2018-7-18

中大人社区

软件设计文档

15331247 欧阳景潮　15331073 方虹　　153310751 方晓吟

15331128 黄梓锋　15331297 王茂源　15331067 邓子漫　15331138 赖达强

目录

中大人社区软件设计文档

1. 技术选型理由

1.1 用例图

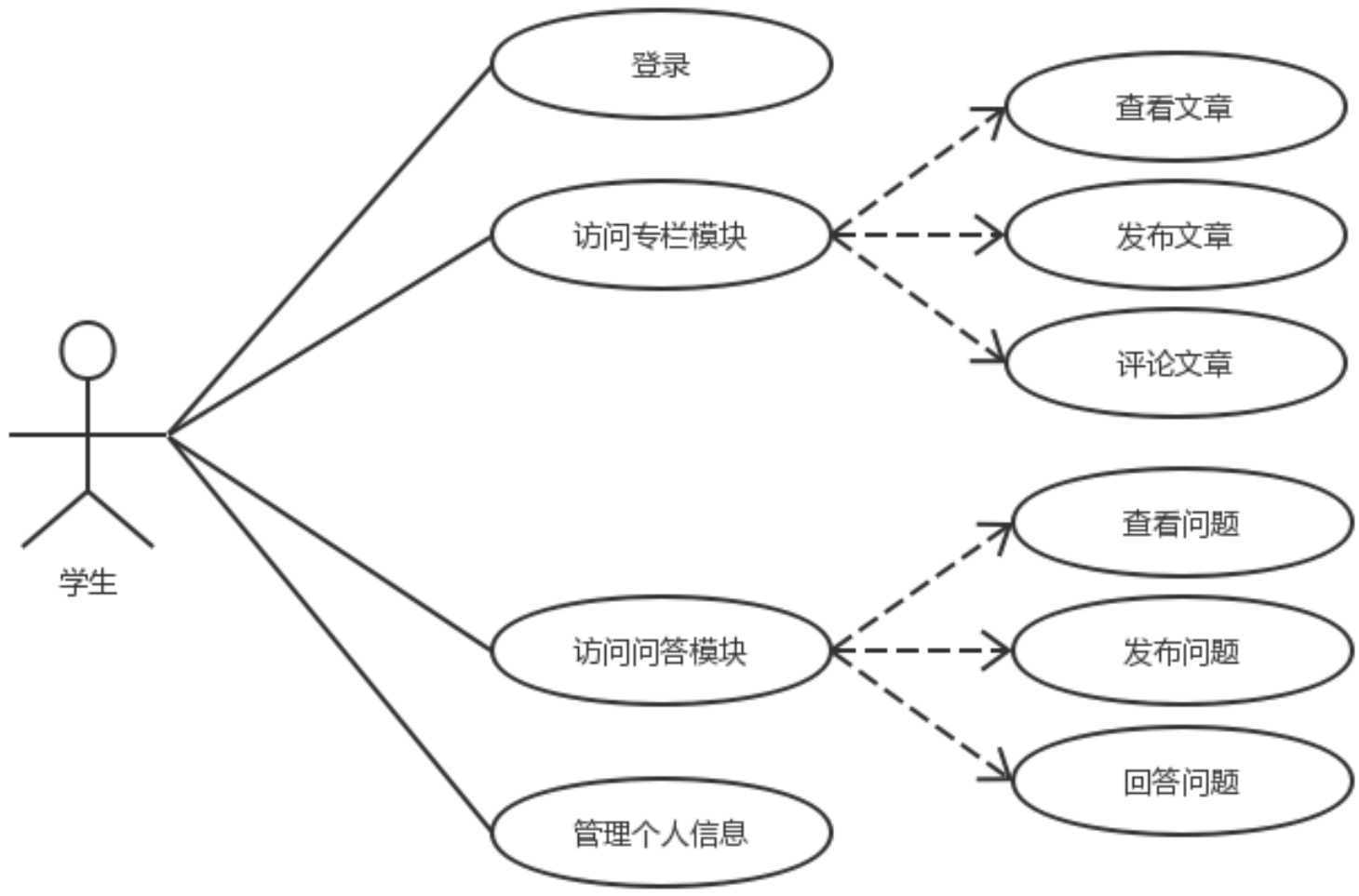

图 1- 1 用例图

1.2 技术选型

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

对于 Structure Programming，用到了三种基本程序结构，抽取了各种功能。

在源代码的 Service package 中，对于 UserService，CommentService，AnswerService，QuestionService，ArticleService 都用到了 Structure Programming，如下：

登录模块：

```
public boolean login(String cid, String password) {
    User user = userDao.findUserById(cid);
    if(user != null) {
        if(user.getPassword().equals(password))
            return true;
    }
    return false;
}
```

编辑用户资料模块：

```
public void edit(User user) {
    try {
        JdbcUtils.beginTransaction();
        User originalUser = userDao.findUserById(user.getUser_id());
        User newUser = implementUser(user, originalUser);
        userDao.editUser(newUser);
        JdbcUtils.commitTransaction();
    } catch (SQLException e) {
        try {
            JdbcUtils.rollbackTransaction();
        } catch (SQLException e1) {
        }
    }
}
```

删除问题、答案模块：

```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;
    }
```

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

在源代码中，有着 User、Comment、Answer、Article、Question 对象，对其有状态和行为的操作。

在 Entity pacakge 中，五个类均为面向对象

```java
public class User {
    String user_id;
    String username;
    String password;
    String icon;// 暂时先用作url
    String sex;
    String college;
    String grade;
    String email;
    List<Article> articles;
    List<Question> questions;
    List<Answer> answers;
    List<Comment> comments;
    public List<Article> getArticles() {
        return articles;
    }
