个人中心

## 15.1 个人中心概览

个人中心实现效果如下图所示：



功能主要包含如下:

个人文章管理、发布文章。

## 15.2 发布文章

### 15.2.1 要求

发布文章要求内容支持富文本编辑，输入项目包含文章标题、标题图片、文章内容、频道和分类。在频道和分类支持联动效果。

### 15.2.2 前端

页面使用bootstrap+jsp实现。富文本编辑使用第三方KindEditor实现。

页面加载完成后立即初始化kindEitor 组件，代码如下所示：

KindEditor.ready(function(K) {  
  
window.editor1 = K.create('textarea[name="content1"]', {  
  
//指定样式  
  
cssPath : '/resource/kindeditor/plugins/code/prettify.css',  
  
//指定文件管理器  
  
uploadJson : '/resource/kindeditor/jsp/upload\_json.jsp',  
  
//文件显示  
  
fileManagerJson : '/resource/kindeditor/jsp/file\_manager\_json.jsp',  
  
//允许文件上传  
  
allowFileManager : true,  
  
//对象创建完成的回调函数  
  
afterCreate : function() {  
  
var self = this;  
  
K.ctrl(document, 13, function() {  
  
self.sync();  
  
document.forms['example'].submit();  
  
});  
  
K.ctrl(self.edit.doc, 13, function() {  
  
self.sync();  
  
document.forms['example'].submit();  
  
});  
  
}  
  
});  
  
prettyPrint();  
  
});

文件上传前端代码直接使用如下格式即可：

<input type=\*"file" class=\*"form-control" id="file" name="file">

联动效果的实现

这里的频道内容是固定的，既使用异步加载的方式，也就是在进入页面以后利用ajax技术获取所有的频道内容然后填入到下拉框当中。也可以直接渲染，也就是在进入页面之前在Controller层中获取到所有频道内容，存入到request对象中，jsp渲染的时候从request中获取内容，利用C标签将内容填充到下拉框当中。

异步加载实现频道列表下拉代码：

//自动加载文章的栏目（频道）

//自动加载文章的栏目

$.ajax({  
 type:"get",  
 url:"/article/getAllChn",  
 success:function(list){  
 $("#channel").empty();  
 for(var i in list){  
 if(${article.channelId}==list[i].id){  
 $("#channel").append("<option selected value='"+list[i].id+"'>"+list[i].name+"</option>")

// 频道的回显  
 $("#category").empty();  
 //根据ID 获取栏目下的分类  
 $.get("/article/getCatsByChn",{channelId:${article.channelId}},function(catlist){  
   
 for(var cati in catlist){  
 if(catlist[cati].id==${article.categoryId}){  
 $("#category").append("<option selected value='"+catlist[cati].id+"'>"+catlist[cati].name+"</option>")  
 }else{  
 $("#category").append("<option value='"+catlist[cati].id+"'>"+catlist[cati].name+"</option>")  
 }  
 //处理回显  
   
 }  
   
 })

}else{  
 $("#channel").append("<option value='"+list[i].id+"'>"+list[i].name+"</option>")  
 }  
   
 }  
 }  
   
})

直接渲染频道列表下拉代码：

<div class="form-group row ">  
 <label for="channel">文章栏目</label>  
 <select class="custom-select custom-select-sm mb-3" id="channel" name="channelId">  
 <option value="0">请选择</option>   
 <c:forEach items="${channels}" var="channel">  
 <option value="${channel.id}" ${channel.id==article.channelId?"selected":""} > ${channel.name}</option>   
 </c:forEach>  
 </select>  
 </div>

频道与分类实现联动效果：

为频道组件绑定change事件，当频道发生改变以后，利用ajax技术获取分类列表，然后清楚分类列表中的内容，最后将重新获取到的分类列表追加到分类列表组件当中。实现的代码如下：

/\*\*  
 \* 函数用于根据频道内容获取分类列表内容  
 \*  
 \*/  
 function changeChannel(){  
   
  
 //先清空原有的栏目下的分类  
 $("#category").empty();  
 var cid =$("#channel").val();//获取当前的下拉框的id  
 //根据ID 获取栏目下的分类  
 $.get("/article/listCatByChnl",{chnlId:cid},function(resultData){  
 if(resultData.result==1){ //后端处理正确  
 var list = resultData.data; //得到分类列表的数据  
 //遍历分类列表数据  
 for(var i in list){  
 //该分类就是文章的分类  
 if(list[i].id==${article.categoryId}){  
 // 该项处于选中状态  
 $("#category").append("<option value='"+list[i].id   
 +"' selected >"+list[i].name+"</option>")  
 }else{  
 //  
 $("#category").append("<option value='"+list[i].id+"'>"  
 +list[i].name+"</option>")  
 }//end if  
 }//end for  
 }//end if   
 })// end $.get(  
  
