



**本科实验报告**

|  |  |
| --- | --- |
| 姓名： | 钟心柳 |
| 学院： | 生物医学工程与仪器科学学院 |
| 系： | 生物医学工程 |
| 专业： | 生物医学工程 |
| 学号： | 3190105744 |
| 指导教师： | 耿晨歌 |

2021年 12 月 28 日

**实验报告**

课程名称： 高级程序设计 实验类型： vscode

实验项目名称： Image-FileHost大作业

学生姓名：钟心柳专业：生物医学工程 学号：3190105744

指导老师： 耿晨歌 实验日期： 2021 年 12 月 28 日

# 实验目的和要求（必填）

# 后端语言限定java，

# 前端语言不限定

# 后端框架推荐springboot，spring

# 大作业要求上交实验报告与项目源码文件夹打包上传

# 实验内容和原理（必填）

1. **采用经典MVC架构**

**controller层:**

**FileHostController:文件处理(上传、下载、删除)**

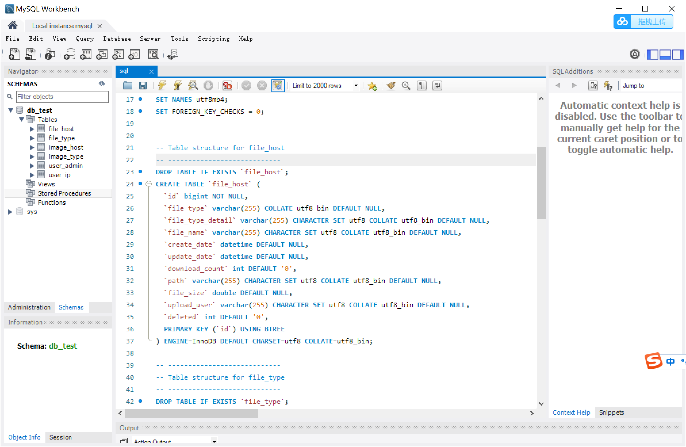
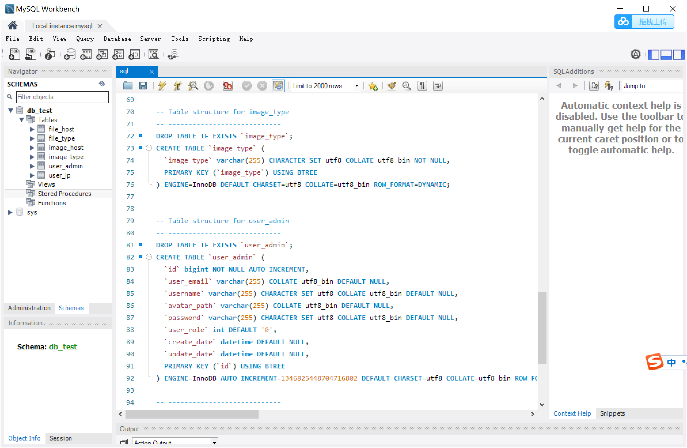
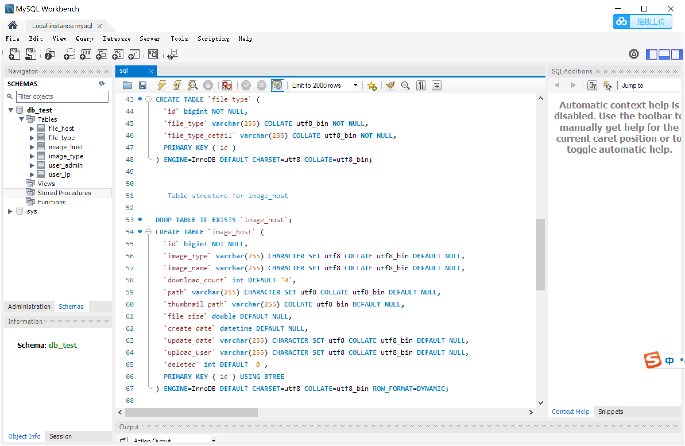
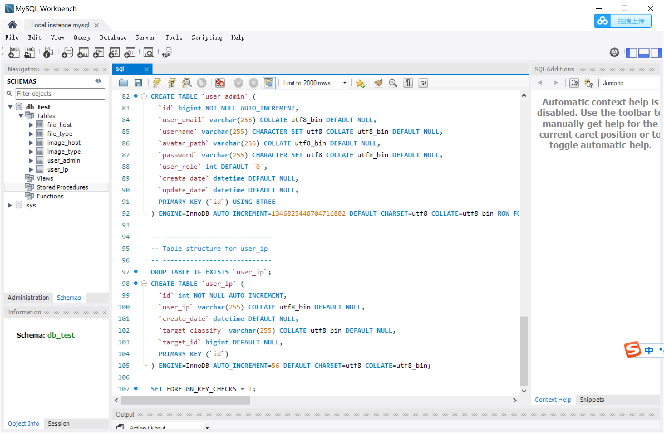
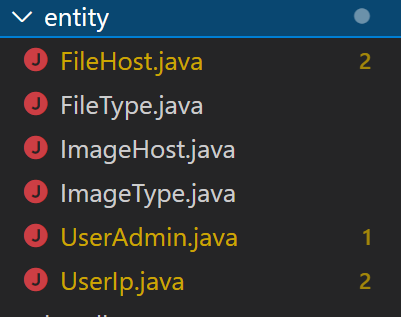
**FileTypeController:文件分类(查询分类、增加分类等)**

