**WEB开发基础**

**轻博客小站 软件项目开发文档**

**目录**

1. **项目需求………………………………3**
2. **开发环境………………………………4**
3. **技术开发方案…………………………5**
4. **功能设计………………………………6**
5. **数据库设计……………………………8**
6. **系统设计 ……………………………11**

**七、系统测试 ……………………………20**

**一、项目需求**

题目:轻博客小站

1、基本需求

开发一个轻量级的个人博客小站，具有博客网站的基本功能，网站会员可以发布文章、管理文章、文章评论与关注等。

2、用户角色

游客(匿名用户或Internet用户)、会员和系统管理员

3、应用场景

(1)游客可以浏览会员公开发布的文章；

(2)游客可以注册为会员用户；

(3)会员用户可以管理其个人基本资料；

(4)会员用户在登录状态下，可以创建个人博客空间、发布文章(可设定公开或不公开，开放或禁止文字评论)、管理文章(创建和管理文章分类、标注索引关键字、编辑、修改或删除文章等)、文章评论(可对文章跟评、关注文章、删除自己文章的评论等)

(5)系统管理员可以进行用户管理(浏览或更新、删除用户资料)，网站文章管理(删除博客文章和评论等)

4、数据需求

(1)用户信息[用户名，密码，姓名，性别，年龄，出生年月，电话，家庭住址]

(2)文章信息[标题，内容，分类，索引关键字，评论，发布时间，最后修改时间]

(3)管理员信息[账号，密码]

**二、开发环境**

1、IDE为Intellij IDEA 2020.3.3，依赖管理及打包工具为maven，代码托管使用github，仓库地址为：

https://github.com/PikaCat-OuO/light-blog-station。

2、JDK版本16，语言级别为16，打包bytecode版本16，打包方式为jar方式。

**三、技术开发方案**

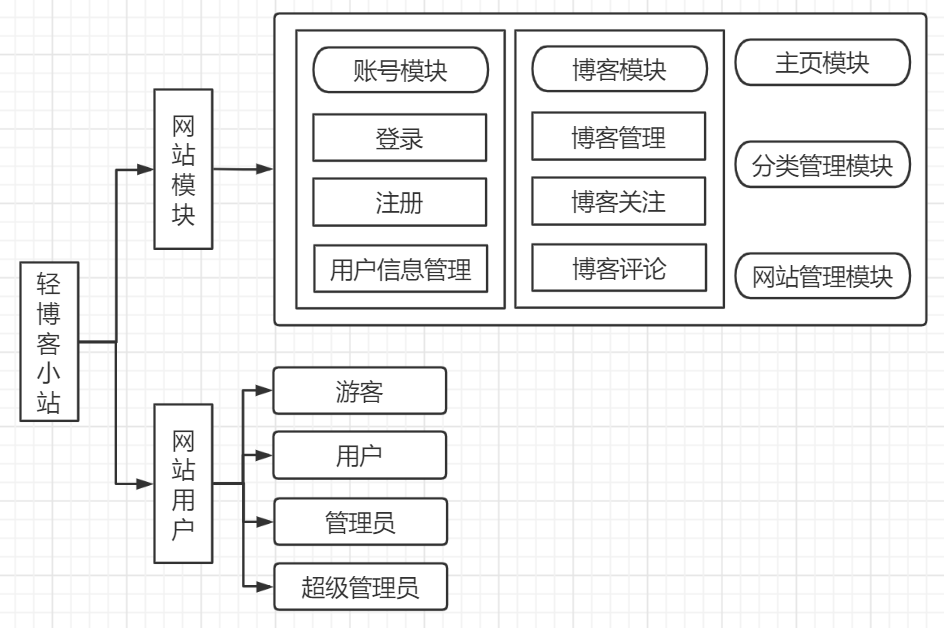
1、前端页面基于html、css、js、thymeleaf模板引擎(JSP的升级版，spring官方推荐的页面构建方式)、layui样式进行开发。

2、后端采用springboot、springsecurity、mybatis-plus进行开发。

3、数据库采用MySQL。(注意：在真实场景中，应该采用MongoDB作为文档类的NoSQL，此处为了简化工作量而直接采用MySQL来做文档的直接存储。)

**四、功能设计**

1、网站用户及模块组成:



