知识分享系统设计与实现

智慧科技：唐英福

综述

前端发出请求之后，后台解析并去访问数据库得到需要展示的数据，并动态的修改前端的显示效果，从而完成动态页面的创建与修改。

一、前端功能设计与实现

前端需要与用户交互，用户通过界面切换和点击相关API获取相关功能，通过功能的实现来与后端交互，实现数据的流动，并实现用户需要的功能。

1.1 接口设计

访问URL	目的
/login	登录
/register	注册
/home	主页
/home/console	主控制台
/home/datamine	我的资料

1.2 逻辑设计

主题切换：深色与浅色系，通过将状态保存在父组件，使用内联样式实现

语言切换：通过react-intl-universal组件实现，配置json文件实现语言切换

上述的切换效果，都以结果的方式将字段缓存到本地浏览器，从而实现用户端的状态保持，后期可以将用户习惯上传到后台，实现主题、语言习惯的云存储。

1.3 主页设计

由于本系统是知识分享系统，所以首先进入的不应该是主页，而应该打开文章列表，以供用户查看和阅读。因此现在我的设计想法是，登录成功以后不显示菜单，用户首先看到的是文章列表，可以选择对应的文章打开，顶部有文章类别标签可以切换文章类别分类查看，留出打开菜单API，当用户点击之后打开菜单并调整屏幕适配大小。

1.4 菜单设计与功能对应

主控制台：显示相关统计等图表和排行榜等总体数据

我的资料：与点击头像处的“我的资料”作用相同，展示用户的相关资料，并提供用户信息修改和校验的入口，由于需要有用户权限的区别，所以还要添加用户认证的入口。

文章管理：

积分管理：

设置：

1.5 权限管理

权限管理包括普通用户、管理员、超级管理员。其中，需要有几点需要关注的地方：

1. 普通用户：无主动权限查看用户列表，即仅可以显式的查看到其他用户的状态，而不能查看其他用户的统计数据等隐私数据，更不能操作用户数据和进行相关的用户增删改查操作；面板无管理员操作界面。
2. 管理员：拥有普通用户具有的功能的同时，面板上显示出用户管理面板，其中能够查看用户的统计数据和相关增删改查操作的接口，有权限对用户数据进行操作，无权限操作其他管理员的数据，更无权限操作超级管理员数据；
3. 超级管理员：具有所有功能，目前设定不可删除自身，必须保证超级管理员对整个系统的掌控。

1.5.1 界面设计与实现

普通用户：目前具有主控制台、我的资料、文章管理、积分管理和设置菜单，功能为基础功能；

管理员：新增 **用户管理** 菜单，其间能够查看用户总数量、文章总量、日注册量、日访问量，用饼图的方式展示各权限用户的数量统计图、下方以列表的形式将所有用户数据罗列，并可滚动查看，列表需要提供API接口供点击进行 **修改和删除**，初定使用**右侧抽屉**的方式实现操作面板；版面右侧留下用户操作接口（包括：添加用户、查询用户、添加文章类别、发布公告等），同样采用右侧抽屉实现。

超级管理员：操作权限与管理员一致，只是无论谁都不能删除管理员。

问题

这就导致了一个问题，虽然说可以让非管理员不显示操作面板，但是实质上只是对普通用户隐藏了该面板，假如用户通过浏览器去调整参数，那还是可以把管理员界面弄出来，还是可以打开管理员界面。当然，我们肯定需要在后台去校验用户身份，如果出现用户数据操作，必须保证操作的是管理员，这在后台可以做。现在的问题是，我们不希望前端渲染出普通用户不希望看到的界面，有几个方法：

```
1．每次打开该界面都需要发一个网络请求，携带当前用户的身份信息和密钥，去后台校验身份后再获得返回值，如果是非法进入，后台就将页面重定向到登录界面，避免非法停留；
2．用户每登录或上线，通过校验用户身份，设置状态值，决定是否显示操作面板，这样在此前的验证加持下，用户即使通过浏览器打开了管理员面板，他也过不了后台的那一关，保证了权限的统一。
```

