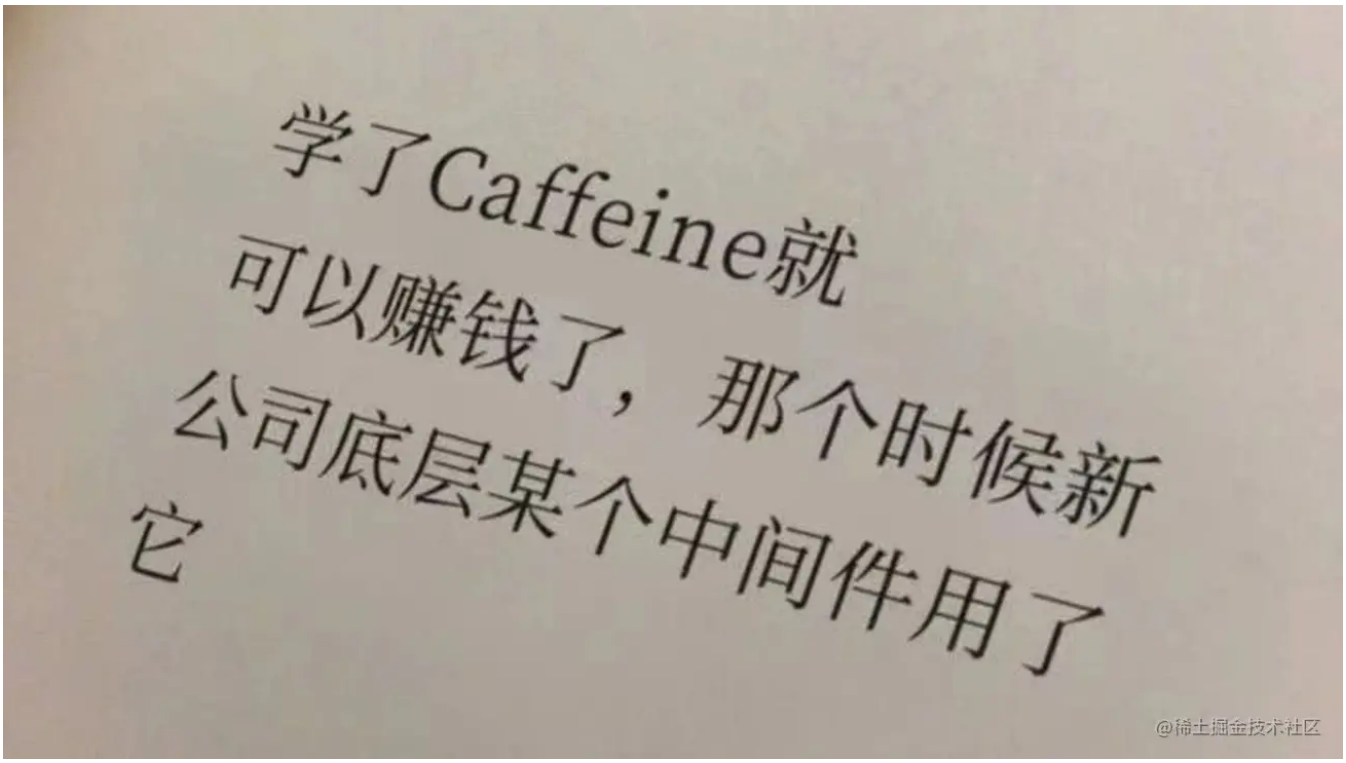
Spring官方都说废掉GuavaCache用Caffeine，赶快换系列三

小饭饭饭饭饭饭 Lv3
2020年12月25日 09:23 · 阅读 1610

关注



最近来了一个实习生小张，看了我在公司项目中使用的缓存框架Caffeine，三天两头跑来找我取经，说是要把Caffeine吃透，为此无奈的也只能一个个细心解答了。

后来这件事情被总监直到了，说是后面还有新人，让我将相关问题和细节汇总成一份教程，权当共享好了，该份教程也算是全网第一份，结合了目前我司游戏中业务场景的应用和思考，以及踩过的坑。

