课程管理后台系统实现

1. 用户登录

包含用户登录和登出功能；

登录参考代码：

public User adminLogin(@RequestBody UserVo userVo) throws Exception {  
 if (userVo == null) {  
 return new User();  
 }  
 User user = userRepository.findByAccountAndPasswordAndStatus(userVo.getAccount(), userVo.getPassword(), 1);  
 if (user == null) {  
 throw new ServiceException("账号或密码错误");  
 }  
 if (RoleEnum.STUDENT.getValue().equals(user.getRole())) {  
 throw new ServiceException("您不是教师或者管理员,无法登录后台,请联系管理员");  
 }  
 return user;  
}

1. 用户管理

完成用户的添加、修改、删除等功能，根据登录用户的角色实现不同的权限；管理员可以添加学生和老师，老师登录只能添加学生；

查询学生列表功能：

public PagerHelper<User> list(String key, String role, Integer pageNo, Integer pageSize) {  
 PagerHelper<User> result = new PagerHelper<>();  
 StringBuffer sql = new StringBuffer("select \* from s\_user where 1=1");  
 List<Object> params = new ArrayList<>();  
 if (StringUtils.isNotEmpty(key)) {  
 sql.append(" and (user\_name like ? or account like ?)");  
 params.add("%" + key + "%");  
 params.add("%" + key + "%");  
 }  
 if (StringUtils.isNotEmpty(role)) {  
 sql.append(" and role = ?");  
 params.add(role);  
 }  
  
 //判断是否为管理员,管理员不进行过滤; 如果是老师，则查询对应班级的学生  
 User tempUser = userRepository.getOne(GaeaContext.getAdminUserId());  
 if (!RoleEnum.ADMIN.getValue().equals(tempUser.getRole())) {  
 List<Classes> classesList = classesRepository.findByTeacherId(GaeaContext.getAdminUserId());  
 if (CollectionUtils.isEmpty(classesList)) {  
 return result;  
 }  
 sql.append(" and class\_id in (" + buildParam(classesList, params) + ")");  
 }  
 StringBuffer totalSql = new StringBuffer("select count(1) from (" + sql.toString() + ") a");  
 Integer total = jdbcTemplate.queryForObject(totalSql.toString(), params.toArray(), Integer.class);  
 int startIndex = (pageNo - 1) \* pageSize;  
 sql.append(" limit ?,?");  
 params.add(startIndex);  
 params.add(pageSize);  
 List<User> users = jdbcTemplate.query(sql.toString(), params.toArray(), new BeanPropertyRowMapper<User>(User.class));  
 if (CollectionUtils.isNotEmpty(users)) {  
 Map<Integer, Classes> classesMap = getAllClassesMap();  
 for (User user : users) {  
 user.setClassName(user.getClassId() == null || classesMap.get(user.getClassId()) == null ? "" : classesMap.get(user.getClassId()).getClassName());  
 }  
 }  
 result.setTotal(total);  
 result.setItems(users);  
 return result;  
}

创建用户参考代码：

public User create(User user) throws Exception {  
 User tempUser = userRepository.findByAccount(user.getAccount());  
 if (tempUser != null) {  
 throw new ServiceException("账号【" + user.getAccount() + "】已存在");  
 }  
 if (StringUtils.isEmpty(user.getPassword())) {  
 user.setPassword(MD5Utils.MD5Encode("123456", "utf8"));  
 }  
 if (StringUtils.isEmpty(user.getRole())) {  
 user.setRole(RoleEnum.STUDENT.getValue());  
 }  
 user.setCreateTime(new Date());  
 user.setStatus(1);  
 return userRepository.save(user);  
}

