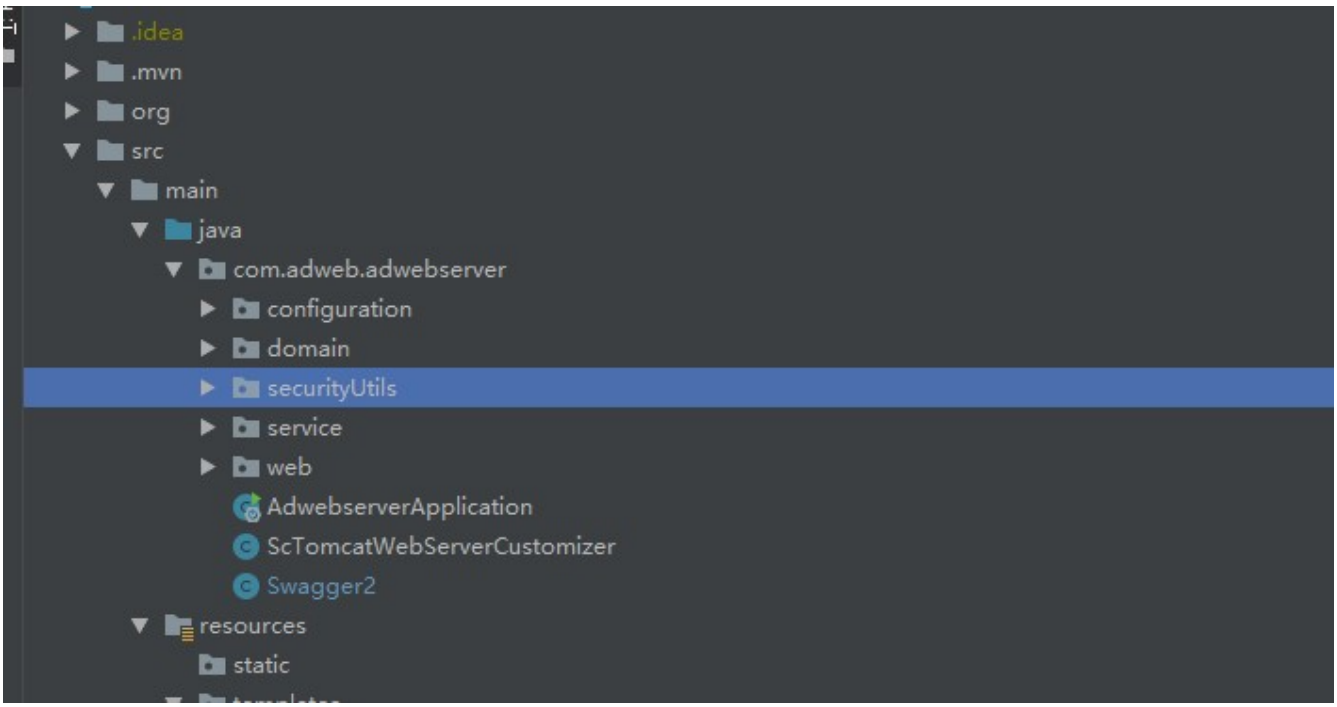
后端一些简单说明

[16302010016 雷玮杰 16302010020 凌锋]:

项目组织以及其中每个文件的说明



具体的目录太过冗长，不做介绍，configuration里涉及Jackson的配置，全局安全过滤器的设置，杜宇图片资源访问的resourceHandler设置以及对于tomcat的特殊字符[]{}校验的处理的RFCConfiguration部分，与之可以放到一起来看的还有下面的Swagger2（用来生成文档的，但是由于太懒了，最终接口文档还是没能好好优化，留下一个半成品），ScTomcatWebServerCustomizer和RFC起到了相同的作用，~可以弃用一个，因为懒~没改

