“技术生活交流社交Web应用”软件开发文档

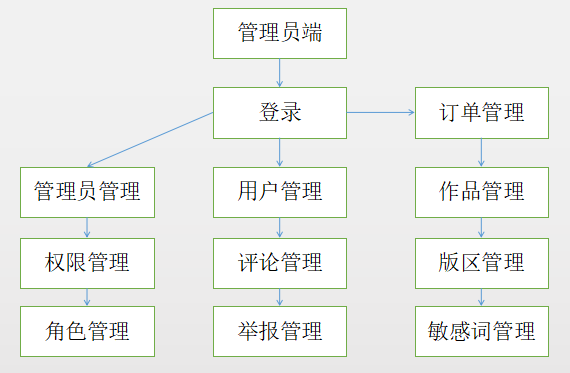
一、需求分析

本项目是基于分享技术生活，提供打分，推荐互感兴趣用户的网页应用。实现的功能分为两部分，一部分是用户一部分是管理员。对于用户来说，应用提供多个版区，用户根据喜爱的版区发布个人的技术分享或生活的作品，作品必须上传至少一张照片或视频(视频链接)，作品描述加上#号相当于加上标签。那当然也可以浏览其他用户的作品，给作品打分、收藏作品、举报作品、评论作品、打赏作品(只作学习测试，不会开放接口)、点赞评论、举报评论，点赞高的评论会出现在流简介上，而作品的热度将会根据作品的发布时间、评分参与人数、评论参与人数、评分高低通过我们自定义的Feed流公式进行排序，在热流中优先显示。用户可以邮箱注册，或使用邮箱或注册ID和密码登录，完善和维护个人信息(上传头像、个人简介等)，修改密码，修改邮箱。可以关注/取消关注其他用户，查看个人关注帐号和被关注帐号，查看收藏的作品，查看站内信。同时提供搜索用户，搜索标签的功能。

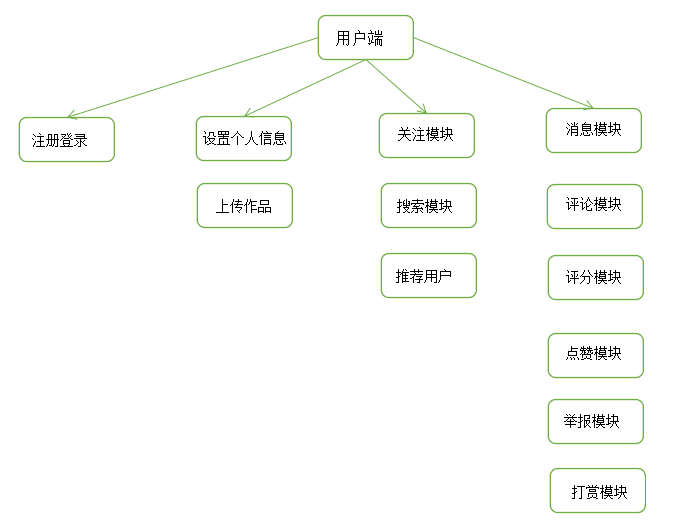
对于管理员来说，应用提供经典的管理员权限管理，可以进行对各版块管理员的管理，管理员分三种，版块管理员，用户管理员，主管理员。版块管理员可以管理版块信息，如删除被举报或违规的作品和评论，移动发错版区的作品。用户管理员可以冻结用户，为用户重置密码。主管理员可以管理其它管理员和用户的权限，敏感词池管理。删改操作将以日志纪录形式在管理员页展示。

二、概要设计

1.系统结构设计如图1图2



管理员端系统架构图-图2



用户端系统架构图-图2

2.数据设计：

数据库采用了MySQL关系型数据库和Redis非关系型缓存数据库，用户端和管理员端各自的前后端，用户端和管理员端共同操作MySQL和Redis数据库。

用户或管理员通过访问前端项目Frontend获取页面，Frontend通过用户和管理员的操作调用Backend的后端接口获取数据，Backend通过分层处理业务逻辑并根据Frontend的要求操作MySQL和Redis数据库且返回数据。系统结构设计图如图1：

