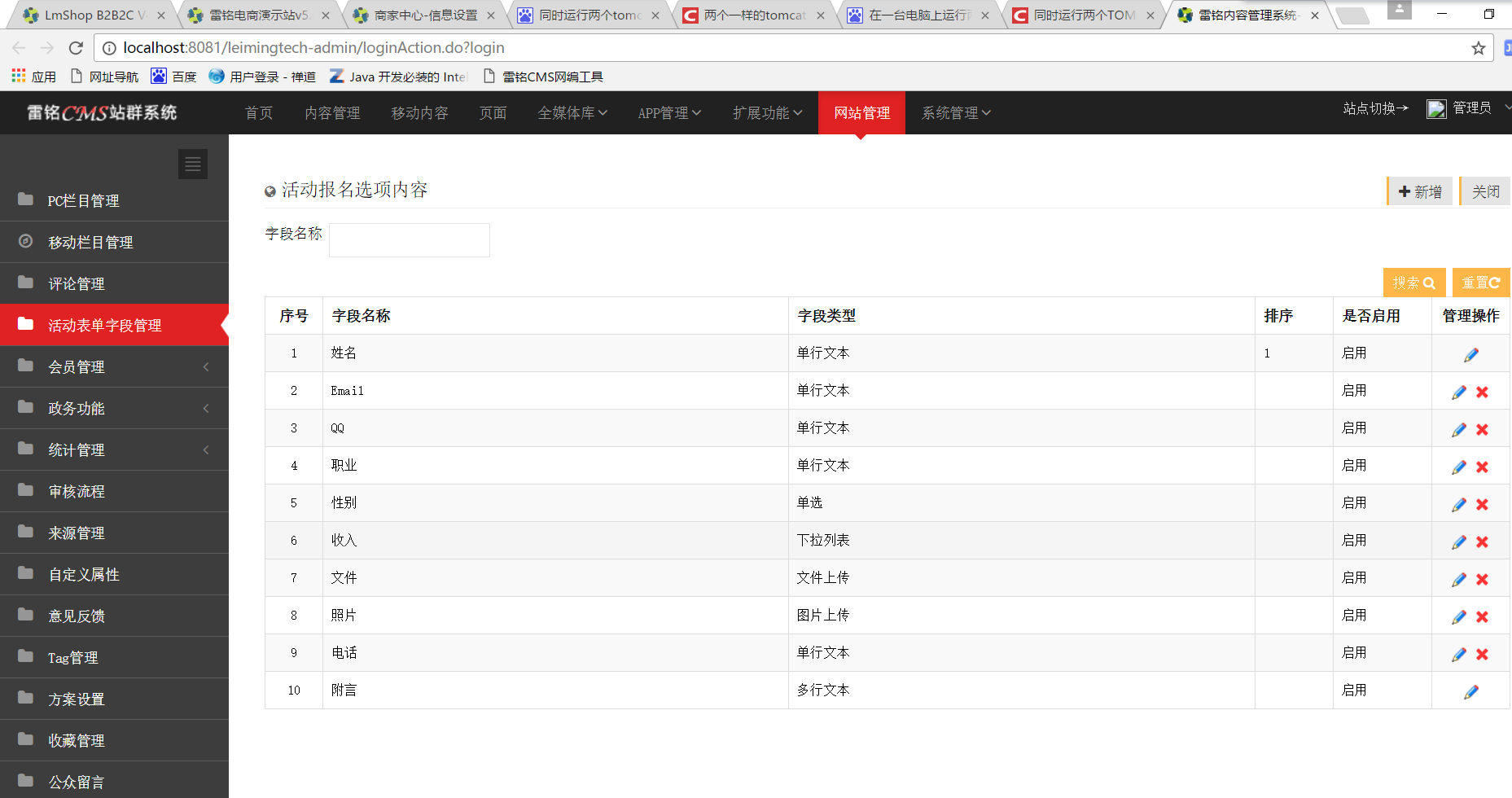


如果改变上述的内容需要重新进行发布，视自己的情况而定，发布站点，栏目，或只发布一个内容。

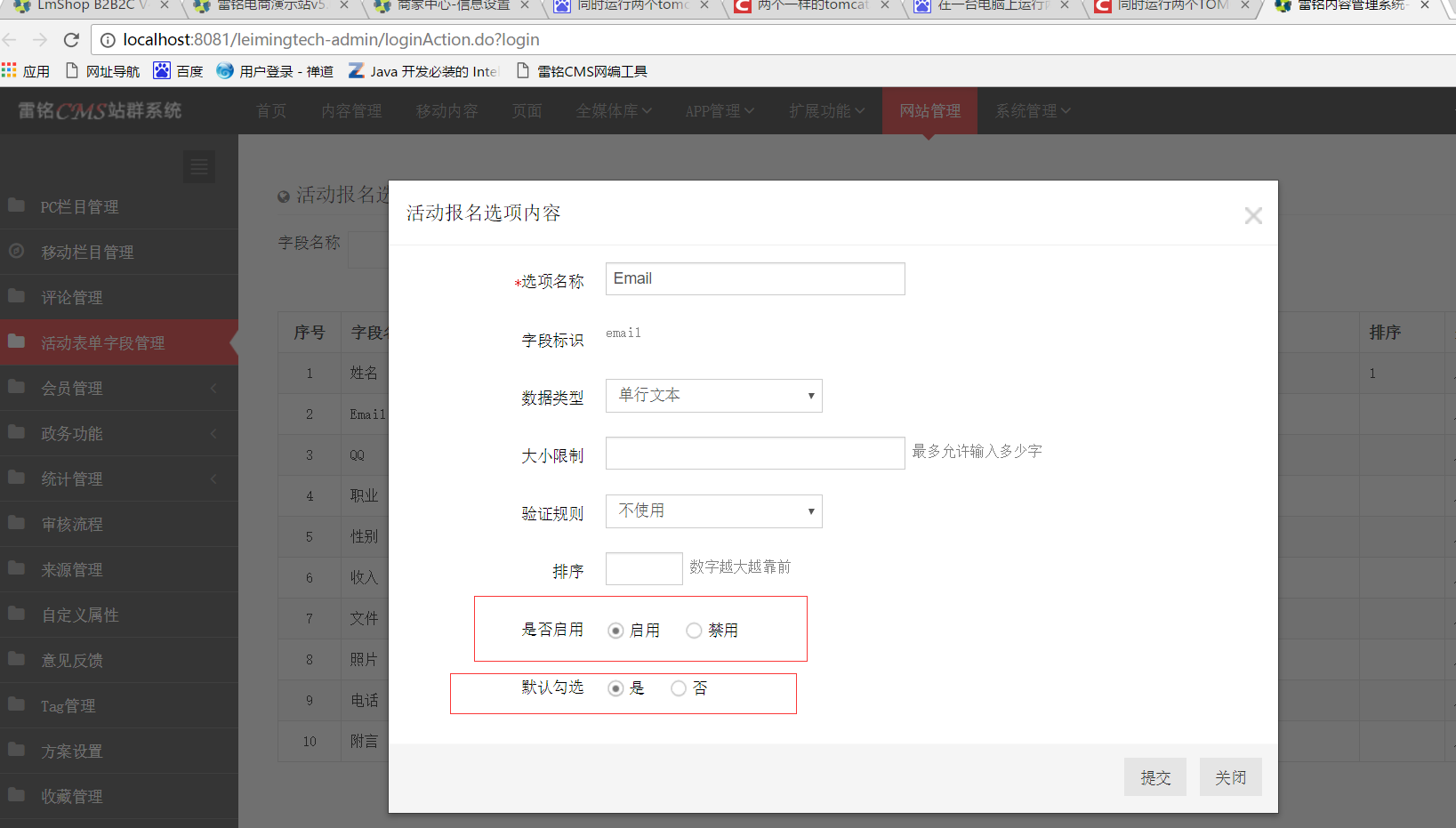
### 4.6活动

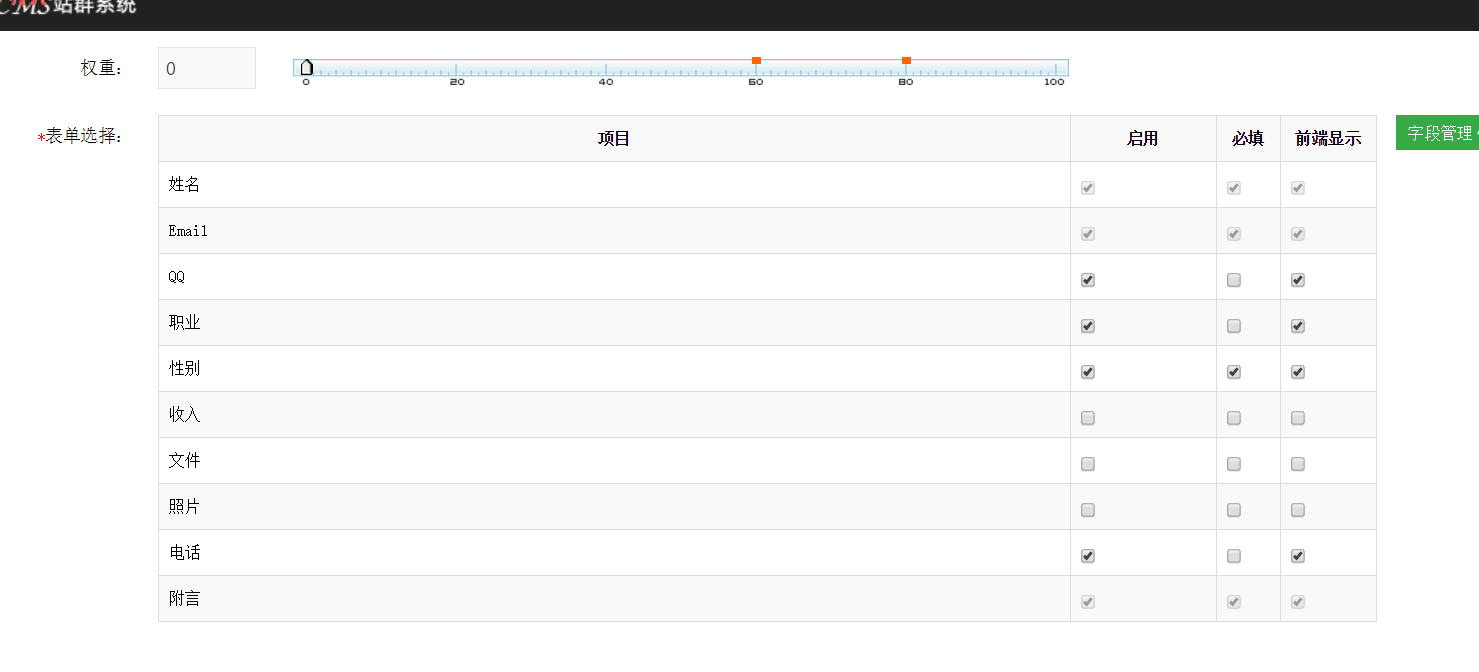
