GUAVA Cache源码阅读-cache、AbstractCache 、AbstractLoadingCache、CacheBuilder

熊熊要更努力 关注
2017.08.01 15:22:41 字数 1,032 阅读 641

1. cache 接口
   主要有这几个方法：

- V getIfPresent(Object var1);
- V get(K var1, Callable<? extends V> var2) throws ExecutionException;
- ImmutableMap<K, V> getAllPresent(Iterable<?> var1);
- void put(K var1, V var2);
- void putAll(Map<? extends K, ? extends V> var1);
- void invalidate(Object var1);
- void invalidateAll(Iterable<?> var1);
- void invalidateAll();
- long size();
- CacheStats stats();
- ConcurrentMap<K, V> asMap();
- void cleanUp();

AbstractCache 是个抽象类，继承了Cache

1. 如果调用AbstractCache的get、put、size、invalidate、invalidateAll()、stats()、asMap()方法，会抛出UnsupportedOperationException()异常。
2. getAllPresent 返回的ImmutableMap为LinkedHashMap。
   当插入一串新数据时，会首先判断是否当前插入的是否存在于之前已经插入的数据中，如果不存在，继续判断是否已经当前对象中，如果不是，就插入结果集合。最后返回结果集合。
3. invalidateAll(Iterable<?> keys)，调用invalidate删除数据
4. 里面有一个SimpleStatsCounter内部类，实现了AbstractCache.StatsCounter接口。
   同时定义了一个LongAddable 接口。
   statsCounter 接口
   public interface StatsCounter
   {

熊熊要更努力 关注
总资产2

谷歌Python代码风格指南 -学习笔记
阅读 123

《JAVA程序性能优化》-ArrayBlockingQueue的简单实现
阅读 92

热门故事

闺蜜的一条朋友圈，结束了我和老公5年的婚姻

娘家拆迁分了两套新房，婆婆让我把房本写上小叔子的名？

前任一哭，现任必输

老婆偷偷拿30万给小舅子买豪车，被我一招“制服”

推荐阅读

Swift实现KVO
阅读 533

TableFactory
阅读 138

加油2022——新 iOS 面试题大全（附答案）
阅读 5,628

10X单细胞（10X空间转录组）数据分析之Consensus Non-negative...
阅读 323

Seurat的subset,数据提取方法
阅读 161

void recordLoadException(long var1);
void recordEviction();
CacheStats snapshot();
}
LongAddable 接口
interface LongAddable
{
void increment();
void add(long var1);
long sum();
}
SimpleStateCounter中定义了一系列LongAddAble操作。包括命中计数、未命中计数、装载成功计数、装载异常计数、总共装载计数、换出计数。（
long hitCount, long missCount, long loadSuccessCount, long loadExceptionCount, long totalLoadTime, long evictionCount)
snapshot()函数将返回当前所有计数的数量。
incrementBy(AbstractCache.StatsCounter other)可以直接在一系列的计数中加上另外一个AbstractCache.StatsCounter的数量。

AbstractLoadingCache是个抽象类，继承了AbstractCache.

方法	含义
V getUnchecked(K key)	暂时不理解
ImmutableMap<K, V> getAll(Iterable<? extends K> keys)	在内部创建了一个链表哈希map,将传入的值放入linkHashMap中，用迭代器进行遍历。最后返回ImmutableMap类型的对象。
V apply(K key)	调用getUnchecked返回对象
refresh	抛出UnsupportedOperationException异常

CacheBuilder
在cacheBuilder的文档中提到，一个LoadingCache和Cache的实例的构造者应该具有以下特点：
A builder of LoadingCache and Cache instances having any combination of the following features:

- automatic loading of entries into the cache
- least-recently-used eviction when a maximum size is exceeded
- time-based expiration of entries, measured since last access or last write
- keys automatically wrapped in [weak]references
- values automatically wrapped in [weak] or [soft] references
- notification of evicted (or otherwise removed) entries
- accumulation of cache access statistics

中文翻译就是

- 自动装载实体到内存。
- 当超过最大内存时，会清除**最近最少使用的**

- 删除或换出数据时有通知
- 积累缓存访问的数据

These features are all optional; caches can be created using all or none of them. By default cache instances created by CacheBuilder will not perform any type of eviction.

这些功能都是可选的;可以使用全部或不创建高速缓存。**默认情况下，CacheBuilder创建的缓存实例不会执行任何类型的逐出。**
一个使用cachebuilder的实例：

```
LoadingCache<Key, Graph> graphs = CacheBuilder.newBuilder()
 .maximumSize(10000)
 .expireAfterWrite(10, TimeUnit.MINUTES)
 .removalListener(MY_LISTENER)
 .build(
     new CacheLoader<Key, Graph>() {
       public Graph load(Key key) throws AnyException {
         return createExpensiveGraph(key);
       }
     });
```

还有一种创建方式

```
 String spec = "maximumSize=10000,expireAfterWrite=10m";
 LoadingCache<Key, Graph> graphs = CacheBuilder.from(spec)
   .removalListener(MY_LISTENER)
   .build(
   new CacheLoader<Key, Graph>() {
     public Graph load(Key key) throws AnyException {
       return createExpensiveGraph(key);
     }
   });
```

这两种创建方式是等价的。

The returned cache is implemented as a hash table with similar performance characteristics to ConcurrentHashMap
. It implements all optional operations of the LoadingCache and Cache interfaces. The asMap
view (and its collection views) have *weakly consistent iterators*. This means that they are safe for concurrent use, but if other threads modify the cache after the iterator is created, it is undefined which of these changes, if any, are reflected in that iterator. These iterators never throw ConcurrentModificationException
.