1.5.2 用户管理页面数据获取（网络请求）

管理页面需要的数据包括用户总量、文章总量、日注册量、日访问量、用户权限比例以及所有用户数据列表。

我现在的想法是：基本信息做一次网络请求，用户数据列表单独做一次请求，即在次页面做两次请求来获取到所有的数据。因此数据库需要创建一个视图，专门返回用户总量、文章总量、日注册量、日访问量（日注册量和日访问量暂时定为0,后期实现）以及用户数据比例（普通用户、管理员、超级管理员且权限只能有一个，向下兼容）；

为避免视图多次去查询其他表格，我定义了一张基本表，现在就存储以上数据，并通过附着触发器来动态修改该表内容，而后在用户管理界面直接访问此表数据来获取相应数据即可。（创建记录见数据库操作部分）。

1.6 设置页面

设置页面设置属性：

主题切换	语言切换		
修改资料	重置系统	关于系统	关于我们

1.7 创作中心

用户可以在此页面进行文章创作，选择封面、选择分类、配置文章标题和概述等。

1. 集成了markdown组件`md-editor-rt`，基于markdown进行文章创作；
2. 需要学习相关markdown语法，否则就是一个纯文本编辑器了；
3. 创作页面可以打开文章配置进行封面、标题、分类、概述的配置。（目前封面暂未提供修改，采用默认图片）；
4. 上传封面：用户自选本地图片作为封面上传，（预期是选择以后先在前端显示，而后上传文章的时候同步上传，但由于游览器的虚拟文件路径问题，目前暂未找到解决办法）；
5. 预期用户点击`发布`以后，就将文章发布到后台，目前暂时未实现，由于发布文章需要或得用户的身份，所以当前需要先完善文章显示部分和用户身份以后再进行设计实现。

1.7.1 发布文章时选择分类

目前实现是管理员加入文章分类，用户在发布文章的时候可以在预选框中选择分类并确认发布，因此，数据库中因当加入`摘要、分类、标签`字段。其中，分类和标签可能有多个，需要确定一个在数据库中存储其的数据结构，同时要兼容数据库表是文章分类在一个表而文章数据在一个表的情况。

1.7.1.1 文章分类的设计与实现

分类选项：

```
{
    "edu"：{          //教育
        "SC"：{       //计算机
            "java"："JAVA",
            "python":"python",
            "dataStructure":"数据结构"
        }
    },
    "military":{         //军事
        "news":{         //新闻
            "international",     //国际
            "domestic"           //国内
        },
        "weaponry":{             //武器装备
            "tank",
            "aiwarcraft",
            "warship"
        }
    },
    "finance":{                  //金融财政
        "internaltional",
        "domestic",
        "tax"                    //税收
    }
}
```

分类策略：

建三个表，主表存顶级分类如：教育、军事、金融、自定义等；次表存储二级分类如：计算机、化学、战争、军事设备等；最后一个表存储具体的分类如：java、C++、飞机、坦克、税收等。

字段统计：

主表	id	分类名	数量			
次表	id	链接主分类ID	分类名	数量		
终表	id	链接主分类ID	链接次分类ID	分类名	数量	

各表概述：

rootType:

id	分类名	数量
int	varchar(100)	int
id	typeName	num

secondType:

id	链接主ID	分类名	数量
int	int	varchar(100)	int
id	linkRootId	typeName	num

finalType:

id	链接主ID	链接次ID	分类名	数量
int	int	int	varchar(100)	int
id	linkRootId	linkSecondId	typeName	num

字段统计：

主表	教育（education)；新闻(news)；军事(military)；金融(finance)；自定义(customize)	
次表	计算机(CS)；生活(life)、出行trip)；武器装备(weaponry)；税收(tax)、股票stock)；	
终表	java(java)、python(python)、数据结构(dataStructure)；商品(merchandise)、车票ticket)；战斗机(warCraft)、坦克(tank)；	

1.7.2 文章方面的积分

发布一篇文章积累3分，而这个操作也可以直接使用触发器来实现，当对数据库表`article`执行一次插入，就自动的到对应积分项之下进行积分+3操作，从而实现积分增长。（触发器代码见数据库操作）

