2018-6-15

中大人社区

软件设计文档

目录

中大人社区软件设计文档

技术选型理由

用例图

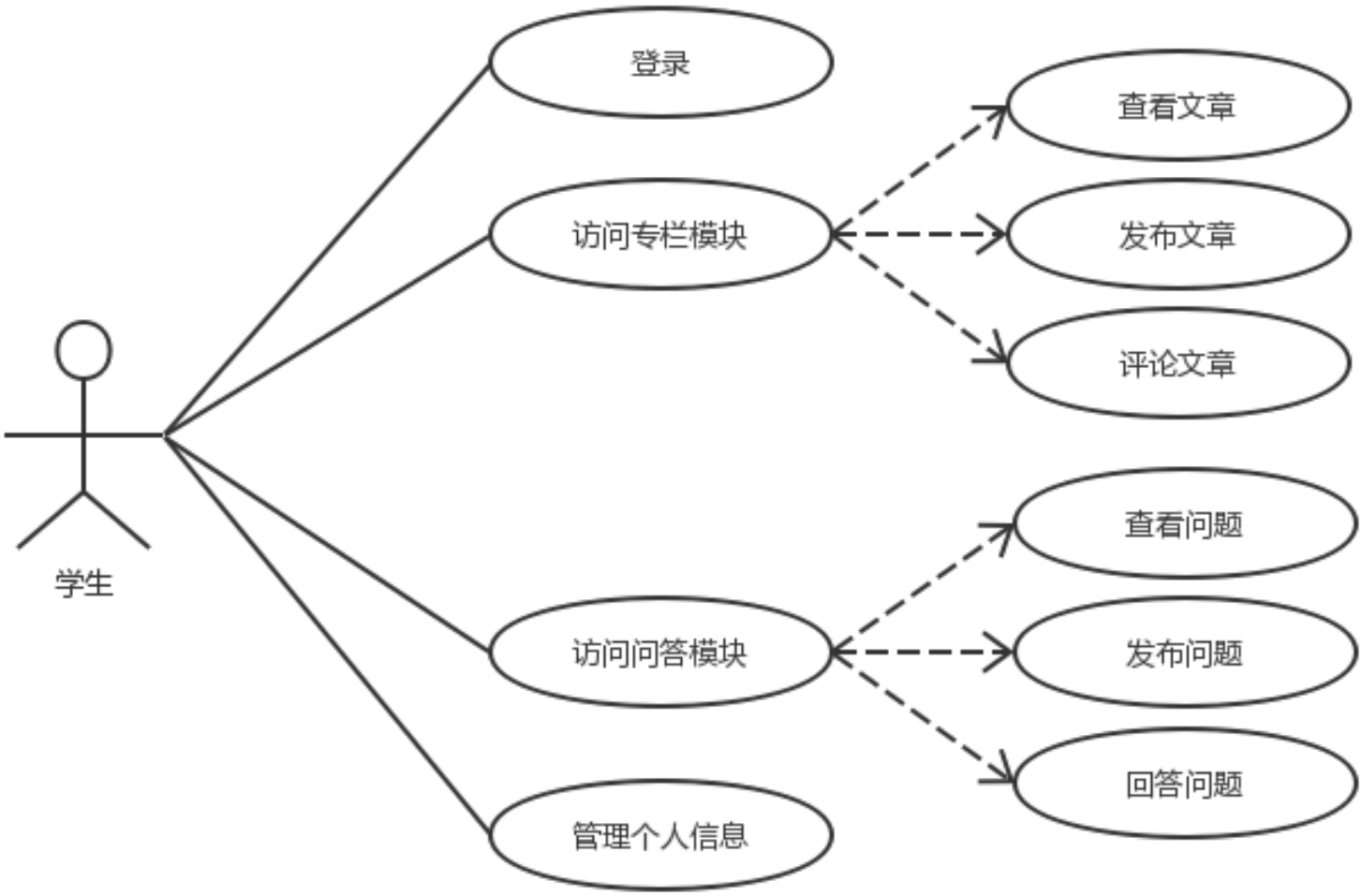


图 1- 1 用例图

技术选型

项目中用到的软件设计技术包括 Structure Programming、Object-Oriented Programming。

```java
public boolean delete(int ques_id, String userid) {
    /**
     * 如果作者是该作者 或 用户为管理员才能删除
     */
    if(questionDao.findQuetionById(ques_id).getAuthorId().equals(userid)
            || userid.equalsIgnoreCase("admin")){
    try {
        JdbcUtils.beginTransaction();
        questionDao.deleteQuestion(ques_id);
        JdbcUtils.commitTransaction();
        return true;
    } catch (SQLException e) {
        try {
            JdbcUtils.rollbackTransaction();
            return false;
        } catch (SQLException e1) {
            throw new RuntimeException(e);
        }
    }
} else {
    return false;
}
}
```

```java
public boolean delete(int c_id, String userid) {
    Comment comment = commentDao.findCommentById(c_id);
    if(comment.getAuthorId().equals(userid) || userid.equalsIgnoreCase("admin")){
        try {
            JdbcUtils.beginTransaction();
            commentDao.deleteComment(c_id);//只能删除当前的
            JdbcUtils.commitTransaction();
            return true;
        } catch (SQLException e) {
            try {
                JdbcUtils.rollbackTransaction();
                return false;
            } catch (SQLException e1) {
                throw new RuntimeException(e);
            }
        }
    }
    else {
        return false;
    }
```

寸于 Object-Oriented Programming，则是对对象的封装。

主源代码中，有着 User、Comment、Answer、Article、Question 对象，对其有状态和行为

主 Entity pacakge 中，五个类均为面向对象

```java
public class User {
    String user_id;
```

```java
public class Answer {
    int ans_id;
    String content;
    String authorId;
    int ques_id;
    Date releaseTime;
    public int getAns_id() {
        return ans_id;
    }
    public void setAns_id(int ans_id) {
        this.ans_id = ans_id;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
```

司时还有实现了接口，实现了解耦，将实体动作抽取出来。

在 Dao Package 中，AnswerMapper、ArticleMapper、UserMapper、CommentMappe tionMapper 是接口，定义了实体的动作。

```java
ublic interface AnswerMapper {
    /*
     * 利用回答内容关键词查找回答
     * 传入参数：content
     * 返回List<Answer>
     */
    public List<Answer> findAnswerByContent(String content);