活动表单字段管理

这个功能是活动提交表单所设置的选项，这个是站点全局的设置，不是针对一个活动的



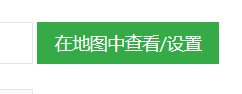
下图，是否启用和默认勾选，在新建活动界面的表单处，会变成默认选中

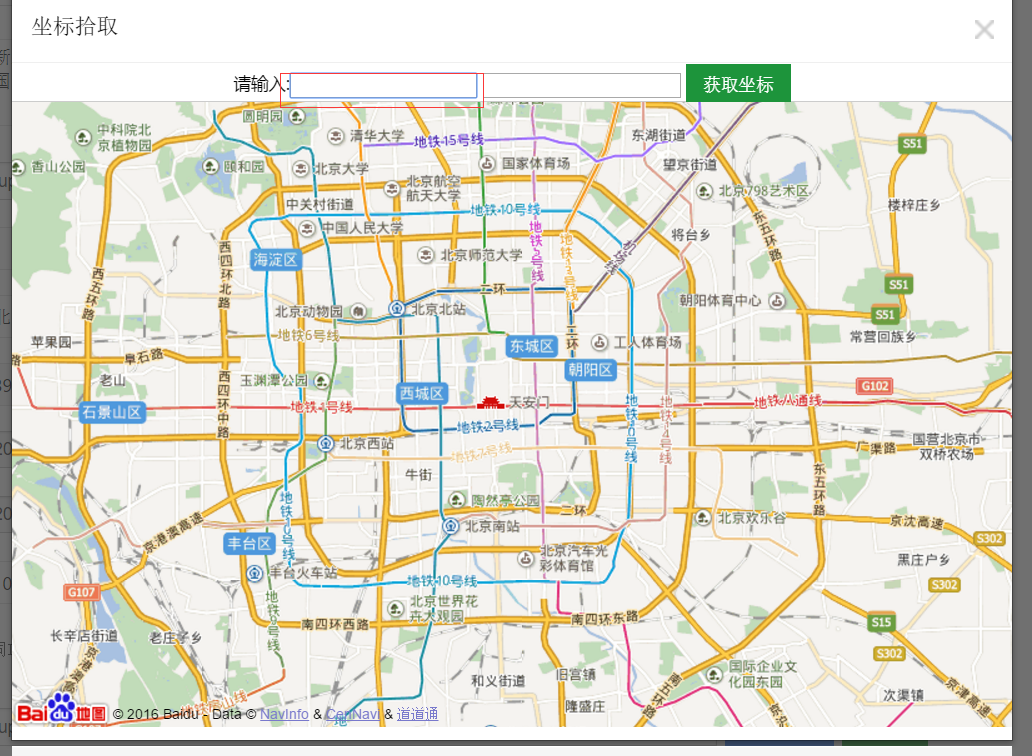




新建活动界面