大概意思如下：

1. 返回的cache的实现类似于ConcurrentHashMap的哈希表。
   它实现了LoadingCache和Cache的所有接口。
2. asMap视图是弱一致的迭代器。这意味着他们是线程安全的。它没有定义这些改变，当影响到迭代器时，这些迭代器绝不会抛出ConcurrentModificationException异常。
3. **已经换出的数据也可能被Cache.size()方法在计数时计算，但是不能对其进行读写操作。**
4. CacheBuilder<K,V> maximumWeight(long weight)这个方法确定了缓存包含的对象的最大权重。
   注意权重只是用来决定缓存是否已经超过其最大容量。它对于接下来要将哪个对象换出内存

- IllegalArgumentException假如设置的权值为负
- IllegalStateException假如已经设置了最大权重或者最大容量

6.

函数	返回值	说明
build()	Cache<K1,V1>	创建缓存，此缓存在请求关键字时，不会自动加载值
build(CacheLoader<? super K1,V1> loader)	LoadingCache<K1, V1>	创建缓存。它可以返回给定键的已经加载的值，也可以使用提供的CacheLoader来进行原子计算或检索它
ticker	CacheBuilder<K,V>	指定一个纳秒精度时间源，用于确定条目何时过期。
concurrencyLevel(int concurrencyLevel)	CacheBuilder<K,V>	指导更新操作之间允许的并发性。
initialCapacity(int initialCapacity)	CacheBuilder<K,V>	设置内部哈希表的最小长度。
maximumSize(long size)	CacheBuilder<K,V>	指定缓存可能包含的最大条目数。
maximumWeight(long weight)	CacheBuilder<K,V>	指定缓存可能包含的条目的最大权重。
newBuilder()	static CacheBuilder<Object,Object>	构造一个新的CacheBuilder实例，具有默认设置，包括强大的键，强大的值，并且没有任何自动驱逐。
recordStats()	CacheBuilder<K,V>	在缓存的操作期间启用CacheStats的累积
refreshAfterWrite(long duration, TimeUnit unit)	CacheBuilder<K,V>	指定在条目创建后经过固定持续时间或最近更换其值后，活动条目符合自动刷新的条件。
from(String spec)	static CacheBuilder<Object,Object>	使用规范中指定的设置构造新的CacheBuilder实例
removalListener(RemovalListener<? super K1,? super V1> listener)	<K1 extends K,V1 extends V> CacheBuilder<K1, V1>	指定一个监听器实例，高速缓存在任何情况下，每次删除一个条目时应该通知它。
softValues()	CacheBuilder<K,V>	指定存储在缓存中的每个值（非键）都应该包装在SoftReference中（默认情况下，使用强引用）。

熊熊要更努力 我与深渊相逢，而我活了下来
总资产2 共写了3.0W字 获得48个赞 共29个粉丝

关注

写下你的评论...

全部评论 0 只看作者 按时间倒序 按时间正序

被以下专题收入，发现更多相似内容

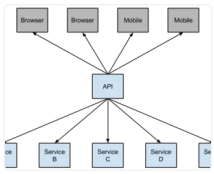
+收入我的专题 guava源码...

推荐阅读

更多精彩内容>

Spring Cloud

Spring Cloud为开发人员提供了快速构建分布式系统中一些常见模式的工具（例如配置管理，服务发现，断路器，智...



卡卡罗2017 阅读 128,596 评论 18 赞 137

为谁而作?

为谁而作？是一件非常重要的事情。 考虑到作是一个非常宽泛的动词，我们暂且把此处的“作”，看作是创作的作。如果分得再...

岗鉴 阅读 121 评论 0 赞 3

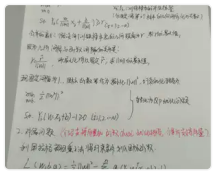
是坑也得跳，电信运营商应成为智能家居的领军力量

我们每个人都有一个梦想，希望未来的家庭生活充满了智能，就如同科幻小说里描写的那样，至少也应该有个“大白”陪伴。实际...

花捧娱乐vlog视频植入 阅读 288 评论 0 赞 0

支持向量机的简单理解

各位小伙伴们大家好，这几天弱弱的看了看老掉牙的支持向量机(Support Vector Machine, SVM)...



云时之间 阅读 1,051 评论 1 赞 3

2017-07-30

今天和大米结束了她喜欢的《窗边的小豆豆》，开始了《疯狂的学校》。一所日本学校的学校生活，一所美国的学校生活；两个国...

悦米时光 阅读 53 评论 0 赞 0

盈痴小尼 阅读 339 评论 0 赞 0