1.8 博客星系

博客星系需要展示用户的头像、账户、签名、文章内容、封面、展示数、点赞数、评论数，这需要同时展示。

1.8.1 文章详情

文章详情目前需要展示`文章标题、作者、文章封面、文章标题`，后续需要加入文章评论等数据。创建一个视图，通过文章ID去查找文章数据以及用户数据。

1.8.2 发布文章

1.8.3 修改文章

1.8.4 删除文章

1.8.5 查询文章

目前查询实现的是局部模糊搜索，搜索的结果可点击跳转到文章详情进行查看。

1.9 用户数据的增删改查与相关操作

用户数据需要有增删改查，同时在增删改查的同时有附带的操作，比如插入一个用户数据，就需要为该用户新增积分表中的对应项，同样的这也可以采用触发器来实现。（触发器实现见数据库操作）。

1.10 角色关联菜单实现

因为要实现角色关联菜单，而我目前的功能中有博客星系、主控制台、我的资料、用户管理、创作管理（创作、管理）、积分管理、设置；上述功能中，只有用户管理是管理员特有的，需要鉴别用户权限决定前端展示的页面是否含有该项。

而在项目需求里，需要管理员能够自定义不同用户关联不同的菜单，即每个人都能实现不一样的菜单项，这就要求前端有相应的配置项目；目前这个接口应当是放在管理员添加用户的界面，通过选择标签来实现关联菜单，同时修改用户的时候可以实现修改标签更改用户的关联菜单。

同时，由于最初的面板中的功能都是基础性的，不应当作为自定义的对象，也就是说每个人都应当有这些菜单选项（除了用户管理是分管理员和非管理员），所以需要加入自定义的菜单项时才需要添加时自定义，由于要实现目标功能，因此我要加入拓展功能来实现这个功能，将拓展功能作为可定义的接口来获取用户的操作，如果关联的菜单没有这一项，就不让用户访问这个页面，也不能直接通过网址直接访问，需要鉴别用户权限（token来实现）；

1.10.1 拓展功能

根据系统目的，拓展功能暂定：

1. 公告管理：让用户能够发布公告，公告后续在实现，暂定滚动播放方式；
2. 文件管理：让用户能够查看服务器的静态文件、如招募审核员来审核其他用户上传的文件合法性；
3. 广告管理：在主控制台和用户管理页面有轮播图，这可以作为广告投放点，可自定义让用户上传图片作为轮播资源，作为广告投放员；
4. 评论管理：可以查看所有的评论，如招募评论审核员来检测评论的合法性，不合法的可删除。

公告管理	文件管理	广告管理	评论管理
notice	fileManage	adManage	commentManage

1.10.2 角色关联菜单的实现方法

根据要求，要对每个人都有不同的菜单，除了默认的菜单项目以外，拓展菜单可变。那么现在有两种方法来实现这个效果：

```
1. 用户点击登录，服务器鉴别拥有的权限和关联菜单之后，将整个菜单从服务器返回到前端，前端直接将该数组作为菜单列表并到前端显示菜单列表；而为了避免用户直接通过网址访问该菜单，首先每打开拓展页面都自动发送验证token的请求，如果token验证不通过就代表是非法访问，直接拒绝访问。
2. 另一种方式是前端定义默认菜单数组，在网络请求菜单未完成时显示的是默认菜单，鉴权完毕并返回菜单拓展项目后再将拓展菜单push到菜单数组中，实现动态刷新。同样，每打开页面都会自动发送鉴权请求，验证token是否合法，拥有菜单权限的用户token验证自然能够通过。
```

其中，既然从服务器返回菜单项，那么在中英文切换的时候就需要语言变换，对此我的改进方法：

```
1. 登录请求返回中文的同时返回英文，切换语言就重新push，因为在程序生命周期内变量的值是存在的、持久化的；
2. 切换的时候发出请求，专门请求其他语言的菜单文本，再切换。
```

现在暂时不考虑这个问题，先采用中文实现关联菜单任务，后期再来实现语言切换拓展。

综合比较之下，我的最终方案是：先展示默认菜单，请求完成之后再进行push，避免网络延迟的问题出现，导致用户体验降低。

后端存储方案为：数据库存储菜单项、以逗号分割，此后使用的时候直接进行字符串分割使用即可。

存储例子：

```
张三 | 2 | [{"key":"administrator","name":"用户管理"},{"key":"notice","name":"公告管理"},{"key":"fileManage","name":"文件管理"},{"key":"adManage","name":"广告管理"},{"key":"commentManage","name":"评论管理"}]
```