  
}//end function

发布文章内容：

function publish(){  
  
  
 //序列化表单数据带文件  
 var formData = new FormData($( "#form" )[0]);  
 //改变formData的值  
 //editor1.html() 是富文本的内容  
 formData.set("content",editor1.html());  
   
 $.ajax({  
 type:"post",  
 data:formData,  
 // 告诉jQuery不要去处理发送的数据  
 processData : false,  
 // 告诉jQuery不要去设置Content-Type请求头  
 contentType : false,  
 url:"/article/update",  
 success:function(obj){  
 if(obj)  
 {  
 alert("修改成功!")  
 // location="/article/listMyArticle";  
 $("#center").load("/user/myarticlelist")  
 }else{  
 alert("修改失败")  
 }  
   
 }  
 })  
}

### 15.2.3 后端

#### 工具包依赖

项目pom.xml 文件中增加如下属性和依赖：

<!-- 定义主要版本号 -->

<properties>

<commons-io.version>1.3.1</commons-io.version>

<commons-fileupload.version>1.3.1</commons-fileupload.version>

</properties>

commons-fileupload  
 commons-fileupload  
 ${commons-fileupload.version}  
  
  
 commons-io  
 commons-io  
 ${commons-io.version}

#### 配置文件

spring-mvc.xml 文件需要配置如下内容，用于处理文件的上传

<!-- 上传下载配置 -->   
 <bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">  
 <property name="defaultEncoding" value="utf-8"></property>  
 <property name="maxUploadSize" value="10485760000"></property>  
 </bean>

#### 控制层

###### 进入发布页面

/\*\*跳转到添加的页面

* @param request
* @return
* \*/

@RequestMapping(value = "add",method=RequestMethod.GET)  
public String add(HttpServletRequest request) {  
 List allChnls = chanService.getAllChnls();  
 // 获取所有的频道  
 request.setAttribute("channels", allChnls);  
 return "article/publish";

}

###### 获取文章频道列表

/\*\*  
  
- 获取所有的频道  
- @return  
  
\*/  
  
@RequestMapping("getAllChn")  
  
@ResponseBody  
  
public List<Channel> getAllChn() {  
  
List<Channel> channels = channelService.getChannels();  
  
return channels;  
  
}

###### 列表

/\*\*  
  
- 根据频道获取相应的分类 用户发布文章或者修改文章的下拉框  
- @param chnlId 频道id  
- @return  
  
\*/  
  
@RequestMapping(value="listCatByChnl",method=RequestMethod.\*GET\*)  
  
@ResponseBody  
  
public ResultMsg getCatByChnl(int chnlId){  
  
CmsAssertJson.\*Assert\*(chnlId>0,"频道id必须大于0");  
  
List<Cat> chnlList = catService.getListByChnlId(chnlId);  
  
return new ResultMsg(1, "获取数据成功", chnlList);  
  
}

##### 发布文章请求

发布文章需要处理上传文件和文章的发布用户。

处理文件上传的代码为：

/\*\*  
  
- 处理每一个图片集合中的文件  
 - @param file  
 - @param article  
 - @throws IllegalStateException  
 - @throws IOException  
 \*/  
 private String processFile(MultipartFile file) throws IllegalStateException, IOException {  
  
  
 // 原来的文件名称  
 System.out.println("file.isEmpty() :" + file.isEmpty() );  
 System.out.println("file.name :" + file.getOriginalFilename());  
   
 if(file.isEmpty()||"".equals(file.getOriginalFilename()) || file.getOriginalFilename().lastIndexOf('.')<0 ) {  
 return "";  
 }  
   
 String originName = file.getOriginalFilename();  
 String suffixName = originName.substring(originName.lastIndexOf('.'));  
 SimpleDateFormat sdf= new SimpleDateFormat("yyyyMMdd");  
 String path = "d:/pic/" + sdf.format(new Date());  
 File pathFile = new File(path);  
 if(!pathFile.exists()) {  
 pathFile.mkdir();  
 }  
 String destFileName = path + "/" + UUID.randomUUID().toString() + suffixName;  
 File distFile = new File( destFileName);  
 file.transferTo(distFile);//文件另存到这个目录下边  
 return destFileName.substring(7);  
  
