数据库大作业报告

张天祎 18340215

项目概述

- **http://106.53.110.132:8080/Test/welcome.html**
- 本网站为一个签到系统，可以让老师发布课程签到，学生可以加入课程并进行签到。

开发环境和开发工具

- 腾讯云服务器 Debian 6.3
    - JDK 1.8
    - MySQL 5.7
    - Tomcat 8.5
- 本地Windows 10

- IDEA 2019.2
- VScode 1.52
- Remote SSH 0.62
- Java 15（开发时为适配只使用了8）
- Tomcat 9.0

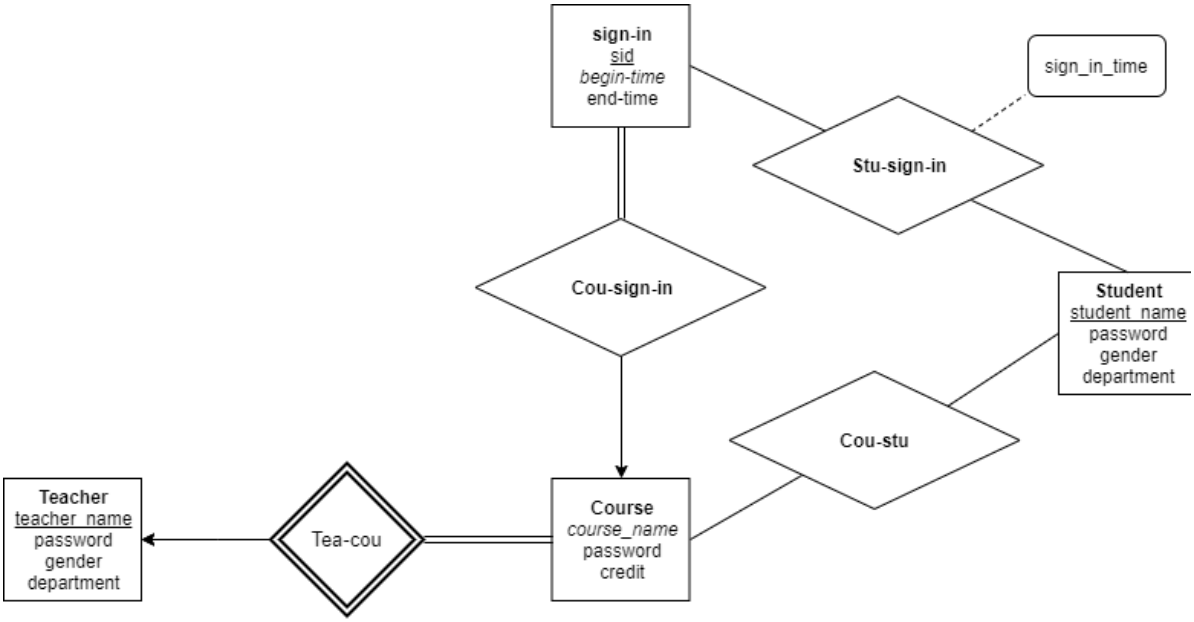
数据库设计及Java Web开发

需求分析

- 老师
  - 注册
  - 登入
  - 创建课程
  - 发布签到
- 学生
  - 注册
  - 登入
  - 加入课程
  - 进行签到

概念结构设计

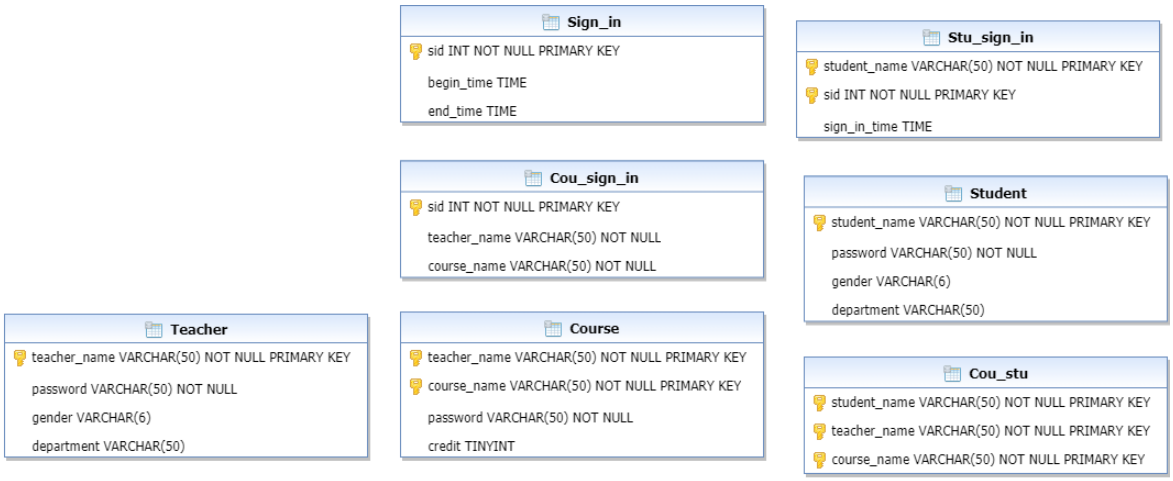
- 设计E-R图如下



- 注：这里没有虚线下划线所以用斜体替代了弱实体集分辨符。
- Course 表是依赖Teacher表的弱实体集，因为每一门课都必须有老师上，即课时属于老师的，这里产生了外键依赖。两者是多对一关系。
- 仅用课程名标识Course是不够的，不同的老师可能会上同名的课，所以还需要老师名属性共同作为主码。
- Course和Sign-in间是一对多关系。
- Course和Student间是多对多关系。
- Student和Sign-in间是多对多关系，二者联系集还有一个签到时间的额外属性。

逻辑结构设计

- 设计MySQL关系模式，EER图如下



- 各主码都是非空的。
- 为增强鲁棒性，字符串类型基本都设置为VARCHAR(50)。
- gender属性只有'male'，'female'两种，故用CHAR(6)。
- 签到开始、结束时间，学生签到时刻使用了Time类型。

物理结构设计

- 索引设计：需求上这里的表内元组都比较少，使用默认的主键索引就比较充分了。
- 表引擎，使用默认引擎InnoDB
    - 支持事务操作和设置隔离级别，默认可重复读
    - 支持外键操作

环境搭建

云服务器

- 这里使用了腾讯云服务器
- 使用了OneinStack的JAVA环境镜像
    - 默认安装好了JDK+MySQL+Tomcat环境
- 安全组配置放开端口
    - MySQL：3306
    - SSH：22
    - HTTP：80
    - Tomcat：8080
    - 默认配置放开了一些端口，但上述几个必须确保放开
    - 没有对入站IP进行限制，都可以访问

VScode + Remote SSH 远程免密登入服务器

- 如何登入云服务器终端？腾讯云默认网页终端和SSH都太弱了。
- 这里使用了VScode的Remote SSH插件，该插件一站式解决了登入终端和文件传输问题（只需要将文件拖进用户根目录即可），甚至支持本地打开根目录下的文件并进行修改。
- 具体配置步骤如下：
    - 在腾讯云默认网页终端登入
    - 关闭防火墙（因为后续可能需要多次开放端口，这里索性一次性处理）