**ImageHostController: 图床管理(直链获取、图片下载、图片删除、图片上传等)**

**ImageTypeController:图片分类**

**UserAdminController:用户模块**

**UserIpController:IP记录处理**

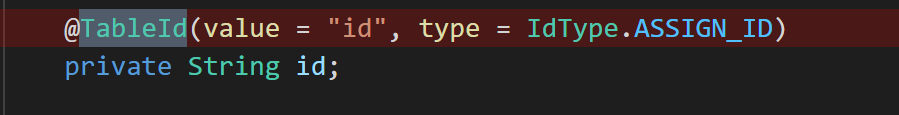
1. **配置数据库：MySQL**
2. **实体类：entity（使用Lombok简化代码）**

**相关注释：**

**主键专属：@TableId**

**比如数据中的表中的字段是id，但是实体类是userId，那么就需要在userId上打上这个注解。**

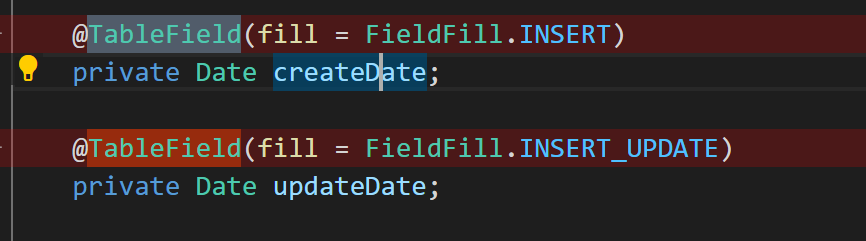
**用法：设置主键映射 value映射主键字段的名字，type 设置主键类型 主键的生成策略。**