  
}

//获取原文件名称

String originName = file.getOriginalFilename();

//获取扩展名

String suffixName = originName.substring(originName.lastIndexOf('.'));

//根据日期获取存放文件的相对路径名

SimpleDateFormat sdf= **new** SimpleDateFormat("yyyyMMdd");

//计算文件存放的绝对路径

String path = uploadPath + "/" + sdf.format(**new** Date());

File pathFile = **new** File(path);

//如果路径不存在，则创建文件夹

**if**(!pathFile.exists()) {

pathFile.mkdir();

}

//计算文件存放位置以及文件名称

String destFileName = path + "/" + UUID.*randomUUID*().toString() +  
suffixName;

File distFile = **new** File( destFileName);

file.transferTo(distFile);//文件另存到这个目录下边

//文章中保存相对路径

article.setPicture(destFileName.substring(uploadPath.length()+1));

}

处理文章内容上传部分需要考虑从session获取当前用户，当前用户的id存入文章对象的作者字段中。然后调用服务层发布文章成功，代码为：

/\*\*  
  
- 处理发布文章请求  
- @param request  
- @param article  
- @param file  
- @return  
- @throws IllegalStateException  
- @throws IOException  
  
\*/  
  
@RequestMapping(value = "add",method=RequestMethod.\*POST\*)  
  
public boolean add(HttpServletRequest request,Article article,  
  
MultipartFile file) throws IllegalStateException, IOException {  
  
//处理标题图片  
  
processFile(file,article);  
  
//获取作者  
  
User loginUser =  
(User)request.getSession().getAttribute(ConstClass.\*SESSION\_USER\_KEY\*);  
  
article.setUserId(loginUser.getId());  
  
//发布文章  
  
return articleService.add(article)>0;  
  
}

#### 服务层

###### 获取频道

略

###### 获取分类

略

###### 发布文章

略

##### 数据层

###### 获取频道

\@Select("select \* from cms\_channel order by id")

\@ResultType(Channel.**class**)

List<Channel> getChannels();

###### 获取分类

/\*\*

\* 根据频道获取分类

\* **\@param** cid

\* **\@return**

\*/

\@Select("SELECT id,name,channel*id channelId FROM cms*category "

+ "WHERE channel\_id = #{value} ")

List<Category> getCategoryByChId(Integer cid);

###### 发布文章

发布文章中发布时间直接使用书库服务器的系统时间；

发布文章要求返回自动生成的主键id；

#### 数据层

具体的数据层代码为：

<!-- 添加文章 EnumOrdinalTypeHandler 是对枚举的处理-->

<insert id=*"add"* useGeneratedKeys=*"true"* keyColumn=*"id"*  
keyProperty=*"id"*>

INSERT INTO  
cms*article(title,content,picture,channel*id,category*id,user*id,hits,hot,

status,deleted,created,updated,commentCnt,articleType)

values(#{title},#{content},#{picture},#{channelId},#{categoryId},#{userId},0,

0,0,0,now(),now(),0,

#{articleType, typeHandler=org.apache.ibatis.type.EnumOrdinalTypeHandler,

jdbcType=INTEGER, javaType=com.mmcro.cms.comon.ArticleType})

</insert>

## 15.3 我的文章

实现效果如图所示：



### 15.3.1 前端

前端内容显示包含文章列表，其中每一条数据中显示内容包含标题图片、标题、发布时间、频道以及页码。显示顺序要求按照发表时间倒叙排列。

功能包含查看文章详情和删除文章。

分页显示部分代码的为：

<ul class="pagination">

<li class="page-item">

<a class="page-link" href="javascript:void(0)" data="1"  
aria-label="Previous">

<span aria-hidden="true">«</span></a>

</li>

</ul>

为分页绑点点击事件的js代码为：

$(**function**(){

$('.page-link').click(**function** (e) {

//获取点击的的*url*

**var** url = $(**this**).attr('data');

//在中间区域显示地址的内容

$('#content-wrapper').load(url);

});

})

删除文章功能要求删除后能够自动刷新当前的列表，不必考虑删除后的分页问题，也就是删除后只需要跳转到第一页即可。

### 15.3.2 后端

#### 控制层

文章列表

我的文章列表部分需要先获取当前用户id，然后根据用户id查询相应的文章列表。页码参数默认值为1。

/\*\*

\* 进入个人中心 获取我的文章

\* **\@param** request

\* **\@return**

\*/

\@RequestMapping("myarticlelist")