点击上方的选择活动地址，点击红框处输入地址，点击获取坐标，获取它的坐标，最后点击确定



人数限制，限制参加活动的人数

IP限制，限制同一个ip地址几个小时不能重复参加该活动

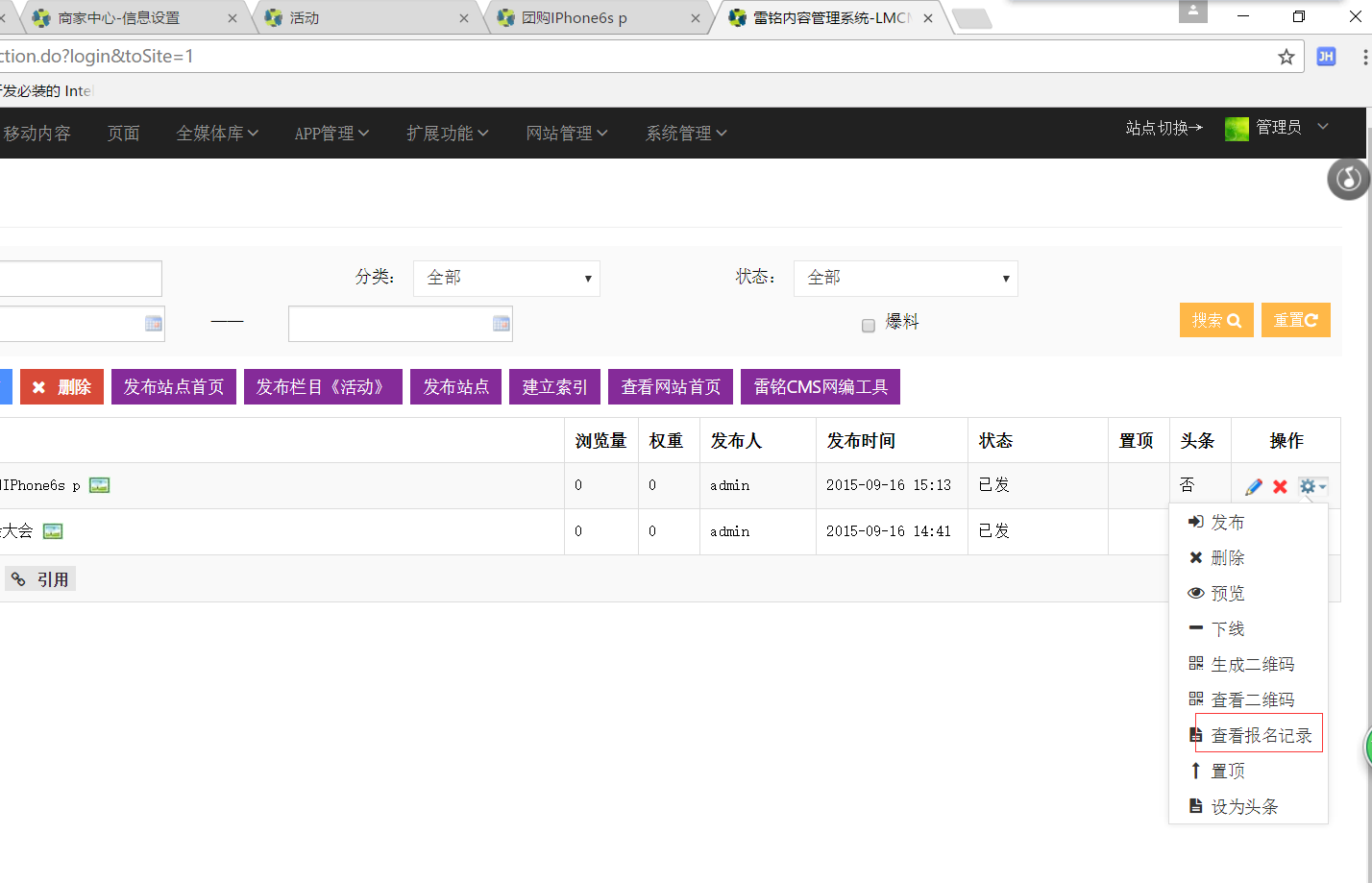
表单选择，选择启动和前端显示，该表单才可以在活动详情界面的表单处显示，必填选择，表示在前端表单处填写该选项为必填。