2、对于前台用户页面，使用layui组件搭建基本的页面，用thymeleaf做渲染，用js做异步请求通讯获取后端数据并显示到前端。其中包括：

(1)主页：上面展示一些公开的博客的标题和用户的用户名，点击用户名可以进入用户信息页。如果是游客，则展示“请登录”的信息，页尾有版权信息。

(2)用户登录页：当用户点击登录按钮时就跳转到这个页面，用于给用户登录网站。

(3)用户注册页：当用户点击注册按钮时就跳转到这个页面，提供给用户注册成为会员。

(4)用户信息页：上面展示登录用户的个人信息以及其所发表过的公开博客(如果查看的是自己的个人信息页面则把不公开的博客也显示出来)。

(5)个人信息编辑页面：可以编辑个人的信息，如账号，密码，性别等。

(6)博客查看页面：可以查看别人发表的博客的具体内容，如果登陆了在此页可以关注别人的博客。

(7)发表博客页面：可以发表博客。

(8)我的博客页面:可以看到自己发布过的博客，可以对博客做管理，比如进入编辑页面，或删除博客。

(9)编辑我的博客页面:用于给会员发布文章或编辑已发布的文章，在这里可以设定文章公开或不公开，开放或禁止文字评论，标注索引关键字。

(10)我的关注页面：可以看到自己所有关注的博客，可以对关注的博客做管理，比如取消关注。

(11)分类贡献页面：用户可以贡献分类到分类库。

3、对于后台管理员页面，采用layui组件来搭建基本的后端管理页面，用thymeleaf做渲染。其中包括:

(1)后台登录页面:该页面与前台的用户登录页属于同一个页面，后台管理员登陆后在导航栏处将显示“网站管理”一个选项。

(2)后台管理页面:用于进行用户管理(浏览或更新、删除用户资料)，网站文章管理(删除博客文章和评论等)，以及用于超级管理员对普通管理员进行增删改查操作。

(3)管理员信息编辑页面：用于编辑管理员的信息

(4)编辑用户信息页面：用于编辑用户信息

(5)编辑分类：用于管理员编辑分类库中的分类

(6)新增管理员页面：用于超级管理员添加新的管理员

(7)管理员信息编辑页面：用于超级管理员编辑管理员的信息

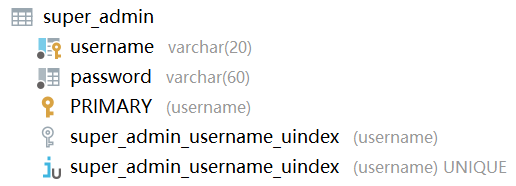
(8)超级管理员信息编辑页面：用于编辑超级管理员的信息

4、对于后端，采用springboot搭建mvc(Entity，Mapper，Service，Controller)，使用mybatis-plus做数据库访问，用springsecurity做用户登录和访问控制。

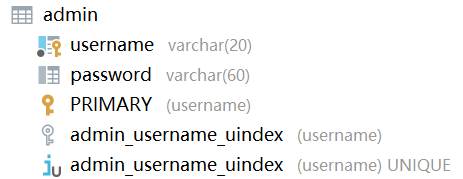
**五、数据库设计**

数据库中，表格的设计如下(图中包含表的索引，具体的数据库创建sql在源码的sql文件夹下):

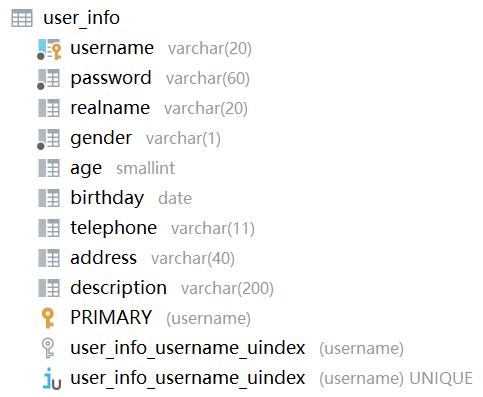
1. 超级管理员信息[超级管理员用户名(主)，超级管理员密码]



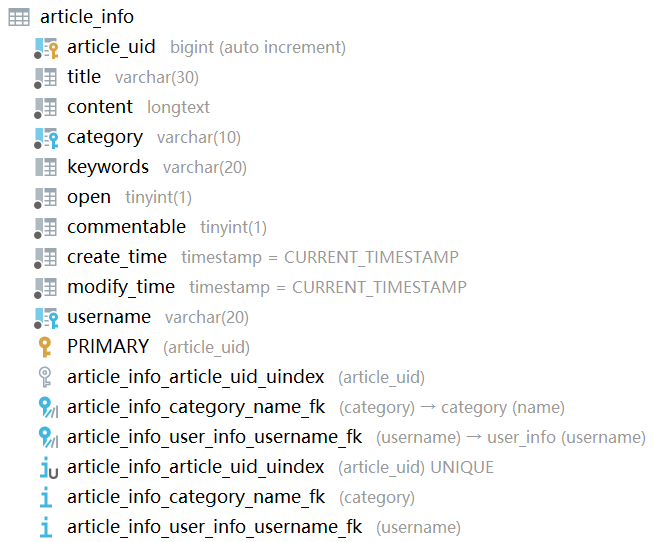
1. 管理员信息[管理员用户名(主)，管理员密码]



