* 1. 创新点

本文基于github上的博客项目进行了二次开发和创新，加入了文章收藏、消息通知、栏目管理、角色管理、主题管理。优化了文章管理，加入了置顶和推荐功能。

* 1. 系统功能概述

本博客系统主要有以下功能：

登陆界面：实现用户的注册与登录。

页面上边框：包含用户主页、博客、随笔、分享、标签五个模块，分别可以跳转到对应页面。另有用户头像模块，有编辑资料、退出登录等功能。并有搜索文章功能。用户会根据是否为管理员显示后台管理的按钮。

用户主页：展示总的视图，可以看到用户自己发的文章并进行管理，显示出自己的文章发布数量与评论数量。可以跳转到用户发表的文章、评论页面和收藏的文章页面。并有通知提示。

编辑文章界面：用户写好文章后，可以选择相应的栏目，并可标注标签。其中文章编辑可选择字体、增加图片，附件文件等。

文章页面：可将文章进行收藏、分享，并进行评论。

个人资料编辑：可供用户修改头像与名称等。

后台管理员界面：首先有仪表盘页面显示总的数据，如文章数等。其次实现了栏目管理、文章管理等功能，可对相应文章进行修改、删除。并有系统管理模块，将系统配置如站点信息等显示出，可供管理员进行修改。

* + 1. 用户个人信息
* 资料编辑

1. 修改昵称与个性签名，点击提交按钮，即可修改成功。
2. 修改邮箱，将新邮箱输入并获取验证码，输入验证码点击提交即可。
3. 修改头像，点击按钮选择本地图像即可。
4. 修改密码，首先输入当前密码，再输入两次新密码后即可，注意前后密码要保持一致，不然会有错误信息提示。

* 首页信息

用户主页有发表的文章等按钮，可跳转到相应页面。

* 消息通知

可以在用户主页左边栏消息通知模块看到是否有新消息，点击可跳转到通知列表查看信息详情。

* + 1. 文章的发布与收藏
* 文章发布

点击页面右上角写文章按钮，进入文章编辑页面，选择相应栏目，并添加标签，点击发布按钮即可发布文章。

* 评论

在文章详情页面评论区编写评论发布即可。

* 文章收藏

点击文章详情页面右侧栏的收藏按钮即可收藏文章，并显示当前文章收藏数。

* 文章转发

文章下方有各个社交平台按钮，点击即可跳转页面转发至相应平台。

* + 1. 后台管理
* 仪表盘

总体数据显示，如文章数量、栏目数量等，点击可跳转到相应管理页面。并显示系统占用情况。

* 栏目管理

显示当前的各个栏目并可修改栏目状态为隐藏或显示，也可对栏目进行增加或删除。

* 文章管理

显示当前发布的所有文章，可以根据关键字和栏目类别对文章进行检索，并可对文章进行置顶或推荐。也可对文章进行和后台修改或删除。

* 角色管理

可以显示当前管理员与用户，并支持增删管理员并分配其权限，将管理员职责具体化。

* 主题管理

可以选择应用哪种主题，并支持上传新的主题，新主题将改变用户页面的颜色等。并可以根据关键字检索主题。

1. 项目管理

以思维导图的形式将个人独立博客系统软件项目工作项的层级结构展示出来，更直观的看 到父子关系，在“规划”中新建的 Story (工作项)，会自动生成到后面的”工作项”和”迭代”页面。

* 1. 规划

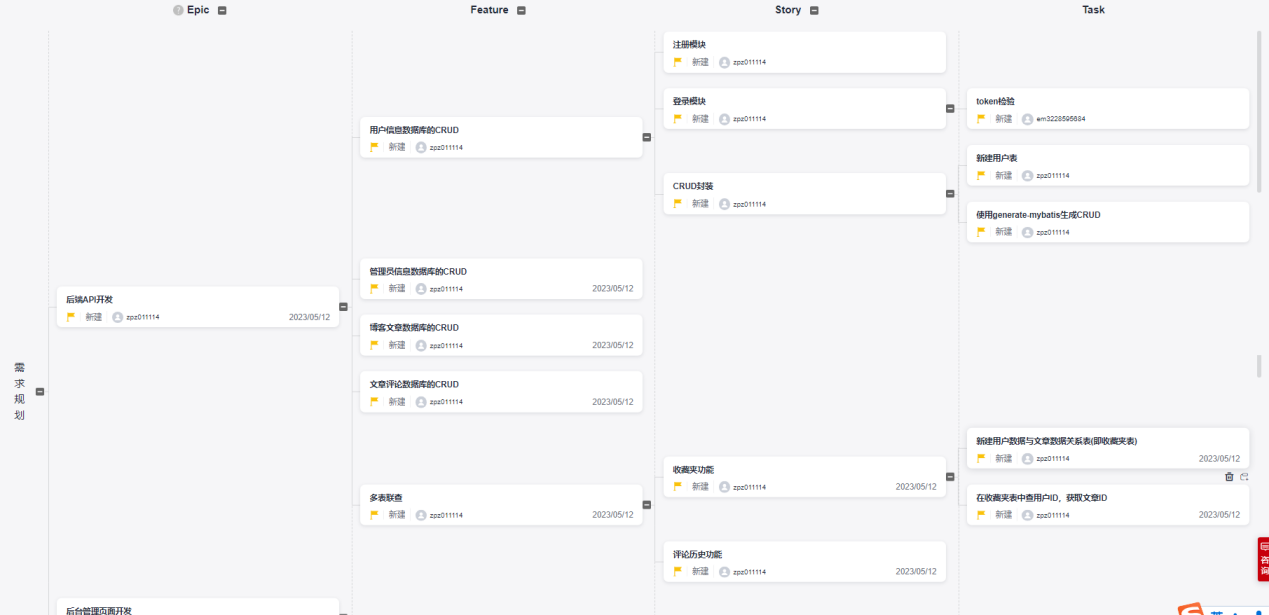
Product Owner（产品经理）将从客户获取的所有需求信息，逐一落实到项目规划中，分解细化，逐步形成 story (工作项)。

本文设置了四项epic进行迭代开发，将项目划分为了后端数据库与功能api，前端主页面和登录页面，具体项目规划如下。



1. Epic总览

每个epic划分为了若干具体的小task进行完成，更符合开发规范。四个epic的具体分工展示如下。



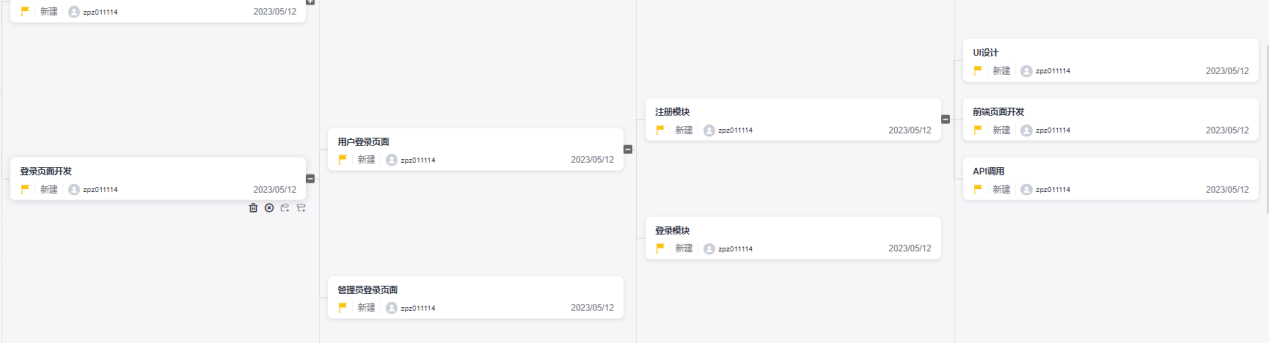
1. 后端API部分task展示



1. 后端数据库部分task展示

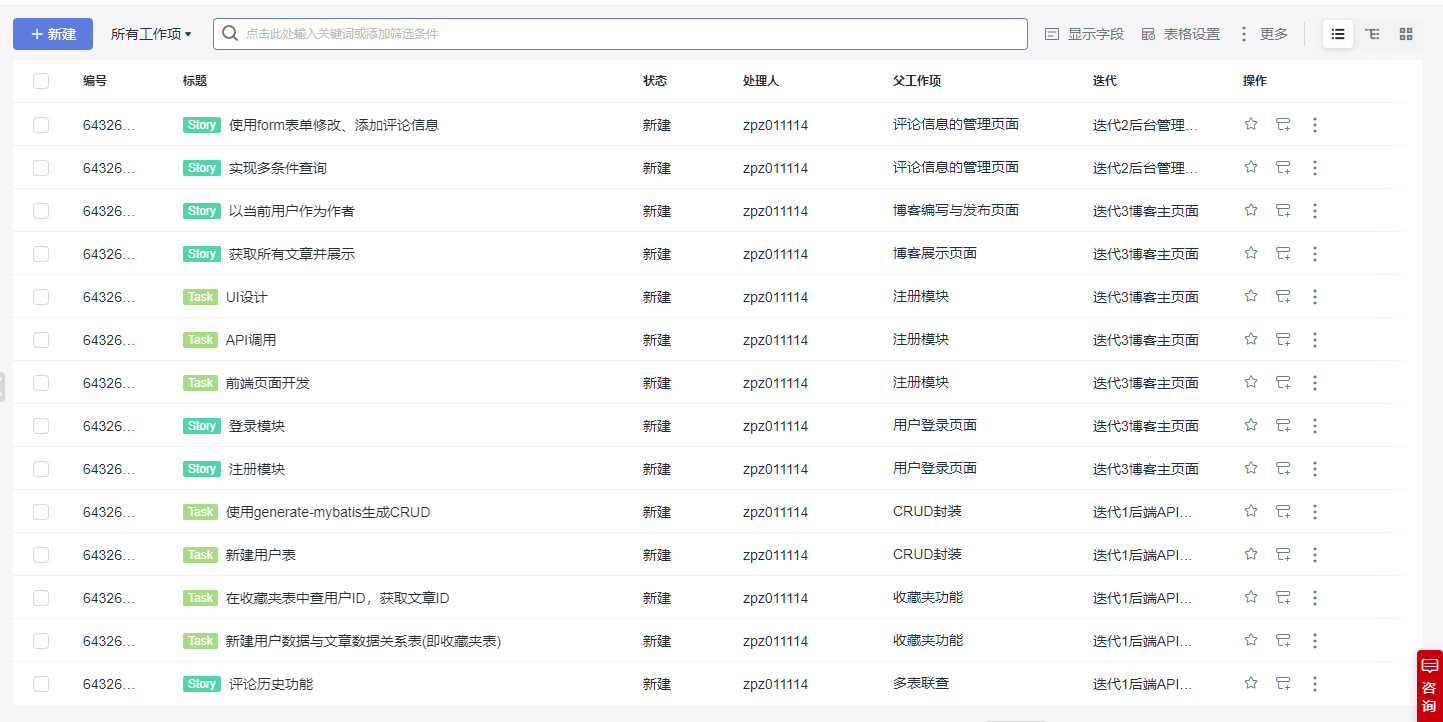


1. 主页面部分task展示



1. 登录页面部分task展示
   1. 工作项

根据需求分析，设置各个 Story 的优先级。输入用户故事描述信息、开始日期、结束日期、预计工时、优先级、重要程度字段信息。



1. 工作项汇总
   1. 迭代

根据项目研发周期，划分迭代及确定迭代周期。《软工综合实践》项目要求 创建迭代，根据规划情况，按照优先级从高到低，重要程度减弱的顺序设置迭代开发。本项目按照工作划分分为了四个迭代，每个迭代完成一个epic。



1. 迭代划分
2. 代码托管

软件源码可由同学们自行开发，亦可利用网上相应的开源资源进行二次开发。

* 1. 开发提交

如果使用本地 IDE 开发，在完成初版软件代码后需上传 DevCloud 代码仓库进行后续开发；也可使用 DevCloud 平台上的 CloudIDE 直接在云端进行代码开发。