下面为所展示的活动详情界面



当内容为活动时，选择操作里面的查看报名记录，可以查看报名情况



## 5.区块的引用

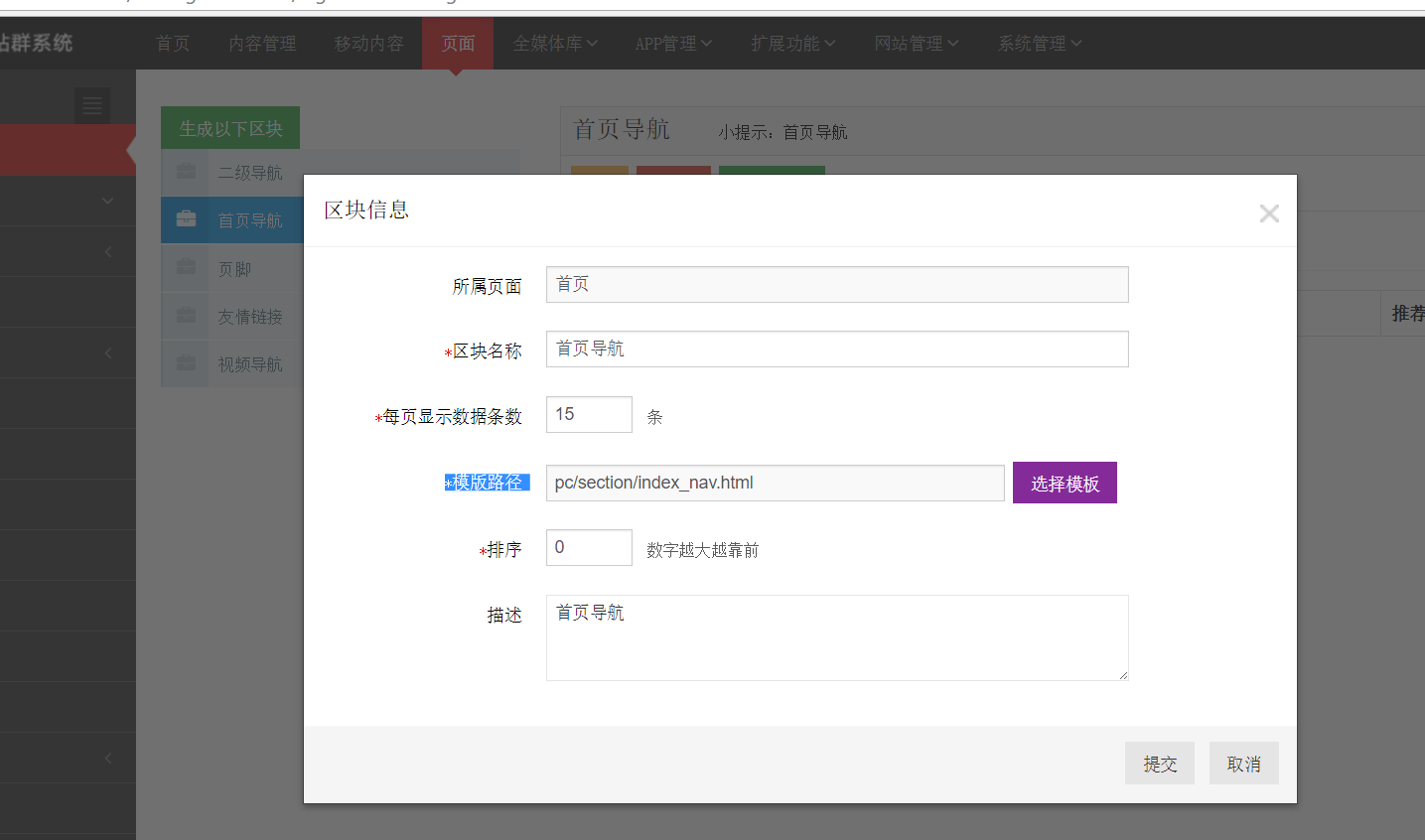
当不同的模板处，里面都有一个相同的区块，那么我们可以将相同的位置单独拿出去。只需要在模板处引用就可以

操作步骤：

首先在后台管理的页面处

添加一个区块

如图所示



例如：我添加一个首页导航，那么在首页模板处引入这个区块页面

<#include "/wwwroot/${sitePath}/template/common/index\_nav.html">

如果引入之后页面并没有看到引入效果，可能是由于nginx里面少配置了

gzip on;

ssi on;

ssi\_silent\_errors on;

ssi\_types text/shtml;

上面这个就是nginx引入include

区块之后，点击生成区块按钮，进行发布

## 6.模板标签的使用

LMCMS标签大致为两种，一种为在页面调用标签，一种为通过发布流程传入在页面直接使用，后台代码的调用层为PublishAction ,它的 实现层为 StaticImpl.

### 6.1栏目标签

参数：

参数 类型 值

name String 栏目id或栏目名称

siteid String (site.id)站点id

levels String 等级(self,root, child)

count String 栏目显示数量，

声明为两种

一种：返回栏目实体，传入name参数，参数值为栏目id或栏目名称,（栏目id可以在PC栏目管理处查看）

leves（等级）参数值为self代表自身

<#assign name ="catalogTag">

<#assign catalogTag =newTag("${name}")>

<#assign catalog =catalogTag("{'levels':'self','name':'31'}")>

catalog 为栏目的实体，可以返回栏目实体内的字段。

一种：返回栏目列表，

1. 查看栏目一级导航，出现场景为排列规范的栏目导航

传入参数siteid(站点id)参数值为${site.id}

levels(等级)参数值为root

<#assign name ="catalogTag">

<#assign catalogTag =newTag("${name}")>

<#assign catalogList =catalogTag("{'levels':'root','siteid':'${site.id}'}")>

catalogList 为栏目的列表（一级栏目）循环返回栏目实体的字段

1. 查看栏目的子栏目，出现场景为栏目下有子栏目

传入name参数，为父栏目id

levels 参数值为child代表子

<#assign name ="catalogTag">

<#assign catalogTag =newTag("${name}")>

<#assign catalogListchild=catalogTag("{'name':'${catList.id!''}','levels':'child'}")>

catalogListchild 为子栏目列表，（返回的子栏目）循环返回栏目实体的字段

模板显示常用字段

|  |  |  |
| --- | --- | --- |
| **名称** | **代码** | **类型** |
| 栏目id | id | varchar() |
| 栏目名称 | name | varchar() |
| 栏目地址 | url | varchar() |

