一篇让你熟练掌握Google Guava包(全网最全)

程序员面试之道 Lv2
2021年05月20日 15:18 · 阅读 2211

关注

Google Guava

guava开源库的地址：github.com/google/guav...

概述

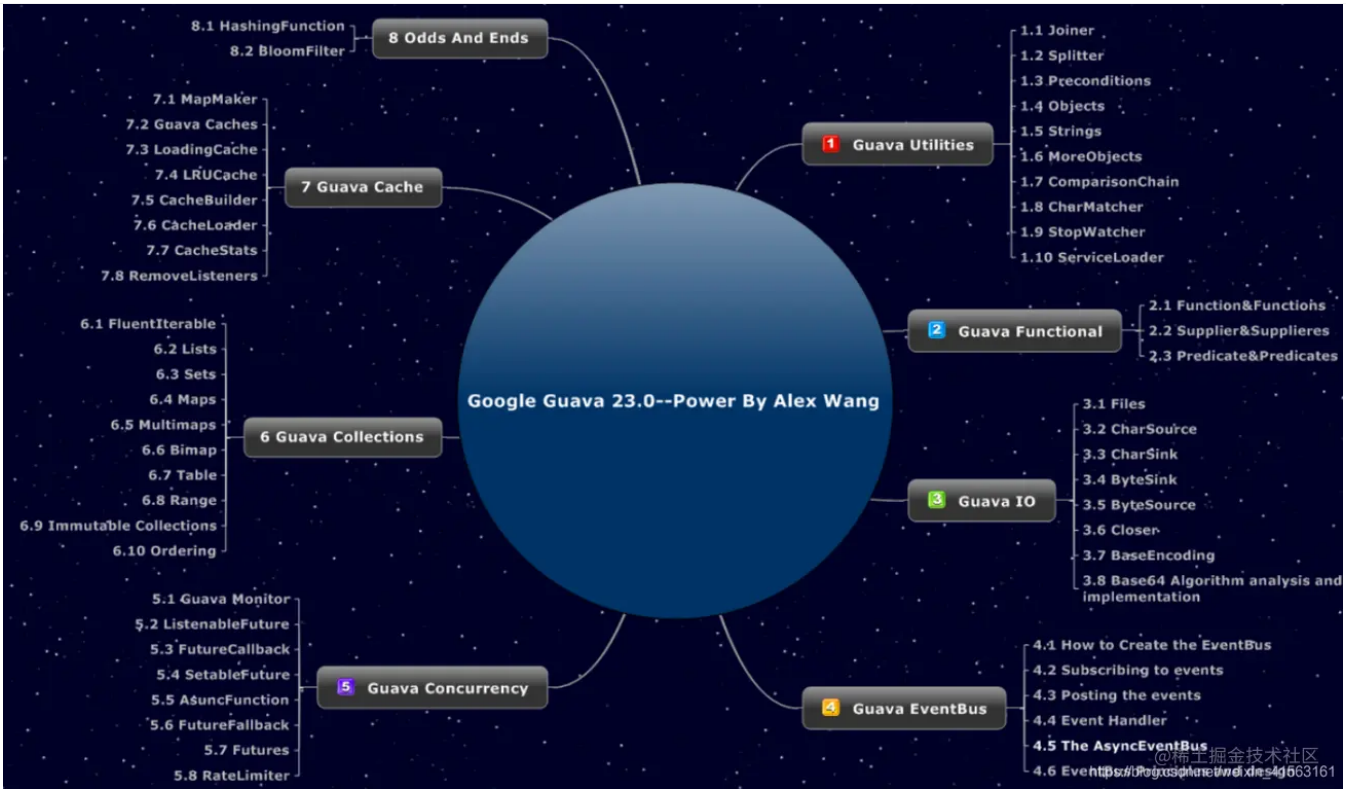
工具类 就是封装平常用的方法，不需要你重复造轮子，节省开发人员时间，提高工作效率。谷歌作为大公司，当然会从日常的工作中提取中很多高效率的方法出来。所以就诞生了guava。

guava的优点：

- 高效设计良好的API，被Google的开发者设计，实现和使用
- 遵循高效的java语法实践
- 使代码更刻度，简洁，简单
- 节约时间，资源，提高生产力

guava的核心库：

- 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的使用

引入gradle依赖（引入Jar包）

java 复制代码

```
compile 'com.google.guava:guava:26.0-jre'
```

java 复制代码

```
<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>21.0</version>
</dependency>
```