二、数据库设计与实现

数据库需要有用户管理表，文章管理表、文章分类表和评论管理表以及积分管理表。

2.1用户管理表(sysUser)

这是给系统使用的数据库表，其查看权限属于管理员。对用户是间接性透明的，即普通用户不能直接查看到数据库表中的数据，而只能查看到系统给予用户的查看数据。

系统表需要记录用户的ID、帐号、密码、签名、性别、电话、邮箱、文章数、被赞总数、评论数、个人排名：

拓展：token、头像路径、拓展菜单

ID	帐号	密码	签名	性别	电话	邮箱	文章数	被赞数	评论数	排名	权限	头像	拓展菜单
int	char(32)	char(32)	varchar(32)	int	char(15)	varchar(20)	int	int	int	int	int	varchar(50)	varchar(100)
id	acc	pass	sign	gender	phone	email	articleNum	beLiked	comment	ranking	authority	avatar	extendMenu
							0	0		0	0	default.png	""

其中ID和帐号唯一，性别**[1-男，0-女]**，权限**[1-管理员，0-用户]**。

2.2 文章管理表(article)

文章表：ID、发布人ID、文章类别ID、点赞数、评论数、浏览数、文章内容

ID	发布人ID	文章类别	点赞数	评论数	浏览数	文章内容	文章标题	文章封面	文章标签	文章摘要
int	int	int	int	int	int	text	varchar(128)	varchar(50)	varchar(50)	text
id	pub	articleType	beLiked	beComment	beScan	content	title	cover	articleTag	articleAbstract
		默认	0	0	0				默认	

2.3 文章分类表(articleType)

文章分类表：ID、类别、数量

ID	类别	数量
int	char(20)	int
id	type	num
		0

分类：教育、军事、财政、金融

rootType:

id	分类名	数量
int	varchar(100)	int

secondType:

id	链接主ID	分类名	数量
int	int	varchar(100)	int

finalType:

id	链接主ID	链接次ID	分类名	数量
int	int	int	varchar(100)	int

2.4 评论表(comment)

主评论表：ID、对应文章ID、用户ID、评论内容

ID	对应文章ID	点赞数	评论内容
int	int	int	text
id	linkArticle	beLiked	commentContent
		0	

次评论表：ID、链接主评论、用户、回复对象、评论内容

ID	链接主评论	用户	回复对象	评论内容
int	int	int	int	text

2.5 积分管理表(credit)

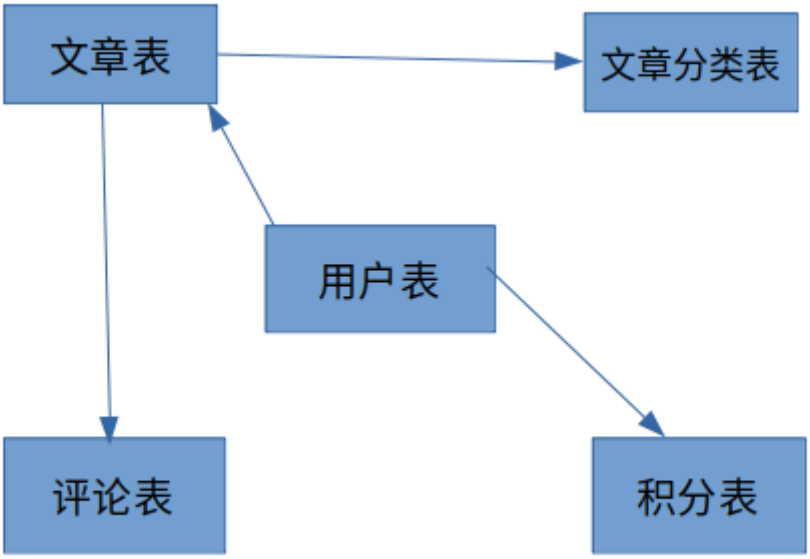
积分表：ID、用户ID、积分数、排名

ID	用户ID	积分数	排名
int	int	int	int
id	linkUser	totalCredit	ranking
		0	0

2.6 基本信息表

基本信息表：id，用户总量，文章总量，日注册量，日访问量，普通用户量、普通管理员量、超级管理员量

id	用户总量	文章总量	日注册量	日访问量	普通用户量	普通管理员量	超级管理员量
int	int	int	int	int	int	int	int
	0	0	0	0	0	0	0