****



**非主键字段使用：@TableField**

**比如数据中的表中的字段是name，但是实体类是userName，那么就需要在userName上打上这个注解。**



**映射非主键字段 value映射字段名****，exist 表示是否为是数据库字段。**

**在VO DTO的时候,经常会查询一些数据库中没有的字段,如果没有打上标签，那么查询会报错(如果实体类中的成员变脸在数据库中没有对应字段,可以视同)需要设置为false。**

**select 表示是否查询该字段，加上不参与查询,并且返回null。**

**fill 表示是否自动填充，将对象存入数据的时候,由mybatis-plus自动给某些字段赋值。**

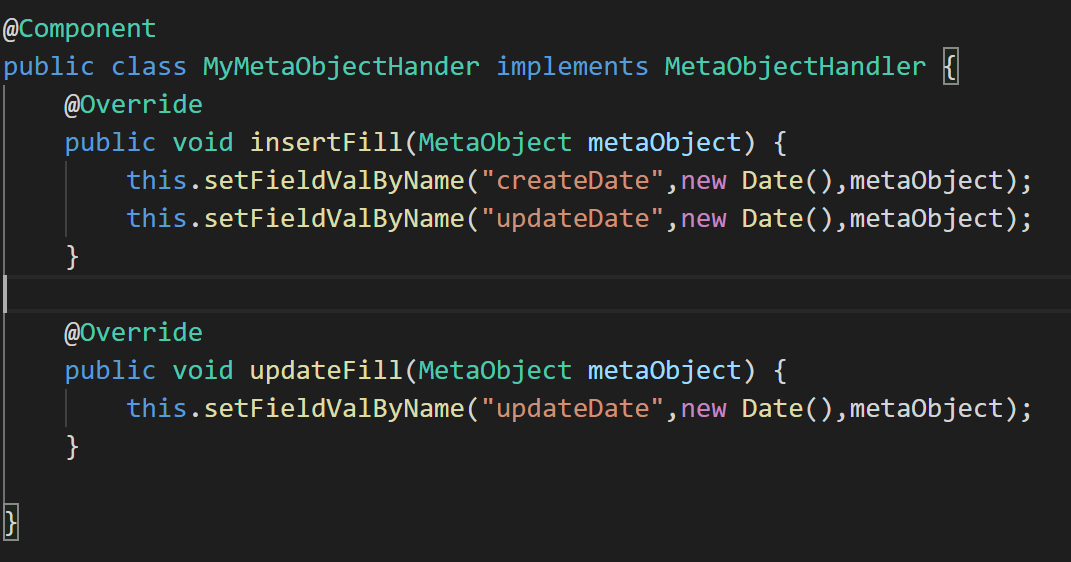
**DEFAULT --默认不处理**

**INSERT --插入时填充字段(第一次添加的时候)**

**INSERT\_UPDATE --插入和更新时填充字段**

**(最近一次更新的时候,也就是插入和更新都满足)**

**UPDATE --更新时填充字段**

**同时创建MyMetaObjectHander.java: 利用这个接口在我们插入或者更新数据的时候，为一些字段指定默认值**

1. **Mapper类：在对应的mapper上继承BaseMapper**
2. **Spring Boot 启动类：添加 @MapperScan 注解，扫描 Mapper 文件夹。**
3. **配置类：继承WebMvcConfigurer:采用JavaBean的形式来代替传统的xml配置文件形式进行针对框架个性化定制，可以自定义一些Handler，Interceptor，ViewResolver，MessageConverter。基于java-based方式的spring mvc配置，需要创建一个配置类并实现WebMvcConfigurer 接口**

**管理员拦截器配置，并指定拦截路径 /fileHost/\*\***

@Configuration

public class MyAdminCheckInterceptorConfig implements WebMvcConfigurer {

    @Override

    public void addInterceptors(InterceptorRegistry registry) {

        //注册TestInterceptor拦截器

        InterceptorRegistration registration = registry.addInterceptor(new AdminInterceptor());

        registration.addPathPatterns("/fileHost/\*\*");                      //所有路径都被拦截

        registration.excludePathPatterns("/fileHost/download/\*\*"

        );

    }

}

**管理员拦截器AdminIntercepter**

public class AdminInterceptor implements HandlerInterceptor {

    @Override

    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {

        //        System.out.println("执行了TestInterceptor的preHandle方法");

        try {

            //统一拦截（查询当前session是否存在user）(这里user会在每次登陆成功后，写入session)

            String role= String.valueOf(request.getSession().getAttribute("userRole"));

            if(role.equals("99")){

                return true;

            }

            response.sendRedirect(request.getContextPath()+"/");

        } catch (IOException e) {

            e.printStackTrace();

