博客系统设计

1. API部分

主要工具：使用了Vue+springboot+mysql+redis

数据库设计：

普通用户表：通过注册即可成为里面的用户，拥有写博客的权限。

文章信息表：存放了文章的各类信息（评论数、创建日期之类……）。

文章内容表：存放了对应文章的内容。

文章类别表：存放了文章的几个类别。

文章标志表：存放了文章的几个标志。

文章对应类别表：存放了每篇文章对应的类别信息。

文章对应标志表：存放了每篇文章对应的标志信息。

文章评论表：存放了文章的评论。

Dao层设计：使用了mybatis-plus技术

Dao层的工作流程：

引入mybatis-plus依赖，并在config下添加mybatis-plus的配置文件。

在dao层下建立pojo类（与数据库的每张表一一对应）。

在dao层下建立mapper文件夹，为每张表编写对应mapper文件；mybatis-plus提供的basemapper类已经为我们提供了足够多的sql查询功能。如果有新的需求，可以在resources文件夹下添加与mapper对应的xml映射文件。

Controller层：接收前端的数据，并调用相应的service得到相应的结果并返回给前端。

一些常用注解：

@Requestbody：把前端请求body体里面的信息映射成后端指定的对象类。

VO:里面存放着一些与前端交互的类（pojo类为数据库中的原生类，直接与前端交互难免有些需求不一致的地方）

几个主要功能：

首页：

全部文章查询功能：

常规的流程，从controller到service到mapper，再从mapper到service到controller（返回给前端）；在service层，需要将pojo转化为vo,这个转化有些麻烦，具体看项目的代码。

其余的功能比较简单，不再赘述。

文章分类页/标签分类页/归档页：

根据不同的查询条件对文章做出分类并返回，比较简单，不再赘述，具体细节看代码。

普通用户注册：

前端将注册的用户信息传到后端，service将注册信息存放到数据库；然后借助jwt生成token；将token与被序列化后的用户信息以键值对的形式存入Redis（后面就可以凭借token从Redis中获取用户信息），将token返回给前端；加密盐：在用户注册的密码后加了一截字符串，保证了用户数据的安全；通俗地说，JWT的本质就是一个字符串，它是将用户信息保存到一个Json字符串中，然后进行编码后得到一个JWT token，并且这个JWT token带有签名信息，接收后可以校验是否被篡改，所以可以用于在各方之间安全地将信息作为Json对象传输。（先了解到这里，后面需要更深入了解时再去拿！）

普通用户登录：

前端用户登录信息被传递到对应service，若信息在数据库中真实存在，拿出该用户的信息，使用jwt生成token，并将token与用户信息存入Redis，然后将token返回给前端（前端将token放入store中保存起来）。

前端携带token到后台拿用户信息：首先借助jwt工具校验token是否正确；若正确，从Redis中拿到用户数据并返回给前端。

登录拦截器（有些功能需要登录后才能使用）：

WebMvcConfigurer配置类其实是Spring内部的一种配置方式，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。

拦截实现：

首先在配置文件夹中添加一个配置类（继承WebMvcConfigurer接口）并实现addInterceptors接口;

在handler文件夹下配置loginIntercepter类（继承HandlerInterceptor接口）：

preHandle（在业务处理器处理请求之前被调用。预处理，可以进行编码、安全控制、权限校验等处理）：配置拦截路径下的所有controller请求在执行之前都必须先过这一关：

被拦截的请求先把token拿出来校验；如果token正确，放行（return true1）并把取出的用户数据放入UserThreadLocal作为当前登录用户；ThreadLocal主要用来为当前线程存储数据，这个数据只有当前线程可以访问。可以将ThreadLocal看成是一个map，而当前的线程就是map中的key。

我的文章：在原有文章的基础上加了一个当前用户名参数，仅此而已！

写文章：

图片上传（我选择上传到本地）：

首先通过addResourceHandlers将本地文件夹的绝对路径映射成指定的路径以便访问。

文件从前端传过来后，后端的service将其存放到指定文件夹，并将图片的访问路径返回给前端。（等以后查看文章时，图片的路径就会转化为图片，从而实现图片上传）。

文章浏览数量（每次浏览后，文章的浏览量加1）：

这里有一个重点，线程池操作：文章在被浏览为主进程，浏览量加1为次要的线程（这个操作在幕后做就行了，没必要让用户等着）

实现：首先在config中配置线程池；然后在service中配置线程服务；然后哪里需要，就直接调用它即可。（具体见代码）

评论：需注意两个点

评论分为子评论与父评论；

在创建新的评论时，用到了UserThreadLocal获取当前登录用户。

大致的要点就在这里，其他的细节看代码！