

统计网站的独立访客与日活跃用户（本项目一天访问过一次就算），这两个感觉像是同一个意思，但独立访客只要是访问过本网站就算，包括游客访问（所以不能通过userId来统计数据需要通过ip地址），而日活跃用户只统计注册登录的用户（用userId来统计），所以还是有点区别，而且你的日活跃用户改下逻辑（访问过多次才算活跃用户，或特定页面），就和uv有很大的区别了，统计uv时只用统计大概就好，不用过于精准，所以采用HyperLogLog（当网站访问量大时，可以节省空间），而日活跃用户需要精准一些（感觉是便于以后奖励客户，活跃度达到一定程度就有奖励，不精准的话可能把某些活跃用户落下，造成客户缺失），用Bitmap（userId为Bitmap的索引位，例如userId为101，则在第101为设置为true）

因为这些都是通过redis来操作的，所以不用写dao层，直接去配置对应的key：

单日的uv为uv:date 区间的uv为uv:dateStart:dateEnd

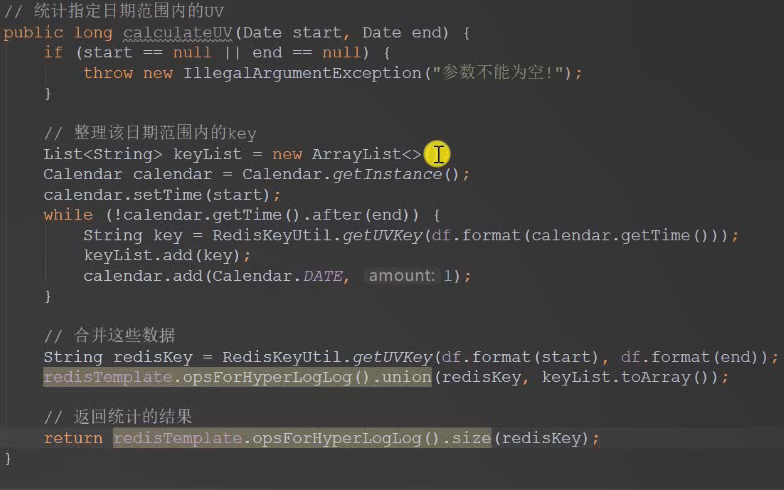
单日的dau为uv:date 区间uv:dateStart:dateEnd

Service层：

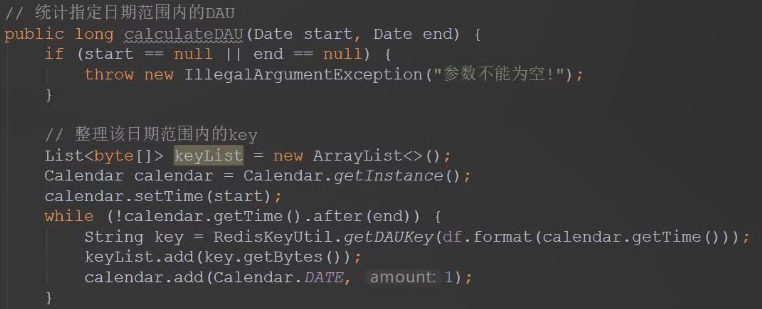
将指定的ip记入uv，首先获取到当前日期对应的key，之后将<key，ip）>键值对存入redis中，数据类型为HyperLogLog，注意现在的key是将date转换为字符串获取到的

在查询uv时，不用编写单独查询一天uv的方法，因为网站查询数据时如下图所示只用范围查询，想要查询一天可以在起始日期和终止日期写成同一天，而且也没有必要再编写一个查询一天的方法，太麻烦（在前端页面还要在设计一个查询一天的表单，控制层还要再加一个查询一天的请求处理，服务层还需要编写查询一天的方法）

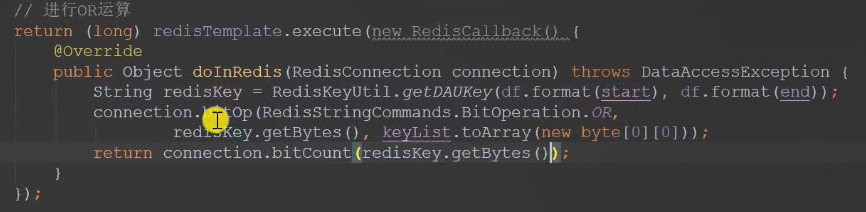
统计指定日期范围内的nv：由于在合并数据时，需要传入合并数据后数据对应的key，以及需要合并数据的key，而合并数据的key是通过日期存的，我们像上节课一样，挨个传入过于麻烦（而且还要通过给起始日期加1来获得之后的日期，已经有循环了），所以肯定是循环遍历加入到集合中（我想的是加入到数组，长度为最后一天-起始天+1，但日期类型的加减没想出来，因为直接转为int的话数据太大了，浪费性能，集合不用涉及到长度问题），视频中通过calender.gerTime（）.after（end）（如果calender日期大于end则为true）来控制循环，注意在合并数据时，不能传入集合类型的key团体，方法规定必须是数组，所以需要转换一下，最后返回redisKey的size，HyperLogLog的size方法返回的是long类型



