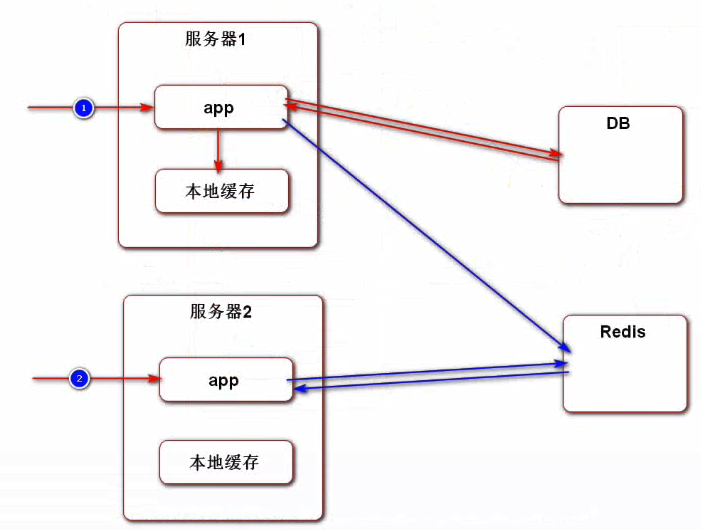


在存储登录凭证时，不能选用本地缓存，因为假如你在本地缓存中存储了登录凭证，第一次登陆账号时，可能将登录凭证存给了服务器1，但可能下一次访问请求，就是去服务器2访问的，服务器2的本地缓存里是没有你这个登录凭证的（可以想想以前课程中将登录凭证存给redis，登录凭证只在redis中存储，不存储在数据库中），而且他也不会去访问数据库，那你这次的请求就直接被拦截器给拦截了，重定向到了登录页面

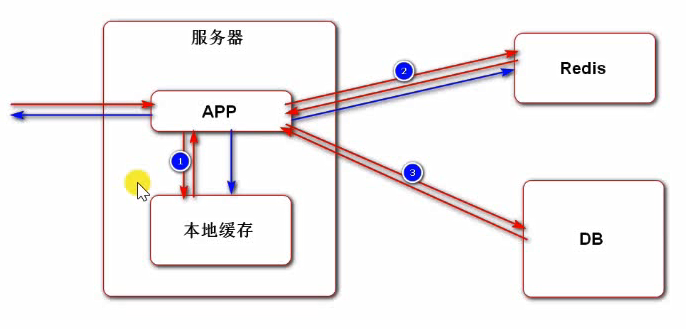
本地缓存比分布式缓存的好处是，直接部署在本机上，速度快，因为分布式缓存，是存储在其他计算机上的，所以需要额外的网络开销，比本地缓存慢，但分布式缓存可以解决咱们上一条说的问题

多台服务器访问数据（redis包括服务器1存数据，服务器2访问数据）：



一个服务器的数据访问顺序：首先访问本地缓存，如果没有就访问分布式缓存，还没有就访问数据库，app访问数据库得到数据之后，会将数据传给redis，再将数据传给本地缓存，以供下一次访问方便

分布式缓存比本地缓存的第二个好处：本地缓存占用的是服务器的空间，不能太大，比较小，所以存不下太多数据，可能你存了一个数据后，又因为空间的问题把前面的某一个数据给挤出去了（取决于你的淘汰策略，可能是存放时间，可能是使用率），这时候如果需要挤出去的这个数据就需要去redis中取，因为redis中的空间足够存下



对主页中热门的帖子排名页进行优化：

通过存放在本地缓存优化，因为热门帖子和客户无关，每个客户看到的都是一样的（是同一个数据），你这次访问了热门帖子主页后，下次请求在同一台服务器上，就算是不同客户访问同一台服务器也能得到正确的响应，如果访问到了其他的服务器在这台服务器再存一次数据就好，本地缓存的空间够使，使得性能更好；这不像是登录凭证，不同的用户去访问同一台服务器，所需要的数据不是同一个，本地缓存存不下所有用户的登录凭证

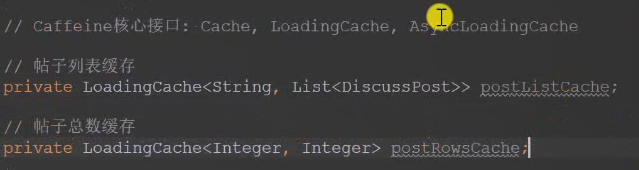
对于为什么不将按时间顺序的主页存放在本地缓存中：因为变化的频率太快了，随时都有客户可能发布新的帖子，不固定，而热门帖子是通过定时任务，经过一定的时间间隔之后，计算分数，重新排序，在一定的时间之内（项目中设置为两个小时算一次）保持稳定不变的