**public** String myarticles(HttpServletRequest request,

\@RequestParam(defaultValue="1") Integer page) {

// 获取当前用户信息

User loginUser =(User)  
request.getSession().getAttribute(ConstClass.*SESSIONUSERKEY*);

//获取一页文章

PageInfo<Article> pageArticles =  
articleService.listArticleByUserId(loginUser.getId(),page);

//利用工具类生成页码信息

PageUtils.*page*(request, "/user/myarticlelist", 10,

pageArticles.getList(), (**long**)pageArticles.getSize(),

pageArticles.getPageNum());

request.setAttribute("pageArticles", pageArticles);

**return** "/my/list";

}

* 删除文章

只能删除自己的文章，删除他人文章需要报异常。

/\*\*

\* 删除用户自己的文章

\* **\@param** id 文章id

\* **\@return**

\*/

\@RequestMapping("delArticle")

\@ResponseBody

**public boolean** delArticle(HttpServletRequest request,Integer id) {

//判断文章是否存在

Article article = articleService.findById(id);

**if**(article==**null**)

**return false**;

//判断文章是否属于自己的

User loginUser =(User) request.getSession().getAttribute(

ConstClass.*SESSIONUSERKEY*);

**if**(loginUser.getId()!= article.getUserId()) {

**return false**;

}

//删除文章

**return** articleService.remove(id)>0;

}

#### 服务层

* 获取文章列表

/\*\*

\* 根据用户id查找文章列表

\* **\@param** id 用户id

\* **\@param** page

\* **\@return**

\*/

\@Override

**public** PageInfo<Article> listArticleByUserId(Integer userId, Integer  
page) {

// **TODO** Auto-generated method stub

//设置分页

PageHelper.*startPage*(page, 10);

//返回分页数据

**return new** PageInfo<Article>(articleMapper.listByUserId(userId));

}

* 删除文章

/\*\*

\* 删除文章

\* **\@param** id 文章id

\* **\@return**

\*/

\@Override

**public int** remove(Integer id) {

// **TODO** Auto-generated method stub

**int** result = articleMapper.deleteById(id);

// 删除中间表，如果还有其他需要继续处理，如评论等。。。。

// 这里按照实际情况处理即可

articleMapper.delTagsByArticleId(id);

**return** result;

}

#### 数据层

* 我的文章列表

<!-- 根据用户id查找文章 根据发表的时间倒叙排列-->

<select id=*"listByUserId"* resultMap=*"articleMapper"*>

SELECT id,title,picture,user*id,channel*id,category\_id,

hits,hot,status,created,updated,commentCnt

FROM cms\_article

WHERE user\_id=#{value}

AND deleted=0

ORDER BY id desc

</select>

* 删除文章

删除文章使用逻辑删除，也就是文章内容依然保存在数据库当中。

/\*\*

\* 根据文章id删除文章

\* **\@param** id

\* **\@return**

\*/

\@Update("UPDATE cms\_article SET deleted=1 WHERE id=#{value} ")

**int** deleteById(Integer id);

## 15.4 文章修改

### 15.4.1 前端

与添加文章页面类似，但是需要数据回显。这里数据回显的难点是下拉框联动的数据回显和页面中文章内容的回显。

* 文章内容的回显

首先使用java代码对文章内容进行如下处理，用于特殊字符的转义，以便数据能在页面正常显示，这里的代码可以写在控制层或jsp页面中。

**private** String htmlspecialchars(String str) {

str = str.replaceAll("&", "&amp;");

str = str.replaceAll("<", "&lt;");

str = str.replaceAll(">", "&gt;");

str = str.replaceAll("\"", "&quot;");

**return** str;

}

将得到的数据显示在jsp页面中。

<div class=*"form-group row "*>

<textarea name=*"content1"* cols=*"100"* rows=*"8"*

style="width: *860px*; height: *250px*; visibility: *hidden*;"  
><%=htmlspecialchars(htmlData)%></textarea>

<br />

</div>

* 频道回显

频道内容在进入修改页面的时候，控制层将数据存储在request中，jsp页面利用c标签循环遍历显示，如果某项频道数据的id恰好等于被修改文章的频道id属性值，则将该项数据指定为选中状态。代码实现格式如下：

<div class=*"form-group row "*>

<label for=*"channel"*>文章栏目</label>

<select class=*"custom-select custom-select-sm mb-3"* id=*"channel"*  
name=*"channelId"*>

<option value=*"0"*>请选择</option>

<c:forEach items=*"*${channels}*"* var=*"channel"*>

<option value=*"*${channel.id}*"*  
${channel.id==article.channelId?"selected":""} >  
${channel.name}</option>