本项目新建了两个分支，master和dev。在dev上进行开发工作，master上是稳定版的系统。



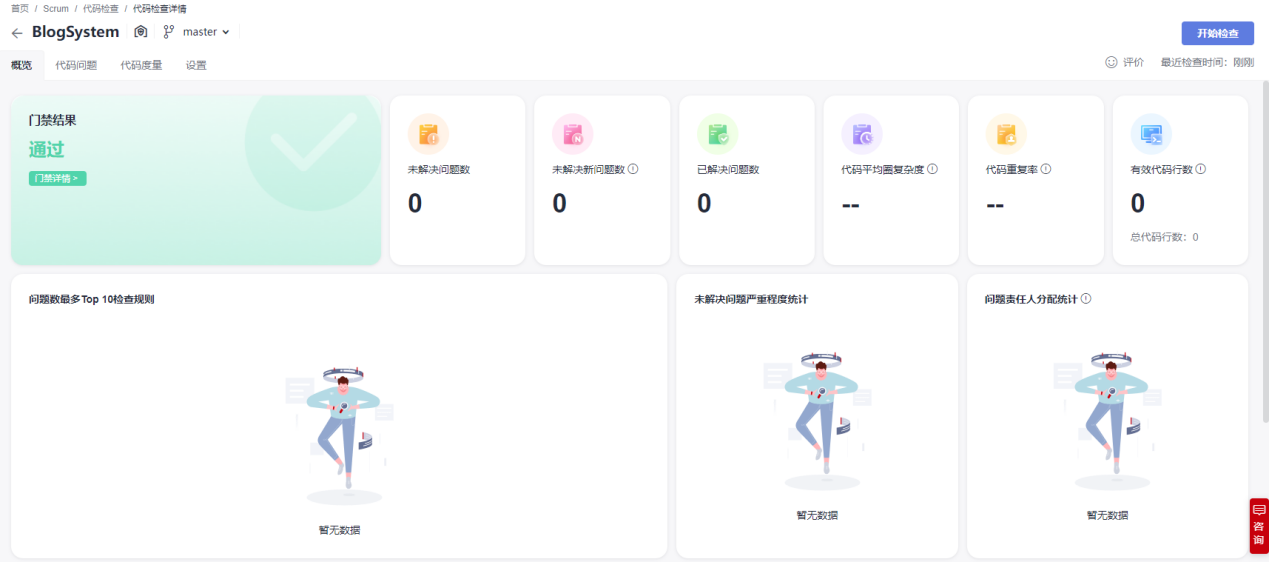
1. 提交到master分支
   1. 代码检查

为确保代码开发的质量，本项目使用 DevCloud 平台的代码检查服务对软件代码进行检查。



1. 代码检查过程

检查结果如下，代码通过了检查不存在问题。



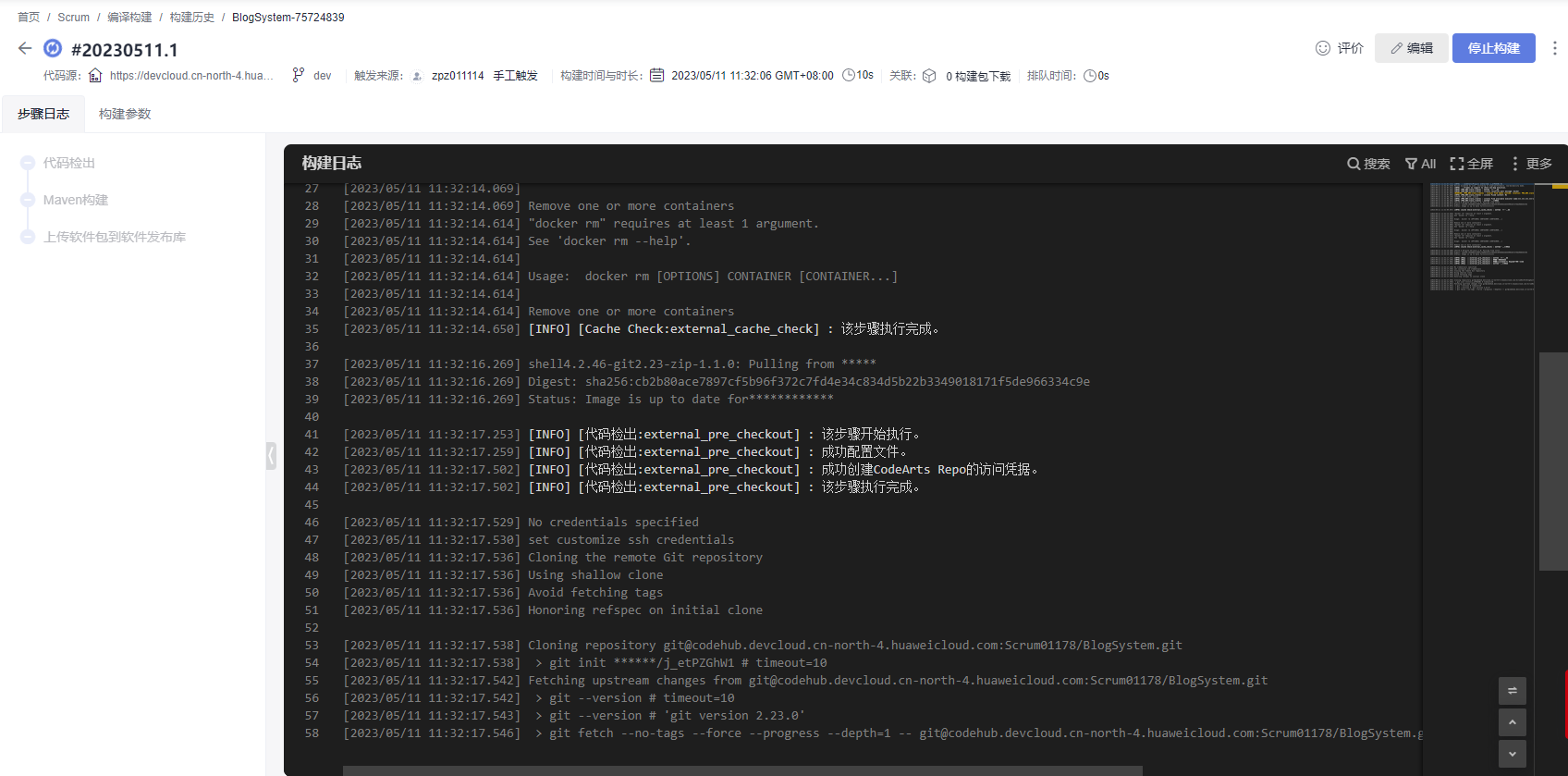
1. 代码检查结果
2. 编译构建
   1. 配置

创建构建任务，并设置本项目环境配置，代码下载，Maven构建等。



1. 项目构建配置

执行构建成功并生成发布包。



1. 代码自动构建
   1. 持续集成

启动持续集成，实现代码变更后自动触发。使用CodeArts提供的功能模块，实现代码流水线开发。

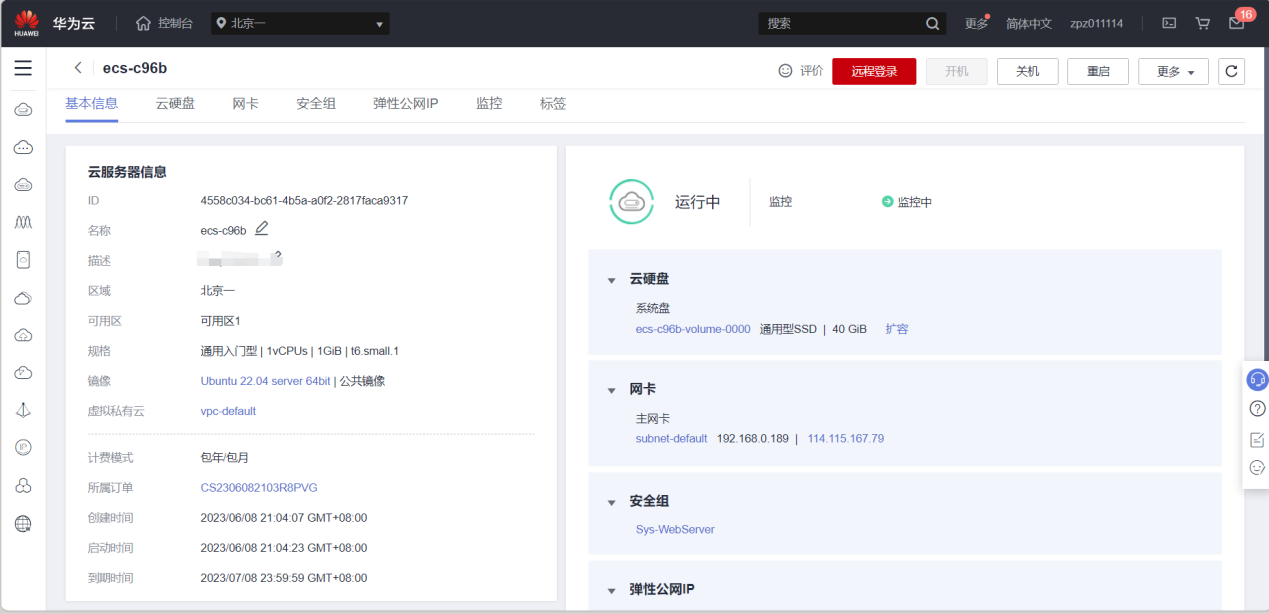


1. 持续集成过程
2. 部署

个人博客软件需要部署到服务器上才可以运行并被访问，这里将软件部署到华为 ECS 弹性云服务器。DevCloud 在使用这台云主机之前需要先对其授信。

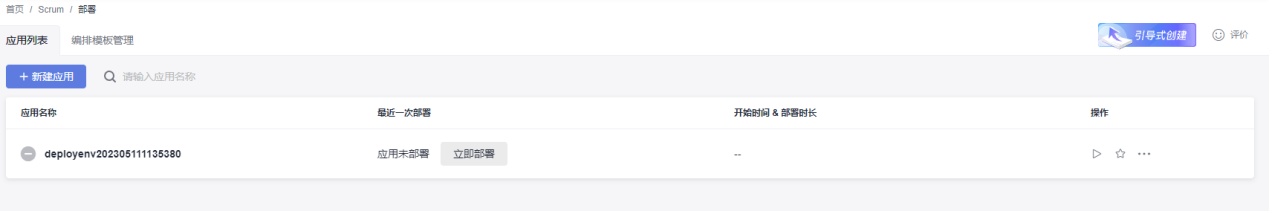
* 1. 主机设置

购买设置弹性云主机。下面是本项目购买并使用的服务器。



1. 服务器购买
   1. 部署配置

创建部署任务，使用springboot应用部署。



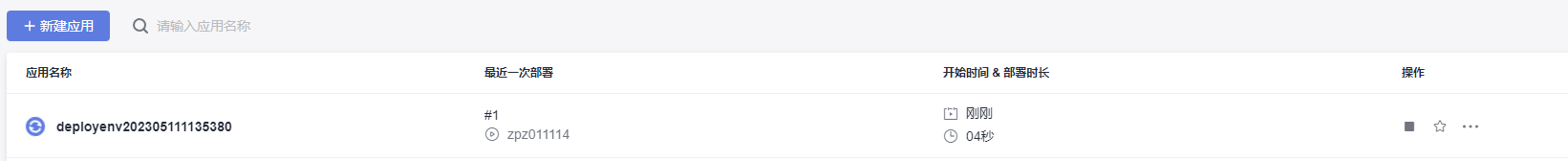
1. 部署任务创建

配置部署步骤，设置主机为购买的华为云主机。



1. 设置运行参数

最后点击执行部署，并成功执行部署。



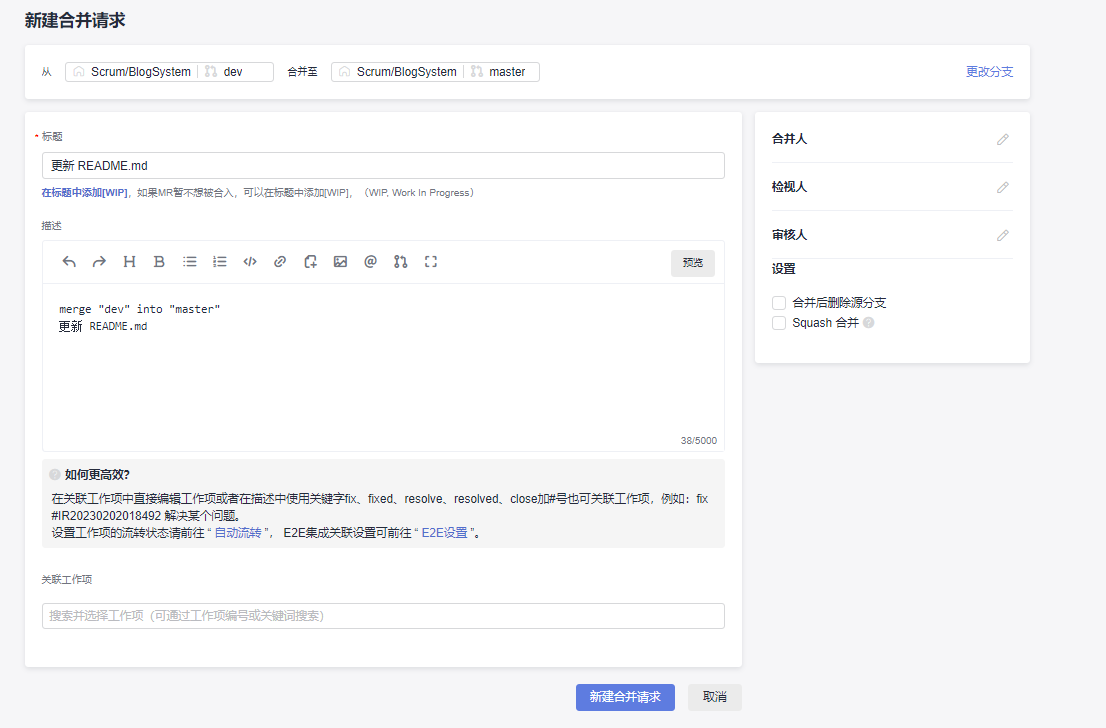
1. 运行部署任务
2. 分支合并和持续交付
   1. 分支合并

为了保证项目的稳定运行，需要有一个稳定的持续可用的分支“Master”，因此本项目不允许任何人直接在 Master 分支上进行代码开发，统一使用功能分支+合并请求的方式进行功能的开发与合并。