### 6.2.文章列表标签

参数：

|  |  |  |
| --- | --- | --- |
| **属性** | **允许的值** | **说明** |
| **标签名** | articleTag | 文章标签 |
| **name** | 栏目id | 栏目名称，显示指定名称的栏目的列表，可以为ID  当level属性值为self，name值是文章名称/ID，返回单条文章 |
| **Level**  **分页时指定无效** | root | 站点一级栏目下文章，如果指定了name则此属性无效 |
| current | 当前栏目的同级栏目中文章 |
| child | 当前栏目下的子栏目中文章，默认属性 |
| self | 获取文章本身 |
| all | 获取当前栏目下包括子集的文章 |
| **count** | 1~n | 整数，列表显示数目  不分页时默认10条 |
| **order** | top | 头条文章 |
| is\_top | 置顶文章 |
| recent | 配合新闻使用，最新新闻，按发布时间排序,默认 |
| pv | 配合新闻使用，按点击率排序 |
| **page** | ture/false | 是否分页 |
| **pageindex** | 'pageindex':'${pageindexF}' | page=true有效  索引固定写法 'pageindex':'${pageindexF}' |
| **isGetImage** | ‘true’/’false’ | 显示带图片的文章 |

大致分为四种情况

<#assign articleTag =newTag("articleTag")>

<#assign articleLis =articleTag("{'name':'${catalogidF}','page':true,'pageindex':'${pageindexF}','level':'参数等级'}")>

传入参数name，如果该模板该栏目的模板写成固定值${catalogidF}或${catalogF.id}即可

**如果查询的栏目不是本栏目或不是本栏目子栏目的，需要传入栏目id**

1. 显示栏目下的栏目文章列表

level：self 代表的是查询本栏目下的文章列表

1. 显示栏目下的栏目文章列表包含子栏目的文章列表

传入参数level 值为all

1. 显示栏目下子栏目的文章列表不包含本栏目

传入参数level 值为child

1. 显示有缩略图的文章列表

传入参数**isGetImage 值为true 默认为不填写，‘isGetImage’：‘true’**

如果对文章进行分页，需要输入

'page':true,

'pageindex':'${pageindexF}'，'pageindex'参数的值为后台发布流程传入的参数为固定写法${pageindexF}

分页的界面是通过java编写的界面，在发布流程处将其作为参数传入

位置在TemplateData处，只需在列表页面的下方写入${pagebar}

如果需要改分页的样式或结构在TemplateData 处

分页目前的效果为下图



限制文章列表的数量，输入参数count,值为整数

上述返回的都为内容实体的列表，循环显示内容实体。(contentsEntity)

**模板显示的常用字段,即**contentsEntity实体下的字段。

|  |  |  |
| --- | --- | --- |
| **名称** | **代码** | **类型** |
| 文章id | id | varchar() |
| 标题 | title | varchar() |
| 短标题 | subtitle | varchar() |
| 标题字体颜色 | color | varchar() |
| 缩略图 | thumb | varchar() |
| 标签 | tags | varchar() |
| 作者 | author | varchar() |
| 来源id | sourceid | varchar() |
| URL路径 | url | varchar() |
| 宽度 | weight | int |
| 发布时间 | published | datetime |
| 浏览量 | pv | int |
| 是否允许评论 | allowcomment | varchar() |
| 摘要 | digest | varchar() |
| 详情 | atriles | varchar() |

### 6.3导航标签

<#assign navTag =newTag("navTag")>

<#assign navTagList =navTag("{'name':'${catalogF.id}'}")>

参数

|  |  |  |
| --- | --- | --- |
| **属性** | **允许的值** | **说明** |
| 'name' | ${catalogF.id} | 栏目id |

模板处常显示字段其实为栏目实体的字段。

|  |  |  |
| --- | --- | --- |
| **名称** | **代码** | **类型** |
| 栏目地址 | url | varchar() |
| 栏目名称 | name | varchar() |

常运用到的场景为下图所示



### 6.4广告标签

参数

|  |  |  |
| --- | --- | --- |
| **属性** | **允许的值** | **说明** |
| **标签名** | advTag | 广告标签 |
| **name** |  | 指定广告位，可以为ID |
| **count** | 1~n | 整数，列表显示数目 |

<#assign name ="advTag">

<#assign advTag =newTag("${name}")>

<#assign advList= advTag ("{'name':'广告位id','count':'10'}")>

count可以不填写，不填写查询广告位下方所以得广告

advList 循环的为AdvertisingEntity 的实体字段

模板常用的显示字段为

|  |  |  |
| --- | --- | --- |
| **名称** | **代码** | **类型** |
| 广告id | id | varchar() |
| 广告名称 | adName | varchar() |
| 图片上传地址 | imgUrl | varchar() |
| 高度 | iheight | varchar() |
| 宽度 | iweight | varchar() |
| 链接地址 | urladress | varchar() |
| 链链提示 | urlpoint | varchar() |
| 链链目标 | urltarget | varchar() |
| 文字内容 | wordcontent | varchar() |
| 文字大小 | wordsize | varchar() |
| 文字颜色 | wordcolor | varchar() |
| 广告位名 | advertisingSpace.adName | varchar() |