2.6 数据库操作

1. 排名计算

```
update credit c set c.ranking=(select d.pm from (select linkUser,@rank:=@rank+1
as pm from credit a,(select @rank:=0) b order by totalCredit desc ) d where
c.linkUser=d.linkUser);
```

2. 排名赋值

```
update sysUser a set ranking=(select ranking from credit b where
a.id=b.linkUser);
```

3. 创建排名获取的视图

```
create view queryTopTen as (select * from sysUser order by ranking limit 10);
select * from queryTopTen;
```

4.
> 博客星系需要展示用户的头像、账户、签名、文章内容、展示数、点赞数、评论数、文章类别

所以，需要创建视图，将用户的头像、账户、签名与文章ID，文章标题，文章内容、封面、展示数、点赞数、评论数、文章类别拼接起来：

```
create view queryBlogGalaxy as (select a.id as
userID,a.acc,a.sign,a.avatar,b.id,b.title,b.content,b.cover,b.beScaned,b.beLi
ked,b.beComment,b.articleType from sysUser a,article b where a.id=b.pub);
```

5. 创建视图，使根据文章ID获取到文章的标题、封面、内容以及作者的账户

```
create view queryArticleById as (select
b.id,a.acc,b.pub,b.title,b.cover,b.content,b.articleType,b.articleTag,b.artic
leAbstract from sysUser a,article b where a.id=b.pub);
```

6. 由于需要每加入一篇文章就增加基本文章数量，所以使用触发器，当对文章表执行插入操作，就自动将文章类型表的`默认`字段做+1操作，实质上也是记录所有文章的数量。

```
create trigger autoUpdateTypeNum after insert on article for each row update
articleType set num=num+1 where id=5;
```

7. 针对数据库基本表，当添加用户后需要触发修改用户数量，删除用户也需要修改数量；当文章发生变化也需要修改文章总量；先目前访问量和注册量先暂时不管；插入或删除用户数据时也需要修改用户比例数据；采用触发器实现以上功能。

```
# 添加用户
delimiter ]]
create trigger autoIncreaceUserNum after insert on sysUser for each row
begin
update basicInfor set totalUser=totalUser+1;
update basicInfor set commonUser=commonUser+1 where new.authority=0;
update basicInfor set commonAdmin=commonAdmin+1 where new.authority=1;
update basicInfor set superAdmin=superAdmin+1 where new.authority=2;
end]]
delimiter ;
# 删除用户
delimiter ]]
create trigger autoDecreaceUserNum after delete on sysUser for each row
begin
update basicInfor set totalUser=totalUser-1;
update basicInfor set commonUser=commonUser-1 where old.authority=0;
update basicInfor set commonAdmin=commonAdmin-1 where old.authority=1;
update basicInfor set superAdmin=superAdmin-1 where old.authority=2;
delete from credit b where old.id=d.linkUser;
end]]
delimiter ;
# 添加文章
delimiter ]]
create trigger autoIecreaceArticleNum after insert on article for each row
begin
update basicInfor set totalArticle=totalArticle+1;
end]]
delimeter ;
# 删除文章
delimiter ]]
create trigger autoDecreaceArticleNum after delete on article for each row
begin
update basicInfor set totalArticle=totalArticle-1;
end]]
delimiter ;
```

这下面的触发器完全可以合并到上面的触发器，但目前先实现功能，后期再来维护

---

8. 插入用户数据之后自动添加对应的积分项目，实现同步表数据：

```
create trigger autoInsertCreditItem after insert on sysUser for each row
insert into credit(linkUser) values(new.id);
```

