还是经典的三层架构，controller、service、mapper/DAO。

entity、dto是实体类。

Expection是异常

Common里面有写了个拦截的注释和好多看不懂的东西。

Utils.generator是…好像是为了不写sql语句弄的。

现在要实现对接的有…文章的查询。用户登录注册。文章评论发布。

看看，然后…写点。如果有现成的就直接axios的post一下完事。还有…这个请求是不是也有很多学问。

有个P！

吐槽一下，找来的这个后台可真是…不知道怎么说，几乎一行自己的代码都没写，全是在导包导别人的函数用…这样…这样…看起来好爽但我爽不了啊啊啊啊

还有，所有东西几乎全写在了controller里面，service，还有给sql语句准备的那堆xml文件就是个摆设。

数据放在了后端静态文件里。。。也好这样不用再多整服务器了。

Element ui里还有upload的标签。。这也太方便了，就是要跨一下域。这样弄的话，就算上传到服务器也简单好多吧。

可能，可以作为一个小组件。

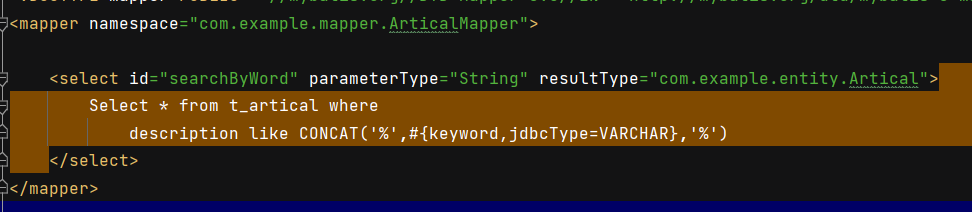
嗯，目前没做的还有查询了。

done！现在有查询了！

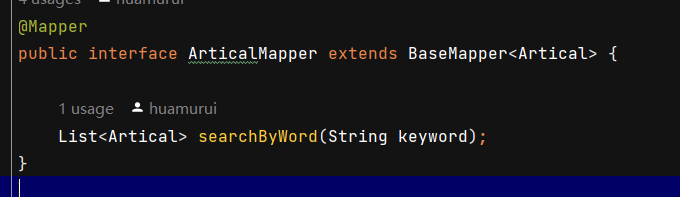
springboot mybtias

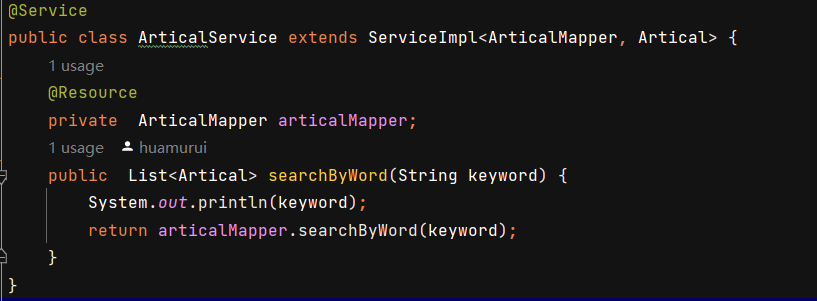
我给它通了一遍。

从数据库一边开始说，首先是在xml文件里写查询

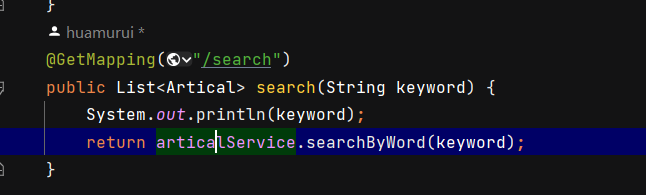


这里的select id就是DAO或者说mapper里的方法名。





然后在service中重写DAO/mapper接口中的方法，此处可以添加业务逻辑。



然后在controller中写何时调用service

反过来看。

Controller是最先接收请求的，接收之后对接受到的请求与写在controller里的请求去匹配，匹配到了就把Service叫起来干活，匹配不到就告诉前台404。

Service干活可能就要访问数据库，访问只能通过Dao提供的接口才行，别问，可能是为了项目方便管理，为了程序健壮性，低耦合高内聚啥的，我也不知道。

Dao这边就是直接对数据库操作了。

src里面还有一堆文件夹，bo、vo、po、dto、pojo、model、entity…但我看里面的东西都和java bean长得差不多？

写的java bean是和数据库里的表对应、写了一堆属性和它们的set get。最简单的是一个bean对一张表，作用是方便复制粘贴、一下子拿出一个东西的全部东西…或者说一条数据的全部数据项。

我看到里面的dto似乎是那种只拿一部分数据项不拿全部、可以减少不必要的数据传输…但那一堆东西的区别我现在是分不清。

@ 注释这个东西我是第一次见，作用可能是告诉框架哪里是干什么的，写在类、实体、方法上的好像都有；这个作用和vue里直接规定的那一堆data、methods、name、component有点像…但我就是看java不爽。

实例化，指类（或者接口）的实现，类是抽象，实例化具体之后才能起作用…一个类在另一个类里实例化（具体而言，就是entity在DAO中实例化、DAO在Service中实例化、Service在Controller中实例化。），这可能是java最像vue组件注册的那种写法的地方，然而这里的两个类并在java好像并不被看作parent-child…

可能也不是对java有意见，是那天交作业弄得太急给我整破防了…我在想以后的工作生活都是这个吊样吗…nmmd我受不了…