### 6.5.友情链接分类标签

基本语法：

<#assign name ="friendLinkCtgTag">

<#assign friendLinkCtgTag=newTag("${name}")>

<#assign friendLink= friendLinkCtgTag ("{'siteid':'${site.id}'}")>

参数

|  |  |  |
| --- | --- | --- |
| **属性** | **允许的值** | **说明** |
| **标签名** | friendLinkCtgTag | 友情链接分类标签 |
| **name** |  | 指定链接分类，可以为ID |
| **count** | 1~n | 整数，列表显示数目 |
| **siteid** | ${site.id} | 站点id |

有两种情况

1.输入name参数，值为分类id

返回的为friendLink 实体（FriendLinkCtgEntity）

2.输入siteid 参数值为${site.id}

返回的为友情分类列表

模板常显示字段：

|  |  |  |
| --- | --- | --- |
| **名称** | **代码** | **类型** |
| 友情链接分类id | id | varchar() |
| 友情链接分类名称 | friendlinkctgName | varchar() |

### 6.6.友情链接标签

基本语法：

<#assign name ="friendLinkTag">

<#assign friendLinkTag =newTag("${name}")>

<#assign friendLinkList= friendLinkTag ("{'name':'(分类id)','count':'10'}")>

参数：

|  |  |  |
| --- | --- | --- |
| **属性** | **允许的值** | **说明** |
| **标签名** | friendLinkTag | 友情链接标签 |
| **name** |  | 指定友情链接，可以为ID |
| **count** | 1~n | 整数，列表显示数目 |

模板常显示字段，即为（FriendLinkEntity） 实体的字段

|  |  |  |
| --- | --- | --- |
| **名称** | **代码** | **类型** |
| 友情链接id | id | varchar() |
| 网站名称 | siteName | varchar() |
| 网站地址 | domain | varchar() |
| 图标 | logo | varchar() |
| 站长邮箱 | email | varchar() |
| 描述 | description | varchar() |
| 链接类型 | friendLinkCtg.friendlinkctgName | varchar() |

### 6.7.活动标签

<#assign activityTag =newTag("activityTag")>

<#assig activity =activityTag("{'contentId':''}")>

参数：

|  |  |  |
| --- | --- | --- |
| **属性** | **允许的值** | **说明** |
| 'contentId' |  | 内容Id |

返回的为活动实体

模板常显示字段，即活动实体下的字段（ActivityEntity）

|  |  |  |
| --- | --- | --- |
| **名称** | **代码** | **类型** |
| 报名时间(开始) | activityapplystarttime | Date |
| 报名时间(结束) | activityapplyendtime | Date |
| 活动时间(开始) | activitystarttime | Date |
| 活动时间(结束) | activityendtime | Date |
| 活动地址 | activityaddress | varchar() |
| 参加人数 | activityplenum | varchar() |
| 性别 | activitysex | varchar() |

### 6.8.上一遍标签

<#assign backTag =newTag("backTag")>

<#assign backList=backTag("{'cid':'${contentidF}'}")>

参数

|  |  |  |
| --- | --- | --- |
| **属性** | **允许的值** | **说明** |
| cid | ${contentidF } | 内容Id |

模板常显示字段

|  |  |  |
| --- | --- | --- |
| **名称** | **代码** | **类型** |
| 文章id | id | Varchar() |
| 标题 | title | varchar() |
| 地址 | url | varchar() |

### 6.9.下一遍标签

<#assign belowTag=newTag("belowTag ")>

<#assign belowList =backTag("{'cid':'${contentidF}'}")>

参数

|  |  |  |
| --- | --- | --- |
| **属性** | **允许的值** | **说明** |
| cid | ${contentidF } | 内容Id |

模板常显示字段

|  |  |  |
| --- | --- | --- |
| **名称** | **代码** | **类型** |
| 文章id | id | Varchar() |
| 标题 | title | varchar() |
| 地址 | url | varchar() |

上边两个标签在文章详情处显示

如下图所示



### 6.10.模板详情

**所以模板详情的固有参数为${ contentF} 该参数为内容实体(contentsEntity)**

**传入的参数在ContentTagMng下的方法getContent处**

#### 6.10.1文章模型

文章的详情在模板表示为${ articleDataF.content} 该参数为articleDataF为文章实体（ArticleEntity）

文章详情附件显示为

<#if accessoryList?exists && accessoryList?size gt 0>

<#list accessoryList as accessory>

<a href=*"${contextpath}/front/newsDeatilController.do?download&path=${uploadPath}${accessory.accessoryUrl!}"*><b>${accessory.accessoryName!}</b></a><br/>