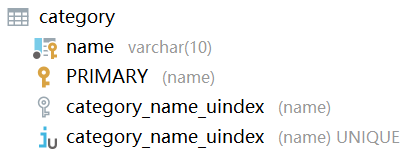
1. 用户信息[用户名(主)，密码，姓名，性别，年龄，出生年月日，电话，家庭住址，个人简介]



1. 文章信息[文章UID(主)，标题，内容，分类(外)，索引关键字，对外公开，开放评论，创建时间，修改时间，所属用户名(外)]



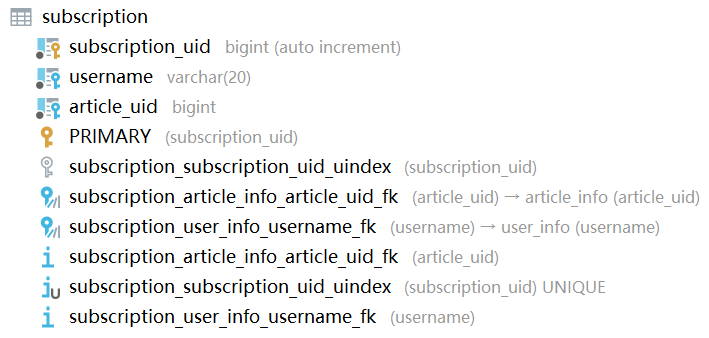
1. 分类信息[类名(主)]



1. 评论信息[评论信息UID(主)，文章UID(外)，评论内容，评论所属用户名(外)，评论时间]

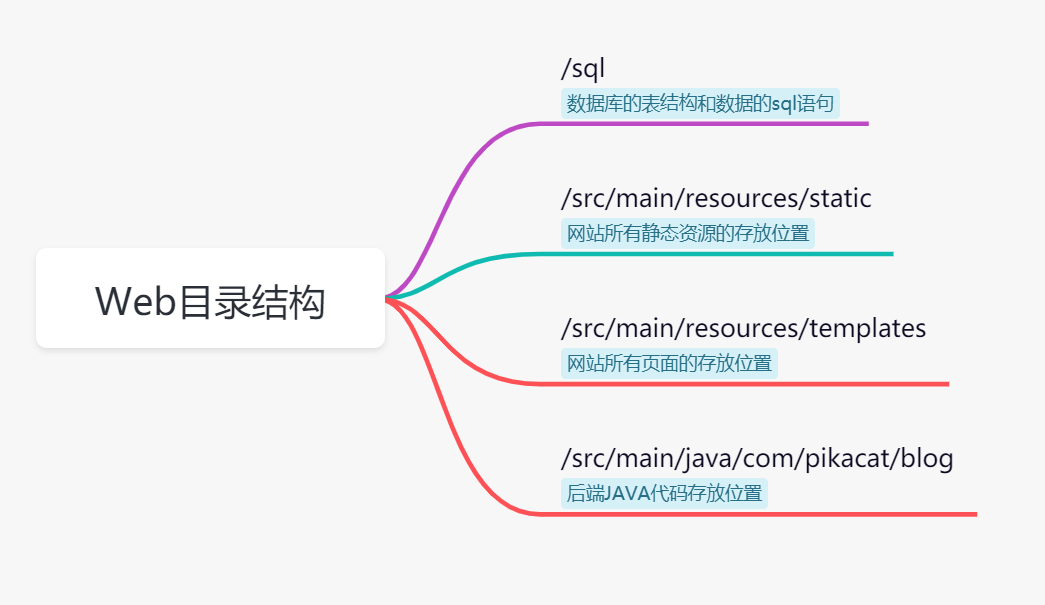


1. 关注信息[关注信息UID(主)，用户UID(外)，关注的文章UID]



**六、系统设计**

1、网站的目录结构 :



2、网站用户组织形式：

(1)游客：可以浏览主页，可以贡献分类到分类库，可以查看别人的个人主页与公开的博客。

(2)用户：除了拥有游客的全部权限外，还可以对自己的个人信息进行管理，可以发表博客，对发表的博客进行管理(包括评论部分的管理)，可以关注别人的博客，可以对别人的博客进行评论。

(3)管理员：拥有对数据库内现有的所有用户，所有博客，所有评论，以及所有的分类信息进行管理的权力。拥有修改自己的用户名和密码的权利。

(4)超级管理员：除了拥有管理员的全部权限外，还可以对数据库内所有的管理员账户进行管理。拥有修改自己的用户名和密码的权利。

3、各个应用模块的设计：

(1)账号模块：

登录页面为login.html，数据提交到后台验证的接口为/login

注册页面为register.html，数据提交接口为/register，用户名重名验证的接口为/register/{username}，提交上去之后后台还会对用户名做二次检查。