    public void setArticles(List<Article> articles) {
        this.articles = articles;
    }
    public List<Question> getQuestions() {
        return questions;
    }
    public void setQuestions(List<Question> questions) {
        this.questions = questions;
    }
    public List<Answer> getAnswers() {
        return answers;
    }
    public void setAnswers(List<Answer> answers) {
        this.answers = answers;
    }
    public List<Comment> getComments() {
        return comments;
    }
```

```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;
    }
```

同时还有实现了接口，实现了解耦，将实体动作抽取出来。

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

```java
public 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);
    /*
     * 加载问题的答案
     */
    public List<Answer> loadAnswers(int ques_id);
    public Answer findLastInsert();
```

```java
public interface UserMapper {
    /*
     * 通过UserId查询User
     * 用于登录
     */
    public User findUserById(String id);
    /*
     * 通过用户名查找用户
     */
    public List<User> findUserByName(String username);
    /*
     * 编辑用户，更改用户信息
     */
    public void editUser(User user);
    /*
     * 增加用户 管理员可用
     */
    public void addUser(User user);
}
```

而 AnswerMapperImpl、ArticleMapperImpl、UserMapperImpl、CommentMapperImpl、QuestionMapperImpl 则是对接口的实现，实现了实际的动作。

每个实现中的各个函数都是动作的实现，各个模块都体现了接口、面向对象。

```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=?, grade=?,"
                + " 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);
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
```

2. 架构设计

2.1 描述架构

中大人社区将采用 MVC(Model View Controller)模式进行架构设计。MVC 将软件系统分为三个基本部分：模型（Model）、视图（View）和控制器（Controller）。该系统的使用实现了应用程序的分层管理，简化了后续对程序的修改和扩展，并且使程序某一部分的重复利用成为可能。