当项目开发完毕后，新建一个合并请求将开发版dev分支与稳定支持版master进行合并。

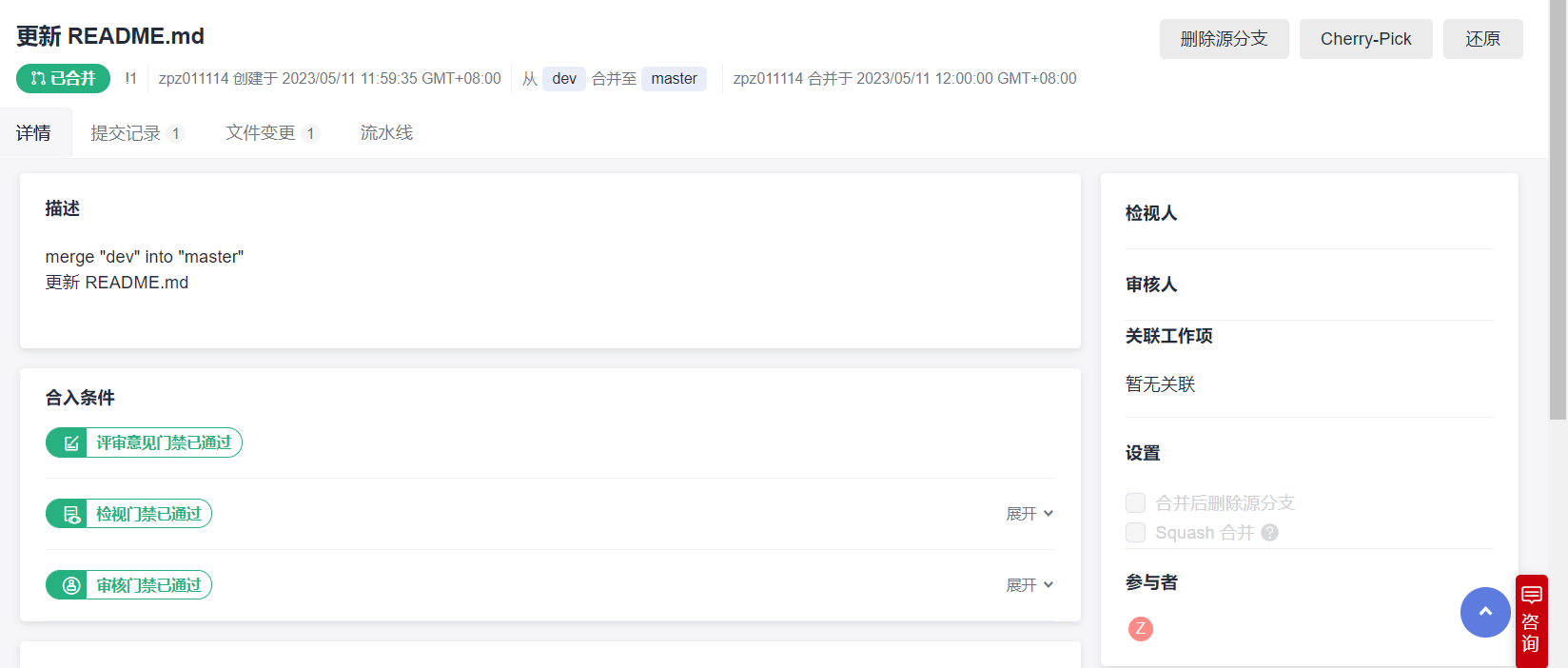


1. 合并开发分支与持久化分支



1. 合并分支配置

执行合并，并成功合并。



1. 合并结果
   1. 持续交付流水线

为了使得软件项目可以持续高效地集成和部署，本项目创建流水线任务，包括代码检查，编译构建，部署子任务等。使用华为云提供的子项目模块和人工审核模块进行审查。



1. 配置流水线

运行持续交付流水线，结果如下。



1. 运行流水线
2. 数据库设计
   1. 概念结构
      1. 实体分析

用户：用户名，密码，状态，管理员种类，评论数，文章数。

文章：创作时间，评论数量，收藏数量，状态，摘要，标签，文章图片存储路径，题目。

标签：标签id，名字。

收藏：收藏id，创建时间。

评论：评论id，内容。

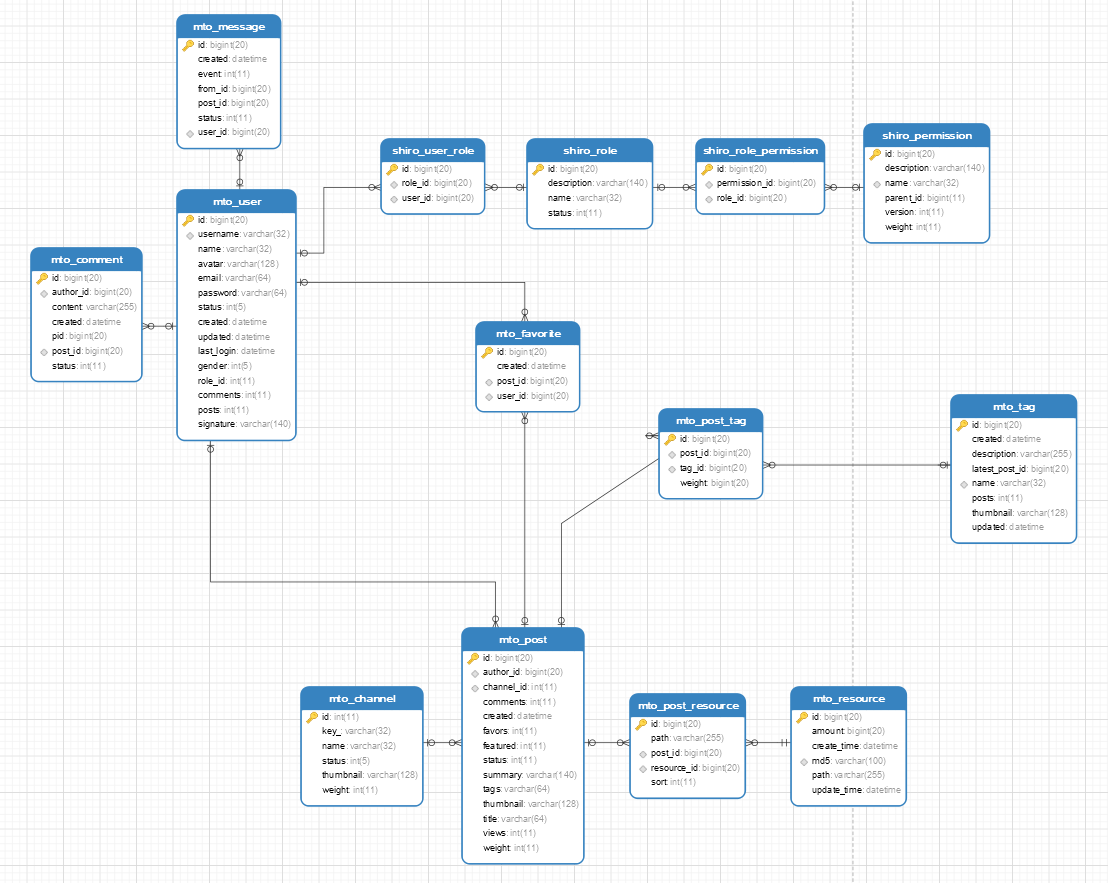
资源：文章中图片，用户头像图片的位置。

管理员：管理员种类。

权限信息：权限种类。

* + 1. 实体之间的联系分析

1. 用户和文章是一对多的关系。
2. 一篇文章能被多个用户收藏，一个用户也能收藏多篇文章。
3. 一篇文章或评论能被多个用户评论，一个用户也能评论多篇文章或评论。
4. 文章和标签是多对多的关系。
5. 栏目和文章是一对多的关系。
6. 管理员和权限之间是多对多的关系。
7. 文章中的图片和用户头像是一对一的关系。
   * 1. ER图



1. ER图
   1. 逻辑结构

用户（用户名id，昵称，头像，邮箱，密码，状态，创建时间，更新时间，最后登录时间，性别，管理员种类，评论数，文章数，个性签名）

文章（id，用户id，栏目id，评论数量，创作时间，收藏数量，推荐状态，状态，摘要，标签，文章图片存储路径，题目，访问数，权重）

栏目（id，唯一标识，名字，是否隐藏，图片存储位置，权重）

标签（id，创建时间，描述，最近一篇文章的id，名字，文章数，图片位置，更新时间）

收藏（id，创建时间，文章id，用户id）

评论（id，对应用户id，内容，评论时间，对评论的id，文章id，状态）

权限（id，描述，权限值，对应子功能，版本，权重）

管理员（id，描述，名字，状态）

权限设置（权限id，管理员id）

* 1. 物理结构

对应的栏目，比如博客，随笔等，可以通过管理员添加，隐藏和删除。

1. mto\_channel

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 备注 |
| Id | int(11) | Auto Increment | id |
| key | varchar(32) | Allow Null | 唯一标识 |
| name | varchar(32) | Allow Null | 名字 |
| status | int(5) |  | 是否隐藏 |
| thumbnail | varchar(128) | Allow Null | 图片存储位置 |
| weight | int(11) |  | 权重 |

记录用户写的评论，其中pid记录是否时对评论的回复，如果是0，则是对文章的评论，是其他数组则是对应的对改id评论的回复。

1. mto\_comment

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 备注 |
| id | bigint(20) | Auto Increment | Comment的id |
| author\_id | bigint(20) | Allow Null | 对应用户id |
| content | varchar(255) | Allow Null | 内容 |
| created | datetime | Allow Null | 评论时间 |
| pid | bigint(20) |  | 对评论的评论的id，为0则是对文章的评论 |
| post\_id | bigint(20) | Allow Null | 对应的文章id |
| status | int(11) |  | 状态 |

mto\_favorite记录用户收藏的文章。

1. mto\_favorite

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 备注 |
| id | bigint(20) | Auto Increment | Favorite的id |
| created | datetime | Allow Null | 创建时间 |
| post\_id | bigint(20) | Allow Null | 对应文章id |
| user\_id | bigint(20) | Allow Null | 对应用户id |

mto\_message记录每个用户操作的信息，比如是否有人回复，收藏文章和创建的文章等。

1. mto\_message

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 备注 |
| Id | bigint(20) | Auto Increment | Message的id |
| created | datetime Allow | Null | 创建时间 |
| event | int(11) |  | 事件类型，代表收藏，评论，回复等 |
| from\_id | bigint(20) | Allow Null | 消息来源用户Id |
| post\_id | bigint(20) | Allow Null | 对应文章id |
| status | int(11) |  | 阅读状态 |
| user\_id | bigint(20) | Allow Null | 对应用户id |

mto\_post记录文章的所有信息。

1. mto\_post

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 备注 |
| id | bigint(20) | Auto Increment | id |
| author\_id | bigint(20) | Allow Null | 用户ID |
| channel\_id | int(11) | Allow Null | 栏目ID |
| comments | int(11) |  | 评论数量 |
| created | datetime | Allow Null | 创作时间 |
| favors | int(11) |  | 收藏数量 |
| featured | int(11) |  | 推荐状态 |
| status | int(11) |  | 状态 |
| summary | varchar(140) | Allow Null | 摘要 |
| tags | varchar(64) | Allow Null | 标签 |
| thumbnail | varchar(128) | Allow Null | 文章图片存储路径 |
| title | varchar(64) | Allow Null | 题目 |
| views | int(11) |  | 访问数 |
| weight | int(11) |  | 权重 |

mto\_post\_attribute记录文章内容。

1. mto\_post\_attribute

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 备注 |
| id | bigint(20) |  | Id |
| content | longtext | Allow Null | 内容 |
| editor | varchar(16) | Allow Null | 编译方式 |

mto\_post\_resource记录文章对应的上传图片等文件资源。

1. mto\_post\_resource

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 备注 |
| id | bigint(20) | Auto Increment | Id |
| path | varchar(255) | Allow Null | 路径 |
| post\_id | bigint(20) | Allow Null | 文章id |
| resource\_id | bigint(20) |  | 资源id |
| sort | int(11) |  | 分类 |

mto\_post\_tag记录文章和对应标签。

1. mto\_post\_tag

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 备注 |
| id | bigint(20) | Auto Increment | Id |
| post\_id | bigint(20) | Allow Null | 文章id |
| tag\_id | bigint(20) | Allow Null | 标签id |
| weight | bigint(20) |  | 权重 |

mto\_resource记录上传头像，文章上传图片，或标签的图片等文件的位置等资源信息。

1. mto\_resource

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 备注 |
| id | bigint(20) | Auto Increment | Id |
| amount | bigint(20) |  | 引用次数 |
| create\_time | datetime | Allow Null | 创建时间 |
| md5 | varchar(100) |  | 签名 |
| path | varchar(255) |  | 存储路径 |
| update\_time | datetime | ON UPDATE CURRENT\_TIMESTAMP | 更新时间 |