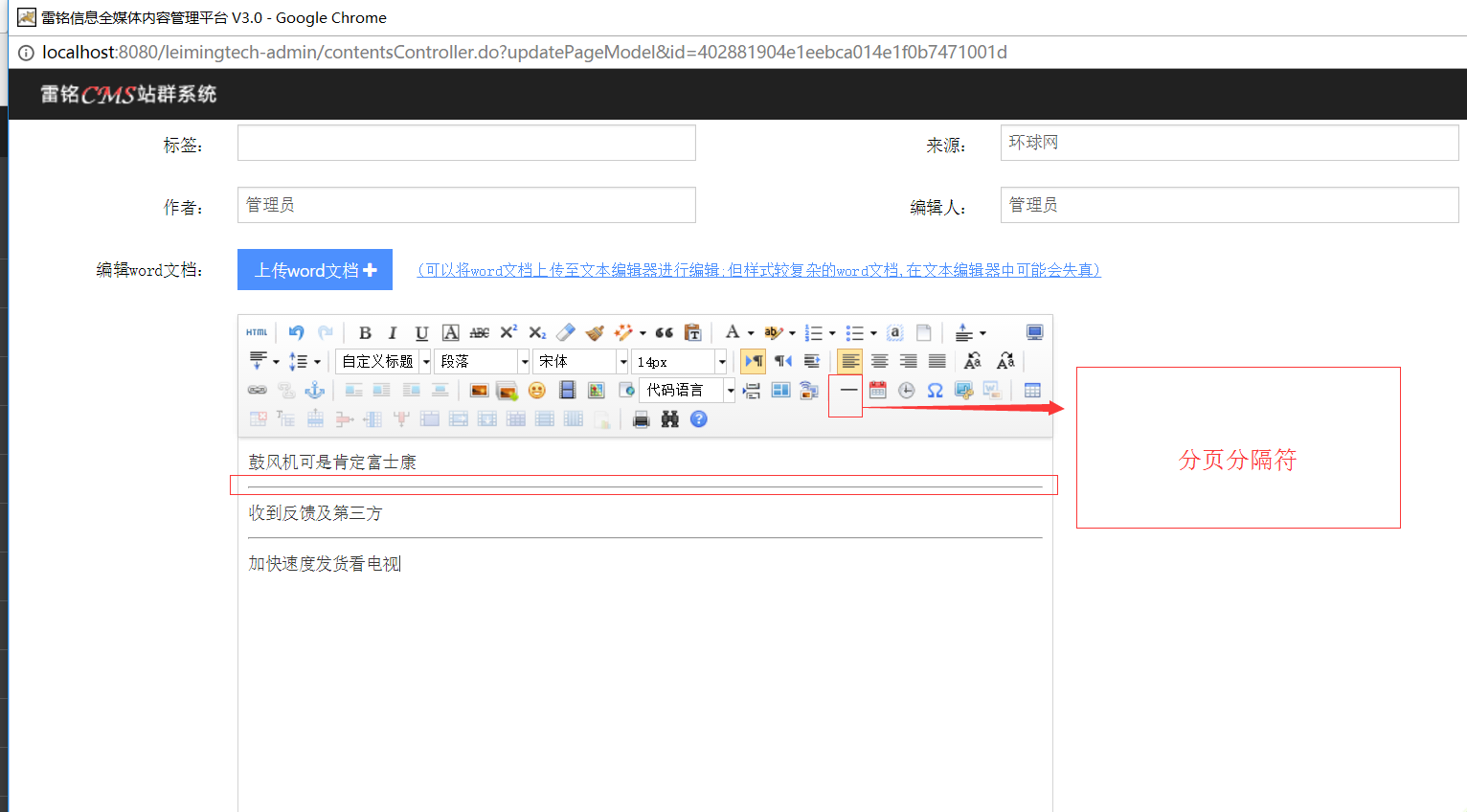
</#list>

<br/>

</#if>

其中accessoryList它即位文章附件列表，实体为（ContentAccessoryEntity）

如果进行文章分页，需要现在编写文章时以分页分隔符隔开，如下图所示



在详情处输入${PageBreakBar}即可，

#### 6.10.2组图模型

${ pictureGroupF }为组图实体(PictureGroupEntity)

${ pictureAloneListF}为组图下的组图列表，实体为(PictureAloneEntity)

示例：

<#**if** pictureAloneListF?exists>

<#list pictureAloneListF as pictureAlone>

'${pictureAlone.pictureUrl}',

</#list>

</#**if**>

#### 6.10.3视频模型

${videoF}为视频实体(ContentVideoEntity)

示例：

${videoF. videourl}// 视频播放地址

#### 6.10.4调查模型

${surveyF} 为实体调查（SurveyEntity）

optionList 为实体选项列表（SurveyOptionEntity）

optionextList为实体选项内容（SurveyOptionExtEntity）

示例：

<#list surveyF.optionList as option>

<tr>

<td height=*"30"* colspan=*"2"* align=*"left"*>

<span style="color: *#1c77ba*;">${option\_index+1}、${option.optionname}</span>

</td>

</tr>

<#if option.dataType=='radio'>

<tr>

<td height=*"30"* colspan=*"2"* align=*"left"*>

<#list option.optionextList as ext>

<input type=*"radio"* name=*"${option.id}"* value=*"${ext.id}"* id=*"${ext.id}"*/>${ext\_index+1}.${ext.extOptionname}

</#list>

</td>

</tr>

</#if>

<#if option.dataType=='checkbox'>

<tr>

<td height=*"30"* colspan=*"2"* align=*"left"*>

<#list option.optionextList as ext>

<input type=*"checkbox"* name=*"${option.id}"* value=*"${ext.id}"* id=*"${ext.id}"* />${ext\_index+1}.${ext.extOptionname}

</#list>

</td>

</tr>

</#if>

<#if option.dataType=='text'>

<tr>

<td height=*"30"* colspan=*"2"* align=*"left"*>

<input name=*"${option.id}"* type=*"text"* id=*"${option.id}"* /></td>

</tr>

</#if>

<input name=*subIds* type=*"hidden"* value=*"${option.id}"* />

</#list>

其中dataType为类型，分为radio,checkbox,text, textarea这几种类型

extOptionname //内容名称

optionname //为选项名称



提交方法在

${contextpath}/front/surveyLogController.do?subSurvey&surveyId=${surveyF.id}处

查看调查结果

${contextpath}/front/surveyLogController.do?surveyPcTag

#### 6.10.5投票模型

${voteF}为投票实体（VoteEntity）

${ voteOptionListF }为投票选项列表(VoteOptionEntity)

示例

<#if voteOptionList?exists>

<#if voteF.votepattern=="10">

<ul>

<#list voteOptionList as voteOption>

<#if voteOption.optionimg?exists && voteOption.optionimg!="">

<li>

<center>

<div class=*"vote\_name"*>

<input type=*"<#if* voteF*.votetype='20'>checkbox<#else>radio</#if>"* name=*"voteitem"* id=*"${voteOption.id}"* value=*"${voteOption.id}"*/>

<a href=*'${voteOption.optionlink!'*javascript:void(0)'}' >${voteOption.optionname}</a>

</div>

</center>

</li>

<#else>

<li>

<div class=*"vote\_name"*>

<input type=*"<#if* voteF*.votetype='20'>checkbox<#else>radio</#if>"* name=*"voteitem"* id=*"${voteOption.id}"* value=*"${voteOption.id}"*/>

<a href=*'${voteOption.optionlink!'*javascript:void(0)'}' >${voteOption.optionname}</a>