        }

        return false;//如果设置为false时，被请求时，拦截器执行到此处将不会继续操作

        //如果设置为true时，请求将会继续执行后面的操作

    }

}

**登录拦截器配置，并配置拦截路径 /”imageHost/\*\* " , "/userAdmin/\*\*" , "/fileType/\*\*" , "/fileHost/\*\*”**

@Configuration

public class MyLoginInterceptorConfig implements WebMvcConfigurer {

    @Override

    public void addInterceptors(InterceptorRegistry registry) {

        //注册TestInterceptor拦截器

        InterceptorRegistration registration = registry.addInterceptor(new LoginInterceptor());

        registration.addPathPatterns("/imageHost/\*\*","/userAdmin/\*\*","/fileType/\*\*","/fileHost/\*\*"); //所有路径都被拦截

        registration.excludePathPatterns("/imageHost/imageDetail/\*\*","/imageHost/getImage/\*\*","/imageHost/download/\*\*"

                ,"/userAdmin/login","/userAdmin/enter","/userAdmin/register","/userAdmin/email"

                ,"/fileHost/download/\*\*","/fileType/dataInfo","/fileType/homebrew","/fileType/fileTypeDetail/\*\*"

                ,"/userAdmin/logout"

        );

    }

}

**登录拦截器**

public class LoginInterceptor implements HandlerInterceptor {

    @Override

    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {

        //        System.out.println("执行了TestInterceptor的preHandle方法");

        try {

            //统一拦截（查询当前session是否存在user）(这里user会在每次登陆成功后，写入session)

            String user= String.valueOf(request.getSession().getAttribute("sessionUser"));

            if(user==null||!(user.equals("null"))){

                return true;

            }

            response.sendRedirect(request.getContextPath()+"/userAdmin/login");

        } catch (IOException e) {

            e.printStackTrace();

        }

        return false;//如果设置为false时，被请求时，拦截器执行到此处将不会继续操作

        //如果设置为true时，请求将会继续执行后面的操作

    }

}

**MyBatisPlusConfig.java:构造乐观锁方法**

**OptimisticLockerInnerInterceptor 当要更新一条记录的时候，希望这条记录没有被别人更新。乐观锁实现方式： 取出记录时，获取当前version 更新时，带上这个version 执行更新时， set version = newVersion where version = oldVersion 如果version不对，就更新失败**

1. **Spring cloud的Maven插件：repackage目标**

**repackage：创建一个自动可执行的jar或war文件。它可以替换常规的artifact，或者用一个单独的classifier附属在maven构建的生命周期中。在原始Maven打包形成的jar包基础上，进行重新打包，新形成的jar包不但包含应用类文件和配置文件，而且还会包含应用所依赖的jar包以及Springboot启动相关类（loader等），以此来满足Springboot独立应用的特性；将原始Maven打包的jar重命名为XXX.jar.original作为原始文件。**

**执行测试用例的插件：surefire-plugin，配置skipTests,true则跳过测试用例。**

   <build>

        <plugins>

            <plugin>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-maven-plugin</artifactId>