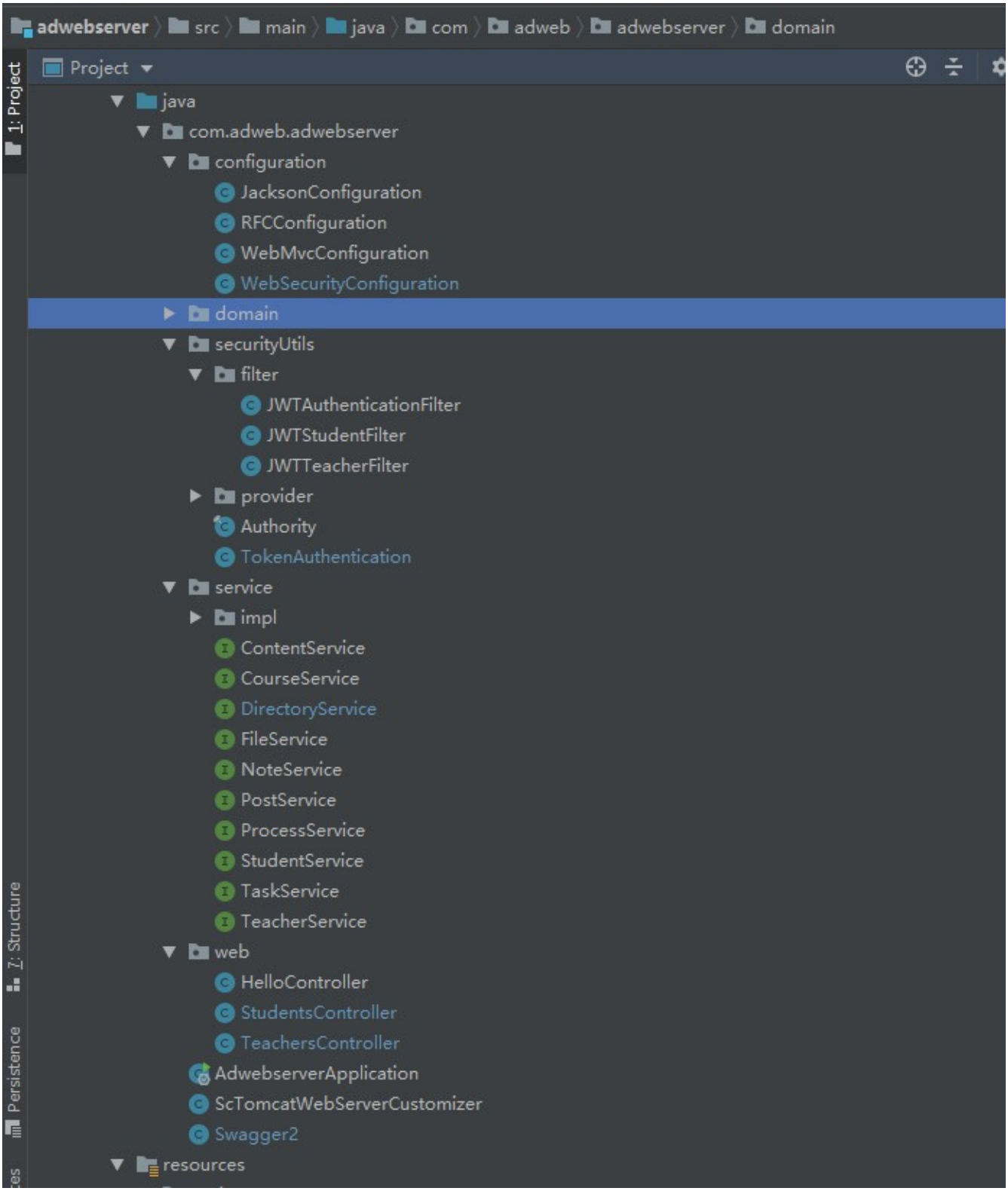
domain存的数据库的实体类，通过lombok简化代码量，repository节省需要写SQL的时间，面向对象编程；同时存储了一些生成复杂数据类型的元节点数据类型，作为一个使用参照，期望使用反射对于对象做处理分析，但是Jackson太好用了~~不习惯，还是fastJson最后划划水String转字符串很好。

sercurityUtils部分是安全实现相关的part，通过几个全局的过滤器，实现Spring Boot Security的UserNameAndPassword校验的token认证，认证详情下面细说吧，大概吧~顺带一提，过滤器是个好东西，正好合理的解决了跨域的问题

service层的东西，就是对domain层的封装，内容，课程，目录，笔记，文件，帖子，学生，任务，老师的服务，将下层数据访问层包装起来，使得上层页面显示的更加restful~~

web里是两个controller，学生和老师（用于测试的就不谈了），经过两种身份认证之后~~通过调用功能，实现各种~非常非常relax的功能（面向接口编程好！！！面向配置编程辣鸡！！！！