9. 发布一篇文章积分+3分，同步数据：

```
create trigger autoIncreasePubCredit after insert on article for each row
update credit set totalCredit=totalCredit+3 where linkUser=new.pub;
```

10. 对于评论，因为评论表存储的是id值，所以需要将其转换为字符串（具体文本），因此通过创建视图来实现，对与主评论，主要是将评论主体帐号转化为具体文本；对于次评论，主要是将评论主体和回复对象转化为具体文本：

```
# 主评论
create view mainCommentView as (select
a.id,a.linkArticle,b.acc,a.linkUser,b.avatar,a.commentContent from sysUser
b,mainComment a where a.linkUser=b.id);
# 次评论
create view subCommentView as (select a.id,a.linkComment,(select c.acc from
sysUser c where c.id=a.linkUser) as linkUser, a.linkUser, b.avatar, (select d.acc
from sysUser d where d.id=a.replyTo) as replyTo, a.commentContent from subComment
a,sysUser b where a.linkUser=b.id);
```

11. 每插入一条评论，就自动更新文章绑定的评论数，采用触发器实现：

```
# 自动更新用户的评论数
create trigger autoIncreaseUserCommentNum after insert on mainComment for
each row update sysUser set userComment=userComment+1 where new.linkUser=id;
# 自动更新文章的评论数
create trigger autoIncreaseArticleCommentNum after insert on mainComment for
each row update article set beComment=beComment+1 where new.linkArticle=id;

# 同理，子评论的添加也是一样的
# 自动更新用户的评论数
create trigger autoIncreaseUserSubCommentNum after insert on subComment for
each row update sysUser set userComment=userComment+1 where new.linkUser=id;
# 自动更新文章的评论数
create trigger autoIncreaseArticleSubCommentNum after insert on subComment
for each row update article set beComment=beComment+1 where
new.linkArticle=id;
```

三、后端逻辑设计与实现

后端需要接受前端的访问请求，并作出响应。本课题采用spring mvc作为技术以来实现后端逻辑，将逻辑分成controller,dao,service几层，避免了后端逻辑的高耦合度，提高编码实现清晰度。

3.1 请求地址解析

请求地址	目的
/user/*	操作用户数据相关
/article/*	操作文章数据相关
/init/*	界面初始化数据需求相关
/user/login.map	登录
/user/register.map	注册
/user/loginout.map	退出登录
/init/articleType.map	获取文章类型
/article/detail	文章详情

四、身份验证

需要身份验证，目前采用cookie实现，因为可以后端生成cookie并拿给前端，前端获取到之后，在每次后续的请求都会获取到cookie并发送给后端，那么假如说我将token值存储在cookie中，这样前端可以获取到，每次都通过这个token判断是否是当前用户在进行操作即可。在记住密码这一项上，需要考虑到cookie的有效期，可以后续实现。

4.1 cookie设计

暂定cookie需要存储token值，用户id和账户名。

```
token=dftyuikertyui13;id=1;acc=张三;
```

所以此时在个人信息就要增加一个token字段，以验证用户的身份，每个响应都要先去读取token值作出比较才能进行实际操作，如果说token值不匹配，那么就不允许用户的操作。

（目前）每次登录都会更新token值，每次退出登录都会重置token值（数据库中）。

五、实现效果

附录

```
//备份
    "homeMenu":"[{\"key\":\"blog\",\"name\":\"博客星系\"},
{\"key\":\"console\",\"name\":\"主控制台\"}, {\"key\":\"profile\",\"name\":\"我的资
料\"}, {\"key\":\"edit\",\"name\":\"创作管理\"}, {\"key\":\"credit\",\"name\":\"积
分管理\"}, {\"key\":\"setting\",\"name\":\"设置\"}]",
    "homeMenuAdmin":"[{\"key\":\"blog\",\"name\":\"博客星系\"},
{\"key\":\"console\",\"name\":\"主控制台\"}, {\"key\":\"profile\",\"name\":\"我的资
料\"}, {\"key\":\"edit\",\"name\":\"创作管理\"}, {\"key\":\"credit\",\"name\":\"积
分管理\"}, {\"key\":\"setting\",\"name\":\"设置\"},
{\"key\":\"admin\",\"name\":\"用户管理\"}]",
```