这是Caffeine教程的第三篇，主要讲解Caffeine和二级缓存的结合使用。

> “
>
> 实习生小张：主管说直接给Caffeine设置了最大缓存个数，会存在一个隐患，那便是当同时在线的玩家数超过最大缓存个数的情况下，会导致缓存被清，之后导致频繁读取数据库加载数据，让我在Caffeine的基础上，结合二级缓存解决这个问题。

可以的，目前来说Caffeine提供了整套机制，可以方便我们和二级缓存进行结合。

在具体给出例子前，要先引出一个CacheWriter的概念，我们可以把它当做一个回调对象，在往Caffeine的缓存put数据或者remove数据的时候回调用。

复制代码

```
/**
 * @author xifanxiaxue
 * @date 2020/12/5 10:18
 * @desc
 */
public class CaffeineWriterTest {

    /**
     * 充当二级缓存用，生命周期仅活到下个gc
     */
    private Map<Integer, WeakReference<Integer>> secondCacheMap =
            new ConcurrentHashMap<>();

    @Test
    public void test() throws InterruptedException {
        // 设置最大缓存个数为1
        LoadingCache<Integer, Integer> cache = Caffeine.newBuilder()
                .maximumSize(1)
                // 设置put和remove的回调
                .writer(new CacheWriter<Integer, Integer>() {
                    @Override
                    public void write(@NonNull Integer key, @NonNull Integer value) {
                        secondCacheMap.put(key, new WeakReference<>(value));
                        System.out.println("触发CacheWriter.write，将key = " + key + "放入二级缓存中");
                    }

                    @Override
                    public void delete(@NonNull Integer key, @Nullable Integer value, @NonNull RemovalCause cause) {
                        switch (cause) {
                            case EXPLICIT:
                                secondCacheMap.remove(key);
                                System.out.println("触发CacheWriter" +
                                        ".delete，清除原因：主动清除，将key = " + key +
                                        "从二级缓存清除");
                                break;
                            case SIZE:
                                System.out.println("触发CacheWriter" +
                                        ".delete，清除原因：缓存个数超过上限，key = " + key);
                                break;
                            default:
                                break;
```

```
                })
                .build(new CacheLoader<Integer, Integer>() {
                    @Nullable
                    @Override
                    public Integer load(@NonNull Integer key) {
                        WeakReference<Integer> value = secondCacheMap.get(key);
                        if (value == null) {
                            return null;
                        }

                        System.out.println("触发CacheLoader.load，从二级缓存读取key = " + key);
                        return value.get();
                    }
                });

        cache.put(1, 1);
        cache.put(2, 2);
        // 由于清除缓存是异步的，因而睡眠1秒等待清除完成
        Thread.sleep(1000);

        // 缓存超上限触发清除后
        System.out.println("从Caffeine中get数据，key为1，value为"+cache.get(1));
    }
}
```

举的这个例子稍显复杂，毕竟是要和二级缓存结合使用，不复杂点就没办法显示Caffeine的妙，先看下secondCacheMap对象，这是我用来充当二级缓存用的，由于value值我设置成为WeakReference弱引用，因而生命周期仅活到下个gc。

> “
>
> 稀饭：小张，这个例子就可以解决你的二级缓存如何结合的问题，你给我说说看最终打印结果值是null还是非null?
>
> ”

> “
>
> 小张：肯定是null 啊，因为key为1的缓存因为缓存个数超过上限被清除了呀。
>
> ”

对Caffeine的运行机制不够熟悉的人很容易犯了小张这样的错误，产生了对结果的误判。

> “
>
> 触发CacheWriter.write，将key = 1放入二级缓存中 触发CacheWriter.write，将key = 2放入二级缓存中 触发CacheWriter.delete，清除原因：缓存个数超过上限，key = 1 触发CacheLoader.load，从二级缓存读取key = 1 从Caffeine中get数据，key为1，value为1 触发CacheWriter.delete，清除原因：缓存个数超过上限，key = 2
>
> ”