                <!--这里写上main方法所在类的路径-->

                <configuration>

                    <includeSystemScope>true</includeSystemScope>

                    <mainClass>cn.zxl.filecloudplus.FilecloudplusApplication</mainClass>

                </configuration>

                <executions>

                    <execution>

                        <goals>

                            <goal>repackage</goal>

                        </goals>

                    </execution>

                </executions>

            </plugin>

            <plugin>

                <groupId>org.apache.maven.plugins</groupId>

                <artifactId>maven-surefire-plugin</artifactId>

                <version>2.22.2</version>

                <configuration>

                    <skipTests>true</skipTests>

                </configuration>

            </plugin>

        </plugins>

    </build>

1. **模板引擎：Thymeleaf**

**Thymeleaf 是一款用于渲染 XML/XHTML/HTML5 内容的模板引擎。与 Spring MVC 等 Web 框架进行集成作为 Web 应用的模板引擎。**

**能够直接在浏览器中打开并正确显示模板页面，而不需要启动整个 Web 应用。**

**th:id替换id<input th:id="${user.id}"/>**

**th:text文字替换<p >text:="${user.name}">anything</p>**

**th:utext支持html的文字替换<p utext:="${htmlcontent}">content</p>**

**th:object替换Object<div th:object="${user}"></div>**

**th:value替换数值<input th:value="${user.name}" >**

**th:each迭代<tr th:each="student:${user}" >**

**th:href替换超链接<a th:href="@{index.html}">超链接</a>**

**th:src替换src<script type="text/javascript" th:src="@{index.js}"></script>**

**通过标签中的th:text属性来填充该标签的一段内容**

<span

th:text="${#httpServletRequest.getSession().getAttribute('sessionUser')}" class="align-text-top" style="padding-left: 0.625rem;">Zxl</span>

**字符串拼接**

<span class="input-group-text" th:text="'当前分类： ' + ${detailType}">

当前分类：NULL</span>

**用th:with来定义变量**

<th:block th:with="var1=0"></th:block>

th:block th:with="var1=${#var1}"></th:block>

**循环**

**在被循环渲染的元素（这里是<tr>）中加入th:each标签，其中th:each="file : ${fileList}"意味着对集合变量file进行遍历，循环变量file在循环体中可以通过表达式访问。**

<tr th:each="file:${fileList}">

   <td style="vertical-align: middle" th:text="${file.getFileName()}" scope="row">${document.document\_name}</td>

   <td style="vertical-align: middle" th:text="${#dates.format(file.getUpdateDate(),'yyyy-MM-dd HH:mm:ss')}">${document.creation\_date}</td>

   <td style="vertical-align: middle" th:text="${#numbers.formatDecimal(file.getFileSize()\*file.getDownloadCount()/(1024\*1024),1,2)}">${document.file\_size}</td>

   <td style="vertical-align: middle" th:text="${file.getDownloadCount()}">${document.download\_count}</td>

   <th:block th:with="var1=${#var1}"></th:block>

                </tr>

**条件求值**

**If/Unless**

**Thymeleaf 中使用th:if和th:unless属性进行条件判断，下面的例子中，<a>标签只有在th:if中条件成立时才显示：**

<a

th:if="${#httpServletRequest.getSession().getAttribute('sessionUser')=

null}" href="/userAdmin/login" class="nav-link ">登录</a>

**th:unless于th:if恰好相反，只有表达式中的条件不成立，才会显示其内容。**

<div

th:unless="${#httpServletRequest.getSession().getAttribute('sessionUse

')==null}" href="/userAdmin/admin" onclick="goToAdmin()" class="border

bg-light text-dark btn">

<img class="rounded-circle" style=" height:1.9rem;" th:src="@{'/userAdmin/getAvatar/'+${#httpServletRequest.getSession().getAttribute('sessionEmail')}}"> <span th:text="${#httpServletRequest.getSession().getAttribute('sessionUser')}" class="align-text-top" style="padding-left: 0.625rem;">Zxl</span>

                </div>

**Utilities**

**为了模板更加易用，Thymeleaf 还提供了一系列 Utility 对象（内置于 Context 中），可以通过#直接访问：**

**#dates**

<td style="vertical-align: middle"

th:text="${#dates.format(file.getCreateDate(),'yyyy-MM-dd

HH:mm:ss')}">${document.creation\_date}</td>

<td style="vertical-align: middle" th:text="${#dates.format(file.getUpdateDate(),'yyyy-MM-dd HH:mm:ss')}">${document.creation\_date}</td>

**使用内置对象dates的format函数即可对日期进行格式化，在format函数中，第一个参数是日期对象，第二个参数为日期格式（规则跟SimpleDateFormat一样）**

**#numbers**

<span

th:text="${#numbers.formatDecimal(fileSizeTotal/(1024\*1024),1,2)}"

style="background: linear-gradient(to right, #F97794, #623AA2);