发表文章时会生成对应的文章标签，mto\_tag会记录所有种类的标签。

1. mto\_tag

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 备注 |
| id | bigint(20) | Auto Increment | Id |
| created | datetime | Allow Null | 创建时间 |
| description | varchar(255) | Allow Null | 描述 |
| latest\_post\_id | bigint(20) |  | 最近一篇文章的id |
| name | varchar(32) |  | 名字 |
| posts | int(11) |  | 文章数 |
| thumbnail | varchar(128) | Allow Null | 图片位置 |
| updated | datetime | Allow Null | 更新时间 |

mto\_user记录用户的各种信息。

1. mto\_user

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 备注 |
| id | bigint(20) | Auto Increment | Id |
| username | varchar(32) | Allow | 用户名 |
| name | varchar(32) | Allow Null | 昵称 |
| avatar | varchar(128) | Allow Null | 头像 |
| email | varchar(64) | Allow Null | 邮箱 |
| password | varchar(64) | Allow Null | 密码 |
| status | int(5) |  | 状态 |
| created | datetime | Allow Null | 创建时间 |
| updated | datetime | Allow Null | 更新时间 |
| last\_login | datetime | Allow Null | 最后登录时间 |
| gender | int(5) |  | 性别 |
| role\_id | int(11) | Allow Null | 管理员种类 |
| comments | int(11) |  | 评论数 |
| posts | int(11) |  | 文章数 |
| signature | varchar(140) | Allow Null | 个性签名 |

shiro\_permission记录所有权限种类，用来显示菜单列表和根目录。

1. shiro\_permission

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 备注 |
| id | bigint(20) | Auto Increment | Id |
| description | varchar(140) | Allow Null | 权限描述 |
| name | varchar(32) |  | 权限值 |
| parent\_id | bigint(11) | Allow Null | 对应那种功能的子功能 |
| version | int(11) | Allow Null | 版本 |
| weight | int(11) |  | 权重 |

shiro\_role记录所有管理员种类，现在只设置了一种有全部权限的管理员。

1. shiro\_role

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 备注 |  |
| id | bigint(20) | Auto Increment | Id |  |
| description | varchar(140) | Allow Null | 描述 |  |
| name |  | varchar(32) |  | 名字 |
| status | int(11) |  | 状态 |  |

shiro\_role\_permission记录每种管理员拥有的权限，不同的管理员有不同的权限。

1. shiro\_role\_permission

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 备注 |
| id | bigint(20) | Auto Increment | Id |
| permission\_id | bigint(20) | Allow Null | 对应权限id |
| role\_id | bigint(20) | Allow Null | 对应管理员种类id |

shiro\_user\_role记录用户名对应的管理员种类，登录时会查找并赋予对应的权限。

1. shiro\_user\_role

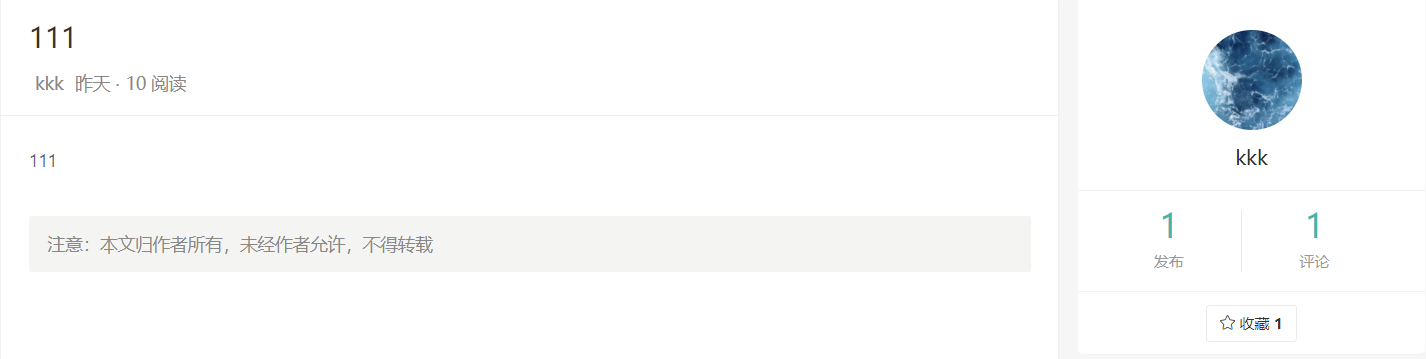
|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 约束 | 备注 |
| id | bigint(20) | Auto Increment | Id |
| role\_id | bigint(20) | Allow Null | 对应管理员种类id |
| user\_id | bigint(20) | Allow Null | 对应用户id |

1. 详细功能设计

本文创新功能有一下六条，在此进行详细的介绍。

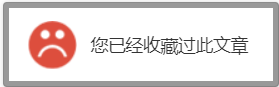
* 1. 文章收藏模块

点击文章页面右边栏的收藏按钮即可实现收藏操作，设计如下：



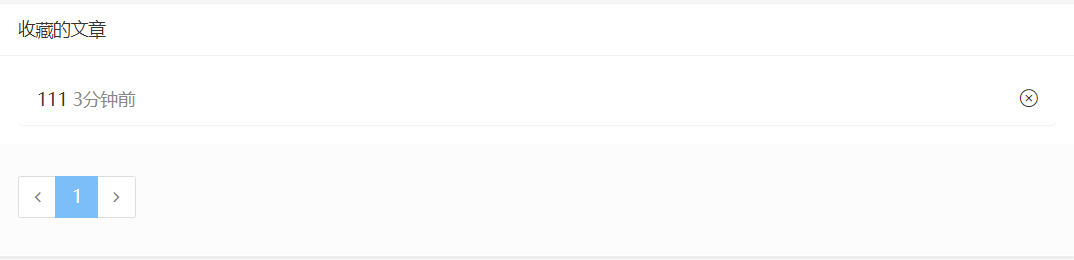
1. 文章收藏

如图会显示当前文章的收藏数量，若重复点击收藏按钮，会有浮窗提示已收藏过当前文章：



1. 重复收藏提示

在用户主页有收藏的文章栏，点击可跳转至收藏管理页面，并对收藏的文章进行删除管理。



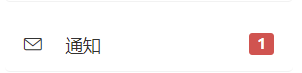
1. 收藏管理

本模块页面采用ftl动态页面+css+js编写：

|  |
| --- |
| <li class="list-group-item" id="loop-${target.id}">  <#if target??>  <a href="${base}/post/${target.id}" class="remove-padding-left">${target.title}</a>  <#else>  <a href="javascript:;" class="remove-padding-left">文章已删除</a>  </#if>  <span class="meta">  <span class="timeago">${timeAgo(row.created)}</span>  </span>  <div class="pull-right hidden-xs">  <#if owner>  <a class="act" href="javascript:void(0);" data-evt="unfavor" data-id="${target.id}">  <i class="icon icon-close"></i>  </a>  </#if>  </div>  </li> |

* 1. 消息通知模块

当有新消息时，用户主页左边栏会有提示：



1. 消息提示

点击通知栏目会跳转到通知列表查看消息，点击消息可查看消息详情：



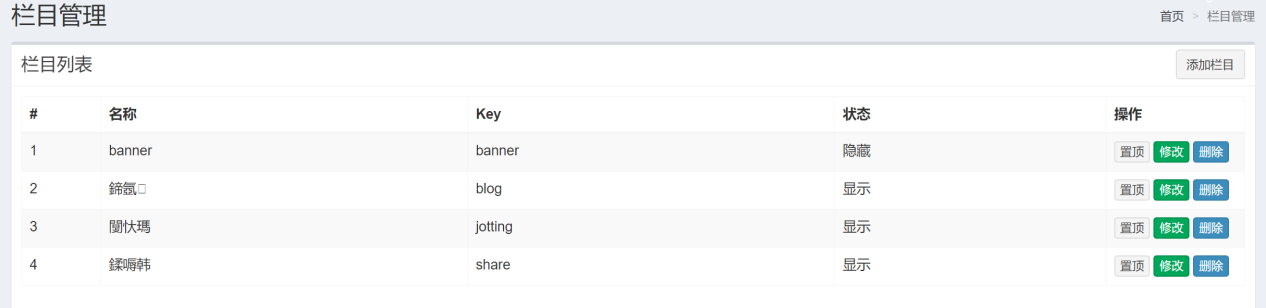
1. 消息详情

前端代码实现：

|  |
| --- |
| <a class="reply\_count\_area hidden-xs pull-right" href="#">  <div class="count\_set">  <abbr class="timeago">${timeAgo(row.created)}</abbr>  </div> </a> <div class="avatar pull-left">  <@utils.showAva row.from "media-object img-thumbnail avatar avatar-middle"/> </div> <div class="infos">  <div class="media-heading">  <#--<spanc lass="hidden-xs label label-rning">${row.channel.name}</span>-->  <a href="${base}/users/${row.from.id}">${row.from.name}</a>  <span>  <#if (row.event == 1)收藏了你的文章 -  <a href="${base}/post/${row.postId}">${row.post.title}</a>  <#elseif (row.event == 3)>评论了你的文章 - <a href="${base}/post/${row.postId}">点击查看详情</a>  <#elseif (row.event == 4)>  回复了你的评论 - <a href="${base}/post/${row.postId}">点击查看详情</a>  </#if>  </span>  </div> </div> |

* 1. 栏目管理模块

具体栏目管理实现的成品如下图所示：



1. 栏目管理实现图

可见我们具体要实现的功能在这个模块中有添加栏目的功能。而关于置顶，通过简单的js代码即可实现。

而关于修改相关的处理，我们可以和增加栏目一起放入一个新的页面中进行处理。具体当前需要处理的用户是哪个我们可以通过id号作为参数传输给相应的页面。

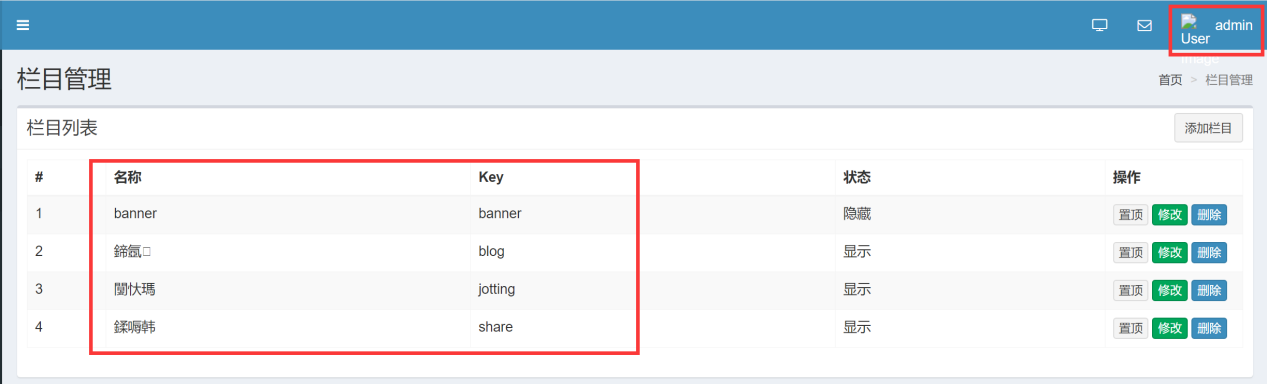
本次我们前端页面依然选择ftl-freemarker模板开发作为前端。

关于这个功能模块我们主要需要使用两个具体的页面。



1. 页面展示

首先是具体的固定标题的代码，依然通过参数的形式进行传入，实现针对不同的用户加载不同的用户数据和用户名。



1. 栏目管理展示

具体关于传入参数的代码如下所示：

1. 参数代码

|  |
| --- |
| <#if view??>  <input type="hidden" name="id" value="${view.id}" />  </#if>  <input type="hidden" name="weight" value="${view.weight!0}">  <input type="hidden" id="thumbnail" name="thumbnail" value="${view.thumbnail}"> |

其他部分变化的代码如下：

1. 参数变化代码

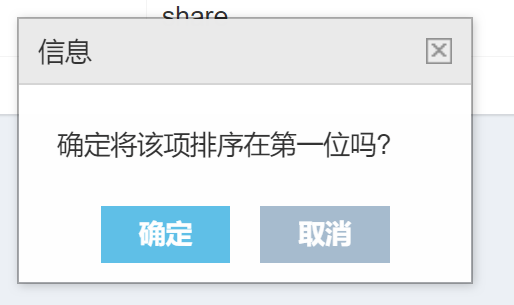
|  |
| --- |
| <div class="form-group">  <label class="col-lg-2 control-label">名称</label>  <div class="col-lg-3">  <input type="text" name="name" class="form-control" value="${view.name}" required></div>  </div>  <div class="form-group">  <label class="col-lg-2 control-label">唯一标识</label>  <div class="col-lg-3">  <input type="text" name="key" class="form-control" value="${view.key}" required>  </div>  </div>  <div class="form-group">  <label class="col-lg-2 control-label">导航栏状态</label>  <div class="col-lg-3">  <select name="status" class="form-control" data-select="${view.status}">  <option value="0">显示</option>  <option value="1">隐藏</option>  </select>  </div>  </div> |

对于页面的刷新我们采用Ajax无刷页面的方式进行刷新，js代码如下：

1. 无刷页面JS代码

|  |
| --- |
| function ajaxReload(json) {  if (json.code >= 0) {  if (json.message != null && json.message != '') {  layer.msg(json.message, {icon: 1});  }  window.location.reload();  } else {  layer.msg(json.message, {icon: 2});  }  } |

