程序清单



项目名称： 表白墙网站开发

专业班级： 软件工程1903

小组成员：吴联想、王义博、郑航舰

指导教师： 杨枨老师

二零二一年十月二十三日

**版本记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 修订日期 | 版本/状态 | 修订人 | 备注 |
| 01 | 2021.12.15 | 0.1 | 王义博 | 登录注册个人中心 |
| 07 | 2021.12.15 | 0.2 | 吴联想 | 管理员模块 |
| 08 |  |  |  |  |
| 09 |  |  |  |  |
| 10 |  |  |  |  |

目录

代码清单

**1注册**

1.1注册

**2登录**

2.1登录

**3个人中心**

3.1评论

3.2收藏

3.3认领

3.4被认领

3.5我的表白墙

3.6修改密码

3.7修改个性签名

3.8跳转

3.9删除

**4管理员模块**

4.1用户管理

4.1.1管理员编辑用户

4.1.2管理员分页查询显示用户列表

4.1.3管理员重置密码

4.2表白墙管理

4.2.1管理员更改表白墙状态

4.2.2管理员分页查询显示表白墙列表

4.2.3管理员删除表白墙

4.3反馈管理

4.2.1管理员更改反馈状态

4.2.2管理员分页查询显示反馈列表

4.2.3管理员删除反馈

1. 注册

Controller：

@RequestMapping(value = "/register", method = RequestMethod.POST)

public ApiResult<Map<String, Object>> register(@Valid @RequestBody RegisterDTO dto) {

user auser = userService.executeRegister(dto);

if (ObjectUtils.isEmpty(auser)) {

return ApiResult.failed("账号注册失败");

}

Map<String, Object> map = new HashMap<>(16);

map.put("user", auser);

return ApiResult.success(map);

}

Server：

@Override

public user executeRegister(RegisterDTO dto) {

//查询是否有相同用户名的用户

LambdaQueryWrapper<user> wrapper = new LambdaQueryWrapper<>();

wrapper.eq(user::getUserId, dto.getStunum());

user user = baseMapper.selectOne(wrapper);

if (!ObjectUtils.isEmpty(user)) {

ApiAsserts.fail("账号已存在！");

}

user addUser = user.builder()

.UserId(dto.getStunum())

.UserName(dto.getOthername())

.UserSex(dto.getSex())

.UserPwd(MD5Utils.getPwd(dto.getPass()))

.UserRegistertime(new Date())

.UserState("正常")

.UserJurisdiction("用户")

.build();

baseMapper.insert(addUser);

return addUser;

}

1. 登录

Controller:

@RequestMapping(value = "/login", method = RequestMethod.POST)

public ApiResult<Map<String, String>> login(@Valid @RequestBody LoginDTO dto) {

String token = userService.executeLogin(dto);

if (ObjectUtils.isEmpty(token)) {

return ApiResult.failed("账号密码错误");

}

Map<String, String> map = new HashMap<>(16);

map.put("token", token);

return ApiResult.success(map, "登录成功");

}

Server:

@Override

public String executeLogin(LoginDTO dto) {

String token = null;

try {

user auser = getUserByUserId(dto.getUsername());

String encodePwd = MD5Utils.getPwd(dto.getPassword());

if(!encodePwd.equals(auser.getUserPwd()))

{

throw new Exception("密码错误");

}

token = JwtUtil.generateToken(String.valueOf(auser.getUserId()));///错误发生点：将username传入token，正确的应该是userid传入，因为登陆时记录的是id和密码

} catch (Exception e) {

log.warn("用户不存在or密码验证失败=======>{}", dto.getUsername());

}

return token;

}

1. 个人中心
2. 获取用户评论：

@GetMapping("/reply/{username}")

public ApiResult<Map<String, Object>> getUserByNameForReply(@PathVariable("username") String username,

@RequestParam(value = "pageNo", defaultValue = "1") Integer pageNo,

@RequestParam(value = "size", defaultValue = "10") Integer size) {

Map<String, Object> map = new HashMap<>(16);

user auser = userService.getUserByUserId(username);

Assert.notNull(auser, "用户不存在");

Page<reply> page = replyService.page(new Page<>(pageNo, size),

new LambdaQueryWrapper<reply>().eq(reply::getReplyUserid, auser.getUserId()));

map.put("user", auser);

map.put("topics", page);

return ApiResult.success(map);

}

前端：fetchUserById2() {

getInfoByNameForReply(this.$route.params.username, this.page.current, this.page.size).then((res) => {

const { data } = res

this.topicUser2 = data.user

this.page2.current = data.topics.current

this.page2.size = data.topics.size

this.page2.total = data.topics.total

this.topics2 = data.topics.records

})

