# 乌云上



**导航**

博客园
首　页
新随笔
联　系
订　阅 XML
管　理

< 2018年9月 >

| 日 | 一 | 二 | 三 | 四 | 五 | 六 |
|---|---|---|---|---|---|---|
| 26 | 27 | 28 | 29 | 30 | 31 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| 30 | 1 | 2 | 3 | 4 | 5 | 6 |

**公告**

昵称：乌云上
园龄：1年9个月
粉丝：1
关注：6
+加关注

**搜索**

找找看
谷歌搜索

**常用链接**

## guava快速入门（三）

Guava工程包含了若干被Google的 Java项目广泛依赖 的核心库，例如：集合 [collections] 、缓存 [caching] 、原生类型支持 [primitives support] 、并发库 [concurrency libraries] 、通用注解 [common annotations] 、字符串处理 [string processing] 、I/O 等等。

guava类似Apache Commons工具集

### Cache

缓存在很多场景下都是相当有用的。例如，计算或检索一个值的代价很高，并且对同样的输入需要不止一次获取值的时候，就应当考虑使用缓存。

Guava Cache与ConcurrentMap很相似，但也不完全一样。最基本的区别是ConcurrentMap会一直保存所有添加的元素，直到显式地移除。相对地，Guava Cache为了限制内存占用，通常都设定为自动回收元素。在某些场景下，尽管LoadingCache 不回收元素，它也是很有用的，因为它会自动加载缓存。

Guava Cache是一个全内存的本地缓存实现，它提供了线程安全的实现机制。

通常来说，Guava Cache适用于：

- 你愿意消耗一些内存空间来提升速度。
- 你预料到某些键会被查询一次以上。
- 缓存中存放的数据总量不会超出内存容量。（Guava Cache是单个应用运行时的本地缓存。它不把数据存放到文件或外部服务器。

如果这不符合你的需求，请尝试Memcached这类工具）

我的随笔
我的评论
我的参与
最新评论
我的标签

**我的标签**

日常积累(110)
Java(26)
数据库(24)
SqlServer(22)
环境搭建(21)
SQL(19)
jQuery(18)
php(16)
Node.js学习(15)
JavaScript(15)
更多

**随笔分类**

.NET(6)
Ajax(1)
BAT(1)
CASCO卡斯柯
Dos命令(1)
Excel(2)
Game(1)
Java(27)
jQuery(17)
jQuery easyUI(1)
Node.js(16)
PHP(14)
Python(15)
SQL(26)
WeUI(10)
总结(4)

**随笔档案**

2018年9月 (19)

Guava Cache有两种创建方式：

- cacheLoader
- callable callback

LoadingCache是附带CacheLoader构建而成的缓存实现

```
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;

import com.google.common.cache.CacheBuilder;
import com.google.common.cache.CacheLoader;
import com.google.common.cache.LoadingCache;

public class LoadingCacheDemo {

    public static void main(String[] args) {
        LoadingCache<String, String> cache = CacheBuilder.newBuilder().maximumSize(100) // 最大缓存数目
                .expireAfterAccess(2, TimeUnit.SECONDS) // 缓存1秒后过期
                .build(new CacheLoader<String, String>() {
                    @Override
                    public String load(String key) throws Exception {
                        return key;
                    }
                });
        cache.put("j", "java");
        cache.put("c", "cpp");
        cache.put("s", "scala");
        cache.put("g", "go");
        try {
            System.out.println(cache.get("j"));
            TimeUnit.SECONDS.sleep(1);
            System.out.println(cache.get("s")); // 1秒后 输出scala
            TimeUnit.SECONDS.sleep(2);
```

2018年8月 (54)
2018年7月 (6)
2018年6月 (43)
2018年5月 (13)
2018年4月 (6)
2018年3月 (1)
2018年1月 (1)

**积分与排名**

积分 - 5839
排名 - 58324

**最新评论**

1. Re:如何把ASP.NET MVC项目部署到本地IIS上
@~雨落忧伤~引用别人能访问吗？可以的，前提是，访问者和你的电脑在同一个局域网上，并且知道你的电脑ip地址，比如，你用手机连上你电脑开出来的wifi，然后再手机的浏览器中输入ip地址也是可以访问的。.....
--乌云上

2. Re:如何把ASP.NET MVC项目部署到本地IIS上
别人能访问吗？
--~雨落忧伤~

**阅读排行榜**

1. 常见浏览器User-Agent大全(773)
2. Python——第一个python程序helloworld(654)
3. jquery里判断数组内是否包含了指定的值或元素的方法(368)

```
28              System.out.println(cache.get("s")); // 2秒后 输出s
29
30
31          } catch (ExecutionException | InterruptedException e) {
32              e.printStackTrace();
33          }
34      }
35
36  }
```