用户信息管理页面为edit-user.html，修改提交的接口为/user-page/{username}/edit-user。页面中对用户的每一个信息都做了各种各样的限制，确保提交到后台的数据符合规范。用户提交后，后台还会对用户提交的数据做二次验证。

用户信息的显示页面为user-page.html，这个页面中的用户数据通过thymeleaf由服务器端直接赋值获得，而页面中该用户博客的部分则通过ajax请求获取，请求的接口为/user-page/get-user-open-articles

(2)博客模块：

发表博客的页面是new-article.html，提交的接口为/my-articles/new-article，后端会对数据做检验。

编辑已发表的博客的页面是edit-article.html，提交的接口为/my-articles/edit-article/{articleUid}，后端会对数据做检验。

查看博客的页面是article.html，数据直接由后端根据articleUid经过thymeleaf渲染到页面上返回给用户。在此页面可以对博客进行关注与评论，对于评论的删除也是在此页面进行。添加评论的接口为/article/{articleUid}/add-comment，删除评论的接口为/article/{articleUid}/remove-comment/{commentUid}，获取评论的接口为/get-comments/{page}，page为页数，用于分页获取。

管理博客关注的页面是my-subscriptions.html，使用ajax技术获取后端接口/my-subscriptions/get-subscriptions的结果显示到前端的表格上，后端根据前端的username返回关注的结果。

我的博客管理页面是my-articles.html，在此页面可以对已发表的博客进行增删改查操作。

(3)主页模块：

轻博客小站的主页是index.html，提交的接口为/get-hot-open-articles，获取当前的公开数据返回到前端的表格进行显示。

(4)分类管理模块:

分类的贡献页面是contribute-category.html，提交的接口为/contribute-category，对分类进行重名检查的接口是/contribute-category/{category}，提交到后台后，后台还会再次检查分类是否已经重复了。

(5)网站管理模块:

网站管理模块的页面是admin.html，下设四个管理子模块。

网站博客管理从后台接口/admin/get-all-articles返回数据，显示到前端的表格上，之后就可以对其执行删改查操作。删除的接口为/admin/delete-article/{articleUid}，修改的接口为/my-articles/edit-article/{articleUid}，对应的页面为edit-article.html，查看的接口为/article/{articleUid}/。

网站用户管理从后台接口/admin/get-all-users中获得数据显示到前端的表格上，并且可以对其进行增删改查操作。删除用户的接口为/admin/delete-user/{username}，修改用户信息的接口为 /user-page/{username}/edit-user，对应的页面为edit-user.html，可以对用户的主页进行查看，接口为/user-page/{username}/，页面为user-page.html。

网站分类管理从后台接口/admin/get-all-category中获得数据，并显示到前端的表格上，可以对其进行增删改查的操作。删除分类的接口为/admin/delete-category/{category}，修改分类的接口为/admin/edit-category/{category}，页面为edit-category.html。

超级管理员管理管理员模块从后台接口/super-admin/get-all-admins中获得管理员的信息，可以对其进行增删改查的操作。修改信息的接口为/super-admin/edit-admin/{username}，页面为super-edit-admin.html，删除管理员的接口为/super-admin/delete-admin/{username}，此外，还可以进入管理员添加页面super-add-admin.html进行添加管理员的操作。

注意：由于项目要求中没有要求实现搜索功能，所以在此不实现该功能，如需实现，需要使用ElasticSearch全文检索引擎与IK分词器。

4、部分功能的实现

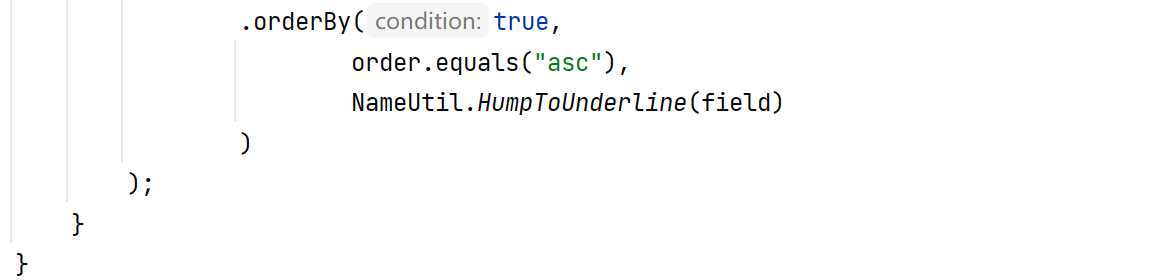
(1)数据获取

带有分页的数据获取都是形如下面的形式取获取:

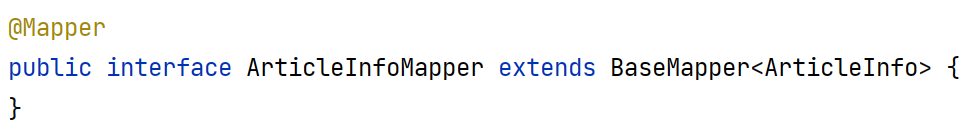


该Controller中的方法将前台传入的当前页page，每页显示的条数limit，排序字段field和升序降序order传入service层，结果返回后封装并返回给前台。





Service层中的该方法根据传入的字段的不同情况来调用Mapper层查询数据库并将结果返回给Controller层。



对于Mapper层，因为有Mybatis-plus，所以不需要写任何方法，只需要继承BaseMapper<>即可获得所有数据库查询方法，开箱即用。如果是采用Vue，则可以使用BaseMapper<>，IService<>，直接将两个层都封装起来，不用手动写代码，开箱即用。更直接可以使用ActiveRecord模式，使实体类继承Model<>即可直接与数据库交互，无需Mapper和Service层。由于本项目不是采用Vue作为前端框架，所以采用了传统的BaseMapper<>方式。

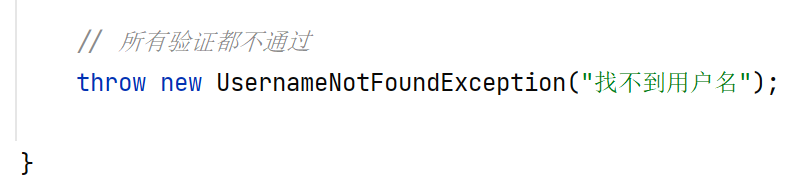
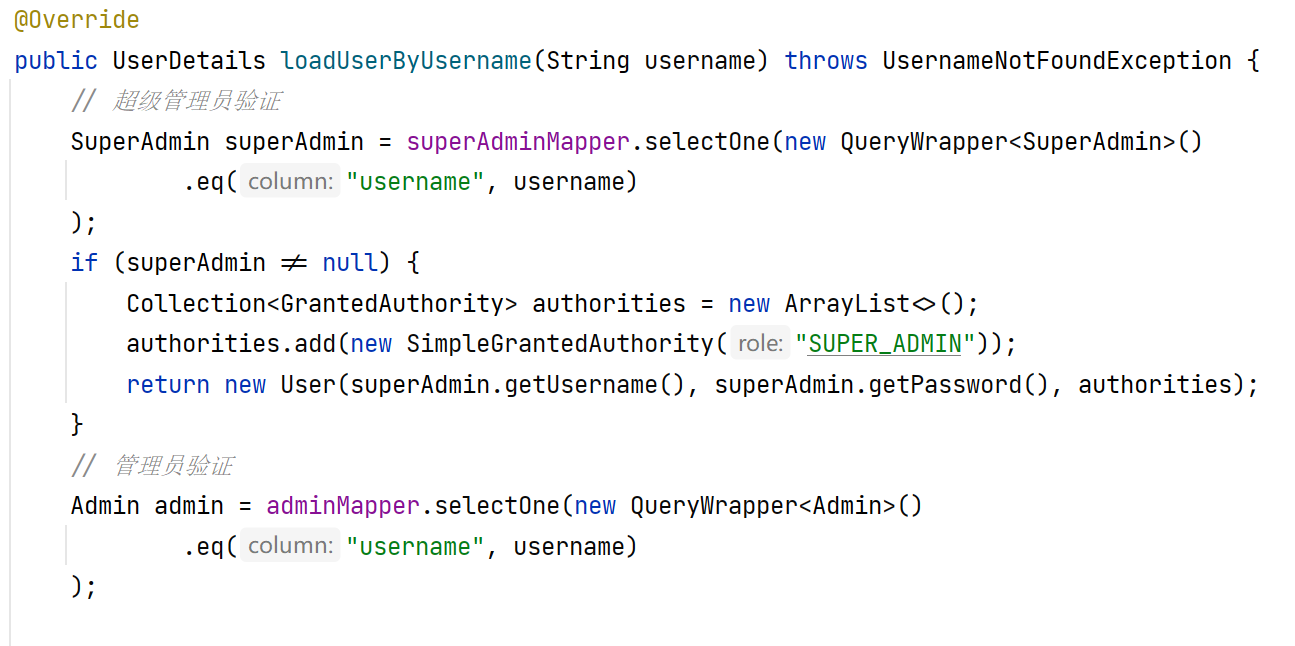
至此，数据获取的功能介绍完毕。

(2)登录与安全

采用springsecurity技术，首先需要写一个类继承UserDetailsService来验证账号密码是否正确，将三个与账号密码有关的Mapper注入到该类中，该类从数据库中读取数据并检验。