关于置顶相应的功能，我们同样通过js代码即可实现，需要注意的是，提供一个弹窗供用户进行确定即可。



1. 用户弹窗

具体的代码实现如下：

1. 弹窗实现JS代码

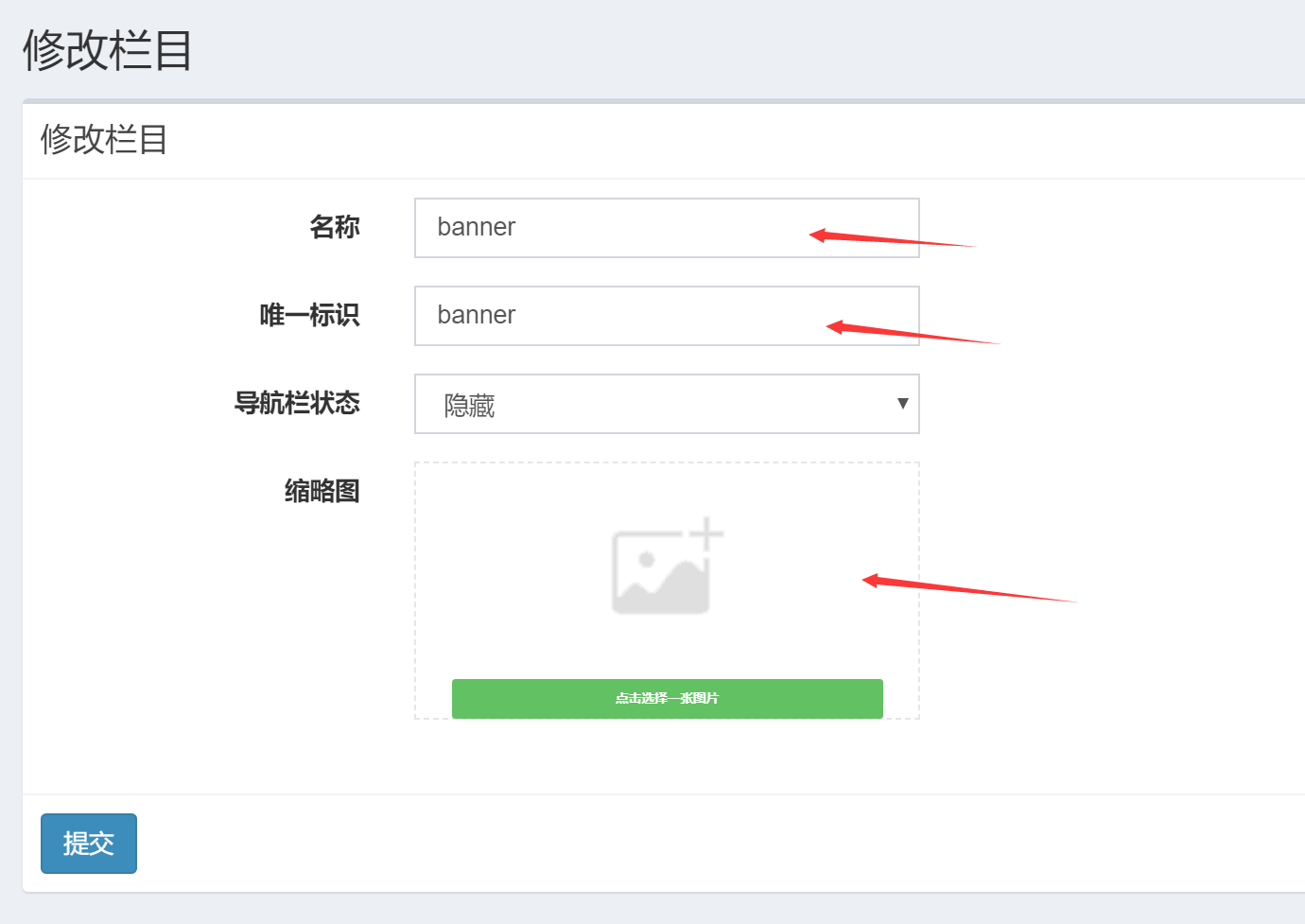
|  |
| --- |
| $(function () {  $('#dataGrid a[data-action="weight"]').bind('click', function(){  var that = $(this);  layer.confirm('确定将该项排序在第一位吗?', {  btn: ['确定','取消'], //按钮  shade: false //不显示遮罩  }, function(){  doUpdateWeight(that.attr('data-id'), 1);  }, function(){  });  return false;  }); |

关于删除功能同样如此：

1. 删除功能代码

|  |
| --- |
| // 删除  $('#dataGrid a[data-action="delete"]').bind('click', function () {  var that = $(this);  layer.confirm('确定删除此项吗?', {  btn: ['确定', '取消'], //按钮  shade: false //不显示遮罩  }, function () {  J.getJSON('${base}/admin/channel/delete', {id: that.attr('data-id')}, ajaxReload);  }, function () {  });  return false;  }); |

而对于相应的新增功能以及修改功能，我们需要使用到一个新的页面，也就是前面展示的view.ftl页面，具体展示的效果如下：

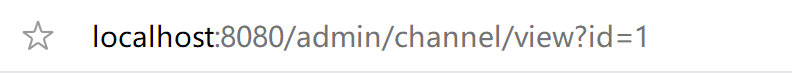


1. 页面展示

此页面对应于前面页面的两个按钮进行跳转，接下来将对这个页面进行分析。

公共存在的部分我们不做过多的赘述，这里主要讲解一下不同的地方。

需要注意的是，具体用户是点击了按钮新增，还是从用户跳转过来需要予以区分，这里重点查看url地址栏的不同。



1. 地址栏展示

可以观察看，对于用户部分修改跳转进来的情况，需要传入参数id，同时相应的修改输入框应该有所体现，具体的代码如下：

1. 输入框参数代码

|  |
| --- |
| <div class="form-group">  <label class="col-lg-2 control-label">名称</label>  <div class="col-lg-3">  <input type="text" name="name" class="form-control" value="${view.name}" required>  </div>  </div>  <div class="form-group">  <label class="col-lg-2 control-label">唯一标识</label>  <div class="col-lg-3">  <input type="text" name="key" class="form-control" value="${view.key}" required>  </div>  </div>  <div class="form-group">  <label class="col-lg-2 control-label">导航栏状态</label>  <div class="col-lg-3">  <select name="status" class="form-control" data-select="${view.status}">  <option value="0">显示</option>  <option value="1">隐藏</option>  </select>  </div>  </div> |

这个部分的重点在于用户可以自行上传相应的缩略图，所以我们需要js控制上传路径等信息。具体代码如下：

1. 上传控制代码

|  |
| --- |
| $(function() {  $('#upload\_btn').change(function(){  $(this).upload('${base}/post/upload?crop=thumbnail\_channel\_size', function(data){  if (data.status == 200) {  var path = data.path;  $("#thumbnail\_image").css("background", "url(" + path + ") no-repeat scroll center 0 rgba(0, 0, 0, 0)");  $("#thumbnail").val(path);  }  });  });  }) |

那么具体实现的结果效果图如下：



1. 页面展示

修改完毕点击提交即可。如此，栏目管理功能成功实现。

* 1. 角色管理模块

首先我将展示角色管理模块的具体样式：

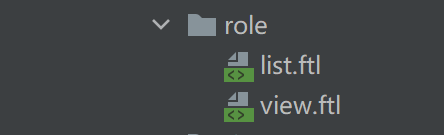


1. 角色管理模块样例图

我们需要重点实现的功能在于关键词的搜索功能，以及添加角色和修改角色信息对应的功能。具体实现这个模块需要使用两个页面。一个是具体的角色管理的展示模块，另一个主要用于新增角色和修改角色信息。

角色管理主要负责对系统中存在的很多用户进行标注，区分一些关键用户，可能后续会为其分配不同的权限等。

具体我们同样为其分配两个静态页面，具体如下：



1. 角色管理模块样例图

针对页面中每一项的修改和删除功能，我们具体通过#list的形式进行实现，具体实现的代码如下：

1. 修改和删除代码

|  |
| --- |
| <#list page.content as row>  <tr>  <td class="text-center">${row.id}</td>  <td>${row.name}</td>  <td class="text-center">  <#if row.id != 1>  <a href="${base}/admin/role/view?id=${row.id}" class="btn btn-xs btn-success">  <i class="fa fa-check-square-o"></i> 修改  </a>  <a href="javascript:void(0);" class="btn btn-xs btn-primary" data-id="${row.id}"  data-action="delete">  <i class="fa fa-bitbucket"></i> 删除  </a>  <#else>  <a href="javascript:void(0);" class="btn btn-xs disabled">  <i class="fa fa-check-square-o"></i> 不可编辑</a>  </#if>  </td>  </tr>  </#list> |

同样对于删除按钮的处理，我们需要设计JavaScript事件来提示用户是否确定进行删除。效果如下：



1. 确定删除图

具体的js代码实现如下：

1. 删除控制代码

|  |
| --- |
| $(function () {  // 删除  $('#dataGrid a[data-action="delete"]').bind('click', function () {  var that = $(this);  layer.confirm('确定删除此项吗?', {  btn: ['确定', '取消'], //按钮  shade: false //不显示遮罩  }, function () {  doDelete(that.attr('data-id'));  }, function () {  });  return false;  });  }) |

那么对于当用户点击了确定按钮以后，我们需要调用函数doDelete，具体该函数通过js的实现代码如下：

1. 删除JS代码

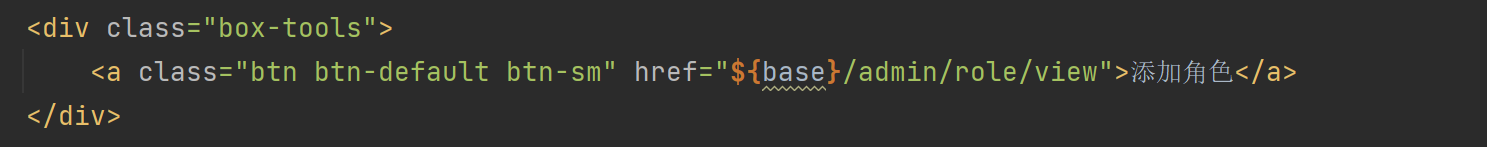
|  |
| --- |
| function doDelete(ids) {  J.getJSON('${base}/admin/role/delete', J.param({'id': ids}, true), ajaxReload);  } |

注意到，在删除的过程中，我们同样调用了函数ajaxReload，这种函数可以通过使用ajax无刷页面来实现删除后的动态修改，立刻就可以看到效果。当然，具体要删除的东西的具体休息也要传给后台进行处理。代码如下：

1. 无刷页面控制

|  |
| --- |
| function ajaxReload(json) {  if (json.code >= 0) {  if (json.message != null && json.message != '') {  layer.msg(json.message, {icon: 1});  }  $('#qForm').submit();  } else {  layer.msg(json.message, {icon: 2});  }  } |

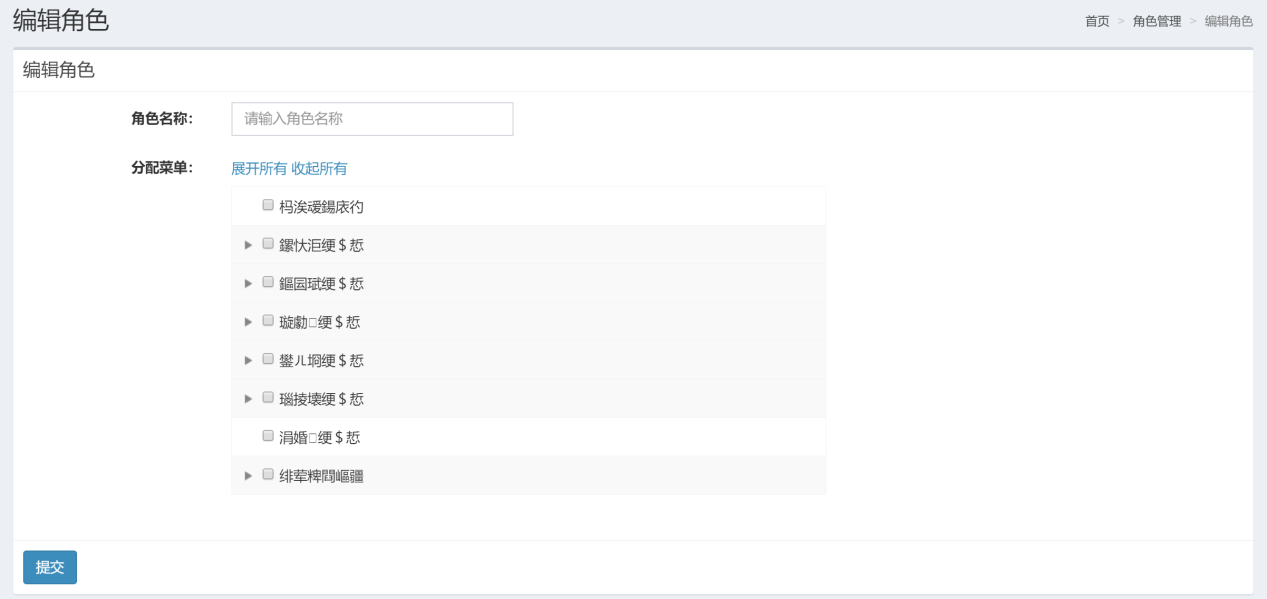
我们同样注意到，在点击添加角色以后，会触发页面的href跳转，具体如下：



1. 代码展示

之后我们会跳转到需要分析的另一个页面，也就是view页面。

具体页面的展示风格如下：



1. 角色编辑样例

页面中的乱码是由于编码问题，仅用于表示某些分组即可。

首先我们需要引入相应的JQuery的包，通过base标签链接当前相对链接。

代码如下：

1. 标签引入

|  |
| --- |
| <link href="${base}/dist/vendors/treetable/css/jquery.treetable.css" rel="stylesheet" type="text/css"/>  <link href="${base}/dist/vendors/treetable/css/jquery.treetable.theme.default.css" rel="stylesheet" type="text/css"/>  <script src="${base}/dist/vendors/treetable/jquery.treetable.js"></script> |