最后是一个相对无关紧要，但是很重要的两个部分，pom.xml和application.propetites，前者是maven的看家法宝，emmmm，很好用，谁用谁知道，后者基本上是spring boot 2.x系列的的产品需要简单配置的文件了，语法简明，但很强大。emmm，涉及到一些未处理的私密信息就不贴图了。但是总的文件目录大概还是要再来一张。



关键功能实现细节

可以说的也没啥东西，值得一提的是初期和组员一起讨论定框架的时候和各种实现方法是多么开心，现实往往不尽如人意，所幸学到很多东西了，吃到的一些苦，踩过的一些坑，终归是有些值得的，以及写在这一篇章之前的话，花钱买时间（买本不错的专业书看），比网上博客翻翻要省事的多，轻松的多。

大概，就写几个遇到的坑和想到的复杂逻辑的描述吧，顺序大概随我高兴

Tomcat的字符问题，编码问题，以及传输问题

Tomcat新版本出现了对于{}[]等字符的校验，认为不安全。虽然我想强行让它过去，毕竟mysql的json存储格式还是极为吸引人的~~最终加了这个但是由于字符的转义啥的，无法直接解析出来一个优美的json。想了想，毕竟总不能都是对象来对象去的美好，还是回归老本行，传字节流，String啥的也是美滋滋，做个校验，生成json挺舒服的，除此之外，也在application.properies稍作编码设置

多媒体文件上传的测试

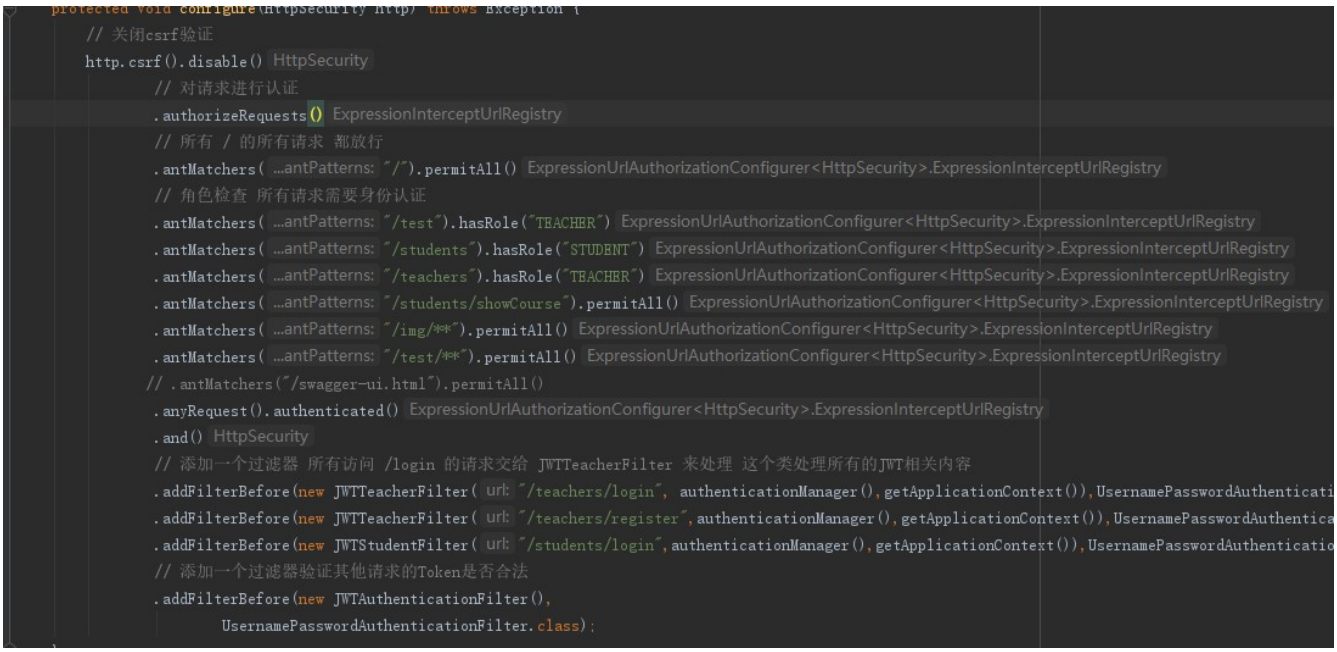
resourseHandler，然后创建文件的时候有时候会有需要权限，在本机测试的时候没有问题，在指定docker地址的时候出问题了，通过进入容器观察，最终决定根目录了~~~，妙啊，对于奇妙的会出现无法创建临时文件的错误，归结下来原因有几种，待上传的文件指向为空，对于multipartFile的传输大小缓存设置不对，还有一些乱七八糟的，，反正，真的是面向配置编程，大概也是不错的经历。