1.集合的创建

1.1

java 复制代码

```
// 普通Collection的创建
List<String> list = Lists.newArrayList();
Set<String> set = Sets.newHashSet();
Map<String, String> map = Maps.newHashMap();
```

```
ImmutableList<String> iList = ImmutableList.of("a", "b", "c");
ImmutableSet<String> iSet = ImmutableSet.of("e1", "e2");
ImmutableMap<String, String> iMap = ImmutableMap.of("k1", "v1", "k2", "v2");
```

创建不可变集合 先理解什么是immutable(不可变)对象

- 在多线程操作下，是线程安全的
- 所有不可变集合会比可变集合更有效的利用资源
- 中途不可改变

java 复制代码

```
ImmutableList<String> immutableList = ImmutableList.of("1","2","3","4");
```

这声明了一个不可变的List集合，List中有数据1，2，3，4。类中的 操作集合的方法（譬如add, set, sort, replace等）都被声明过期，并且抛出异常。 而没用guava之前是需要声明并且加各种包裹集合才能实现这个功能

java 复制代码

```
// add 方法
@Deprecated @Override
public final void add(int index, E element) {
  throw new UnsupportedOperationException();
}
```

1.2 当我们需要一个map中包含key为String类型，value为List类型的时候，以前我们是这样写的

java 复制代码

```
Map<String,List<Integer>> map = new HashMap<String,List<Integer>>();
List<Integer> list = new ArrayList<Integer>();
list.add(1);
list.add(2);
map.put("aa", list);
System.out.println(map.get("aa"));//[1, 2]
```

现在

iava 复制代码

```
map.put("aa", 2);
System.out.println(map.get("aa"));  //[1, 2]
```

1.3

java 复制代码

```
MultiSet: 无序+可重复   count()方法获取单词的次数  增强了可读性+操作简单
创建方式:  Multiset<String> set = HashMultiset.create();

Multimap: key-value  key可以重复
创建方式: Multimap<String, String> teachers = ArrayListMultimap.create();

BiMap: 双向Map(Bidirectional Map) 键与值都不能重复
创建方式:  BiMap<String, String> biMap = HashBiMap.create();

Table: 双键的Map Map--> Table-->rowKey+columnKey+value  //和sql中的联合主键有点像
创建方式: Table<String, String, Integer> tables = HashBasedTable.create();

...等等(guava中还有很多java里面没有给出的集合类型)
```

2.特色工具

字符串连接器Joiner 连接多个字符串并追加到StringBuilder

java 复制代码

```
StringBuilder stringBuilder = new StringBuilder("hello");
// 字符串连接器，以|为分隔符，同时去掉null元素
Joiner joiner1 = Joiner.on("|").skipNulls();
// 构成一个字符串foo|bar|baz并添加到stringBuilder
stringBuilder = joiner1.appendTo(stringBuilder, "foo", "bar", null, "baz");
System.out.println(stringBuilder); // hellofoo|bar|baz
```

连接List元素并写到文件流

java 复制代码

```
FileWriter fileWriter = null;
try{
    fileWriter = new FileWriter(new File("/home/gzx/Documents/tmp.txt"));
}
catch(Exception e){
    System.out.println(e.getMessage());
`
```

```
dateList.add(new Date());
dateList.add(null);
dateList.add(new Date());
// 构造连接器：如果有null元素，替换为no string
Joiner joiner2 = Joiner.on("#").useForNull("no string");
try{
    // 将list的元素的tostring()写到fileWriter，是否覆盖取决于fileWriter的打开方式，默认是覆盖，若有true，则是追加
    joiner2.appendTo(fileWriter, dateList);
    // 必须添加close()，否则不会写文件
    fileWriter.close();
}
catch(IOException e){
    System.out.println(e.getMessage());
}
```