视图层即图形界面，为用户提供交互界面；

控制器负责转发请求，对请求进行处理，它控制着应用程序的流程，处理时间并进行响应；

模型层又称数据层，封装了与应用程序的业务逻辑相关的数据。

2.2 架构图

经分析，中大人社区的架构如图所示：

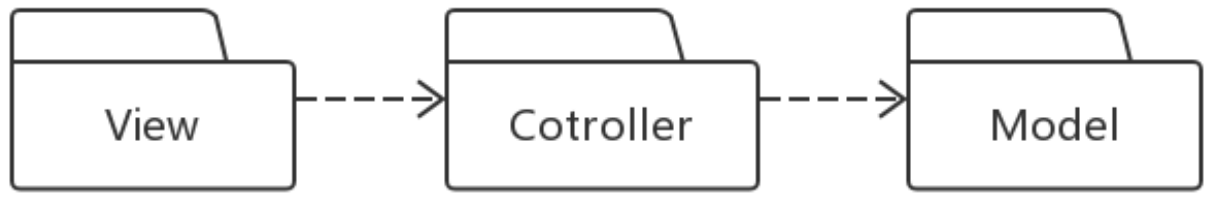

图 2-1　中大人社区架构图

2.3　领域模型

图 2-2　领域模型

3. 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。三个子系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。

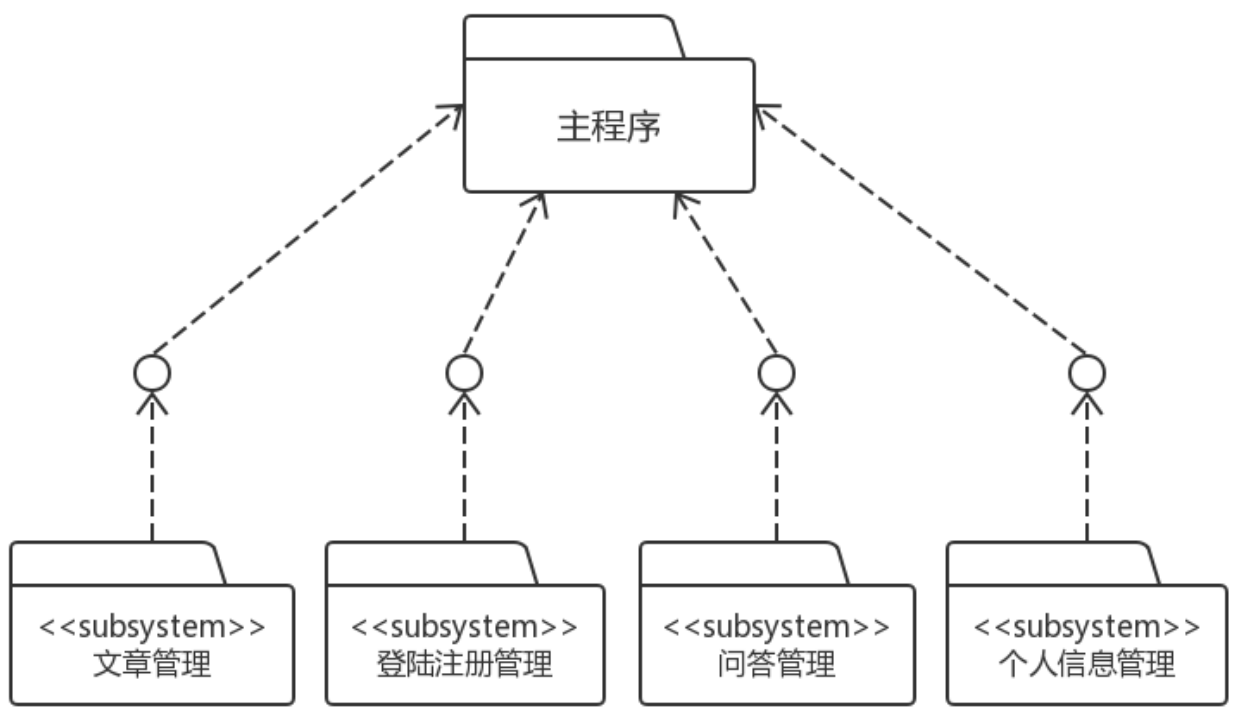

图 3-1　子系统调用子系统接口流程

3.1 文章管理模块设计

根据系统总架构设计图，文章管理模块也采用三层架构模式：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，展示用户文章的相关信息，同时负责对文章进行逻辑处理，处理操作包括文章的的增加、查找、修改、删除。

