软件开发文档--董雨潇

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

系统结构设计：

本项目总共用到MySQL和Redis两种数据库，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：

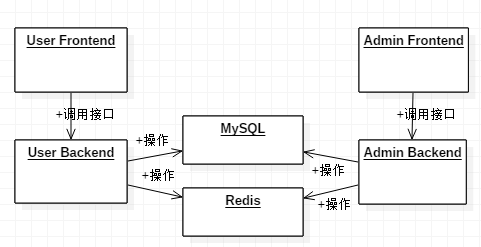


图1

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

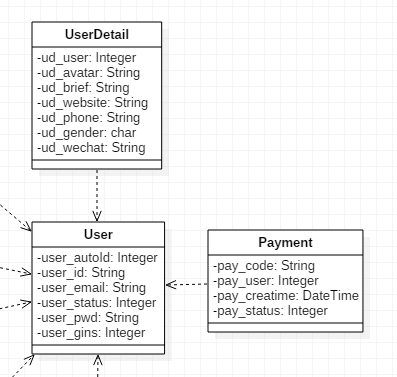
Fans-

key: fans:[user\_id]

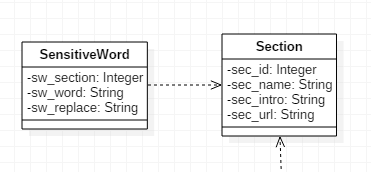
value: //粉丝id

三、详细设计

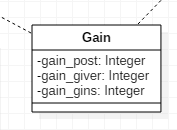
项目是用户有关生活分享的网页应用，对于用户来说，每个用户都应注册录入自己的个人信息。在使用方面：应用提供多个版区，用户根据喜爱的版区发布个人的技术分享或生活的作品，作品必须上传至少一张照片或视频(视频链接)，当然也可以浏览其他用户的作品，给作品打分、收藏作品、打赏作品。因此需要处理图片上传、个人信息、收藏、打赏等模块。在管理端，为了方便用户注册，用户浏览，需要设计管理员的订单管理以及版区管理；在净化网络环境方面设有敏感词管理。



个人信息包括自增ID，用户ID，用户邮箱，用户状态，用户密码以及用户GINS。用户状态可有管理员来控制是否冻结。Payment为用户订单，订单包括所获取的GINS码，创建用户的时间以及用户的状态。用户的个人信息，都是通过数据库的增删改查来添加或删除。



敏感词有敏感词的版区，属于敏感词的词汇以及它的替代词。先看词语属于哪个版区，再判断在此版区中该词语是否属于敏感词汇，如果属于敏感词则将其替代。



在用户打赏支付时可通过判断该用户GINS余额是否足够来判断该用户是否可以对作品进行打赏。可判断打赏来自的用户以及打赏的金额。

部分代码演示：

例如：

@Repository

public interface AdminDao extends BaseDao<Admin> {

String getPassword(String loginId);

int getRole(String loginId);

}

AdminDao继承了BaseDao的如下四个增删改查的方法

public interface BaseDao<Domain> {

int insertOne(Domain domain);

int updateOne(Domain domain);

int deleteByDTO(Domain domain);

Domain[] selectByDTO(@Param("domain") Domain domain,

@Param("offset") int offset,

@Param("length") int length);

}

AdminMapper一共就获得6个方法

<select id="getRole" resultType="int">

SELECT `admin\_role` FROM `admin` WHERE `admin\_loginid`=#{loginId}

</select>

<insert id="insertOne" parameterType="edu.qd.adminbackend.domain.Admin">

INSERT INTO `admin` VALUES (#{loginId},#{pwd},#{role})

</insert>

<update id="updateOne" parameterType="edu.qd.adminbackend.domain.Admin">

UPDATE `admin`

</update>

<delete id="deleteByDTO" parameterType="edu.qd.adminbackend.domain.Admin">

DELETE FROM `admin` WHERE `admin\_loginid`=#{loginId}

</delete>

<select id="selectByDTO" parameterType="map" resultMap="BaseResultMap">

SELECT `admin\_loginid`,`admin\_role` FROM `admin`

</select>

<select id="getPassword" parameterType="string" resultType="string">

SELECT `admin\_pwd` FROM `admin` WHERE `admin\_loginid`=#{loginId}

</select>

整理完对每个Mapper进行测试。

四、软件测试

当我们参照软件详细设计文档编码完成后，接着就会根据我们所实现的功能，进行软件测试文档的编写；大多测试文档有两类，一类是软件单体测试文档，一类是软件结合测试文档；顾名思义，单体测试：就是对软件中每个小的方法，一个独立的方法进行测试的文档；结合测试：就是把多个功能模块组合到一起进行测试，主要是为了检测每个功能模块之前的交互性和功能的结合实现性。

SpringBoot整合JUnit测试框架，测试dao层，service层，controller层。

部分测试代码如下：

AdminDaoTest：

@SpringBootTest

@RunWith(SpringRunner.class)

public class AdminDaoTest {

@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);

}

}

编写完每个DaoMapper都对其进行测试，检查看是否有报错，并进行修改。直至测试通过为止。

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。