字符串分割器Splitter

将字符串分割为Iterable

java 复制代码

```
// 分割符为|，并去掉得到元素的前后空白
Splitter sp = Splitter.on("|").trimResults();
String str = "hello | world | your | Name ";
Iterable<String> ss = sp.split(str);
for(String it : ss){
    System.out.println(it);
}
```

结果为： hello world your Name

字符串工具类Strings

java 复制代码

```
System.out.println(Strings.isNullOrEmpty("")); // true
System.out.println(Strings.isNullOrEmpty(null)); // true
System.out.println(Strings.isNullOrEmpty("hello")); // false
// 将null转化为""
System.out.println(Strings.nullToEmpty(null)); // ""

// 从尾部不断补充T只到总共8个字符，如果源字符串已经达到或操作，则原样返回。类似的有padStart
System.out.println(Strings.padEnd("hello", 8, 'T')); // helloTTT
```

字符匹配器CharMatcher 空白一一替换

java 复制代码

```
// 空白回车换行对应换成一个#，一对一换
String stringWithLinebreaks = "hello world\r\r\ryou are here\n\ntake it\t\t\teasy";
String s6 = CharMatcher.BREAKING_WHITESPACE.replaceFrom(stringWithLinebreaks,'#');
System.out.println(s6); // hello#world###you#are#here##take#it###easy
```

连续空白缩成一个字符

java 复制代码

```
// 将所有连在一起的空白回车换行字符换成一个#，倒塌
String tabString = "  hello   \n\t\tworld   you\r\nare          here  ";
String tabRet = CharMatcher.WHITESPACE.collapseFrom(tabString, '#');
System.out.println(tabRet); // #hello#world#you#are#here#
```

去掉前后空白和缩成一个字符

java 复制代码

```
// 在前面的基础上去掉字符串的前后空白，并将空白换成一个#
String trimRet = CharMatcher.WHITESPACE.trimAndCollapseFrom(tabString, '#');
System.out.println(trimRet);// hello#world#you#are#here
```

保留数字

java 复制代码

```
String letterAndNumber = "1234abcdABCD56789";
// 保留数字
String number = CharMatcher.JAVA_DIGIT.retainFrom(letterAndNumber);
System.out.println(number);// 123456789
```

3.将集合转换为特定规则的字符串

3.1以前我们将list转换为特定规则的字符串是这样写的:

java 复制代码

```
//use java
List<String> list = new ArrayList<String>();
list.add("aa");
list.add("bb");
list.add("cc");
String str = "";
for(int i=0; i<list.size(); i++){
    str = str + "-" +list.get(i);
}
```

```java
//use guava
List<String> list = new ArrayList<String>();
list.add("aa");
list.add("bb");
list.add("cc");
String result = Joiner.on("-").join(list);
//result为 aa-bb-cc
```

3.2把map集合转换为特定规则的字符串

java 复制代码

```java
Map<String, Integer> map = Maps.newHashMap();
map.put("xiaoming", 12);
map.put("xiaohong",13);
String result = Joiner.on(",").withKeyValueSeparator("=").join(map);
// result为 xiaoming=12,xiaohong=13
```

4.将String转换为特定的集合

java 复制代码

```java
//use java
List<String> list = new ArrayList<String>();
String a = "1-2-3-4-5-6";
String[] strs = a.split("-");
for(int i=0; i<strs.length; i++){
    list.add(strs[i]);
}

//use guava
String str = "1-2-3-4-5-6";
List<String> list = Splitter.on("-").splitToList(str);
//list为 [1, 2, 3, 4, 5, 6]
```

guava还可以使用 omitEmptyStrings().trimResults() 去除空串与空格

java 复制代码

```java
String str = "1-2-3-4-  5-  6   ";
List<String> list = Splitter.on("-").omitEmptyStrings().trimResults().splitToList(str);
System.out.println(list);
```

java 复制代码

```
String str = "xiaoming=11,xiaohong=23";
Map<String,String> map = Splitter.on(",").withKeyValueSeparator("=").split(str);
```

5.guava还支持多个字符切割，或者特定的正则分隔

java 复制代码

```
String input = "aa.dd,,ff,,.";
List<String> result = Splitter.onPattern("[.|,]").omitEmptyStrings().splitToList(input);
```

