版本V1

UserVO extend User

SectionVO extend Section

ArticleVO extend Article

private User author;

CommentVO extend Comment

private User author;

ReplyVO extend Reply

private User author;

private User receiver;

NotificationVO extend Notification

private User sender;//触发通知用户

ChatVO extend Chat

private User receiver;//对方用户

MessageVO extend Message

private User receiver;

UserService 对应UserException extend ServiceException只用传String参构造方法

//注册成功则无异常抛出

//1.用户名已被使用则抛出UserNameException

//2.邮箱已注册则抛出UserEmailException

void register(User user);

//需要更新的user属性不为null，否则为null，id不能为null

//更新失败则抛出UserException

void updateUser(User user);

//根据userid 获取对应的UserVO对象

//获取成功返回对应实例，否则返回null

UserVO getUser(Integer userId);

//page为null则不分页查询（注意，跟获取全部是有区别的）

//likeUser 为null时，则不进行模糊查询，

//否则，若它属性不为null时，则对某个属性进行模糊查询

List<UserVO> getUsers(Page page, User likeUser);

SectionService 对应SectionException extend ServiceException只用传String参构造方法

//page为null则不分页查询（注意，跟获取全部是有区别的）

//likeSection 为null时，则不进行模糊查询，

//否则，若它属性不为null时，则对某个属性进行模糊查询

//orderByLastPublishArticleTime 为true则按最新排前，否则默认数据库排序

List<SectionVO> getSections(Page page, Section likeSection

, boolean orderByLastPublishArticleTime)

//根据sectionId 获取对应 sectionVO实例

//存在则返回对应实例，否则返回null

SectionVO getSection(Integer sectionId);

//更新不为null属性，更新失败则抛出SectionException ,其id不能为null

void updateSection(Section section);

//添加失败则返回SectionException 及它子类异常，否则什么都不发生

void addSection(Section section);

//删除对应sectionId的记录，并返回删除的记录数

//这是危险操作，系统不实现级联删除该版块下的文章及评论内容

//执行该操作前，需求该版块下的文章及评论内容已清空

int delectSection(Integer sectionId);

ArticleService 对应ArticleException extend ServiceException只用传String参构造方法

//page为null则不分页查询（注意，跟获取全部是有区别的）

//likearticle 为null时，则不进行模糊查询，否则，肯定有一个属性不为null

//否则，若它属性不为null时，则对某个属性进行模糊查询

//last\_active\_time 最新排前规则

List<ArticleVO> getArticles(Page page, Article likeArticle)

//根据articleId获取对应 articleVo实例

//存在则返回对应实例，否则返回null

ArticleVO getArticle(Integer articleId);

//添加一篇文章，添加成功后，用户文章数+1，所属版块文章数+1

//添加失败则抛出ArticleException 及其子类异常

void addArticle(Article article);

//更新不为null属性，更新失败则抛出ArticleException ,其id不能为null

void updateArticle(Article article)

//删除对应sectionId的记录，并返回删除的记录数

//删除成功后，所属该文章的主次评论连带删除，用户文章数-1，所属版块文章数-1

int deleteArticle(Integer articleId);

CommentService 对应CommentException extend ServiceException只用传String参构造方法

//page为null则不分页查询（注意，跟获取全部是有区别的）

//likeComment 为null时，则不进行模糊查询，

//否则，若它属性不为null时，则对某个属性进行模糊查询

//publish\_time 最新排前规则

List<CommentVO> getComments(Page page, Comment likeComment)

//发表一条主评论，发表失败则抛出CommentException 及其子类异常

void addComment(Comment comment);

//删除对应commentId的记录，并返回删除的记录数(级联删除)

int deleteComment(Integer commentId);

ReplyService 对应ReplyException extend ServiceException只用传String参构造方法

//page为null则不分页查询（注意，跟获取全部是有区别的）

//likeReply 为null时，则不进行模糊查询，

//否则，若它属性不为null时，则对某个属性进行模糊查询

//publish\_time 最新排后规则

List<ReplyVO> getReplies(Page page, Reply likeReply)

//发表一条次评论，发表失败则抛出ReplyException 及其子类异常

//发表成功，所属的主评论的次评论数+1

void addReply(Reply reply);

//删除对应replyId的记录，并返回删除的记录数(级联删除)

int deleteReply(Integer replyId);

//根据replyId获取对应实例，失败返回null

ReplyVO getReply(Integer replyId);

NotificationService 对应NotificationException extend ServiceException只用传String参构造方法

//page为null则不分页查询（注意，跟获取全部是有区别的）

//isReaded == null 则查询全部

//publish\_time 最新排前规则

List<NotificationVO> getNotifications(Page page, Integer userId)

//添加一条，添加失败则抛出NotificationException 及其子类异常

void addNotification(Notification notification);

//删除对应notificationId的记录，并返回删除的记录数

int deleteArticle(Integer notificationId);

//获取userId对应的用户未读通知数量

int countUnreadNotification(Integer userId);

ChatService 对应ChatException extend ServiceException只用传String参构造方法

//page为null则不分页查询（注意，跟获取全部是有区别的）

//last\_active\_time 最新排前规则

//根据当前用户userId获取

List<ChatVO> getChats(Page page, Integer userId);

//删除对应chatId和userid相符的记录，同时级联删除对应的Messages，并返回删除的//记录数

int deleteChat(Integer chatId);

//获取userId对应的用户未读私信数量

int countUnreadMessage(Integer userId);

MessageService 对应MessageException extend ServiceException只用传String参构造方法

//page为null则不分页查询（注意，跟获取全部是有区别的）

//send\_time 最新排前规则

//更新userid的chat信息，即置chat.unread 为 0

List<MessageVO> getMessages(Page page, Integer userId, Integer receiverId)

//添加一条私信，添加失败则抛出MessageException 及其子类异常

//1.1如果我跟对方的会话不存在，则新建我跟对话的会话，并且unread 为 0

//1.2如果对方跟我的会话不存在，则新建对话跟我的会话，并且unread 为 1

//1.2.1 如果对方与我的会话已存在，则unread + 1

//2.插入我跟对方私信内容，并插入对方与我私信内容

//3.更新我跟对方、对方跟我会话最新内容和时间

void addMessage(Message message);

RoleService

//根据roleId获取角色实例,没有则返回null

Role getRole(Integer roleId);

//获取所有的角色实例

List<Role> getAllRoles();

//原理：先全部删除该roleId对应的角色的权限，再重置权限  
 **int** updatePermissions(Integer roleId, List<Integer> permissionIds);

//获取所有的权限实例

List<Permission> getAllPermissions();

//根据roleId获取对应角色已拥有的权限

List<Permission> getPermissionsBy(Integer roleId);