</c:forEach>

</select>

</div>

* 分类回显

分类列表与频道列表要进行联动，也就是当频道选中一条数据以后，分类列表内容需要进行相应的改变，这一点与添加文章内容的处理方式完全相同。由于修改文章需要回显，所以在jsp页面加载完成后就需要根据文章的频道获取该频道下的所有分类列表，实现该功能需要使用ajax技术进行局部刷新跟新分类内容，并且刷新分类的同时选中文章中的分类。具体实现代码参考如下：

/\*\*

\* 函数用于根据频道内容获取分类列表内容

\*

\*/

**function** changeChannel(){

//先清空原有的栏目下的分类

$("#category").empty();

**var** cid =$("#channel").val();//获取当前的下拉框的id

//根据ID 获取栏目下的分类

$.get("/article/listCatByChnl",{chnlId:cid},**function**(resultData){

**if**(resultData.result==1){ //后端处理正确

**var** list = resultData.data; //得到分类列表的数据

//遍历分类列表数据

**for**(**var** i **in** list){

//该分类就是文章的分类

**if**(list[i].id==${article.categoryId}){

// 该项处于选中状态

$("#category").append("<option value='"+list[i].id

+"' selected >"+list[i].name+"</option>")

}**else**{

//

$("#category").append("<option value='"+list[i].id+"'>"

+list[i].name+"</option>")

}//end if

}//end for

}//end if

})// end $.get(

}//end function

/\*\*

\* 预加载函数

\*/

$(**function**(){

//根据频道获取分类

changeChannel();

//为栏目添加绑定事件 触发联动

$("#channel").change(**function**(){

changeChannel();

}) // end of change

})//end $(function

* 文章id处理

在form中添加隐藏变量，以便提交更新数据的时候可以把文章id一起提交到后端。参考代码如下：

<input type=*"hidden"* value=*"*${article.id}*"* name=*"id"*>

* 数据提交

代码参考添加文章，只要将其中的提交地址修改如下地址即可：

[url:"/article/update](url:%22/article/update)",

### 15.4.2 后端

#### 控制层

进入修改页面

/\*\*

\* 跳转到修改的页面

\* **\@param** request

\* **\@return**

\*/

\@RequestMapping(value = "update",method=RequestMethod.*GET*)

**public** String update(HttpServletRequest request,Integer id) {

//获取频道

List<Channel> allChnls = chanService.getAllChnls();

//获取文章

Article article = articleService.findById(id);

//

request.setAttribute("article", article);

request.setAttribute("content1", article.getContent());

request.setAttribute("channels", allChnls);

**return** "my/update";

}

处理提交请求，需要判断文章的存在性、是否登录以及该文是否属于当前用户。具体代码参考如下：

/\*\*

\*

\* **\@param** request

\* **\@param** article

\* **\@param** file

\* **\@return**

\* **\@throws** IllegalStateException

\* **\@throws** IOException

\*/

\@RequestMapping(value = "update",method=RequestMethod.*POST*)

\@ResponseBody

**public boolean** update(HttpServletRequest request,

Article article, MultipartFile file)

**throws** IllegalStateException, IOException {

//获取作者

User loginUser =  
(User)request.getSession().getAttribute(ConstClass.*SESSIONUSERKEY*);

//尚未登录

**if**(loginUser == **null**) {

**return false**;

}

//获取原来的文章

Article srcArticle = articleService.findById(article.getId());

//原来文章存在

**if**(srcArticle == **null**) {

**return false**;

}

//原文的作者不是当前的登录用户

**if**(srcArticle.getUserId()!= loginUser.getId()) {

**return false**;

}

//处理上传文件

processFile(file,article);

//调用服务层保存修改后的文章数据

**int** result = articleService.update(article);

**return** result > 0;

}

#### 服务层

/\*\*

\* 修改文章

\* **\@param** article

\* **\@return**

\*/

\@Override

**public int** update(Article article) {

// **TODO** Auto-generated method tub

**int** result = articleMapper.update(article);

// 删除中间表中的

articleMapper.delTagsByArticleId(article.getId());

// 处理文章的标签

processTag(article);

**return** result;

}

#### 数据层

代码参考如下：

/\*\*

\* 修改文章

\* **\@param** article

\* **\@return**

\*/

\@Update("UPDATE cms\_article set title=#{title},content=#{content},"

+ " picture=#{picture},channel\_id=#{channelId},"

+ "category\_id=#{categoryId},updated=now() "

+ " WHERE id=#{id}")

**int** update(Article article);