关于字符串的操作 都是在Splitter这个类上进行的

java 复制代码

```
// 判断匹配结果
boolean result = CharMatcher.inRange('a', 'z').or(CharMatcher.inRange('A', 'Z')).matches('K'); //true

// 保留数字文本  CharMatcher.digit() 已过时   retain 保留
//String s1 = CharMatcher.digit().retainFrom("abc 123 efg"); //123
String s1 = CharMatcher.inRange('0', '9').retainFrom("abc 123 efg"); // 123

// 删除数字文本  remove 删除
// String s2 = CharMatcher.digit().removeFrom("abc 123 efg");   //abc  efg
String s2 = CharMatcher.inRange('0', '9').removeFrom("abc 123 efg"); // abc  efg
```

6. 集合的过滤

我们对于集合的过滤，思路就是迭代，然后再具体对每一个数判断，这样的代码放在程序中，难免会显得很臃肿，虽然功能都有，但是很不好看。

guava写法

java 复制代码

```
import com.google.common.base.*;
import com.google.common.collect.*;
import com.google.common.collect.Maps;
import org.junit.jupiter.api.Test;

import java.util.*;

/**
```

```
 * @description:
 */
public class Test8 {

    @Test
    public void Test1(){
        //按照条件过滤
        ImmutableList<String> names = ImmutableList.of("begin", "code", "Guava", "Java");
        Iterable<String> fitered = Iterables.filter(names, Predicates.or(Predicates.equalTo("Guava"), Predicates.equalTo
        System.out.println(fitered);
        // [Guava, Java]

        //自定义过滤条件   使用自定义回调方法对Map的每个Value进行操作
        ImmutableMap<String, Integer> m = ImmutableMap.of("begin", 12, "code", 15);
        // Function<F, T> F表示apply()方法input的类型，T表示apply()方法返回类型
        Map<String, Integer> m2 = Maps.transformValues(m, input -> {
            if(input > 12){
                return input;
            }else{
                return input + 1;
            }
        });
        System.out.println(m2);
        //{begin=13, code=15}
    }
}
```

set的交集, 并集, 差集

java 复制代码

```
HashSet setA = newHashSet(1, 2, 3, 4, 5);
HashSet setB = newHashSet(4, 5, 6, 7, 8);

SetView union = Sets.union(setA, setB);
System.out.println("union:");
for (Integer integer : union)
    System.out.println(integer);          //union 并集:12345867

SetView difference = Sets.difference(setA, setB);
System.out.println("difference:");
for (Integer integer : difference)
    System.out.println(integer);        //difference 差集:123

SetView intersection = Sets.intersection(setA, setB);
System.out.println("intersection:");
```

```
System.out.println(integer);  //intersection 交集:45
```

map的交集，并集，差集

java 复制代码

```
HashMap<String, Integer> mapA = Maps.newHashMap();
mapA.put("a", 1);mapA.put("b", 2);mapA.put("c", 3);

HashMap<String, Integer> mapB = Maps.newHashMap();
mapB.put("b", 20);mapB.put("c", 3);mapB.put("d", 4);

MapDifference differenceMap = Maps.difference(mapA, mapB);
differenceMap.areEqual();
Map entriesDiffering = differenceMap.entriesDiffering();
Map entriesOnlyLeft = differenceMap.entriesOnlyOnLeft();
Map entriesOnlyRight = differenceMap.entriesOnlyOnRight();
Map entriesInCommon = differenceMap.entriesInCommon();

System.out.println(entriesDiffering);   // {b=(2, 20)}
System.out.println(entriesOnlyLeft);    // {a=1}
System.out.println(entriesOnlyRight);   // {d=4}
System.out.println(entriesInCommon);    // {c=3}
```

7.检查参数

java 复制代码

```
//use java
if(list!=null && list.size()>0)
'''
if(str!=null && str.length()>0)
'''
if(str !=null && !str.isEmpty())

//use guava
if(!Strings.isNullOrEmpty(str))

//use java
if (count <= 0) {
    throw new IllegalArgumentException("must be positive: " + count);
}

//use guava
Preconditions.checkArgument(count > 0, "must be positive: %s", count);
```