项目中使用的是caffeine当做本地缓存，可以通过spring整合，但不建议用spring整合本地缓存的工具，因为spring整合之后是通过一个缓存管理器去整合所有的缓存（疑问：通过依赖注入？可以设置不同的创建模式使得每次注入的都不是同一个bean对象啊），统一设置过期时间、缓存大小，不太合适，因为我们本地缓存不止有热门帖子，每一个缓存由于缓存的业务不同，管理缓存的逻辑也有差别

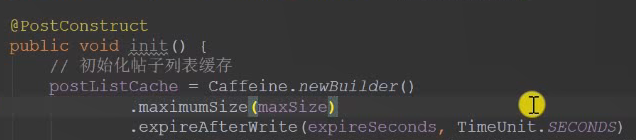
由于缓存列表缓存的热门帖子页，每个帖子在计算分数之前都有可能改变点赞数量等内容，所以我们设置缓存的有效时间为3分钟，过了三分钟则清除掉这个缓存

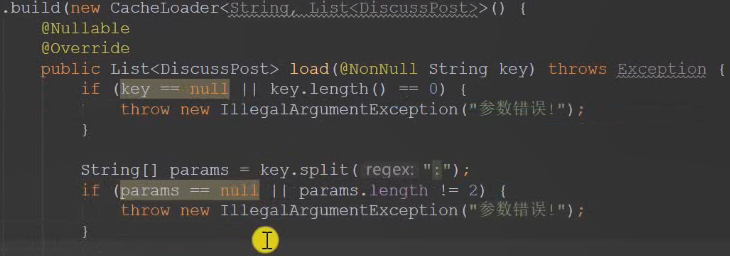
缓存一般是在service层设置：将热门帖子和总行数加入缓存，由于在查询的时候，页面的最下边页码的最后一个是总页码，需要总行数这个参数，所以想要性能更好的热门帖子列表，还需要将总行数加入本地缓存中

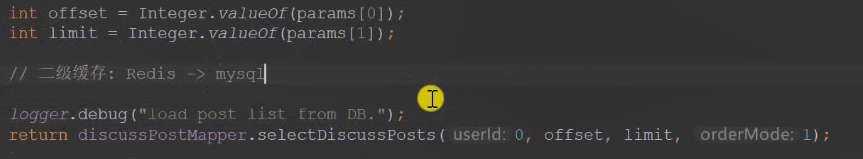
声明帖子列表缓存以及帖子总行数缓存:caffeine有两种核心接口来实现缓存，这里用的是LoadingCache，通过同步的方式，有多个线程同时访问缓存里的数据，如果缓存中的数据没有，会先让线程等待，等到从数据库中得到这个数据后，再让线程执行 （好像取数据也是同步的？）



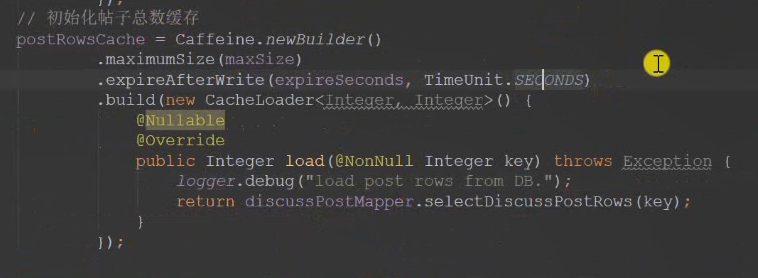
创建帖子列表本地缓存（postListCache）：由于帖子列表的本地缓存只用创建一次，所以我们可以在创建sevice对象时，直接创建（init）帖子列表缓存，设置帖子列表缓存的最大容量（不止要将热帖的第一页加入到本地缓存中，其他的页也要加入进去），设置帖子缓存的过期时间，build方法是解决怎么在数据库中查数据，数据的来源是哪，由于我们在存入数据时的key是offset：limit，所以需要先将其分开，才能调用dao层的查询方法，来获取到数据库中的帖子列表



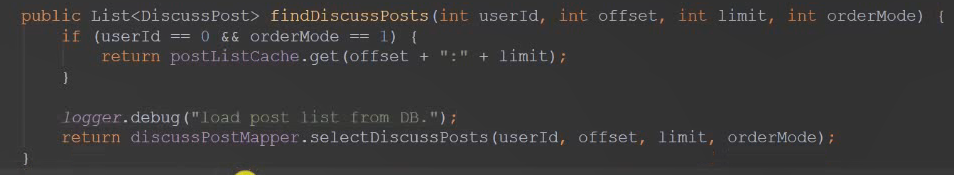




创建帖子总行数缓存（postRowsCache）：



在查询帖子列表，如果是userid=0（查询全部用户的帖子），ordermode=1（按照热门顺序排）则返回缓存中的热门帖子数据，在调用本地缓存的get方法时，如果没有这个数据，本地缓存缓存热门帖子的对象会去调用数据库得到这个数据



在调用总行数时，先查缓存中的帖子总行数：