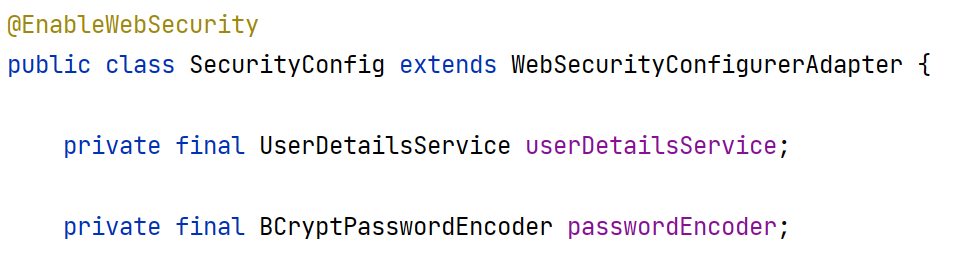


重写其中的loadUserByUsername方法

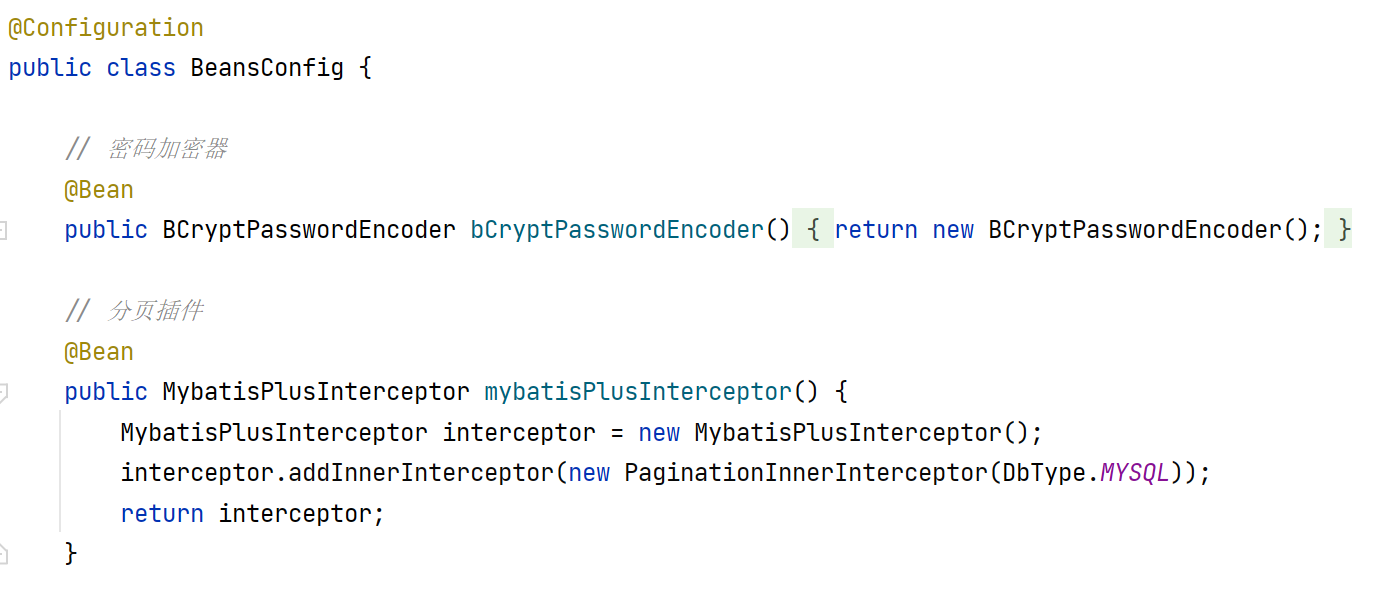


这样就可以实现读取数据库并把密码返回给springsecurity配置类做校验。

接着写一个配置类继承WebSecurityConfigurerAdapter，并将刚才写的UserDetailsService注入其中，同时还需要注入一个密码加密类，该类与其余两个Beans共同配置。



配置密码加密器以及其他Beans的类:



接着配置接口的过滤规则，规定接口的访问权限。



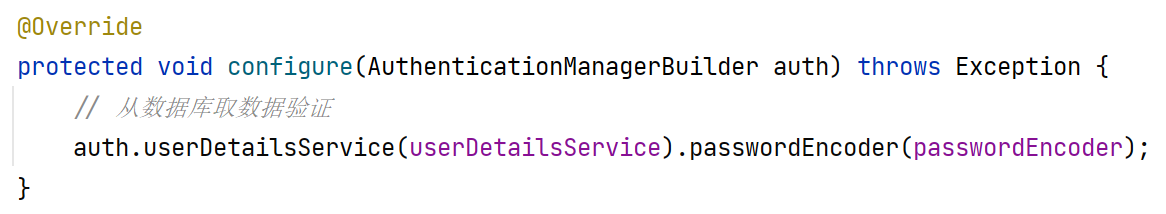
接着配置登录页面以及注销页面



接着配置iframe和csrf，由于本站需要用get方式注销登录，并且开启csrf后每次请求都需要带上csrf\_token，较为麻烦，故此处将其关闭。真实场景中应该将其开启，做好安全防护。