免去了很多麻烦！并且会使你的代码看上去更好看。而不是代码里面充斥着 !=null， !=""

==检查是否为空,不仅仅是字符串类型，其他类型的判断，全部都封装在 Preconditions类里，里面的方法全为静态==

其中的一个方法的源码

java 复制代码

```
@CanIgnoreReturnValue
public static <T> T checkNotNull(T reference) {
  if (reference == null) {
    throw new NullPointerException();
  }
  return reference;
}
```

方法声明（不包括额外参数）	描述	检查失败时抛出的异常
checkArgument(boolean)	检查boolean是否为true，用来检查传递给方法的参数。	IllegalArgumentException
checkNotNull(T)	检查value是否为null，该方法直接返回value，因此可以内嵌使用checkNotNull。	NullPointerException
checkState(boolean)	用来检查对象的某些状态。	IllegalStateException
checkElementIndex(int index, int size)	检查index作为索引值对某个列表、字符串或数组是否有效。 index > 0 && index < size	IndexOutOfBoundsException
checkPositionIndexes(int start, int end, int size)	检查[start,end]表示的位置范围对某个列表、字符串或数组是否有效	IndexOutOfBoundsException

8. MoreObjects

这个方法是在Objects过期后官方推荐使用的替代品，该类最大的好处就是不用大量的重写==**toString**==，用一种很优雅的方式实现重写，或者在某个场景定制使用。

java 复制代码

```
System.out.println(str);
//输出Person{age=11}
```

9.强大的Ordering排序器

排序器[Ordering]是Guava流畅风格比较器[Comparator]的实现，它可以用来为构建复杂的比较器，以完成集合排序的功能。

java 复制代码

```
natural()  对可排序类型做自然排序，如数字按大小，日期按先后排序
usingToString()      按对象的字符串形式做字典排序[lexicographical ordering]
from(Comparator)  把给定的Comparator转化为排序器
reverse()  获取语义相反的排序器
nullsFirst()      使用当前排序器，但额外把null值排到最前面。
nullsLast()       使用当前排序器，但额外把null值排到最后面。
compound(Comparator) 合成另一个比较器，以处理当前排序器中的相等情况。
lexicographical()     基于处理类型T的排序器，返回该类型的可迭代对象Iterable<T>的排序器。
onResultOf(Function)     对集合中元素调用Function，再按返回值用当前排序器排序。
```

示例

java 复制代码

```
Person person = new Person("aa",14);  //String name  ,Integer age
Person ps = new Person("bb",13);
Ordering<Person> byOrdering = Ordering.natural().nullsFirst().onResultOf(new Function<Person,String>(){
      public String apply(Person person){
            return person.age.toString();
      }
});
byOrdering.compare(person, ps);
System.out.println(byOrdering.compare(person, ps)); //1      person的年龄比ps大 所以输出1
```

10.计算中间代码的运行时间

java 复制代码

```
import com.google.common.base.Stopwatch;

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

```
 * @date: 2020/11/18 20:16
 * @description:
 */
public class Test9 {
    public static void main(String[] args) throws InterruptedException {
        // 创建stopwatch并开始计时
        Stopwatch stopwatch = Stopwatch.createStarted();
        Thread.sleep(1980);

        // 以秒打印从计时开始至现在的所用时间，向下取整
        System.out.println(stopwatch.elapsed(TimeUnit.SECONDS)); // 1

        // 停止计时
        stopwatch.stop();
        System.out.println(stopwatch.elapsed(TimeUnit.SECONDS)); // 1

        // 再次计时
        stopwatch.start();
        Thread.sleep(100);
        System.out.println(stopwatch.elapsed(TimeUnit.SECONDS)); // 2

        // 重置并开始
        stopwatch.reset().start();
        Thread.sleep(1030);

        // 检查是否运行
        System.out.println(stopwatch.isRunning()); // true
        long millis = stopwatch.elapsed(TimeUnit.MILLISECONDS); // 1034
        System.out.println(millis);