结合代码，我们可以看到CacheWriter.delete中，我判断了RemovalCause，也就是清除缓存理由的意思，如果是缓存超上限，那么并不清除二级缓存的数据，而CacheLoader.load会从二级缓存中读取数据，所以在最终从Caffeine中加载key为1的数据的时候并不为null，而是从二级缓存拿到了数据。

> “
>
> 实习生小张：那最后的打印“ 触发CacheWriter.delete，清除原因：缓存个数超过上限，key = 2 “ 又是什么情况呢?
>
> ”

那是因为Caffeine在调用*CacheLoader.load*拿到非null的数据后会重新放入缓存中，这样便导致缓存个数又超过了最大的上限了，所以清除了key为2的缓存。

> “
>
> 实习生小张：稀饭稀饭，我这边想具体看到缓存命中率如何，有没有什么方法呢?
>
> ”

有的有的，看源码的话就可以看到，Caffeine内部有挺多打点记录的，不过需要我们在构建缓存的时候开启记录。

复制代码

```
/**
 * @author xifanxiaxue
 * @date 2020/12/1 23:12
 * @desc
 */
public class CaffeineRecordTest {

    /**
     * 模拟从数据库中读取数据
```

```
     * @return
     */
    private int getInDB(int key) {
        return key;
    }

    @Test
    public void test() {
        LoadingCache<Integer, Integer> cache = Caffeine.newBuilder()
                //开启记录
                .recordStats()
                .build(new CacheLoader<Integer, Integer>() {
                    @Override
                    public @Nullable Integer load(@NonNull Integer key) {
                        return getInDB(key);
                    }
                });
        cache.get(1);

        // 命中率
        System.out.println(cache.stats().hitRate());
        // 被剔除的数量
        System.out.println(cache.stats().evictionCount());
        // 加载新值所花费的平均时间[纳秒]
        System.out.println(cache.stats().averageLoadPenalty() );
    }
}
```

「实际应用：上次在游戏中引入Caffeine的时候便用来*record*的机制，只不过是在测试的时候用，一般不建议生产环境用这个。具体用法我是开了条线程定时的打印命中率、被剔除的数量以及加载新值所花费的平均时间，进而判断引入Caffeine是否具备一定的价值。」

> “
>
> 实习生小张：Caffeine我已经用上了，可是会有个问题，如果数据忘记保存入库，然后被淘汰掉了，玩家数据就丢失了，Caffeine有没有提供什么方法可以在淘汰的事情让开发者做点什么？
>
> ”

> “
>
> 稀饭：还真的有，Caffeine对外提供了淘汰监听，我们只需要在监听器内进行保存就可以了。

复制代码

```
/**
 * @author xifanxiaxue
 * @date 2020/11/19 22:34
 * @desc 淘汰通知
 */
public class CaffeineRemovalListenerTest {

    @Test
    public void test() throws InterruptedException {
        LoadingCache<Integer, Integer> cache = Caffeine.newBuilder()
                .expireAfterAccess(1, TimeUnit.SECONDS)
                .scheduler(Scheduler.systemScheduler())
            // 增加了淘汰监听
                .removalListener(((key, value, cause) -> {
                    System.out.println("淘汰通知，key：" + key + "，原因：" + cause);
                }))
                .build(new CacheLoader<Integer, Integer>() {
                    @Override
                    public @Nullable Integer load(@NonNull Integer key) throws Exception {
                        return key;
                    }
                });

        cache.put(1, 2);

        Thread.currentThread().sleep(2000);
    }
```

可以看到我这边使用removalListener提供了淘汰监听，因此可以看到以下的打印结果：

> “
>
> 淘汰通知，key：1，原因：EXPIRED
>
> ”

> “
>
> 实习生小张：我看到数据淘汰的时候是有提供了几个cause的，也就是原因，分别对应着什么呢？
>
> ”