},

1. 获取用户收藏

@GetMapping("/coll/{username}")

public ApiResult<Map<String, Object>> getUserByNameForColl(@PathVariable("username") String username,

@RequestParam(value = "pageNo", defaultValue = "1") Integer pageNo,

@RequestParam(value = "size", defaultValue = "10") Integer size) {

Map<String, Object> map = new HashMap<>(16);

user auser = userService.getUserByUserId(username);

Assert.notNull(auser, "用户不存在");

Page<collection> page = collService.page(new Page<>(pageNo, size),

new LambdaQueryWrapper<collection>().eq(collection::getCollectionUserid, auser.getUserId()));

map.put("user", auser);

map.put("topics", page);

return ApiResult.success(map);

}

前端：

fetchUserById5() {

getInfoByNameForColl(this.$route.params.username, this.page.current, this.page.size).then((res) => {

const { data } = res

this.topicUser5 = data.user

this.page5.current = data.topics.current

this.page5.size = data.topics.size

this.page5.total = data.topics.total

this.topics5 = data.topics.records

})

},

1. 获取用户认领情况：

@GetMapping("/choose/{username}")

public ApiResult<Map<String, Object>> getUserByNameForChoose(@PathVariable("username") String username,

@RequestParam(value = "pageNo", defaultValue = "1") Integer pageNo,

@RequestParam(value = "size", defaultValue = "10") Integer size) {

Map<String, Object> map = new HashMap<>(16);

user auser = userService.getUserByUserId(username);

Assert.notNull(auser, "用户不存在");

Page<choose> page = chooseService.page(new Page<>(pageNo, size),

new LambdaQueryWrapper<choose>().eq(choose::getChooseUserid, auser.getUserId()));

map.put("user", auser);

map.put("topics", page);

return ApiResult.success(map);

}

前端：fetchUserById3() {

getInfoByNameForChoose(this.$route.params.username, this.page.current, this.page.size).then((res) => {

const { data } = res

this.topicUser3 = data.user

this.page3.current = data.topics.current

this.page3.size = data.topics.size

this.page3.total = data.topics.total

this.topics3 = data.topics.records

})

},

1. 获取用户被认领情况：

@GetMapping("/bechoose/{username}")

public ApiResult<Map<String, Object>> getUserByNameForBeChoose(@PathVariable("username") String username,

@RequestParam(value = "pageNo", defaultValue = "1") Integer pageNo,

@RequestParam(value = "size", defaultValue = "10") Integer size) {

Map<String, Object> map = new HashMap<>(16);

user auser = userService.getUserByUserId(username);

Assert.notNull(auser, "用户不存在");

Page<choose> page = chooseService.page(new Page<>(pageNo, size),

new LambdaQueryWrapper<choose>().eq(choose::getChooseBeuserid, auser.getUserId()));

map.put("user", auser);

map.put("topics", page);

return ApiResult.success(map);

}

前端：fetchUserById4() {

getInfoByNameForBeChoose(this.$route.params.username, this.page.current, this.page.size).then((res) => {

const { data } = res

this.topicUser4 = data.user

this.page4.current = data.topics.current

this.page4.size = data.topics.size

this.page4.total = data.topics.total

this.topics4 = data.topics.records

})

},

1. 获取用户表白墙：

@GetMapping("/{username}")

public ApiResult<Map<String, Object>> getUserByName(@PathVariable("username") String username,

@RequestParam(value = "pageNo", defaultValue = "1") Integer pageNo,

@RequestParam(value = "size", defaultValue = "10") Integer size) {

Map<String, Object> map = new HashMap<>(16);

user auser = userService.getUserByUserId(username);

Assert.notNull(auser, "用户不存在");

Page<wall> page = wallService.page(new Page<>(pageNo, size),

new LambdaQueryWrapper<wall>().eq(wall::getWallUserid, auser.getUserId()));

map.put("user", auser);

map.put("topics", page);

return ApiResult.success(map);

}

前端：fetchUserById() {

getInfoByName(this.$route.params.username, this.page.current, this.page.size).then((res) => {

const { data } = res

this.topicUser = data.user

this.page.current = data.topics.current

this.page.size = data.topics.size

this.page.total = data.topics.total

this.topics = data.topics.records

})

},

1. 修改密码：

@PutMapping("/updatepass")

public ApiResult<?> updatepass(@RequestBody user u){

u.setUserPwd(MD5Utils.getPwd(u.getUserPwd()));

userService.getBaseMapper().updateById(u);

return ApiResult.success();

}

前端：submitForm(formName) {

this.$refs[formName].validate((valid) => {

if (valid) {

alert('submit!');

updatepass(this.ruleForm).then(res => {

this.$router.go(0)