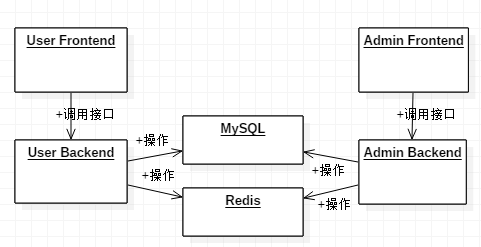


图3

前后端交互接口设计：

本项目根据Restful　API风格设计前后端交互接口。而根据业务逻辑管理员端涉及按条件的CRUD接口较多，而用户端则设计了更多人性化接口。

管理员端大部分是直接增删改查的接口，此处列出用户管理接口作例子：

/user/add 新增用户

Data:{

id: String,

email: String,

pwd: String,

status: Integer,

gins: Integer

}

/user/del/{id} 删除用户

data:{

pathvariable:id:long

}

/user/get/{id} 查看用户

data:{

pathvariable:id:long

}

/user/list 条件查看用户列表

data: {

id: String,

email: String,

pwd: String,

status: Integer,

}

/user/mod 修改用户

data:{

id: String,

email: String,

pwd: String,

status: Integer,

gins: Integer

}

而用户端接口偏向于业务逻辑设计，同样以举用户模块为例子：

/user/checkid/{id} 检查注册ID是否已被占用

data: {

pathvariable:id:String

}

/user/detail 查看个人详情页

data:{}

/user/moddetail 修改个人详情页

data:{

avatar:String,

brief:String,

website:String,

phone:String,

gender:char,

wechat:String

}

/user/modpwd 修改密码

data:{

oldPwd:String,

newPwd:String

}

/user/my 查看我的信息

data:{}

/user/register 注册

data:{

id:String,

email:String,

pwd:String

}

数据设计：由于引入Redis缓存数据库，根据业务逻辑在实现特定功能时会在Redis使用对应数据结构。

同样以用户模块作例子，在MySQL关系型数据库中，表的设计如下：

# User

CREATE TABLE `user`(

`user\_autoid` BIGINT AUTO\_INCREMENT,

`user\_id` VARCHAR(20),

`user\_email` VARCHAR(30),

`user\_status` INT,

`user\_pwd` VARCHAR(200),

`user\_gins` INT,

PRIMARY KEY (`user\_autoid`),

INDEX (`user\_email`),

INDEX (`user\_id`,`user\_pwd`),

INDEX (`user\_email`,`user\_pwd`)

)ENGINE=INNODB CHARSET utf8;

而在Redis非关系型数据库中，涉及用户的结构如下：

Hash-

User:

key: user:[user\_id]

field: id, email, gins, avatar, follows, fans, posts, msgstatus//是否有未读消息

Set-

Follows-

key: follows:[user\_id]

value: //关注用户id

Fans-

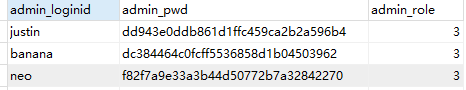
key: fans:[user\_id]

value: //粉丝id

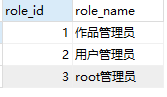
三、详细设计

本项目的开发中，本人主要负责注册登录模块、推荐用户模块、搜索模块、关注模块、探索模块，以及管理员端的管理员角色权限管理。

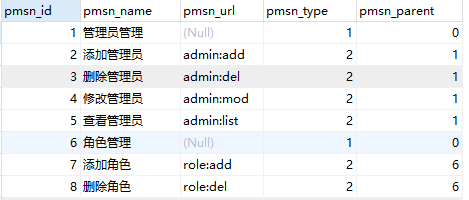
涉及注册登录模块与管理员角色权限管理，本项目使用Apache Shiro安全框架，通过引入Subject概念对用户访问进行固定，通过实现Realm实现授权和验证，其中验证使用了对加入用户ID盐值进行MD5两次散列加密密码，而授权方面，数据库中实现基于角色的权限管理。数据库中实现如下：



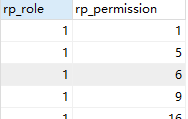
Admin表-图4



Role表-图5



Permission表-图6



RolePermision中间表-图7

根据Admin表的role字段获取角色ID，然后在RolePermission中间表获取该角色拥有的权限ID列表，再关联Permission表获取对应权限，通过Realm实现授权。

以下为实现Realm代码：

public class ShiroRealm extends AuthorizingRealm {

//省略Dao层接口的引入

//授权

@Override

protected AuthorizationInfo

doGetAuthorizationInfo(PrincipalCollection principals) {

SimpleAuthorizationInfo simpleAuthorizationInfo =

new SimpleAuthorizationInfo();

UserInfo userInfo = (UserInfo) principals.getPrimaryPrincipal();

String loginId = userInfo.getUsername();

Admin admin = new Admin(loginId,null,0);

admin = adminDao.selectByDTO(admin,0,1)[0];

int[] perms = rolePermissionDao.selectPermByRole(admin.getRole());