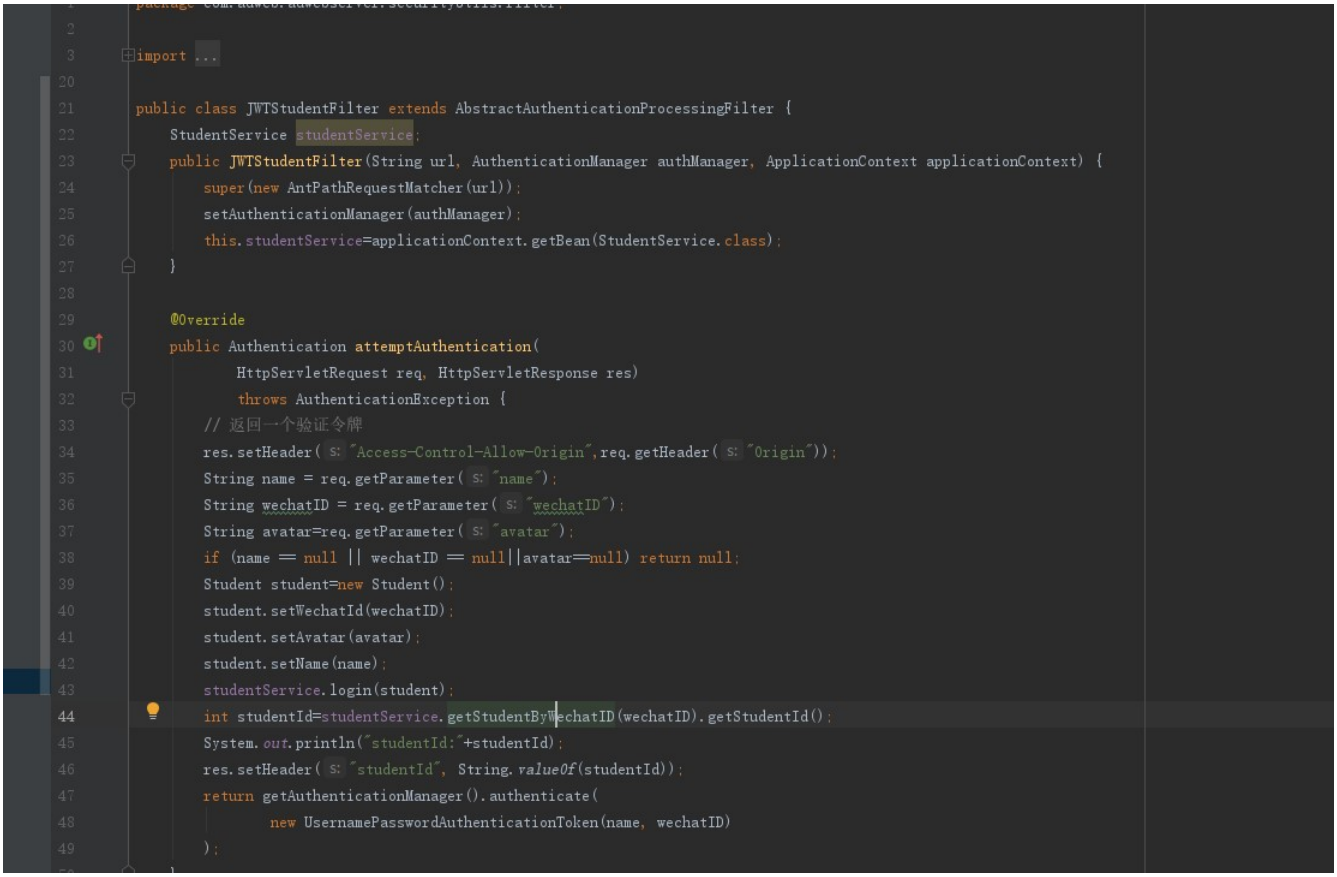
Spring Boot JPA和Hibernate的使用

这是一个非常非常cooooooool的写法，唯一的不好是，对于hibernate的使用者来说，依据实体或者map或者数据库来说，三种都是可以选择的，就偶尔会发生冲突。比较妙的是，通过hibernate的一个扩展库，可以将Mysql5.7之后的JSON类型，通过type注解转化为其他可操作复杂类型，intersting~~。很好玩反正，写起来也很省事。

安全认证有点东西~~Spring Boot Security +JWT Token

emmm，具体的机制，来自于官方文档来着~~Token的生成就不说了，，比较有意思的是过滤器，和给过滤器设置的实现





```java
import ...

public class JWTStudentFilter extends AbstractAuthenticationProcessingFilter {
    StudentService studentService;
    public JWTStudentFilter(String url, AuthenticationManager authManager, ApplicationContext applicationContext) {
        super(new AntPathRequestMatcher(url));
        setAuthenticationManager(authManager);
        this.studentService=applicationContext.getBean(StudentService.class);
    }

    @Override
    public Authentication attemptAuthentication(
            HttpServletRequest req, HttpServletResponse res)
            throws AuthenticationException {
        // 返回一个验证令牌
        res.setHeader( s: "Access-Control-Allow-Origin",req.getHeader( s: "Origin"));
        String name = req.getParameter( s: "name");
        String wechatID = req.getParameter( s: "wechatID");
        String avatar=req.getParameter( s: "avatar");
        if (name == null || wechatID == null||avatar==null) return null;