首先最开始我们需要使用#macro标签来实现循环标记功能，将相应的列表进行展开，并且如果不存在字子集就不继续展开，代码如下：

1. JQuery子集展开代码

|  |
| --- |
| <#macro treeIterator nodes>  <#-- 循环节点-->  <#list nodes as row>  <tr data-tt-id="${row.id}" <#if (row.parentId??) > data-tt-parent-id="${row.parentId}" </#if>>  <td>  <input type="checkbox" name="perms" id="${row.name}-${row.id}" value="${row.id}"> ${row.description}  </td>  </tr>  <#-- 判断是否有子集 -->  <#if row.items??>  <@treeIterator nodes=row.items />  </#if>  </#list>  </#macro> |

相应的角色名称输入框我们除了需要让用户自己输入以外，如果发现是通过修改按钮跳转过来的，还需要预先在相应的框中赋值当前用户的数值，具体代码如下：

1. 角色名称控制代码

|  |
| --- |
| <div class="form-group">  <label for="name" class="col-lg-2 control-label">角色名称：</label>  <div class="col-lg-3">  <input type="text" class="form-control" placeholder="请输入角色名称" name="name"  value="${view.name}" required>  </div>  </div> |

我们注意到有两个按钮，如下：



1. 分配菜单样例

我们具体实现这个代码是通过a标签的onclick事件进行处理，对于这个表格的相应处理可以调用JQuery自带的一些函数实现即可，代码如下：

1. 展开和收起代码实现

|  |
| --- |
| <caption>  <a href="#" onclick="jQuery('#dataGrid').treetable('expandAll'); return false;">展开所有</a>  <a href="#" onclick="jQuery('#dataGrid').treetable('collapseAll'); return false;">收起所有</a>  </caption>  <tbody>  <@treeIterator nodes=permissions />  </tbody> |

后续关于一些表格还需要一些js函数的支持，具体代码如下：

1. 函数支持

|  |
| --- |
| <#list view.permissions as p>  perm.push('${p.id}');  <#if p.children??>  <#list p.children as c>  perm.push('${c.id}');  <#if c.children??>  <#list c.children as a>  perm.push('${a.id}');  </#list>  </#if>  </#list>  </#if>  </#list> |

关于下来菜单我们需要补充一些代码，dataGrid标签需要标识出父子下拉菜单需要展开到什么位置，需要通过js处理一下长度，具体代码如下：

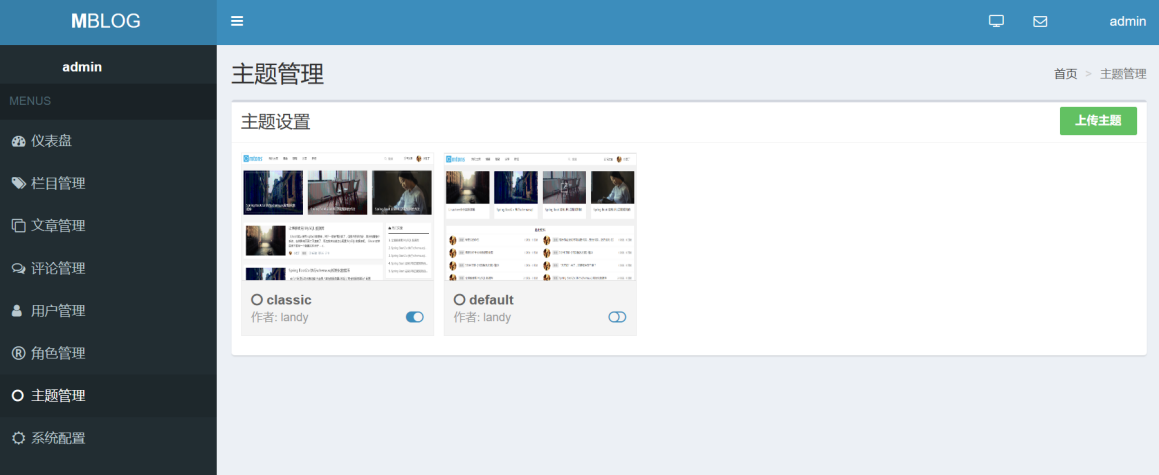
1. 表格长度处理

|  |
| --- |
| $(function () {  var temp = '';  for (var i = 0; i < parentArray.length; i++) {  if (i > 0) {  temp += '.' + parentArray[i];  } else {  temp += parentArray[i];  }  $('tr[data-tt-id="' + temp + '"]>td>input:checkbox').prop("checked", $(this).prop("checked"));  }  }  }  })  }); |

综上，这部分模块的解析就结束了。

* 1. 主题管理模块
     1. 主题管理UI设计展示

主题管理部分的UI展示如下图所示：



1. 主题管理UI展示图
2. 主题管理UI主要代码

|  |
| --- |
| <div class="box">  <div class="box-header with-border">  <h3 class="box-title">主题设置</h3>  <div class="box-tools">  <div class="upload-btn">  <label style="width: 80px; margin: 0;">  <span>上传主题</span>  <input id="upload\_btn" type="file" name="file" accept="application/x-zip-compressed" title="点击添加">  </label>  </div>  </div>  </div> |

* + 1. 主题风格设计

不同的主题对应不同的css文件，在这设计了两种风格，可再通过上传css文件添加。两种主题的设计界面如下：



1. 主题风格1图



1. 主题风格2图

相关css代码如下：

1. 风格1主要代码

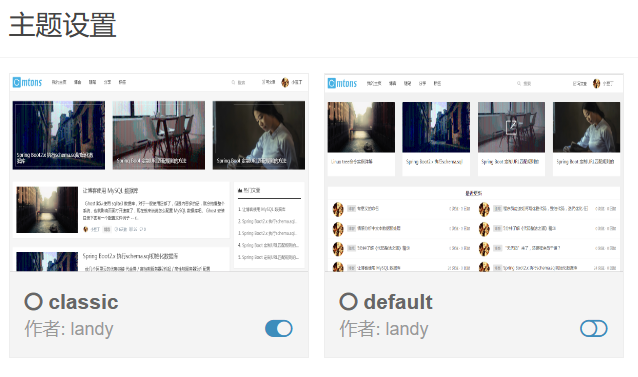
|  |
| --- |
| .layout-top-nav .main-header .navbar {  margin-left: 0;  }  .main-header #navbar-search-input.form-control {  background: rgba(255, 255, 255, 0.2);  border-color: transparent;  }  .main-header #navbar-search-input.form-control:focus,  .main-header #navbar-search-input.form-control:active {  border-color: rgba(0, 0, 0, 0.1);  background: rgba(255, 255, 255, 0.9);  }  .main-header #navbar-search-input.form-control::-moz-placeholder {  color: #ccc;  opacity: 1;  } |

1. 风格2主要代码

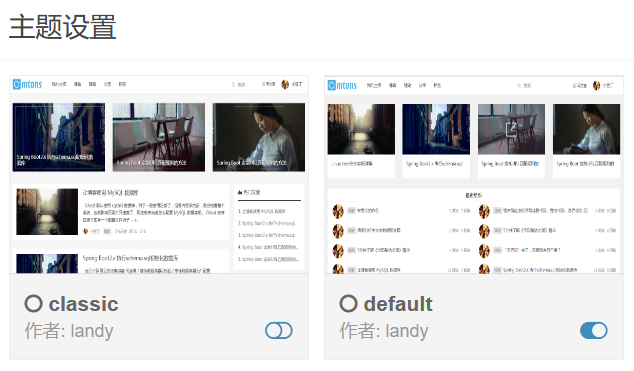
|  |
| --- |
| .layout-top-nav .main-header .navbar {  margin-left: 0;  }  .main-header #navbar-search-input.form-control {  background: rgba(255, 255, 255, 0.2);  border-color: transparent;  }  .main-header #navbar-search-input.form-control:focus,  .main-header #navbar-search-input.form-control:active {  border-color: rgba(0, 0, 0, 0.1);  background: rgba(255, 255, 255, 0.9);  } |

* + 1. 主题切换功能

主题切换功能通过修改options['theme']的值改变，前端设计多个单选按钮，控制全局的css的选择。切换操作如下图所示：



1. 主题1选择图



1. 主题2选择图

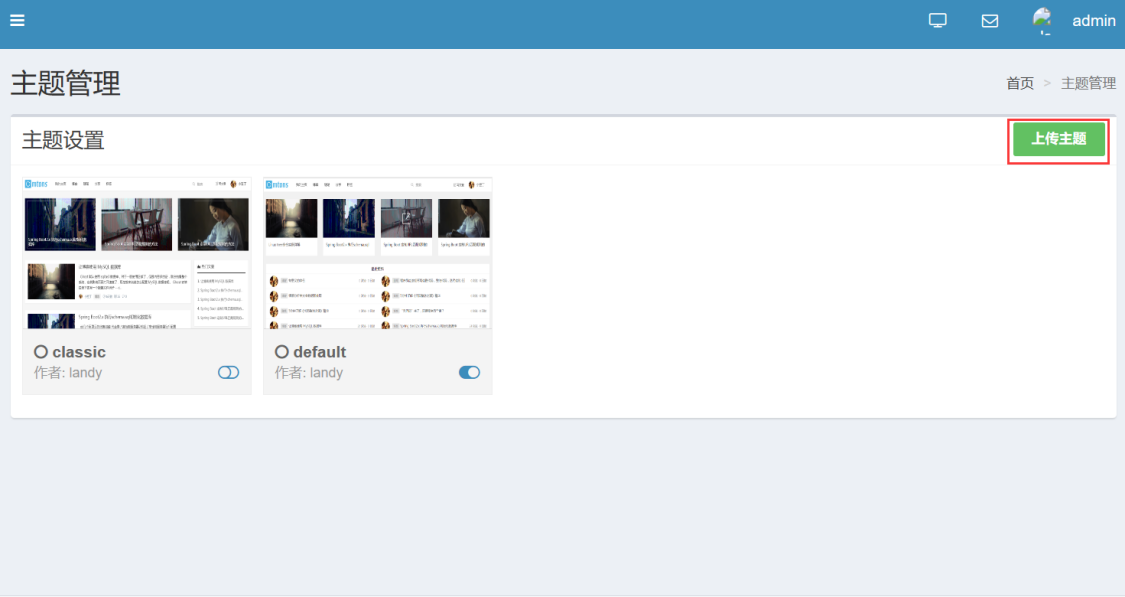
主题切换主要代码如下：

1. 主题切换核心代码

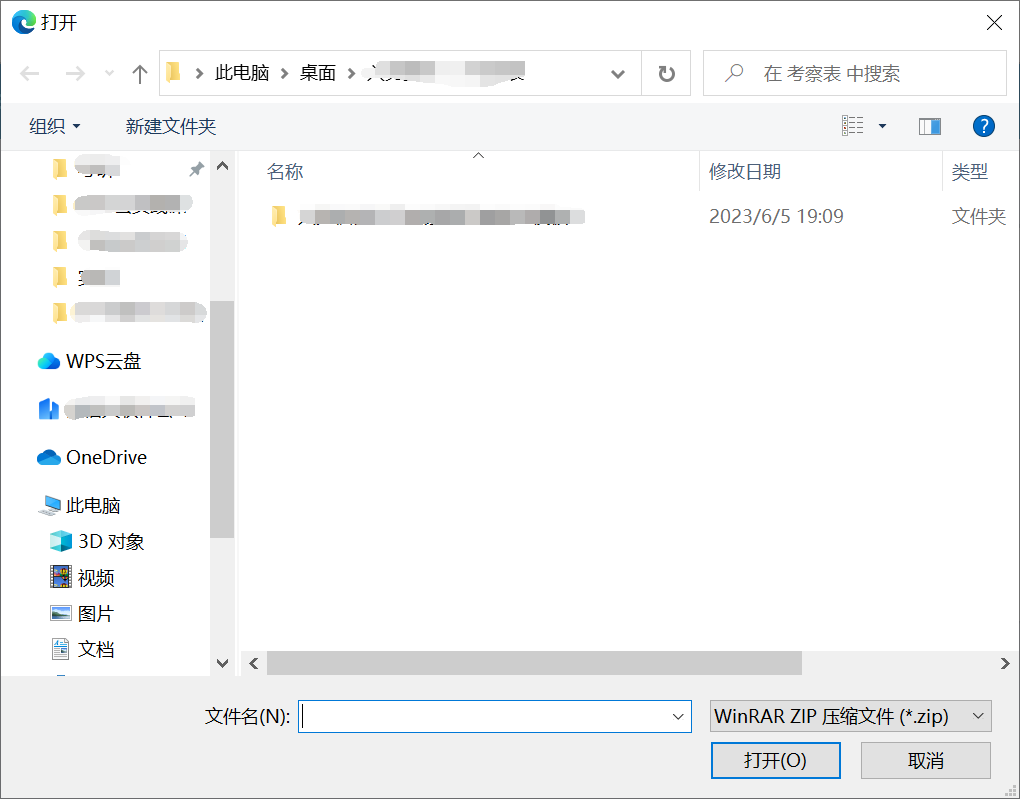
|  |
| --- |
| <#if options['theme'] == row.name >  <a href="javascript:;" class="pull-right">  <i class="fa fa-lg fa-toggle-on"></i>  </a>  <#else><a href="javascript:;" class="pull-right" data-action="toggle" data-value="${row.name}">  <i class="fa fa-lg fa-toggle-off"></i>  </a>  </#if> |

* + 1. 主题上传功能

点击右上角按钮“上传主题”，即可在文档中选择.zip文件进行上传。操作如下图所示：



1. 上传主题操作图



1. 上传.zip文件示意图

上传功能主要代码如下：

1. 上传功能核心代码

|  |
| --- |
| <label style="width: 80px; margin: 0;">  <span>上传主题</span>  <input id="upload\_btn" type="file" name="file" accept="application/x-zip-compressed" title="点击添加">  </label> |