在交互过程中，表示层获取文章的相关信息后，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，转发到相应的业务逻辑层处理类进行处理。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数据访问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。

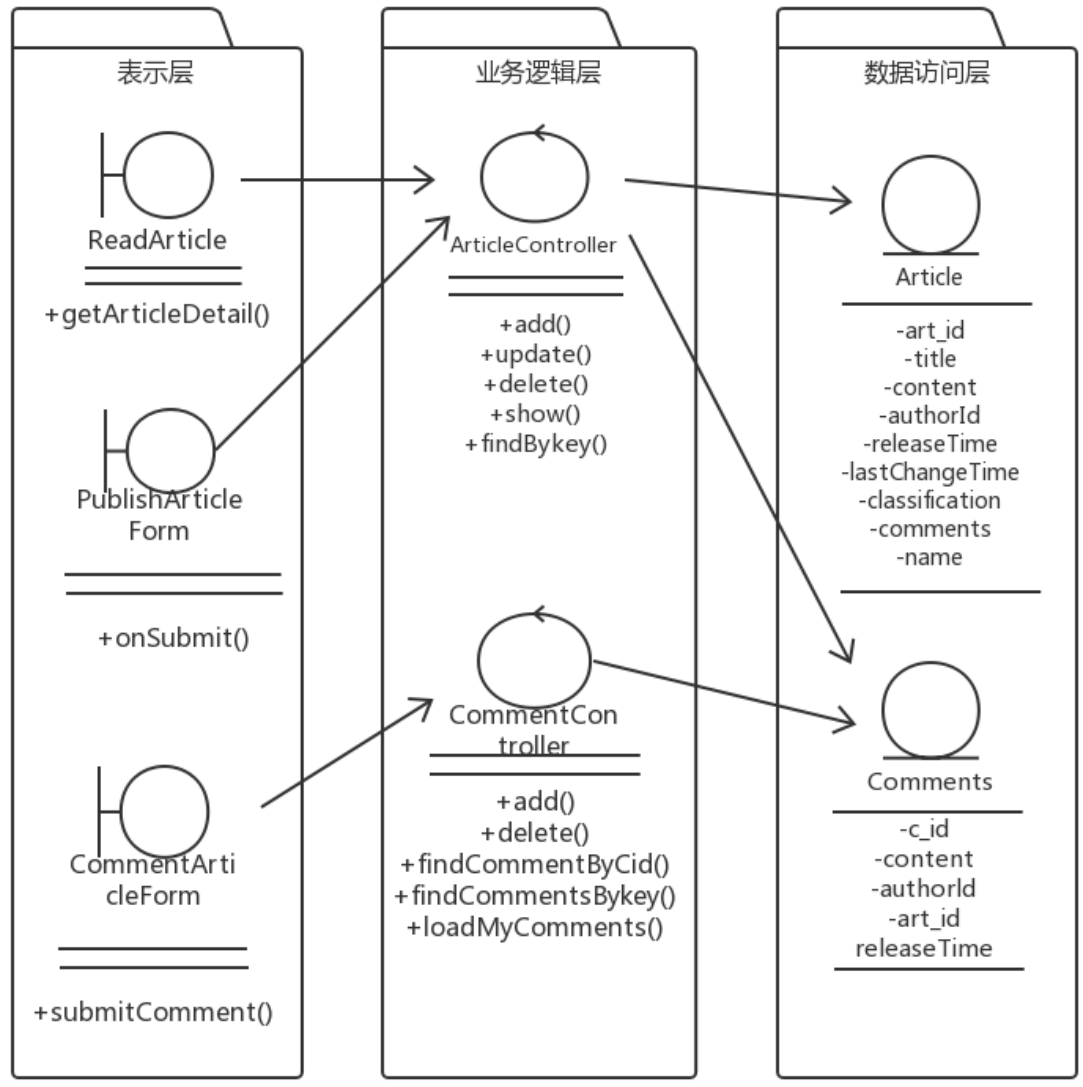

图 3-2　文章管理子系统及其接口设计

3.2 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，包括用户的用户名、密码等等。

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，管理器类负责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，并调用下层的数据访问服务来完成持久化的操作。