        Student student=new Student();
        student.setWechatId(wechatID);
        student.setAvatar(avatar);
        student.setName(name);
        studentService.login(student);
        int studentId=studentService.getStudentByWechatID(wechatID).getStudentId();
        System.out.println("studentId:"+studentId);
        res.setHeader( s: "studentId", String.valueOf(studentId));
        return getAuthenticationManager().authenticate(
                new UsernamePasswordAuthenticationToken(name, wechatID)
        );
    }
```

具体的感觉打字好麻烦，，，还是写代码爽，，，就就就酱紫吧

跨域问题

主要也就Access-Control-Header，还有个允许Origin之类的东西，主要出现的原因是因为发的东西给i有时候会带有自定义的header和一些复杂的数据结构（具体就是复杂~），导致它被当成一个复杂的包，需要服务器给它返回一个什么什么参数来允许，（报的错会告诉你），这个请求我们一般叫它option请求，是用来对复杂请求做预检工作的~~得劲

这个时候，一个全局的filter就美滋滋了，对response和request做了简单的解析和设置，doFilterChain,深藏功与名，啦啦啦

服务器部署配置的详细介绍

部署比较easy，本来亚马逊的rds，今天早上突然崩了，wwww，福气。然后临时租赁6个月阿里的RDS，配上学生阿里机，嗯，比腾讯云良心不少哈哈哈。就使用体验还行，嗯，跑题了。

简单来说，远程主机docker装好，对本机开放2375端口，然后开始干活~~，设置docker可以外部访问。

第二步写好Dockerfile,在intellij里配置好数据卷，运行参数啥的~~

第三步，被前端队友催着看日志，debug，补自己的锅，补别人的锅，嗯，还有补天

其他想说明的问题

大概框架会是一个比较不错的东西，但是想在框架里加上一点自己东西和配置，往往就是戴着镣铐起舞，大概会有一种凄壮美?

多踩踩开源的坑，多踩踩实践的坑，很多东西，融入骨子里，恭喜你，码农爱你噢————~~~~假的，学习框架，更要增强自身解决问题的能力才是真的，毕竟不管编程语言还是框架都只是工具而已