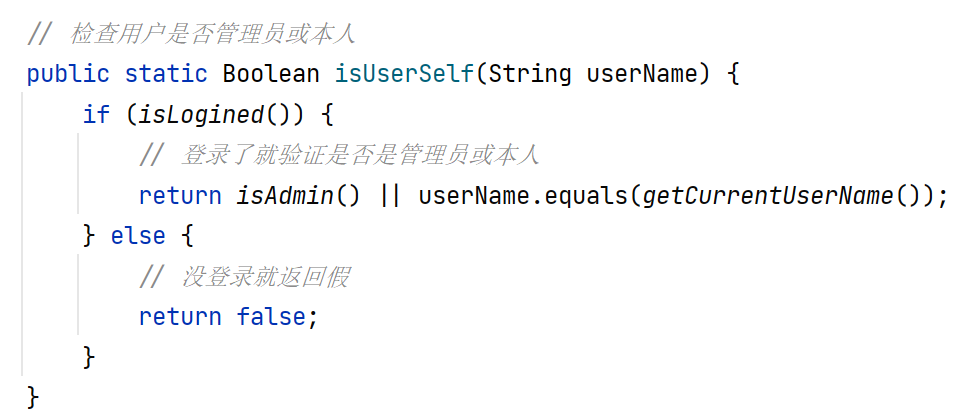


重写configure方式实现密码验证。



至此，所有与springsecurity相关的配置都已经完成，接口也得到了相应的保护，必须拥有相应的权限才能调用相关的接口。有效的阻止了接口的恶意调用。

(3)更安全

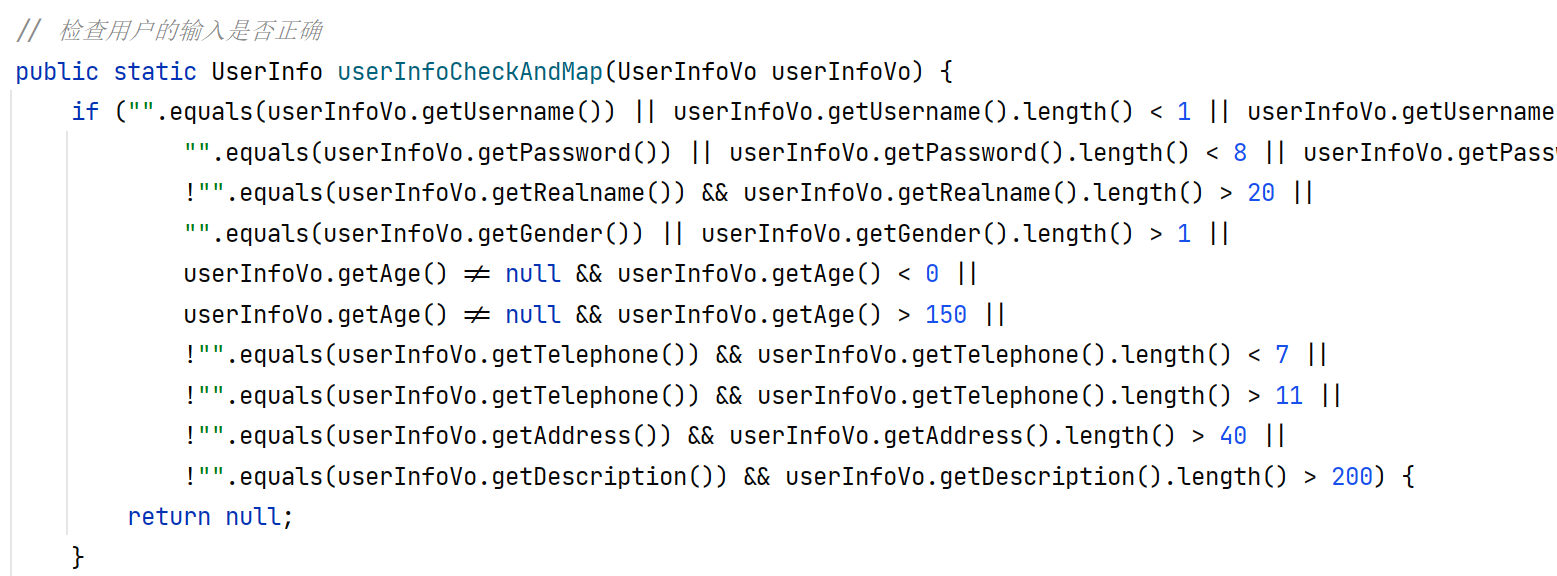
虽然接口已经被保护了，但是还是存在登录了账户后获得USER权限后随意调用接口修改他人文章内容的情况，所以在Service层还需要对用户的身份做进一步的认证。再次之前需要写一个SecurityUtil类来检验用户的身份，这里只是展示其中的一个功能，想要了解全部的功能，读者可以自行阅读源码。