    /*
     * 利用作者ID查找作者的回答
     * 传入参数：authorId
     * 返回List<Answer>
     */
    public List<Answer> findAnswersByAuthorId(String authorId);
    public Answer findAnswerById(int ansId);
    /*
     * 增加回答
   * 传入参数为一个Question对象
     */
    public void addAnswer(Answer answer);
    /*
     * 通过ansId删除回答
     */
    public void deleteAnswer(int ansId);
    /*
     * 更新文章
     * 传入参数为一个Answer对象
     */
    public void updateAnswer(Answer answer);
```

```java
public class UserMapperImpl implements UserMapper {

    private QueryRunner qRunner = new TxQueryRunner();

    @Override
    public User findUserById(String id) {
        String sql = "select * from t_user where user_id = ?";
        try {
            return qRunner.query(sql, new BeanHandler<User>(User.class), id);
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    public List<User> findUserByName(String username) {
        String sql = "select * from t_user where username like ?";
        try {
            return qRunner.query(sql, new BeanListHandler<User>(User.class),
                    "%" + username + "%");
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    public void editUser(User user) {
        String sql = "update t_user set username=?, icon=?, sex=?, password=?, college=?, gra
                + " email=? where user_id=?";
        Object[] params = { user.getUsername(), user.getIcon(), user.getSex(),
                user.getPassword(), user.getCollege(), user.getGrade(),
                user.getEmail(), user.getUser_id() };
        try {
            qRunner.update(sql, params);
```

```java
public class AnswerMapperImpl implements AnswerMapper {

    QueryRunner qRunner = new TxQueryRunner();

    @Override
    public List<Answer> findAnswerByContent(String content) {
        String sql = "select * from answer where content like ?";
        try {
            return qRunner.query(sql, new BeanListHandler<Answer>(Answer.class), "%"+content+"%"
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    public List<Answer> findAnswersByAuthorId(String authorId) {
        String sql = "select * from answer where authorId = ?";
        try {
            return qRunner.query(sql, new BeanListHandler<Answer>(Answer.class), authorId);
```