</div>

</li>

</#if>

</#list>

</ul>

</#if>

其中votepattern //评选模式,10普通20 评选

Votetype //投票类型 10单选 20多选

Optionname //选项内容

Optionimg // 选项图片

Optionlink //选项链接

提交方法

${contextpath}/front/voteLogController.do?voteLogTag&voteId=${voteF.id}

查看投票结果

${contextpath}/front/voteLogController.do?queryPCvoteLog

#### 6.10.6活动模型

${ activityF } 活动实体（ActivityEntity）

${ activityOptionListF } 为联表查询的集合，分别为实体表单选项ActivityOptionEntity和实体活动选项内容

ActivityOptionContentEntity

参加活动提交方法为

${contextpath}/activityOptionExtController.do?subActivity&activityId=${activityF.id}

上述标签里面有些返回的值为固定写法，来源为

### 6.11适用于每个模板

${site} ，代表的是站点实体，适用于每个模板

${contextpath} 代表的是工程目录的例如：localhost/leimingtech-admin

${ sitePath}代表的是站点路径

### 6.12列表模板

${catalogF} 代表的是栏目实体，主要运用于栏目列表的标题，导航标签处。

${ catalogidF}代表的是栏目id

${pageindexF} 代表的是页数，主要用于新闻分页，

主要运用于文章标签处

${pagebar}，分页，是通过后台代码进行处理，不是运用标签。

### 6.13详情模板

${ catalogF }代表的是栏目实体，

${contentF }内容详情实体。

### 6.14常用

站点下的5个目录 html（即模板）页面放进tempelate目录下 css，js,图片放进相应的目录下

在html页面引用css文件或者js文件下的资源时

只需要加‘/’后面跟路径就行了 这是因为在nginx进行了配置



引入登录和注册的div界面，将页面放入到一个div的id名为divLogin'里面，

如下图所示，这个只是一个案例仅供参考，



$(**function**() {

**var** urlStr = "${contextpath}/front/surveyLogController.do?isLogin";

$.ajax({

type : 'post',

url : urlStr,

data : 'data=0',

dataType : 'text',

success : **function**(msg) {

$('#divLogin').empty();

$('#divLogin').append(msg);

}

});

});

文章详情引入访问量

window.onload = **function**(){

**var** contentid = "${contentF.id?if\_exists}";

**var** urlStr = "${contextpath}/contentsController.do?addPv";

$.ajax({

type:'post',

url:urlStr,

data:"contentId="+contentid,

success:**function**(msg){

}

});

}

Lucen查询的方法引用，这个不属于公用的，因为返回的界面为雷铭新闻网这个站点处的界面，如果调用lucen时需要将引入的方法修改下

**function** toSearch4(urlStr) {

**var** keyword=document.getElementById("keyword4").value;

**var** key=keyword.replace(/\s+/g,"")

**var** keyWord4 = encodeURI(key);

keyWord4 = encodeURI(keyWord4);

window.location = urlStr + "&keyword=" + keyWord4+"&siteid="+${site.id};;

}

## 7前台界面调用动态方法注意事项

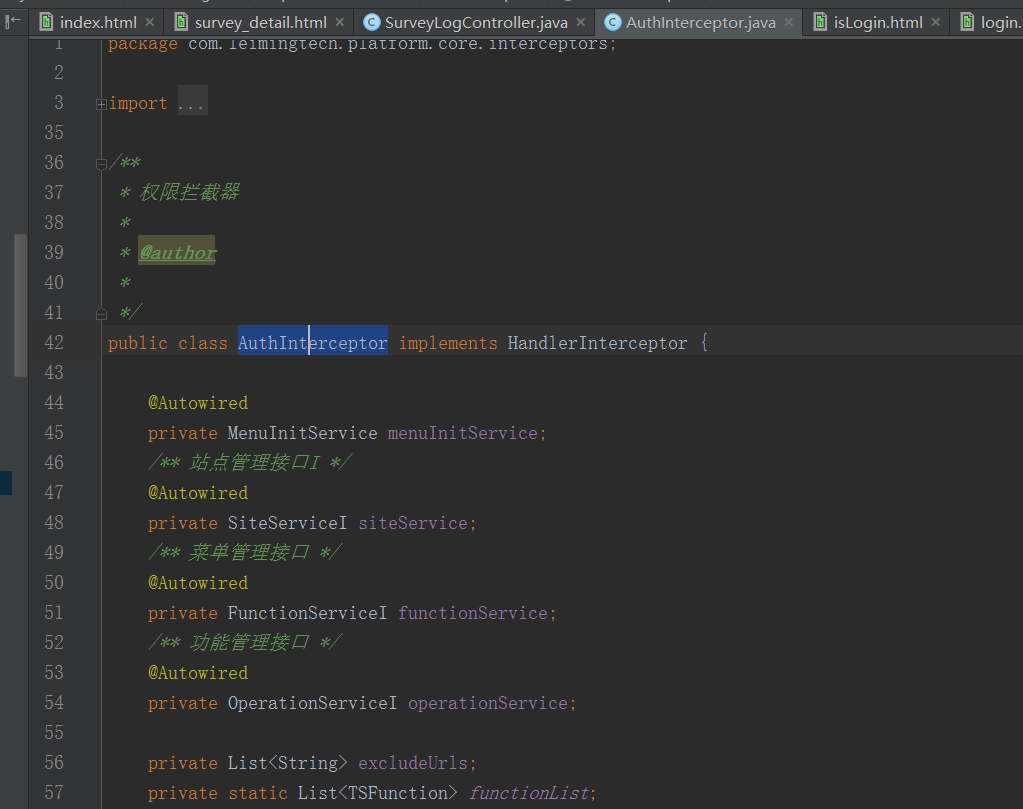
注意，如果在站点界面，即前端页面写动态方法时，一定要在实现层的前面加front或member,如下图所示

比如：

@RequestMapping("/front/surveyLogController")

我们会根据front和member进行权限拦截

参考



在前台页面调用动态方法

var urlStr = "${contextpath}/front/surveyLogController.do?isLogin";