基本在所有应该保证安全的地方都调用了该方法来验证用户身份，此处举一个获取用户文章的例子:



(4)更健壮

用户提交过来的数据不一定经过前台，有可能是直接用postman等工具恶意提交的，所以此处写了另一个工具类做参数合法性验证，这里只是选取其中的一个方法做展示，该类在程序中也有多处运用场景，读者有兴趣可以自行查阅源码。



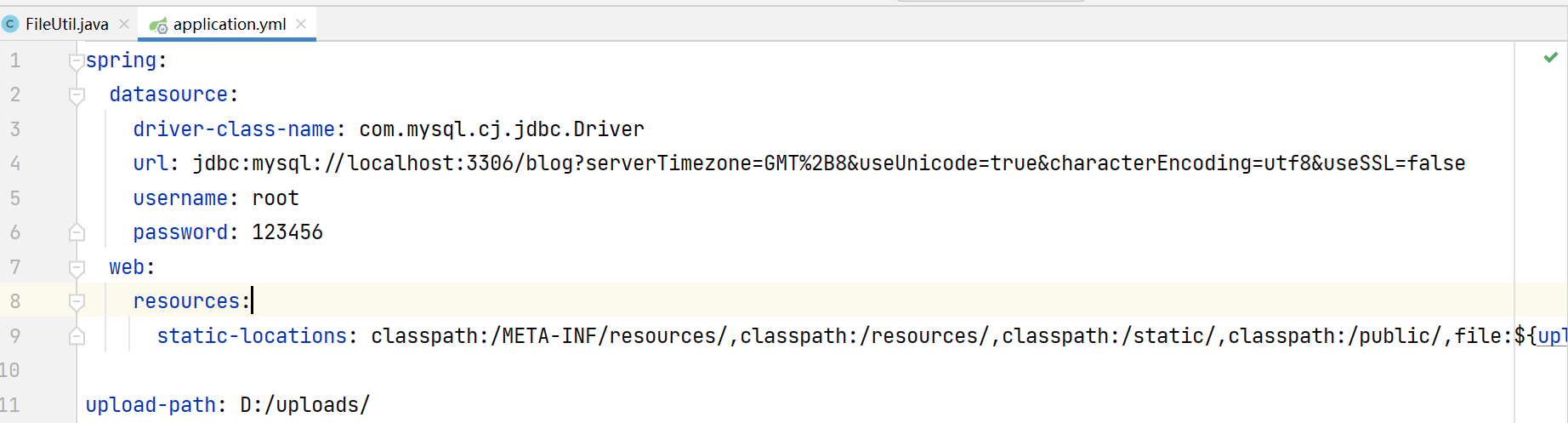
(5)图片上传



该工具类实现了图片上传的保存功能，主要提供给博客图片的保存使用，saveToDisk方法接受一个MultiPart类型的file并对其做检查，如果满足条件将会把该文件存放到磁盘上，扩展名的规则由extensions中指定。

saveImage方法调用saveToDisk方法并指定其接受的扩展名为JPG、JPEG、PNG、GIF的图片类型，保证安全性，防止用户上传Webshell。

而setUploadPath方法则指定了用户文件上传保存的位置，这个位置由容器注入，在配置文件upload-path中指定，如下图:



调用到这个类的接口为/my-articles/upload：



上传成功与否的信息会返回给前端富文本编辑器并在前端做相应的操作。

**七、系统测试与使用说明**

关于本部分在程序配套的演示视频中有具体的展示，读者直接查看视频中的展示即可，在此处不再独自展示网站运行截图。

如果读者想独自尝试该系统，可以用idea打开该项目直接运行，也可以用maven编译该项目为jar文件后使用java -jar xxxx.jar命令运行程序，推荐使用后面的方式运行，比较不那么占用内存。

如果读者没有maven，本程序已经打包成了blog.jar文件，读者亦可直接启动jar。

运行项目前需要先运行sql文件夹内的sql保证数据库内有数据，然后需要修改application.yml中的数据库链接信息为自己的数据库。

默认有两个用户user1和user2，一个管理员admin和一个超级管理员superadmin，密码均为12345678