- 开启SSH服务
- 将Windows下的SSH公钥拷贝至用户根目录的.ssh文件夹下的认证文件中并给予.ssh文件夹合适的权限
- 写Remote SSH配置文件

MySQL

- 安装和初始化过程已由镜像完成
- 根据原始密码登入，修改密码
- 授予所有IP远程登录的权限
- 开启mysqld服务
- 本地测试登入
- 设置字符集为utf8mb4（有时候默认已经为该设置了）

IDEA

- 安装配置Java
- 安装Tomcat
- 为进行本地开发测试，需要在IDEA下配置Java+Tomcat工具链，实现一键项目构建部署，可以在localhost下测试
- 测试好后，只需要打包war把放入服务器的Tomcat webapp目录下并运行Tomcat即可

后端开发

建立数据库

```
create table Teacher(
    teachername varchar(50) not null primary key,
    password varchar(50) not null,
    gender varchar(6),
    department varchar(50));

create table Student(
    studentname varchar(50) not null primary key,
    password varchar(50) not null,
    gender varchar(6),
    department varchar(50));

create table Course(
    teachername varchar(50) not null,
    coursename varchar(50) not null,
    password varchar(50) not null,
    credit tinyint,
    primary key(teachername, coursename));

create table Signin(
    sid int not null primary key,
    begintime time,
    endtime time);

create table CouSign(
    sid int not null primary key,
    teachername varchar(50) not null,
    coursename varchar(50) not null);

create table CouStu(
    studentname varchar(50) not null,
```

```
    teachername varchar(50) not null,
    coursename varchar(50) not null,
    primary key(studentname, teachername, coursename));

create table StuSignin(
    studentname varchar(50) not null,
    sid int not null,
    signintime time,
    primary key(studentname, sid));
```