Permission permission = new Permission();

for ( int perm : perms ) {

permission.setId(perm);

Permission temp = permissionDao.selectByDTO(permission,0,1)[0];

if ( temp.getType() == 1 ) continue;

simpleAuthorizationInfo.addStringPermission(temp.getUrl());

}

return simpleAuthorizationInfo;

}

//验证

@Override

protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token)

throws AuthenticationException {

String loginId = (String) token.getPrincipal();

String password = adminDao.getPassword(loginId);

UserInfo userInfo = new UserInfo(loginId);

if ( password == null || password.equals("") )

throw new UnknownAccountException();

SimpleAuthenticationInfo simpleAuthenticationInfo =

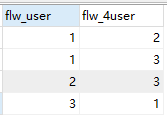
new SimpleAuthenticationInfo(userInfo, password, ByteSource.Util.bytes(loginId), getName());

return simpleAuthenticationInfo;

}

}

推荐用户模块是在关注模块的基础上实现的，关注模块通过记录关注者与被关注者ID实现，直接使用单表记录。



Follow表-图8

推荐用户通过Redis数据结构Set实现，具体实现是检索在用户关注的列表中获取用户列表，并对获取的用户列表的关注中进行Set合并，且与用户所关注的集合作非操作，并随机选取5名作推荐用户推给用户。简图如图7所示：

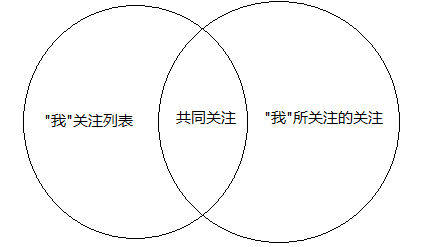


图9

探索模块涉及作品模块在Redis中的数据结构，实现每1小时更新版区热文的功能，实现探索模块而自定义了Feed流公式（热度与时间结合），并进行了粗略计算测试。

作品模块Redis中相关的数据结构如下：

Hash-

Post:

key: post:[post\_id]

field: author, multmedia, text, creatime, section, score, scorers

ZSet-

Post:

key: hotpost:[section\_id]

score: score+cmts\*2+creatime>>7

members: [post\_id]

Comment:

key: cmtrank:[post\_id]

score: //点赞数

members: [cmtid]

自定义的Feed流公式如下：

Score(scorers, comments, creatime) = scorers + comments << 1 + creatime >> 7

根据creatime时间戳右移7位来减少时间对Feed流的影响，增大评论量对Feed流的影响。

四、软件测试

后端开发的测试工具使用的Spring整合的JUnit测试框架和Swagger2.0与SwaggerUI插件，由于Apache Shiro限制的原因，某些接口使用Chrome浏览器插件Restlet Client测试。

JUnit测试框架主要负现Dao层、Service层和其它一些工具类的测试，以下举例管理员管理的Dao层测试类：

@SpringBootTest

@RunWith(SpringRunner.class)

public class AdminDaoTest {

//注入管理员Dao

@Autowired

private AdminDao adminDao;

@Test

public void insertOne() throws Exception {

Admin admin = new Admin("jijiji","asdasd",2);

adminDao.insertOne(admin);

}

@Test

public void updateOne() throws Exception {

Admin admin = new Admin("justin","wwwwww",2);

adminDao.updateOne(admin);

}

@Test

public void deleteByDTO() throws Exception {

Admin admin = new Admin("jijiji",null,0);

adminDao.deleteByDTO(admin);

}

@Test

public void selectByDTO() throws Exception {

Admin admin = new Admin("",null,3);

Admin[] admins = adminDao.selectByDTO(admin,0,1);

for ( Admin adm : admins )

System.out.println(adm);

}

}

service层的测试与dao层测试相似。

Restlet Client接口测试举例登录接口测试，测试访问接口如图8，测试数据体如图9。

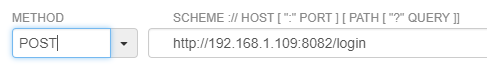


图10

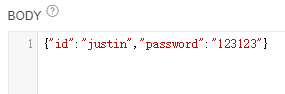


图11

测试结果如图12：



图10

五、软件完成后的总结汇报

项目开发过程中遇到的问题很繁多也很有挑战性，如前后端的跨域问题 、安全框架Apache Shiro与SpringBoot整合时注入的方式，从网上整合各种资料最终得到自己满意的结果，还有前端使用之前未接触过的VueJS和ElementUI，在遇到问题和解决问题中过得很充实受益良多。