        -webkit-background-clip: text;background-clip: text;

        color: transparent;">NULL</span>

1. **前端框架：Bootstrap**

**Bootstrap，来自 Twitter，是目前最受欢迎的前端框架。Bootstrap 是基于 HTML、CSS、JavaScript的，它在jQuery的基础上进行了更为个性化和人性化的完善，形成一套自己独有的网站风格，并兼容大部分jQuery插件。**

**本地调用：**

    <!--    bootstrap套件-->

    <link rel="stylesheet" href="/bootstrap/css/bootstrap.min.css" />

    <script src="/bootstrap/js/bootstrap.min.js"></script>

    <link href="https://libs.cdnjs.net/flat-ui/2.3.0/css/flat-ui.min.css" rel="stylesheet" />

    <link rel="icon" type="image/png" href="/img/icon/favicon.svg" />

**为了让网站对移动设备友好，确保适当的绘制和触屏缩放，需要在网页的 head 之中添加 viewport meta 标签，通过为 viewport meta 标签添加 user-scalable=no 可以禁用其缩放（zooming）功能。通常情况下，maximum-scale=1.0 与 user-scalable=no 一起使用。这样禁用缩放功能后，用户只能滚动屏幕，网站看上去更像原生应用。**

 <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">

**下拉菜单**

            <li class="nav-item dropdown ml-lg-3">

                <a class="nav-link dropdown-toggle" href="#" id="infoDropdown" role="button" data-toggle="dropdown"

                   aria-haspopup="true" aria-expanded="false">

                    数据中心

                </a>

**按钮**

**class .btn**

<div th:unless="${#httpServletRequest.getSession().getAttribute('sessionUser')==null}" href="/userAdmin/admin" onclick="goToAdmin()" class="border bg-light text-dark btn">

1. **持久层框架：MyBatis-Plus**

**支持主键自动生成：支持多达 4 种主键策略（内含分布式唯一 ID 生成器 - Sequence），可自由配置，完美解决主键问题。雪花算法：SnowFlake 算法，是 Twitter 开源的分布式 id 生成算法。其核心思想就是：使用一个 64 bit 的 long 型的数字作为全局唯一 id。在分布式系统中的应用十分广泛，且ID 引入了时间戳，基本上保持自增的。这 64 个 bit 中，其中 1 个 bit 是不用的，然后用其中的 41 bit 作为毫秒数，用 10 bit 作为工作机器 id，12 bit 作为序列号。**

        fileHost.setId(String.valueOf(IdWorker.getId()));

**上传时设置fileHost、imageHost id为雪花id。**

1. **实现上传时生成webp格式图片缩略图功能：thumbnailImage**

**引入依赖：**

        <!--        webp图片处理依赖-->

        <dependency>

            <groupId>com.github.nintha</groupId>

            <artifactId>webp-imageio-core</artifactId>

            <version>0.1.3</version>

            <type>jar</type>

            <scope>system</scope>

            <systemPath>${project.basedir}/lib/webp-imageio-core-0.1.3.jar</systemPath>

        </dependency>

        <!-- Google图片开源压缩包-->

        <dependency>

            <groupId>net.coobird</groupId>

            <artifactId>thumbnailator</artifactId>

            <version>0.4.13</version>

        </dependency>

****

1. **实现注册自动发送邮件验证码功能**

**正则表达式判断邮箱是否合法：**

**^([a-z0-9A-Z]+[-|\_|\\.]?)+[a-z0-9A-Z]@([a-z0-9A-Z]+(-[a-z0-9A-Z]+)?\\.)+[a-zA-Z]{2,}$**

**邮箱账号开启 SMTP 服务**

**引入依赖：**

        <!-- 邮箱依赖-->

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-mail</artifactId>

        </dependency>

**产生随机验证码：**

        StringBuilder str = new StringBuilder();

        Random random = new Random();

        for (int i = 0; i < 6; i++) {

            str.append(random.nextInt(10));

        }

emailServiceCode = str.toString();