- 根据数据库逻辑结构设计创建即可。

Java对象

- 根据数据库中的表建立对应的对象
- 代码看起来多，但实际上只需要定义对象的变量即可，使用IDEA可自动生成规范化的所有常规方法。

```
public class Student {
    String studentname;
    String password;
    String gender;
    String department;

    public Student() {
    }

    public Student(String studentname, String password) {
        this.studentname = studentname;
        this.password = password;
    }

    public Student(String studentname, String password, String gender, String
department) {
        this.studentname = studentname;
        this.password = password;
        this.gender = gender;
        this.department = department;
    }

    public String getStudentname() {
        return studentname;
    }

    public void setStudentname(String studentname) {
        this.studentname = studentname;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getGender() {
        return gender;
```

```
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public String getDepartment() {
        return department;
    }

    public void setDepartment(String department) {
        this.department = department;
    }

    @Override
    public String toString() {
        return "Student{" +
                "studentname='" + studentname + '\'' +
                ", password='" + password + '\'' +
                ", gender='" + gender + '\'' +
                ", department='" + department + '\'' +
                '}';
    }
}
```

JDBC

- druid+Spring
- druid是阿里巴巴的一个数据库连接池框架，使用它可以快速构建数据库连接池，加速开发，同时它也是支持高并发的。

```
public class JDBC_Util {
    private static DataSource ds;

    static {
        try {
            Properties pro = new Properties();
            InputStream is =
JDBC_Util.class.getClassLoader().getResourceAsStream("druid.properties");
            pro.load(is);
            ds = DruidDataSourceFactory.createDataSource(pro);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static DataSource getDateSource(){
        return ds;
    }

    public static Connection getConnection() throws SQLException {
        return ds.getConnection();
    }

}
```

- 和自己手写数据库连接池一样，要想优化项目结构同时让自己少犯错误，最好是使用java的反射机制写如下的配置文件来一站式控制数据库的连接。

```
driverClassName=com.mysql.cj.jdbc.Driver
url=jdbc:mysql://106.53.110.132:3306/DB
username=root
password=*******
initialSize=5
maxActive=10
maxWait=3000
```

- Spring是一个MySQL语句模板框架，使用它就不用自己反复地处理MySQL原生的API接口，加速开发。
- 下面的实例代码展示了一个典型JDBC流程，在该项目中针对不同对象，有大量类似下面代码的JDBC类，其它类可能语句不同，参数和返回值不同，但过程是一样的，故这里只发现一例。
- 首先利用从druid处获得的数据库连接创建一个MySQL语句模板，然后再try-catch语句块中执行MySQL语句。
- 这里示例的是select一行，所以还需要把结果通过JavaBean封装成对象返回。
    - 可以不用JavaBean，但会麻烦一点。
    - 能适用JavaBean要有默认空参构造函数和所有规范化get()，set()方法。
- 若语句执行失败，直接返回null交给调用者处理。
- 当然要使用preparestatement来执行MySQL，即使用'?'来代替查询变量，防止SQL注入问题。

```
public class JDBC_TeacherLogin {
    private JdbcTemplate template = new JdbcTemplate(JDBC_Util.getDateSource());

    public Teacher login(Teacher teacher){
        try {
            String sql = "select * from Teacher where teachername=? and password=?";
            return template.queryForObject(sql, new BeanPropertyRowMapper<>
(Teacher.class), teacher.getTeachername(), teacher.getPassword());
        } catch (DataAccessException e) {
            return null;
        }
    }
}
```

中间件servlet

- servlet是服务器处理请求的实体，它连接了前后端。
- 下方servlet是一个典型实例，项目中有大量的类似servlet，大同小异。
- 使用@WebServlet注解来指定servlet地址。
- 项目所有doGet请求都交给doPost处理。
- 首先将请求编码设置为"utf-8"防止中文乱码
- 然后执行JDBC方法，其中利用request.getParameter获取URL请求参数，根据结果决定下一步的处理。
- 这里如果SQL查询失败了，会将请求转发给LoginFailed这个servlet进一步处理请求，若成功则重定向到student_worktable.html，这里还传递了studentname的URL参数，来让HTML页面“认出”访问者。
    - 转发是在服务器完成的，重定向还需要客户端
    - 转发是一次请求，重定向是两次请求
    - 转发必须在一个服务器下，重定向可以访问不同的服务器（甚至其它资源）

```
@WebServlet("/StudentLogin")
```

```
public class StudentLogin extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        Student student = new JDBC_StudentLogin().login(new
Student(request.getParameter("studentname"), request.getParameter("password")));
        if(student==null){
            request.getRequestDispatcher("/LoginFailed").forward(request, response);
        }else {
            response.sendRedirect("./student_worktable.html?
studentname="+request.getParameter("studentname"));
        }
    }
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        doPost(request, response);
    }
}
```