返回：

```
1  java
2  scala
3  s
```

回调：

```
1  import java.util.concurrent.ExecutionException;
2  import java.util.concurrent.TimeUnit;
3
4  import com.google.common.cache.Cache;
5  import com.google.common.cache.CacheBuilder;
6
7  public class CallbackDemo {
8
9      public static void main(String[] args) {
10         Cache<String, String> cache = CacheBuilder.newBuilder().maximumSize(100)
11                 .expireAfterAccess(1, TimeUnit.SECONDS)
12                 .build();
13         try {
14             String result = cache.get("java", () -> "hello java");
15             System.out.println(result);
16         } catch (ExecutionException e) {
```

4. 如何把ASP.NET MVC项目部署到本地IIS上(329)
5. jquery操作radio单选按钮，实现取值，动态选中，动态删除的各种方法(328)

**评论排行榜**

1. 如何把ASP.NET MVC项目部署到本地IIS上(2)

```
            e.printStackTrace();
        }
    }

}
```

refresh机制:

- LoadingCache.refresh(K) 在生成新的value的时候，旧的value依然会被使用。
- CacheLoader.reload(K, V) 生成新的value过程中允许使用旧的value
- CacheBuilder.refreshAfterWrite(long, TimeUnit) 自动刷新cache

## 并发

### ListenableFuture

传统JDK中的Future通过异步的方式计算返回结果:在多线程运算中可能或者可能在没有结束返回结果，Future是运行中的多线程的一个引用句柄，确保在服务执行返回一个Result。

ListenableFuture可以允许你注册回调方法(callbacks)，在运算（多线程执行）完成的时候进行调用，或者在运算（多线程执行）完成后立即执行。这样简单的改进，使得可以明显的支持更多的操作，这样的功能在JDK concurrent中的Future是不支持的。

```
import java.util.concurrent.ExecutionException;
import java.util.concurrent.Executors;

import com.google.common.util.concurrent.FutureCallback;
import com.google.common.util.concurrent.Futures;
import com.google.common.util.concurrent.ListenableFuture;
import com.google.common.util.concurrent.ListeningExecutorService;
import com.google.common.util.concurrent.MoreExecutors;

public class ListenableFutureDemo {

```

```
12      public static void main(String[] args) {
13          // 将ExecutorService装饰成ListeningExecutorService
14          ListeningExecutorService service = MoreExecutors.
15                  listeningDecorator(Executors.newCachedThreadPool());
16
17          // 通过异步的方式计算返回结果
18          ListenableFuture<String> future = service.submit(() -> {
19              System.out.println("call execute..");
20              return "task success!";
21          });
22
23          // 有两种方法可以执行此Future并执行Future完成之后的回调函数
24          future.addListener(() -> { // 该方法会在多线程运算完的时候,指定的Runnable参数传入的对象会被指定的Exe
25              try {
26                  System.out.println("result: " + future.get());
27              } catch (InterruptedException | ExecutionException e) {
28                  e.printStackTrace();
29              }
30          }, service);
31
32          Futures.addCallback(future, new FutureCallback<String>() {
33              @Override
34              public void onSuccess( String result) {
35                  System.out.println("callback result: " + result);
36              }
37
38              @Override
39              public void onFailure(Throwable t) {
40                  System.out.println(t.getMessage());
41              }
42          }, service);
43      }
44
45  }
```

返回：

```
call execute..
result: task success!
callback result: task success!
```

## IO

```
import java.io.File;
import java.io.IOException;
import java.util.List;

import com.google.common.base.Charsets;
import com.google.common.collect.ImmutableList;
import com.google.common.io.Files;

public class FileDemo {

    public static void main(String[] args) {
        File file = new File(System.getProperty("user.dir"));
        System.out.println(file.getName());
        System.out.println(file.getPath());
    }

    // 写文件
    private void writeFile(String content, File file) throws IOException {
        if (!file.exists()) {
            file.createNewFile();
        }
        Files.write(content.getBytes(Charsets.UTF_8), file);
    }
```

```
24
25      // 读文件
26      private List<String> readFile(File file) throws IOException {
27          if (!file.exists()) {
28              return ImmutableList.of(); // 避免返回null
29          }
30          return Files.readLines(file, Charsets.UTF_8);
31      }
32
33      // 文件复制
34      private void copyFile(File from, File to) throws IOException {
35          if (!from.exists()) {
36              return;
37          }
38          if (!to.exists()) {
39              to.createNewFile();
40          }
41          Files.copy(from, to);
42      }
43
44  }
```

返回：

```
1  collection-others
2  D:\GITHUB\java\code\test01\collection-others
```

参考：Google Guava官方教程（中文版）

guava-importnew

分类: Java

标签: 日常积累, guava, guava基础, Java

好文要顶 关注我 收藏该文

乌云上
关注 - 6
粉丝 - 1
+加关注

0 0

« 上一篇: guava快速入门（二）
» 下一篇: 什么是SSH

posted on 2018-06-20 17:24 乌云上 阅读(46) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

**注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。**



**最新IT新闻**:
· 约谈整改28天后探访：北京租房仍存虚假房源 中介自爆商业内幕