**使用mail.SimpleMailMessage和mail.javamail.JavaMailSenderImpl构造邮件并发送。**

1. **实现获取邮箱头像功能**

**引入依赖：**

        <dependency>

            <groupId>org.projectlombok</groupId>

            <artifactId>lombok</artifactId>

            <optional>true</optional>

        </dependency>

<!--        FastJSON-->

        <dependency>

            <groupId>com.alibaba</groupId>

            <artifactId>fastjson</artifactId>

            <version>1.2.75</version>

        </dependency>

**首先判断是否为qq邮箱：**

public static void main(String[] args) throws Exception {

        System.out.println(saveAvatar("1064220853@qq.com","Small"));

    }

    public static String saveAvatar(String email, String name){

        //判断是否为QQ邮箱

        String type=email.substring(email.indexOf("@"));

        String url=null;

        if(type.equals("@qq.com")){

            url=analyzingQQAvatar(email);

        }

        else {

            url=analyzingGravatarAvatar(email);

        }

        try {

            url=downloadAvatar(url,name);

        } catch (Exception e) {

            e.printStackTrace();

        }

        return url;

    }

**如果是qq邮箱：**

    public static String analyzingQQAvatar(String email){

        String Path;

        String QQNumber=email.substring(0, email.indexOf("@"));

        String url = "https://ptlogin2.qq.com/getface?imgtype=3&uin="+QQNumber+"@qq.com";

        StringBuilder sb = new StringBuilder();

        try {

            URL urlObject = new URL(url);

            URLConnection uc = urlObject.openConnection();

            //设置编码格式 解决中文乱码问题

            BufferedReader in = new BufferedReader(new InputStreamReader(uc.getInputStream(),"UTF-8"));

            String inputLine = null;

            while ( (inputLine = in.readLine()) != null) {

                sb.append(inputLine);

            }

            in.close();

        } catch (MalformedURLException e) {

            e.printStackTrace();

        } catch (IOException e) {

            e.printStackTrace();

        }

        String jsonStr= sb.toString();

        jsonStr = jsonStr.substring(jsonStr.indexOf("(") + 1, jsonStr.lastIndexOf(")"));

        JSONObject jsonObject = JSONObject.parseObject(jsonStr);

        Path=jsonObject.getString(QQNumber);

        if (Path==null){

            Path=jsonObject.getString("10000");

        }

        return Path;

    }

**其他邮箱：**

    public static String analyzingGravatarAvatar(String email){

        email= DigestUtils.md5DigestAsHex(email.getBytes());

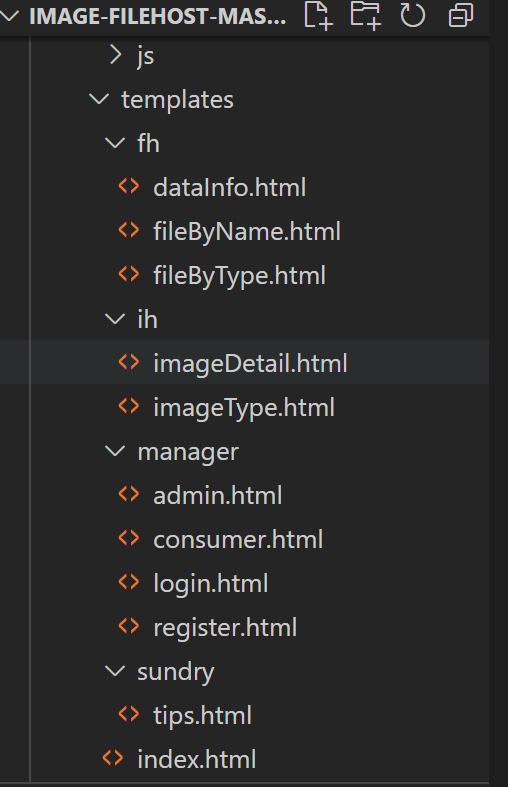
        String url="https://gravatar.loli.net/avatar/"+email;

        return url;

    }

# 五、实验数据记录和处理

Html文件：



最后页面：