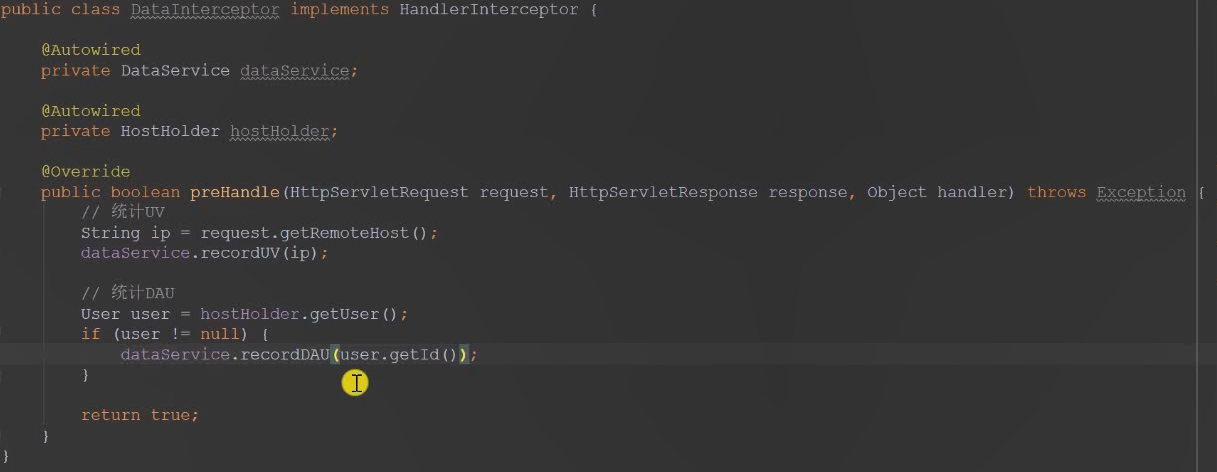
将指定用户记入Dau和uv逻辑一样

统计范围内的dau：前半部分与 uv一样，也是将key存入集合中，便于后面的操作，因为统计范围内的dau的话需要进行or运算，只要这个范围内有一天达到了活跃标准，就算是活跃用户，在进行or运算时也需要将每一个需要运算的key传入进去

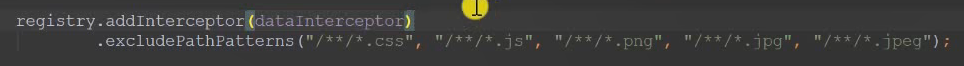
或运算后返回的是long类型的数据

  
因为记录uv和dau在每次请求都需要访问（HyperLogLog有去重，Bitmap的对应位设置为true，重复设置也没关系），所以要将这两个方法加入到拦截器中

编写uv和dau对应的拦截器：

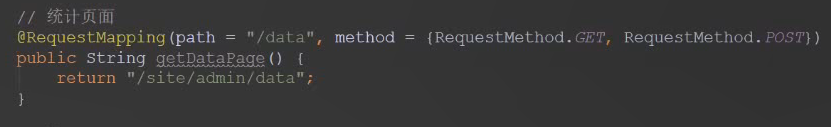


配置拦截器：设置静态资源不用拦截

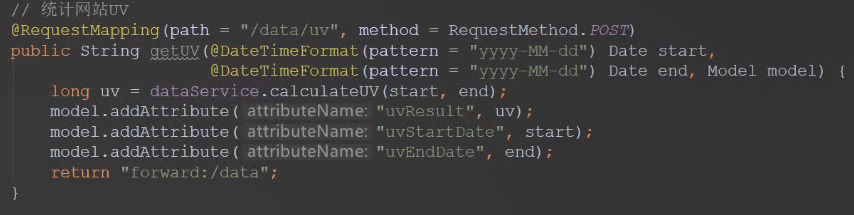


Controller层：

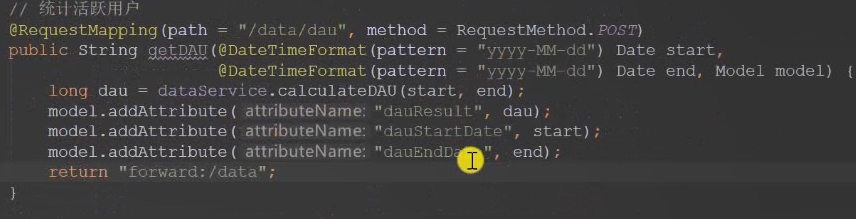
统计页面：由于只能管理员来访问，所以不用在页面另外设置统计的按钮，直接让管理员通过搜索url来实现



统计网站uv：当管理员在网面选择起始日期与截至日期后，将请求传给服务器（post），传过来的日期其实是字符串，还需要转换成为日期类型，通过@DateTimeFormat注解可以实现，最后返回可以返回模板页面，也可以将请求转发给统计页面的请求



统计活跃用户:



因为此功能只能管理员来访问，所以要在security的配置中将这些请求路径设置为管理员权限

感觉这些思路要先主要绕着dao、service、controller来思考，如果需要其他的东西再去调用其他配件，不能刚开始就思考来用到什么配件，想不全也费尽，不如等用到了再去调用