1. 课程管理

获取课程列表参考代码：

public PagerHelper<CourseVo> list(String key, Integer tagId, boolean isFront, Integer pageNo, Integer pageSize) throws Exception {  
 PagerHelper<CourseVo> result = new PagerHelper<>();  
 StringBuffer sql = new StringBuffer("select \* from s\_course where 1=1");  
 List<Object> params = new ArrayList<>();  
 if (StringUtils.isNotEmpty(key)) {  
 sql.append(" and course\_name like ?");  
 params.add("%" + key + "%");  
 }  
 if (tagId != null) {  
 sql.append(" and tag\_id like ?");  
 params.add("%" + tagId + "%");  
 }  
 if (!isFront) {  
 //判断是否为管理员,管理员不进行过滤  
 User user = userService.get(GaeaContext.getAdminUserId());  
 if (!RoleEnum.ADMIN.getValue().equals(user.getRole())) {  
 sql.append(" and teacher\_id = ?");  
 params.add(GaeaContext.getAdminUserId());  
 }  
 }  
 StringBuffer totalSql = new StringBuffer("select count(1) from (" + sql.toString() + ") a");  
 Integer total = jdbcTemplate.queryForObject(totalSql.toString(), params.toArray(), Integer.class);  
  
 int startIndex = (pageNo - 1) \* pageSize;  
 sql.append(" limit ?,?");  
 params.add(startIndex);  
 params.add(pageSize);  
 List<CourseVo> courseVos = jdbcTemplate.query(sql.toString(), params.toArray(), new BeanPropertyRowMapper<>(CourseVo.class));  
 if (CollectionUtils.isNotEmpty(courseVos)) {  
 for (CourseVo courseVo : courseVos) {  
 //获取课程章节信息  
 List<CourseChapter> courseChapters = courseChapterRepository.findByCourseId(courseVo.getId());  
 courseVo.setCourseChapterVos(resolveChapter(courseChapters, courseVo.getId(), false));  
 //解析课程的标签  
 courseVo.setTags(getTagNames(courseVo.getTagId()));  
 }  
 }  
 result.setTotal(total);  
 result.setItems(courseVos);  
 return result;  
}

**创建课程章节代码参考：**

public CourseChapterVo create(CourseChapterVo courseChapterVo) {  
 List<CourseChapter> courseChapters = new ArrayList<>();  
 CourseChapter courseChapter = new CourseChapter();  
 BeanUtils.copyProperties(courseChapterVo, courseChapter);  
 courseChapter.setCreateTime(new Date());  
 courseChapter = courseChapterRepository.save(courseChapter);  
 //保存子章节信息  
 if (CollectionUtils.isNotEmpty(courseChapterVo.getCourseChapterVos())) {  
 for (CourseChapterVo chapterVo : courseChapterVo.getCourseChapterVos()) {  
 CourseChapter chapter = new CourseChapter();  
 BeanUtils.copyProperties(chapterVo, chapter);  
 chapter.setChapterParentId(courseChapter.getId());  
 chapter.setCourseId(courseChapterVo.getCourseId());  
 courseChapters.add(chapter);  
 }  
 }  
 courseChapterRepository.saveAll(courseChapters);  
 return courseChapterVo;  
}

修改章节代码：

public CourseChapterVo update(Integer id, CourseChapterVo courseChapterVo) throws Exception {  
 List<CourseChapter> courseChapters = new ArrayList<>();  
 CourseChapter courseChapter = courseChapterRepository.getOne(id);  
 if (courseChapter == null) {  
 throw new ServiceException("不存在【" + id + "】对应的章节");  
 }  
 BeanUtils.copyProperties(courseChapterVo, courseChapter);  
 courseChapter.setId(id);  
 courseChapterRepository.save(courseChapter);  
 //保存子章节信息  
 if (CollectionUtils.isNotEmpty(courseChapterVo.getCourseChapterVos())) {  
 courseChapterRepository.deleteByChapterParentId(id);  
 for (CourseChapterVo chapterVo : courseChapterVo.getCourseChapterVos()) {  
 CourseChapter chapter = new CourseChapter();  
 BeanUtils.copyProperties(chapterVo, chapter);  
 chapter.setChapterParentId(courseChapter.getId());  
 chapter.setCourseId(courseChapterVo.getCourseId());  
 courseChapters.add(chapter);  
 }  
 }  
 courseChapterRepository.saveAll(courseChapters);  
 return courseChapterVo;  
}

1. 班级管理
2. 标签管理
3. 作业管理
4. 教学管理