* 1. 文章管理优化
     1. 文章管理UI设计展示

文章管理部分的整体UI设计如下图所示：



1. 文章管理部分的整体UI图

文章管理部分UI设计核心代码如下：

1. 文章管理部分UI设计核心代码

|  |
| --- |
| <h1>文章管理</h1>  <ol class="breadcrumb">  <li><a href="${base}/admin">首页</a></li>  <li class="active">文章管理</li>  </ol>  <h3 class="box-title">文章列表</h3>  <div class="box-tools">  <a class="btn btn-default btn-sm" href="${base}/admin/post/view">新建</a>  <a class="btn btn-default btn-sm" href="javascrit:;" data-action="batch\_del">批量删除</a>  </div>  </div>  <div class="box-body">  <form id="qForm" class="form-inline search-row">  <input type="hidden" name="pageNo" value="${page.number + 1}"/>  <div class="form-group">  <select class="form-control" name="channelId" data-select="${channelId}">  <option value="0">查询所有栏目</option>  <#list channels as row> |

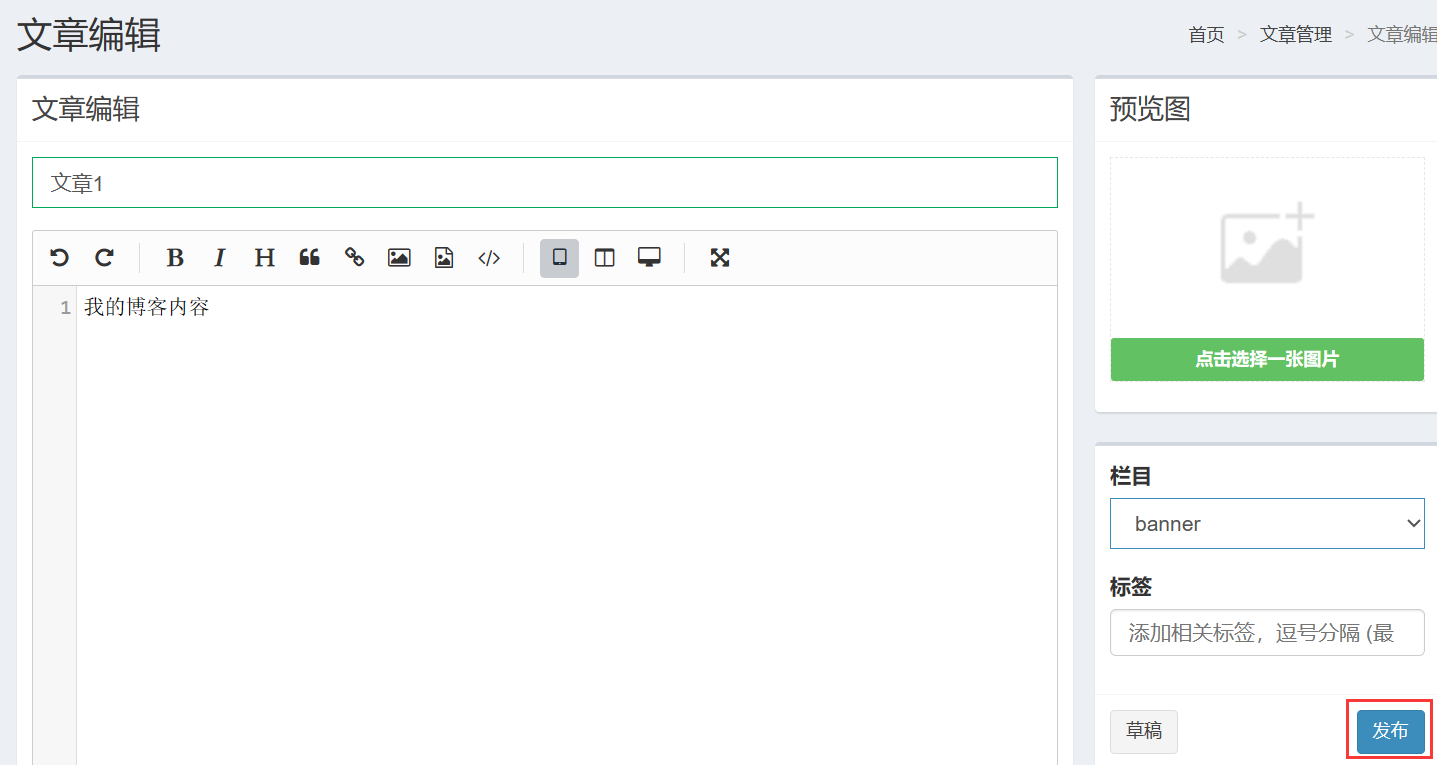
* + 1. 文章的新建功能

点击右上角“新建”按钮，即可新建文章，如下图所示：

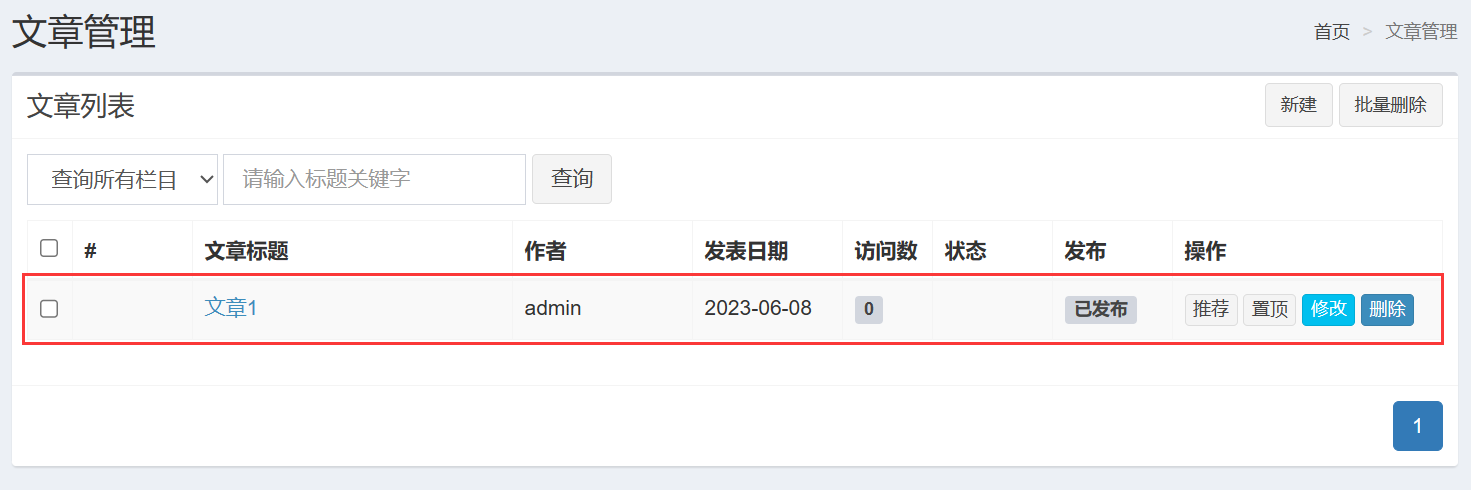


1. 新建文章操作图

进入到新建文章的页面再输入相应内容后点击右下角“发布”按钮即可完成新建一篇文章，如下图所示：



1. 发布操作图



1. 发布完成图

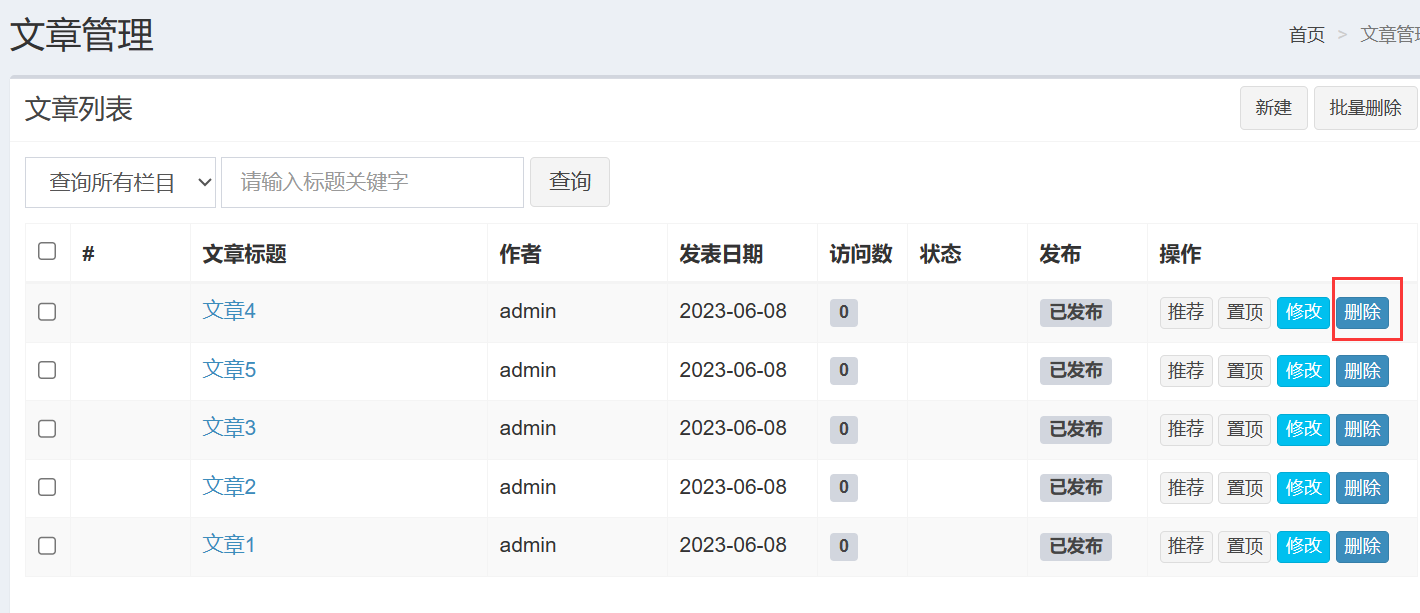
新建文章核心代码如下：

1. 新建文章核心代码

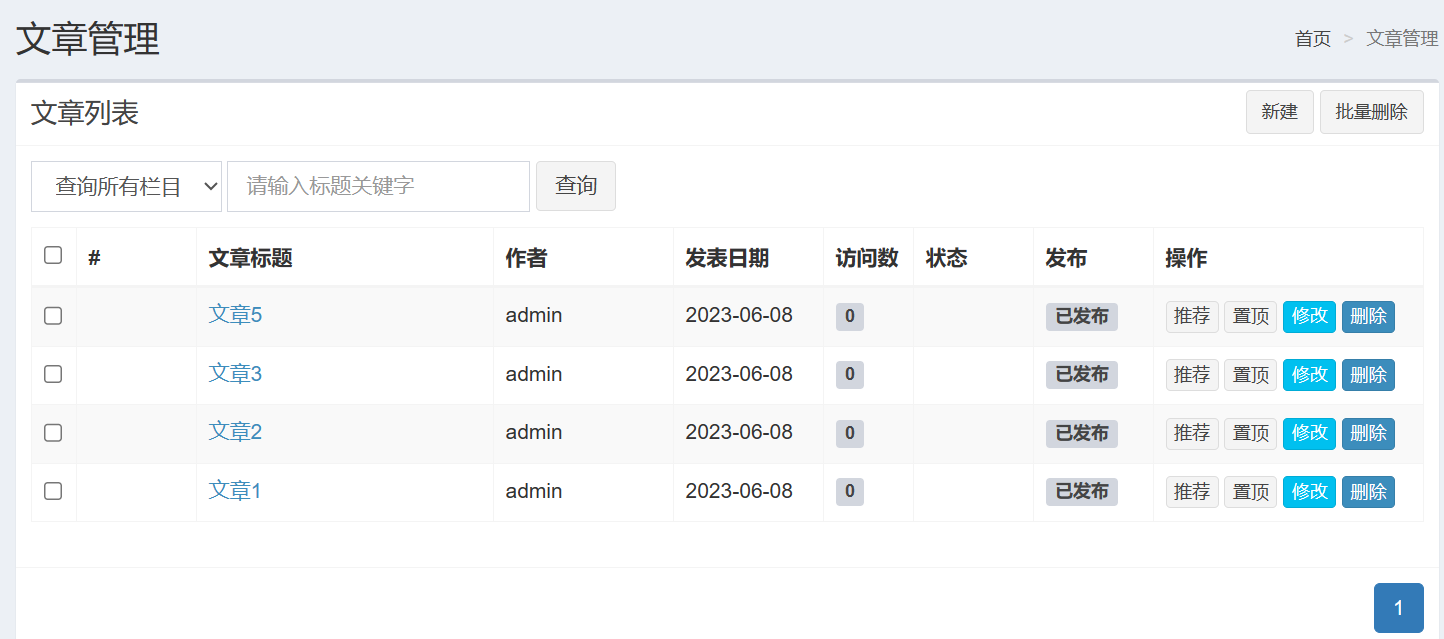
|  |
| --- |
| $(function() {  $('#upload\_btn').change(function(){  $(this).upload('${base}/post/upload?crop=thumbnail\_post\_size', function(data){  if (data.status == 200) {  var path = data.path;  $("#thumbnail\_image").css("background", "url(" + path + ") no-repeat scroll center 0 rgba(0, 0, 0, 0)");  $("#thumbnail").val(path);  }  });  });  $('button[event="post\_submit"]').click(function () {  var status = $(this).data('status');  $("input[name='status']").val(status);  $("form").submit();  }); |