前端开发

静态页面

- HTML+JavaScript
- 下面是一个利用JavaScript+正则表达式实现页面间的跳转的HTML例子，它实现了URL参数的传递。
- 实现的功能是学生端登入成功后选择加入课程还是去签到。

```
<!DOCTYPE html>
<html lang="ch">
<head>
    <meta charset="UTF-8">
    <title>StudentWorkTable</title>
</head>
<body>
    登入成功! <br><br>

    <script>
        function getURLParameter(name) {
            return decodeURIComponent((new RegExp('[?|&]' + name + '=' + '([^&;]+?)
(&|#|;|$)').exec(location.search)||[,""])[1].replace(/\+/g, '%20'))||null;
        }
        var studentname=getURLParameter("studentname");
        document.write("<a href='./signin.html?studentname="+studentname+"'>去签到</a>
<br><br>");
        document.write("<a href='./join_course.html?studentname="+studentname+"'>加入课
程</a><br>");
    </script>
</body>
</html>
```

- 下面是一个提交HTML表单的例子，利用了JavaScript来获取URL参数并提交表单隐藏属性
- HTML表单的input type项完成了很多前端的交互。

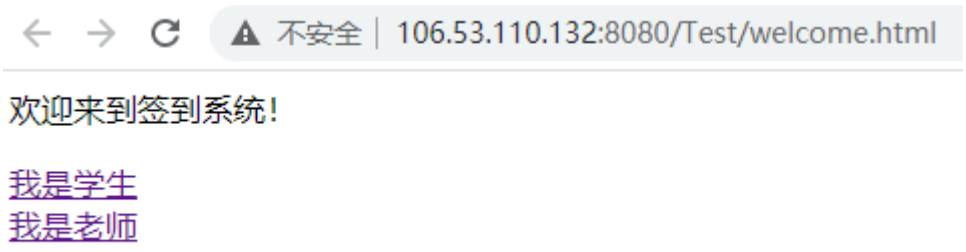
```
<!DOCTYPE html>
<html lang="ch">
```

```
<head>
    <meta charset="UTF-8">
    <title>JoinCourse</title>
</head>
<body>
    请输入课程信息：<br>
    <form action="./JoinCourse" method="post">
        <script>
            function getURLParameter(name) {
                return decodeURIComponent((new RegExp('[?|&]' + name + '=' +
'([^&;]+?)(&|#|;|$)').exec(location.search)||[,""])[1].replace(/\+/g, '%20'))||null;
            }
            var studentname=getURLParameter("studentname");
            document.write("<input name='studentname' type='hidden' value='"+
studentname +"'>");
        </script>
        <label for="teachername">
            老师：
            <input name="teachername" placeholder="请输入用户名" type="text"
id="teachername">
        </label><br>
        <label for="coursename">
            课程名：
            <input name="coursename" placeholder="请输入课程名" type="text"
id="coursename">
        </label><br>
        <label for="password">
            邀请码：
            <input name="password" placeholder="请输入邀请码" type="password"
id="password">
        </label><br>
        <input type="submit" value="加入">
    </form>
</body>
</html>
```