        // 打印
        System.out.println(stopwatch.toString()); // 1.034 s

    }
}
```

11.文件操作

以前我们写文件读取的时候要定义缓冲区，各种条件判断，各种 `$%#$@#`

而现在我们只需要使用好guava的api 就能使代码变得简洁，并且不用担心因为写错逻辑而背锅了

```
try {
        list = Files.readLines(file, Charsets.UTF_8);
} catch (Exception e) {
}

Files.copy(from,to);  //复制文件
Files.deleteDirectoryContents(File directory); //删除文件夹下的内容(包括文件与子文件夹)
Files.deleteRecursively(File file); //删除文件或者文件夹
Files.move(File from, File to); //移动文件
URL url = Resources.getResource("abc.xml"); //获取classpath根下的abc.xml文件url
```

12.guava缓存

==guava的缓存设计的比较巧妙，可以很精巧的使用。guava缓存创建分为两种，一种是CacheLoader,另一种则是callback方式==

CacheLoader:

java 复制代码

```
LoadingCache<String,String> cahceBuilder=CacheBuilder
                    .newBuilder()
                    .build(new CacheLoader<String, String>(){
                        @Override
                        public String load(String key) throws Exception {
                            String strProValue="hello "+key+"!";
                            return strProValue;
                        }
                    });
System.out.println(cahceBuilder.apply("begincode"));  //hello begincode!
System.out.println(cahceBuilder.get("begincode")); //hello begincode!
System.out.println(cahceBuilder.get("wen")); //hello wen!
System.out.println(cahceBuilder.apply("wen")); //hello wen!
System.out.println(cahceBuilder.apply("da"));//hello da!
cahceBuilder.put("begin", "code");
System.out.println(cahceBuilder.get("begin")); //code
```

api中已经把apply声明为过期，声明中推荐使用get方法获取值

关注公众号“程序员面试之道”

回复“面试”获取面试一整套大礼包！！！

本公众号分享自己从程序员小白到经历春招秋招斩获10几个offer的面试笔试经验，其中包括【Java】、【操作系统】、【计算机网络】、【设计模式】、【数据结构与算法】、【大厂面经】、【数据库】期待你加入！！！

1.计算机网络----三次握手四次挥手

2.梦想成真-----项目自我介绍

3.你们要的设计模式来了

4.震惊！来看《这份程序员面试手册》！！！

5.一字一句教你面试“个人简介”

6.接近30场面试分享

分类：　后端　　标签：　后端

评论

输入评论（Enter换行，Ctrl + Enter发送）

相关推荐

_晨曦_　3年前　Android

Java里的Boost——Google Guava官方教程中文版

1332　12　评论

前端要努力　6天前　前端　后端　面试

接了很多私活的感触

3.4w　329　200

4.1w 257 113

华为云开发者社区 1月前 JavaScript

掌握这20个JS技巧，做一个不加班的前端人

6.0w 1028 61

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

程序员阿牛 9月前 架构 后端

领导：谁再用定时任务实现关闭订单，立马滚蛋！

4.3w 347 134

l拉不拉米 4月前 后端 程序员

『2021年终总结』10年深飘，3辆车，3套房

3.6w 138 237

shaoqing 6月前 前端 Vue.js

【前车之鉴】Vue，你真的熟练了么?

2.6w 623 65

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

DHL 1年前 Android Jetpack

[Google] 再见 SharedPreferences 拥抱 Jetpack DataStore

1.9w 184 35

潜行前行 10月前 Java 后端

工具篇：介绍几个好用的guava工具类

2026 32 3

3.1w 208 45

why技术 3月前 前端 后端 Java

请问各位程序员，是我的思维方式有错误吗?

4.9w 253 234

捡田螺的小男孩 11月前 后端 Java

美团二面：Redis与MySQL双写一致性如何保证?

5.0w 599 125

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

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

谁主沉浮oo7 1年前 Java

使用Guava RateLimiter限流入门到深入

3569 7 1

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

Ethan01 2月前 前端 后端

面试题 -- 跨域请求如何携带cookie?

4.8w 633 76

yeyan1996 2年前 JavaScript

字节跳动面试官：请你实现一个大文件上传和断点续传

22.0w 5110 536