目前数据被淘汰的原因不外有以下几个：

- REPLACED：就是替换了，也就是put数据的时候旧的数据被覆盖导致的移除。
- COLLECTED：这个有歧义点，其实就是收集，也就是垃圾回收导致的，一般是用弱引用或者软引用会导致这个情况。
- EXPIRED：数据过期，无需解释的原因。
- SIZE：个数超过限制导致的移除。

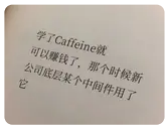
以上这几个就是数据淘汰时会携带的原因了，如果有需要，我们可以根据不同的原因处理不同的业务。

「实际应用：目前我们项目中就有在数据被淘汰的时候做了缓存入库的处理，毕竟确实有开发人员忘记在逻辑处理完后手动save入库了，所以只能做个兜底机制，避免数据丢失。」

> “
>
> **「第四篇将分享Caffeine结合db实现的定时持久化的缓存组件，对Caffeine有兴趣或者想咨询Caffeine相关问题，请关注我」**
>
> **「作者简介：饭谈编程，学技术，学习新技术，学习有用的技术，请微信搜索：稀饭下雪。」**
>
> ”

分类： 后端 标签： Java

文章被收录于专栏：



caffeine全网最全教程

caffeine是一个面向未来的本地缓存框架，该工程提供者了caffein...

关注专栏

评论

输入评论（Enter换行，Ctrl + Enter发送）

全部评论 4

最新　最热

小饭饭饭饭饭饭 Lv3 （作者）　服务端高级开发 @ wx...　1年前

系列：juejin.cn

1　回复

小饭饭饭饭饭饭 Lv3 （作者）　服务端高级开发 @ wx...　1年前

系列一：juejin.cn

1　回复

相关推荐

vivo互联网技术　12月前　Java　后端

如何把 Caffeine Cache 用得如丝般顺滑?

1557　12　1

程序员乔戈里　2年前　Java

美团面试官问我一个字符的String.length()是多少，我说是1，面试官说你回去好好学一下吧

7.0w　312　96

公众号_IT老哥　1月前　Java

为什么不建议用try catch处理异常?

5.1w　353　25

Java小咖秀　1年前　Spring Boot

SpringBoot缓存实战 Redis + Caffeine 实现多级缓存

1651　17　2

小饭饭饭饭饭饭　9月前　后端　Java

我写了个专栏，涵盖了本地缓存框架Caffeine教程、以及应用场景

946　10　2

23.1w 6031 265

程序员小黑 2年前 Spring

Spring Cache 缺陷，我好像有解决方案了

2987 10 2

杰出D 9月前 前端 算法

面试了十几个高级前端，竟然连（扁平数据结构转Tree）都写不出来

16.1w 2763 1483

暮色妖娆丶 1月前 后端 Java

优秀的后端应该有哪些开发习惯?

3.9w 481 187

前端胖头鱼 6月前 前端 正则表达式 JavaScript

就因为这三个知识点，我彻底学废了”正则表达式“

3.5w 1327 169

MacroZheng 1月前 后端 Java Redis

颜值爆表！Redis官方可视化工具来啦，功能真心强大！

4.5w 282 41

大帅老猿 11月前 前端 JavaScript

产品经理：你能不能用div给我画条龙?

11.1w 2948 585

后端架构进阶 6月前 后端

Caffeine Cache进阶本地缓存之王

1344 4 2

r09er 2年前 Spring

SpringDataCache踩坑记

2584 2 评论

薛定谔的狗12123 1年前 Spring Boot

1711 10 2

艺术家阿克曼 1年前 Java EE

分享一个比spring自带的Filter还强的

106 点赞 4

楼下小黑哥 1年前 Java

求求你了，不要再自己实现这些逻辑了，开源工具类不香吗?

3.7w 526 67

红尘炼心 6月前 前端 Vue.js React.js

你会用ES6，那倒是用啊!

25.1w 6367 1058

方石剑 5年前 HTTP PHP

HTTP Cache 为什么让人很困惑

1988 54 2

沉默王二 1年前 Java 后端

Guava - 拯救垃圾代码，写出优雅高效，效率提升N倍

7816 75 7