数据访问层封装了数据库的连接、读取、写入等操作，以提供数据库访问的服务。

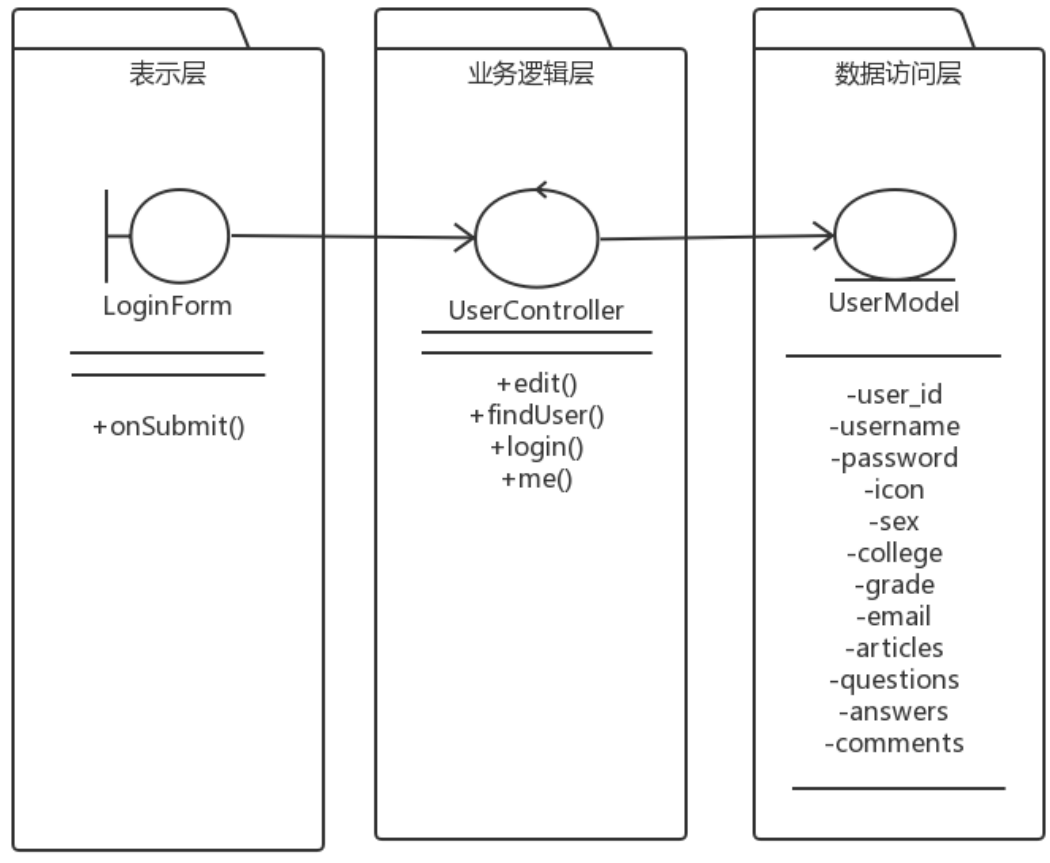

图 3-3　登陆模块设计

3.3 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进行基本操作，处理操作包括问题的回答、查找、修改、删除等。在交互过程中，表示层获取问题的相关信息后，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，转发到相应的业务逻辑层处理类进行处理。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数据访问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。

