链滴

用户名自动完成算法

作者：88250

原文链接：https://ld246.com/article/1459220325886

来源网站：链滴

概述

发帖、回帖有时需要 @ 用户，这个时候需要用自动完成来辅助用户输入。

总的来说有两种思路：

1. 纯前端：自动完成候选数据在前端渲染时就给定，比如回帖时 @ 本贴的参与者，参与者相对与较，是可以在帖子渲染时就填充好的
2. 后端交互：发送请求到后端，由后端在整个用户列表中做匹配

社区目前选择的是第二种思路。

关键点

后端在进行用户名匹配时需要到整个用户列表中进行搜索，如果直接查库性能上肯定是不能接受的，以这个时候引入一个缓存来解决性能问题。

因为需求比较简单，不涉及到写数据和一致性，所以也没必要引入分布式缓存这样高大上的产品，直从数据库用户表中将所有用户名捞出放后放内存，并定时重捞（新用户注册、用户名变更等）。

目前感觉还是蛮好用的，但是有个问题一直没解决好：用户名匹配算法。

整体流程

用户输入 @ 并打出 A 时将发送一个自动完成请求到后端，后端接收到这个查询前缀为 A 的用户名时缓存的用户名列表（按字母排序）中进行二分查找，然后选择命中条目和后面 4 个条目：

```
public List<JSONObject> getUserNamesByPrefix(final String namePrefix) {
    final List<JSONObject> ret = new ArrayList<JSONObject>();

    final JSONObject nameToSearch = new JSONObject();
    nameToSearch.put(User.USER_NAME, namePrefix);

    // 二分查找，userNames 是缓存好的用户名列表
    final int index = Collections.binarySearch(userNames, nameToSearch, new Comparator<JS
NObject>() {
        @Override
        public int compare(final JSONObject u1, final JSONObject u2) {
            final String u1Name = u1.optString(User.USER_NAME).toLowerCase();
            final String inputName = u2.optString(User.USER_NAME).toLowerCase();

            if (u1Name.startsWith(inputName)) {
                return 0;
            } else {
                return namePrefix.compareTo(u1Name);
            }
        }
    });

    if (index >= 0) {
        final int max = index + 5 <= userNames.size() - 1 ? index + 5 : userNames.size() - 1;
```

```
        for (int i = index; i < max; i++) {
            ret.add(userNames.get(i));
        }
    }

    return ret;
}
```

至此，@ 时用户名自动完成的基本原理阐述完毕，下面分享一个细节。

前缀匹配

在进行二分查找时，上面的代码用了 startsWith 进行前缀匹配，乍一看感觉没错，因为自动完成时户就是按照从左到右的输入顺序进行。

实际上，漏了一个非常重要的前提。前面提到过用户名列表是按字母排序的，但这还不够，还需要再长度再排序，否则按前缀匹配就不可能准确。

总结

分享完了，大家还有什么要了解的请跟帖，非常欢迎任何改进！