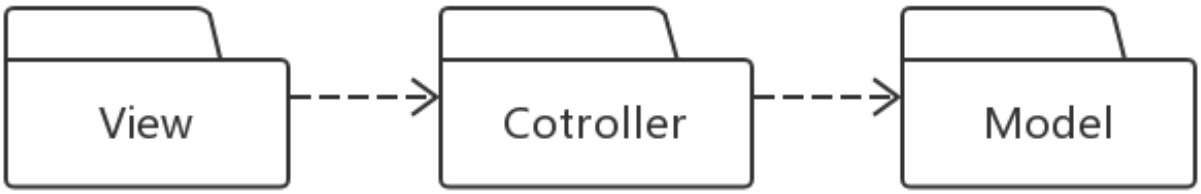


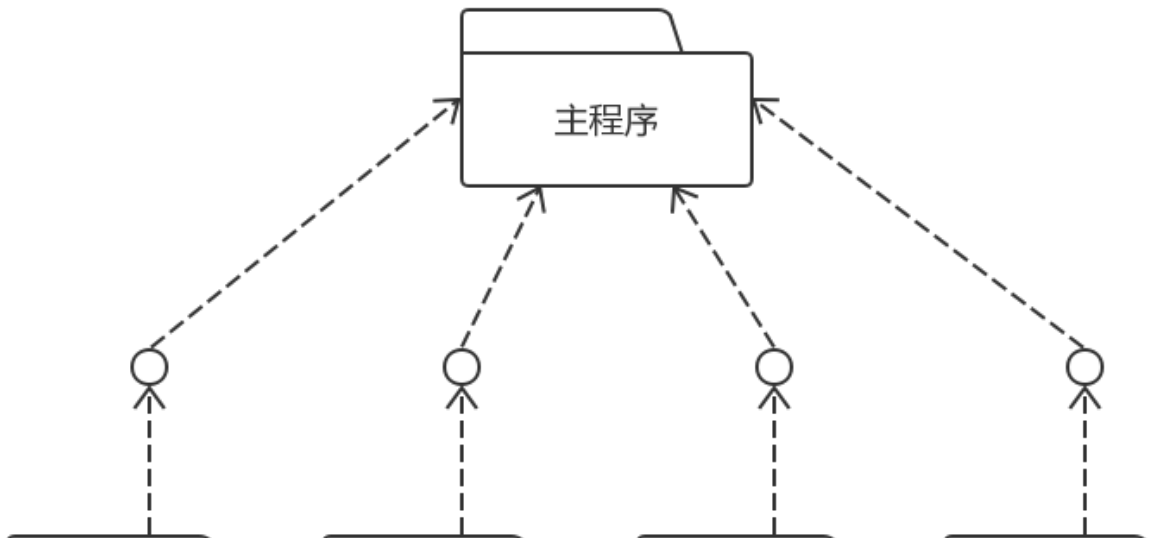
图 2-1　中大人社区架构图

领域模型

图 2-2　领域模型

模块划分

本社区包括四个模块：登录注册模块、文章管理模块、问答管理模块、个人信息管理模块。
统向主程序提供接口服务，主程序调用子系统提供的接口完成主要的流程。



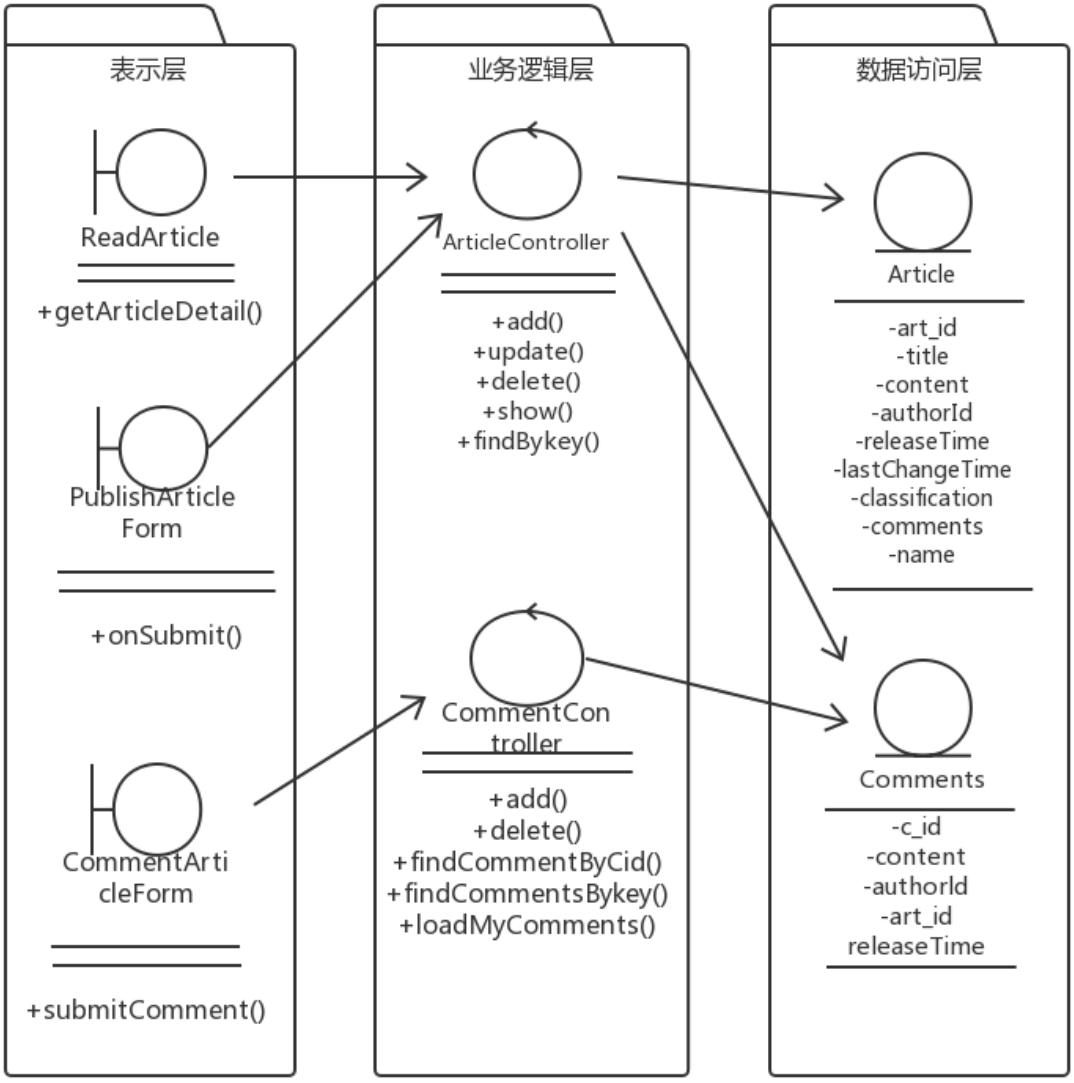


图 3-2　文章管理子系统及其接口设计