})

} else {

console.log('error submit!!');

return false;

}

});

},

1. 修改个性签名：

@PutMapping("/updatesign")

public ApiResult<?> updatesign(@RequestBody user u){

u.setUserSignature((u.getUserSignature()));

userService.getBaseMapper().updateById(u);

return ApiResult.success();

}

前端：submitForm2(formName2) {

this.$refs[formName2].validate((valid) => {

if (valid) {

alert('submit!');

updatesign(this.ruleForm2).then(res => {

this.$router.go(0)

})

} else {

console.log('error submit!!');

return false;

}

});

},

1. 跳转到表白墙：

@GetMapping("/selectonecoll/{wallid}")

public ApiResult<wall> SelectOneColl(@PathVariable("wallid") Integer wallId) {

List<wall> list = wallService.list(new LambdaQueryWrapper<wall>()

.eq(wall::getWallId,wallId));

return ApiResult.success(list.get(list.size()-1));

}

前端：showwall(wallId){

sessionStorage.setItem("wallid",wallId)

this.$router.push({ path: '/CollectionWall' })

},

9．删除表白墙/评论/认领/收藏：

@DeleteMapping("/delete/{replyId}")

public ApiResult<?>deletereply(@PathVariable int replyId){

replyService.getBaseMapper().deleteById(replyId);

return ApiResult.success();

}

前端：handleDeleteReply(id) {

deletereply(id).then(value => {

const { code, message } = value

alert(message)

if (code === 200) {

this.$router.go(0)

}

})

},

1. 管理员模块

4.1 用户管理

4.1.1管理员编辑用户

@PutMapping("/update")

public ApiResult<?> update(@RequestBody user u){

userService.getBaseMapper().updateById(u);

return ApiResult.success();

}

4.1.2 管理员分页查询显示用户列表

@GetMapping("/findPage")

public ApiResult<Page<user>> findPage1(@RequestParam(defaultValue = "1") Integer pageNum,

@RequestParam(defaultValue = "10") Integer pageSize,

@RequestParam(defaultValue = "") String search){

Page<user> userPage=userService.getBaseMapper().selectPage(new Page<>(pageNum,pageSize), Wrappers.<user>lambdaQuery().like(user::getUserId, search));

return ApiResult.success(userPage);

}

4.1.3 管理员重置密码

@PutMapping("/reset")

public ApiResult<?> reset(@RequestBody user u){

u.setUserPwd(MD5Utils.getPwd("123456"));

userService.getBaseMapper().updateById(u);

return ApiResult.success();

}

4.2表白墙管理

4.2.1管理员更改表白墙状态

@PutMapping("/update")

public ApiResult<?> update(@RequestBody wall w){

wallService.getBaseMapper().updateById(w);

return ApiResult.success();

}

4.2.2管理员分页查询显示表白墙列表

@GetMapping("/findWallPage1")

public ApiResult<?> findPage1(@RequestParam(defaultValue = "1") Integer pageNum,

@RequestParam(defaultValue = "1") Integer pageSize,

@RequestParam(defaultValue = "") String search){

Page<wall> wallPage=wallService.getBaseMapper().selectPage(new Page<>(pageNum,pageSize), Wrappers.<wall>lambdaQuery().like(wall::getWallContenttitle, search));

return ApiResult.success(wallPage);

}

4.2.3管理员删除表白墙

@DeleteMapping("/delete/{wallId}")

public ApiResult<?>deletewall(@PathVariable int wallId){

wallService.getBaseMapper().deleteById(wallId);

return ApiResult.success();

}

4.3反馈管理

4.2.1管理员更改反馈状态

@PutMapping("/update")

public ApiResult<?> update(@RequestBody feedback f){

f.setFeedbackState("已处理");

feedbackService.getBaseMapper().updateById(f);

return ApiResult.success();

}

4.2.2管理员分页查询显示反馈列表

@GetMapping("/findPage")

public ApiResult<?> findPage(@RequestParam(defaultValue = "1") Integer pageNum,

@RequestParam(defaultValue = "10") Integer pageSize,

@RequestParam(defaultValue = "") String search){

Page<feedback> feedbackPage=feedbackService.getBaseMapper().selectPage(new Page<>(pageNum,pageSize), Wrappers.<feedback>lambdaQuery().like(feedback::getFeedbackUserid, search));

return ApiResult.success(feedbackPage);

}

4.2.3管理员删除反馈

@DeleteMapping("/delete/{feedbackId}")

public ApiResult<?>deleteuser(@PathVariable Integer feedbackId){

feedbackService.getBaseMapper().deleteById(feedbackId);

return ApiResult.success();

}