实验结果

首页

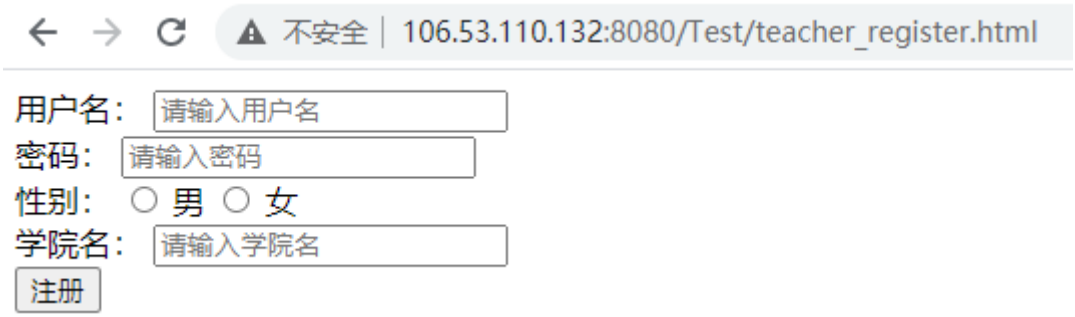
- 如下：

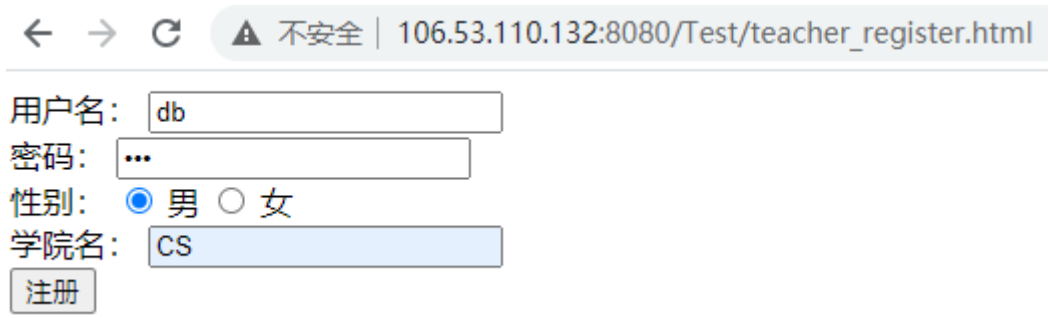


老师

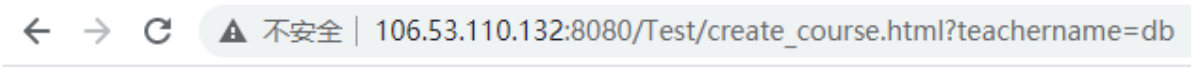
- 登入界面如下：

- 选择注册，密码为123，直接跳转到了登入成功界面：

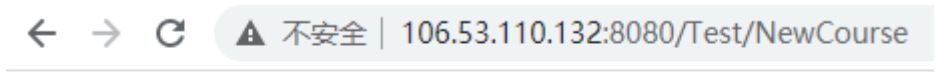




- 创建课程，邀请码为123

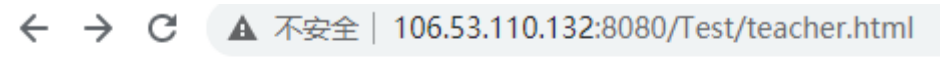


请输入课程信息：
课程名： db
邀请码： •••
学分： 4
创建



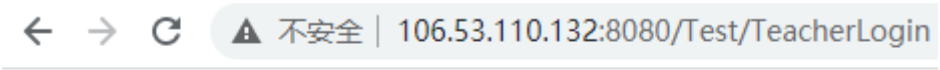
课程创建成功！

- 返回首页登入，先使用错误的密码321，再使用正确的密码123

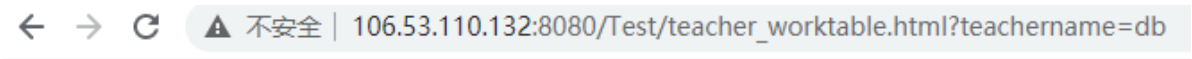


用户名： db
密码： •••
登入

没有账号？注册一个！



登入失败，用户名或密码错误！

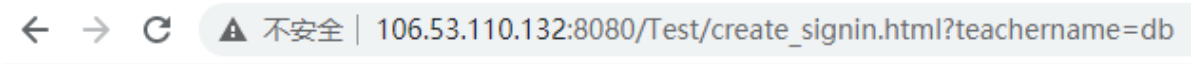


登入成功！

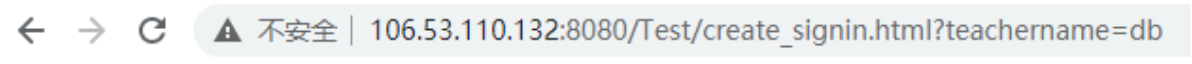
发布一次签到

创建课程

- 发布签到，为方便测试，签到时间设为1000s，签到码需要记住来让学生进行签到。



签到课程名： 请输入进行签到的课程名
签到时长： 请输入时长（s）
开始

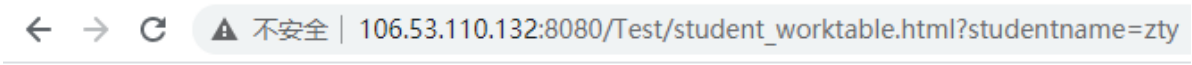


签到课程名： db
签到时长： 1000
开始

签到创建成功！签到码为3076476

学生端

- 登入和注册是和老师类似的，这里就不再展示，这里使用的账户为用户名zty，密码123
- 登入成功界面如下：

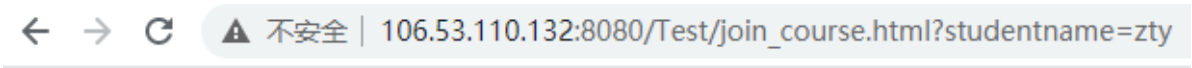


登入成功!