* + 1. 文章的删除功能

若需要删除某一篇文章，可以点击对应文章右边的“删除”按钮即可。操作如下图所示：

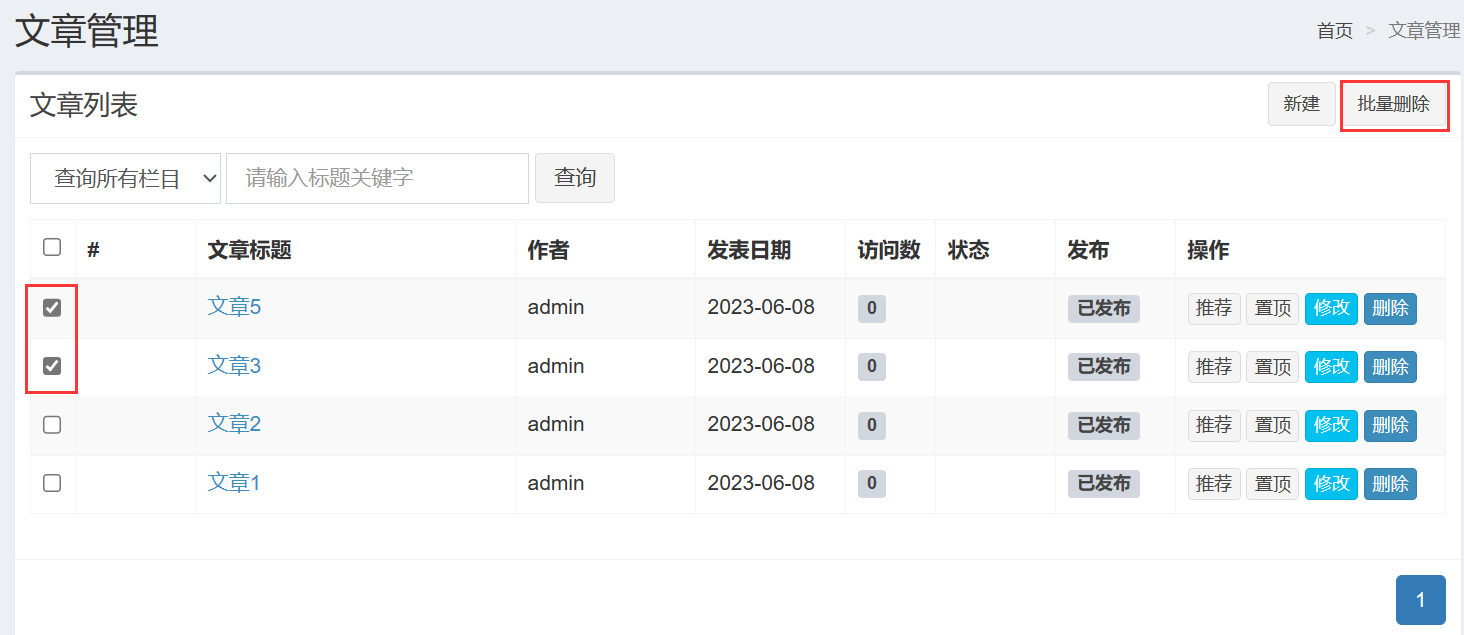


1. 删除操作图



1. 删除完成图

若需要批量处理，可以选定对应文章后点击右上角的批量删除进行处理。操作如下图所示：



1. 批量删除操作图



1. 批量删除完成图

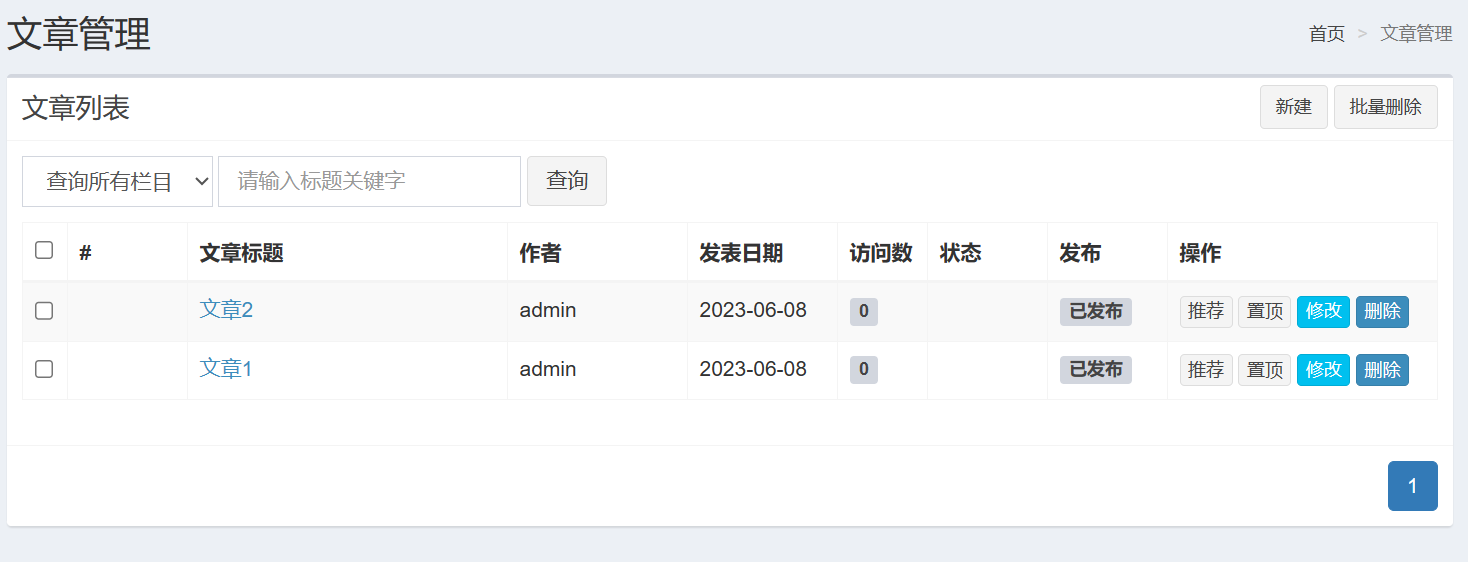
文章删除部分核心代码如下：

1. 文章删除部分核心代码

|  |
| --- |
| <a class="btn btn-default btn-sm" href="javascrit:;" data-action="batch\_del">批量删除</a>  $('a[data-action="batch\_del"]').click(function () {  var check\_length=$("input[type=checkbox][name=id]:checked").length;  if (check\_length == 0) {  layer.msg("请至少选择一项", {icon: 2});  return false;  }  var ids = [];  $("input[type=checkbox][name=id]:checked").each(function(){  ids.push($(this).val());  });  layer.confirm('确定删除此项吗?', {  btn: ['确定','取消'], //按钮  shade: false //不显示遮罩  }, function(){  doDelete(ids);  }, function(){  });  }); |

* + 1. 文章的查询功能

可以通过搜索文章标题关键词查询相应文章，操作如下图所示：



1. 查询前示意图



1. 查询完成示意图

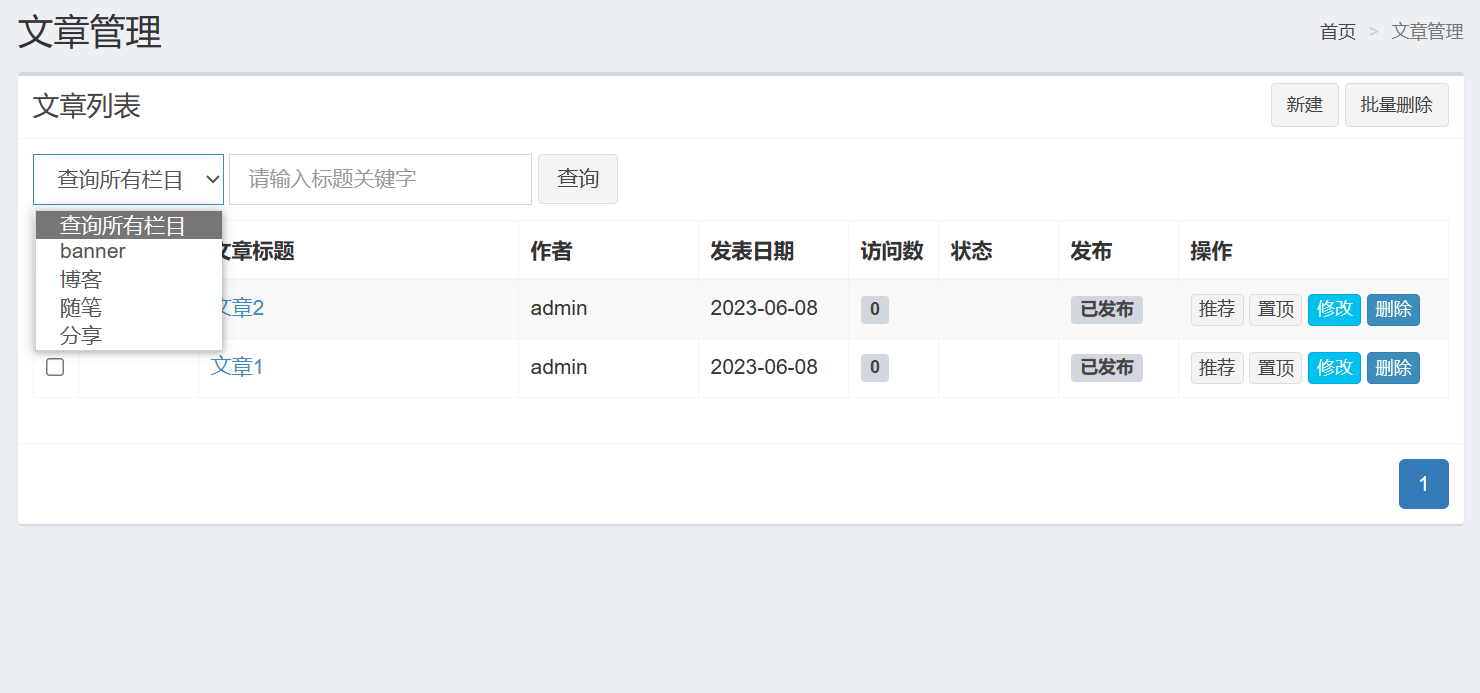
文章查询核心代码如下：

1. 文章查询核心代码

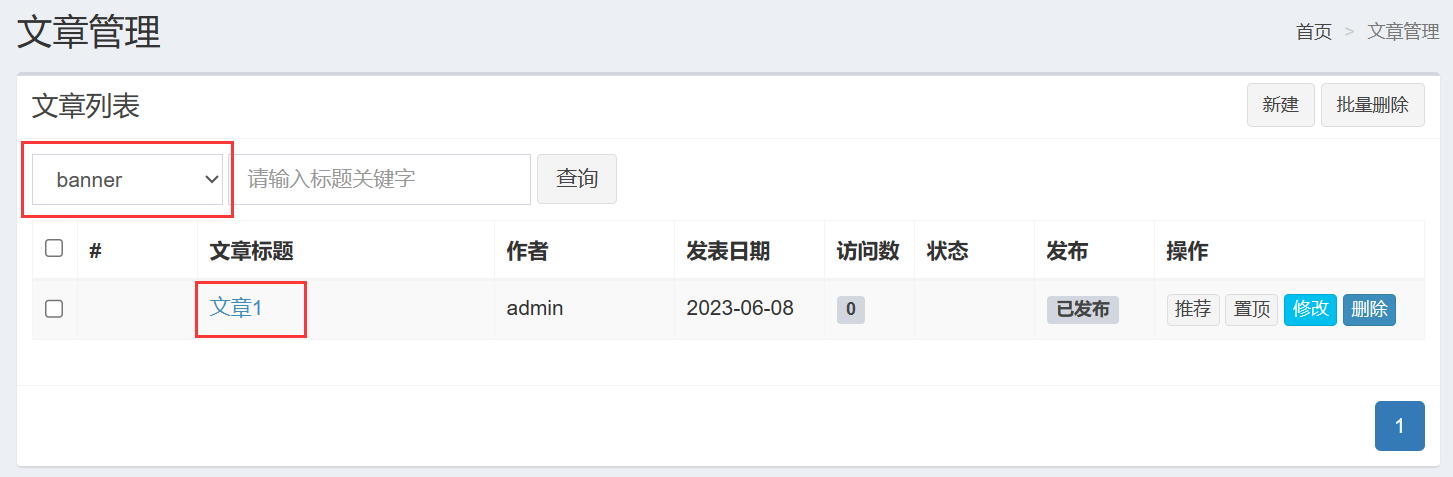
|  |
| --- |
| <div class="form-group">  <input type="text" name="title" class="form-control" value="${title}" placeholder="请输入标题关键字">  </div>  <button type="submit" class="btn btn-default">查询</button> |

* + 1. 文章的筛选功能

文章一共分为五个栏目，可以根据文章栏目对文章进行筛选。操作如下图所示：



1. 各栏目示意图



1. 筛选完成示意图

文章筛选功能主要代码如下：

1. 文章筛选功能核心代码

|  |
| --- |
| <input type="hidden" name="pageNo" value="${page.number + 1}"/>  <div class="form-group">  <select class="form-control" name="channelId" data-select="${channelId}">  <option value="0">查询所有栏目</option>  <#list channels as row>  <option value="${row.id}">${row.name}</option>  </#list>  </select>  </div> |