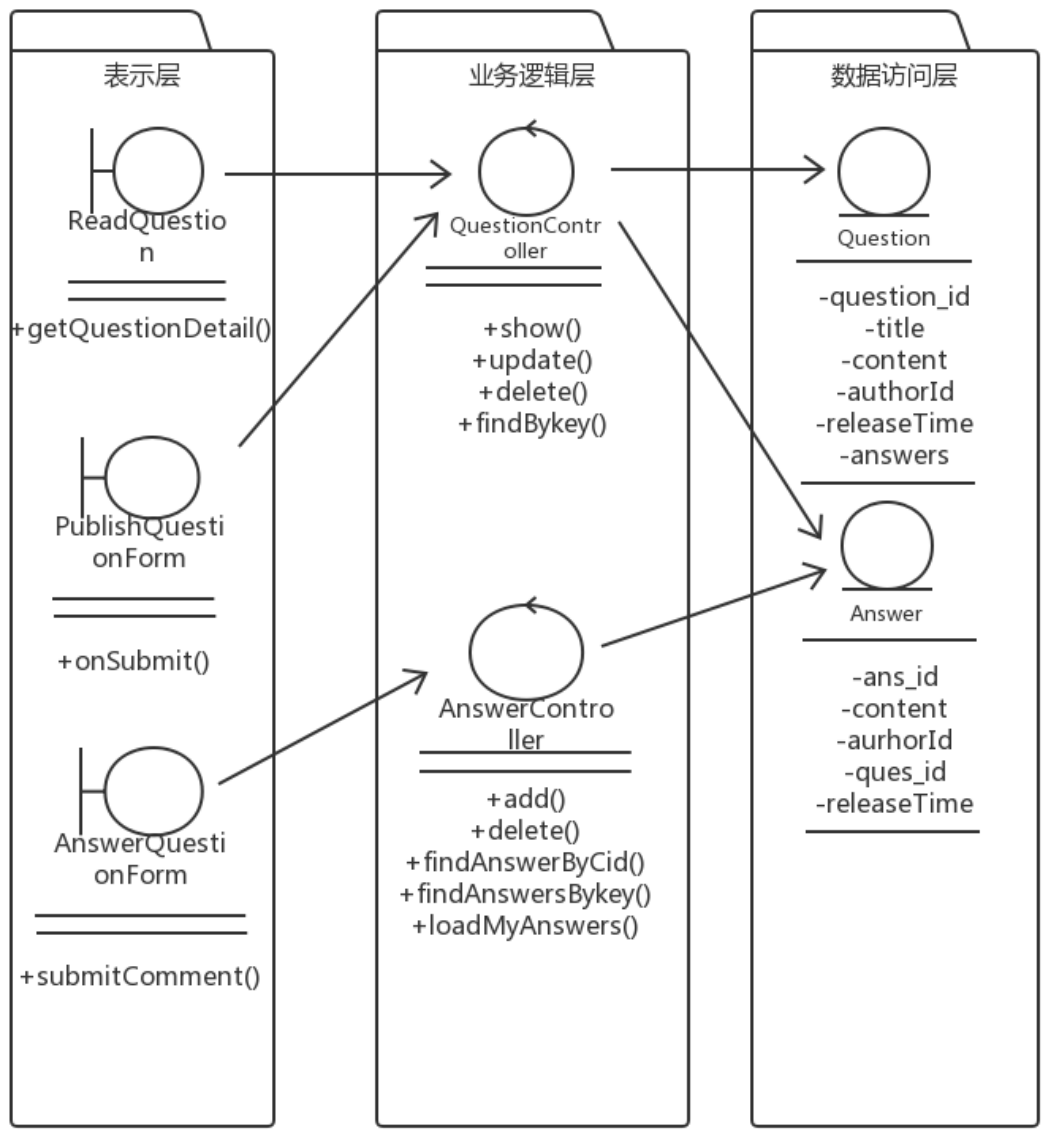

图 3-4　问答管理子系统及其接口设计

3.4 个人信息管理模块设计

个人信息管理模块采用三层架构模式：表示层、业务逻辑层、数据访问层。

表示层以窗口界面展示个人信息。用户可以浏览个人的用户信息，除此之外还可以对信息进行编辑、添加、删除等操作。

在交互过程中，表示层获取获取个人的相关信息后，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，转发到相应的业务逻辑层处理类进行处理。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数据访问层的相应类进行处理。最后，通过数据访问层对信息的更新。

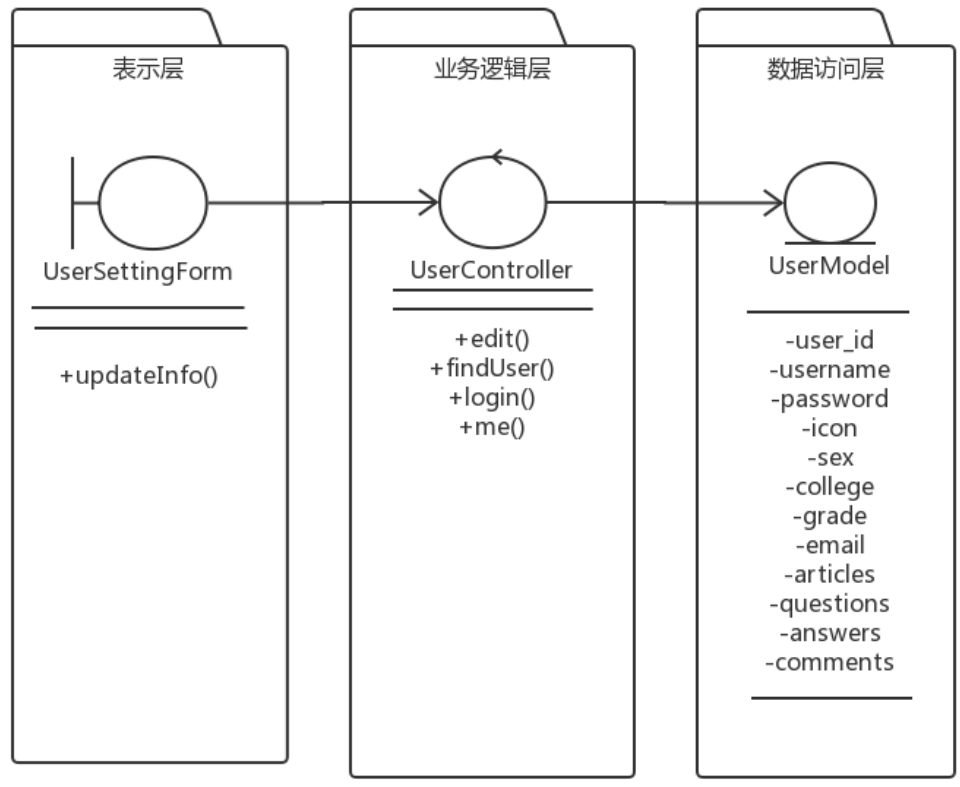

图 3-5　个人信息管理子系统及其接口设计

4. 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类划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：

输入条件	有效等价类	无效等价类
文章标题	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）
文章分类	2 个字符 （7）	少于 2 个字符 （8） 多于 2 个字符 （9）
文章内容	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

编号	输入数据			覆盖等价类	预期输出
	标题	分类	内容		
1	23	生活	今天很热。	1 5 7 10	合法输入
2	11111111111111 11111111111111 11111111111	生活	今天很热。	2 5 7 10	非法输入
3	NULL	生活	今天很热。	3 7 10	非法输入
4	23	人	今天很热。	1 5 8 10	非法输入
5	23	中大人	今天很热。	1 5 9 10	非法输入

表 4-2　测试用例