登录模块设计

登录子系统遵循总体架构的分层策略，分别分为表示层、业务逻辑层与数据访问层。

表示层以窗口的方式与用户进行交互，展示用户登录时需要输入的信息，包括用户的用户

等等。

业务逻辑层负责提供完整的数据流控制服务，对每一个数据实体提供一个管理器类，管理

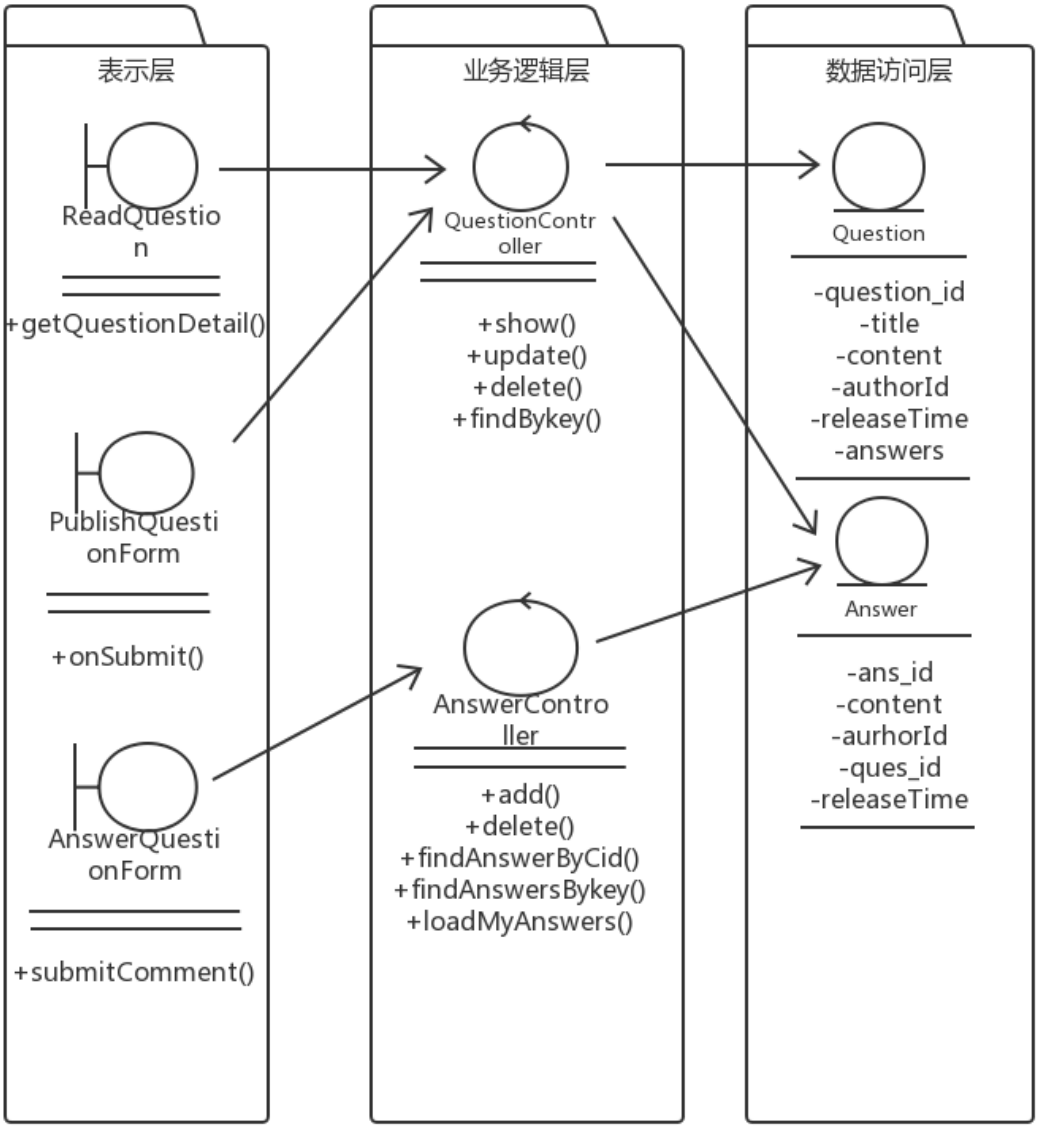
处理数据的更改和转移，是整个系统的中枢。管理器向上层提供调用接口，并调用下层的

服务来完成持久化的操作

问答管理模块设计

问答管理模块采用三层架构模式：表示层，业务逻辑层，数据库访问层。