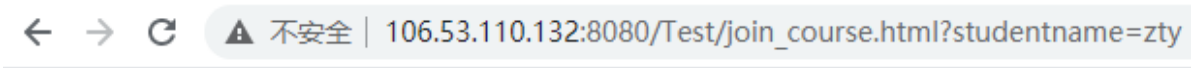
去签到

加入课程

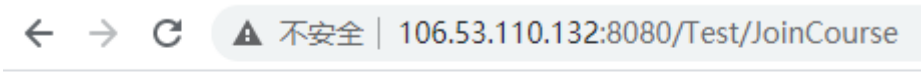
- 加入刚刚创建的课程



请输入课程信息:
老师: 请输入用户名
课程名: 请输入课程名
邀请码: 请输入邀请码
加入

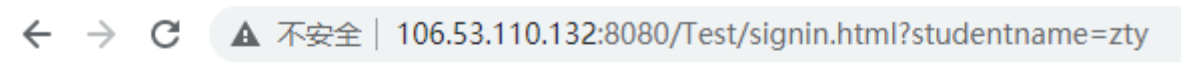


请输入课程信息:
老师: db
课程名: db
邀请码: •••
加入

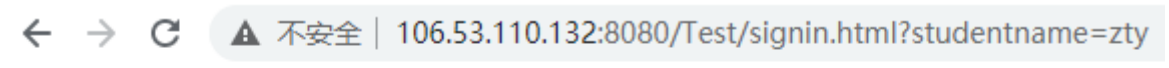


加入课程成功!

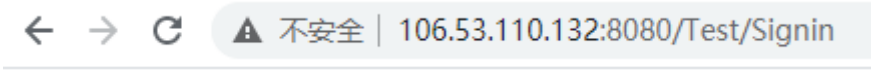
- 进行签到



签到码: 请输入签到码
签到



签到码: 3076476
签到



签到成功!

- 一个正常的操作流程就完成了!

- 错误输入或操作情况很多，这里就不赘述了，可以尝试一下，该项目基本上都进行了一些处理。

数据库运行维护

版本升级展望

- 有一个细节没处理好：时间应该用TimeStamp存，这样就不会出现第二天能签到的bug。
- 一旦提交了表单，客户端就看不到过去提交的信息了，所以最好完善一些查询类的功能。
- 前端比较简陋，希望接下来的版本能使用CSS或Vue框架。

总结和感想

- 最花时间的是学习，以前完全没有接触过Java，真正的从零开始全栈开发，即使是速成学习也花了一个月。
- 最头疼的是搭环境，各种环境的搭建总是令程序员头疼，虽然自己曾接触过不少Linux环境搭建，SSH配置相关的知识，但真的遇到各种稀奇古怪的Bug还是会严重影响开发进度。
    - 一开始自己的云服务器使用的是原生CentOS 8镜像，Remote SSH + Java + MySQL + Tomcat 四件套都是自己搭建的，期间克服了种种困难，成功上线了自己的测试Demo。
    - 但好景不长，在一次平平无奇的更新后服务器便出现了问题，只要启动Tomcat就会宕机。~~果然不能使用太新的版本吗~~
    - 花了一天也没有解决这个问题，最后含泪选择使用配好环境的镜像，熬夜把环境问题解决了。
- 开发反而比较快，一个星期就完成了。
- 开发时间有点和期末考试冲突了，时间有点紧，功能做的比较简单，有点可惜。

参考资料

- **黑马程序员Java Web教学视频**
    - 一个完美的从零开始Java Web开发教学视频
    - 1000P的视频，跳着看大概看来600P左右就能进行开发了
- **w3c在线教程**
    - 需要经常查API文档
- **腾讯云官方文档**
    - 建站相关
- **镜像使用手册**
    - 建站相关
- **CSDN**
    - Bug的可能解决途径