表示层以窗口界面的形式与用户进行交互，展示社区上的所有问题，同时用户可以对问题操作，处理操作包括问题的回答、查找、修改、删除等。在交互过程中，表示层获取问题息后，以消息的形式传到业务逻辑层的控制器，控制器根据消息的类型，转发到相应的业处理类进行处理。处理完毕后，业务逻辑层处理类根据需要将处理后的消息进一步转发到层的相应类进行处理。最后，通过数据访问层对信息的持久化处理。



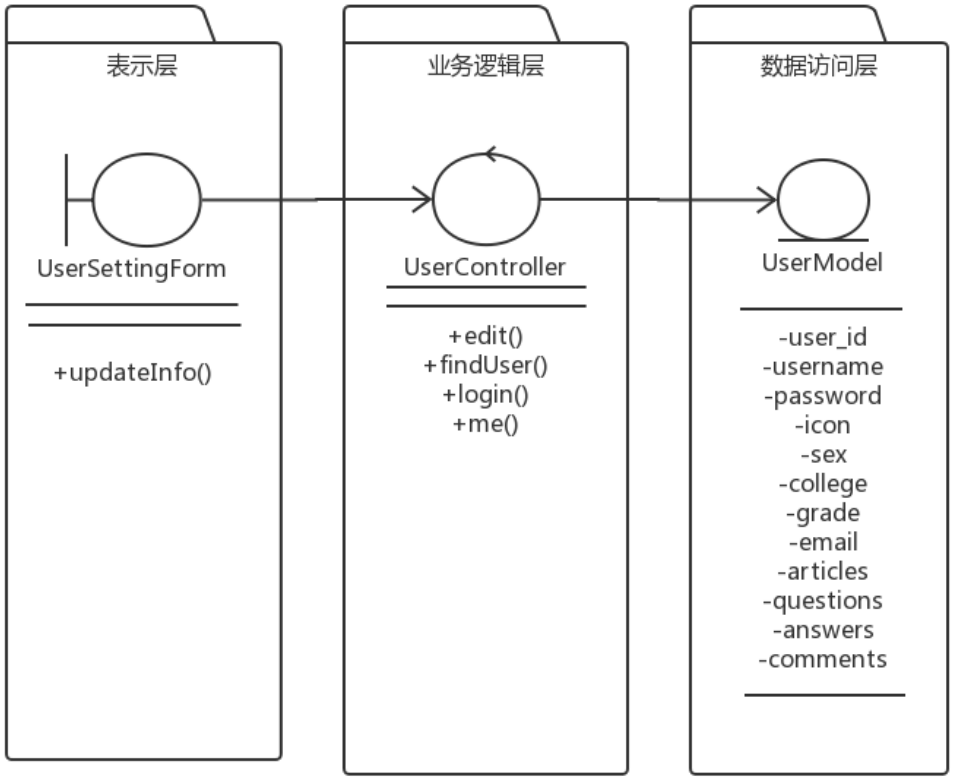


图 3-5　个人信息管理子系统及其接口设计

测试案例设计

对发布文章用例进行测试案例设计。由于发布文章并不涉及太多边界值，因此此处主要使用等价分方法设计测试用例。设计得到的等价类表以及测试用例如下：

入条件	有效等价类	无效等价类
章标题	0-30 个字符　（1）	多于 30 个字符　（2） 空　（3）
	以常见字符、数字开头　（5）	以标点符号或下划线开头　（6）
章分类	2 个字符　（7）	少于 2 个字符　（8） 多于 2 个字符　（9）
章内容	包含常见字符、数字、标点符号　（10）	非打印字符　（11）

表 4-1 等价类表

输入数据			覆盖等价类	预期输出
标题	分类	内容		