

FutureTask详解

原创

SL码路

2019-06-03 22:10:24

45917

收藏 93

版权

分类专栏:

并发编程

读书笔记


文章标签:

Future

FutureTask

FutureTask详解

Future和FutureTask

 并发编程 同时被 2 个专栏收录 ▾

0 订阅 7 篇文章

订阅专栏

FutureTask介绍

一个可取消的异步计算。FutureTask提供了对Future的基本实现，可以调用方法去开始和取消一个计算，可以查询计算是否完成并且获取计算结果。只有当计算完成时才能获取到计算结果，一旦计算完成，计算将不能被重启或者被取消，除非调用runAndReset方法。

除了实现了Future接口以外，FutureTask还实现了Runnable接口，因此FutureTask交由Executor执行，也可以直接用线程调用执行(futureTask.run())。根据FutureTask的run方法执行的时机，FutureTask可以处于以下三种执行状态：

- 1、未启动：在FutureTask.run()还没执行之前，FutureTask处于未启动状态。当创建一个FutureTask对象，并且run()方法未执行之前，FutureTask处于未启动状态。
- 2、已启动：FutureTask对象的run方法启动并执行的过程中，FutureTask处于已启动状态。
- 3、已完成：FutureTask正常执行结束，或者FutureTask执行被取消(FutureTask对象cancel方法)，或者FutureTask对象run方法执行抛出异常而导致中断而结束，FutureTask都处于已完成状态。

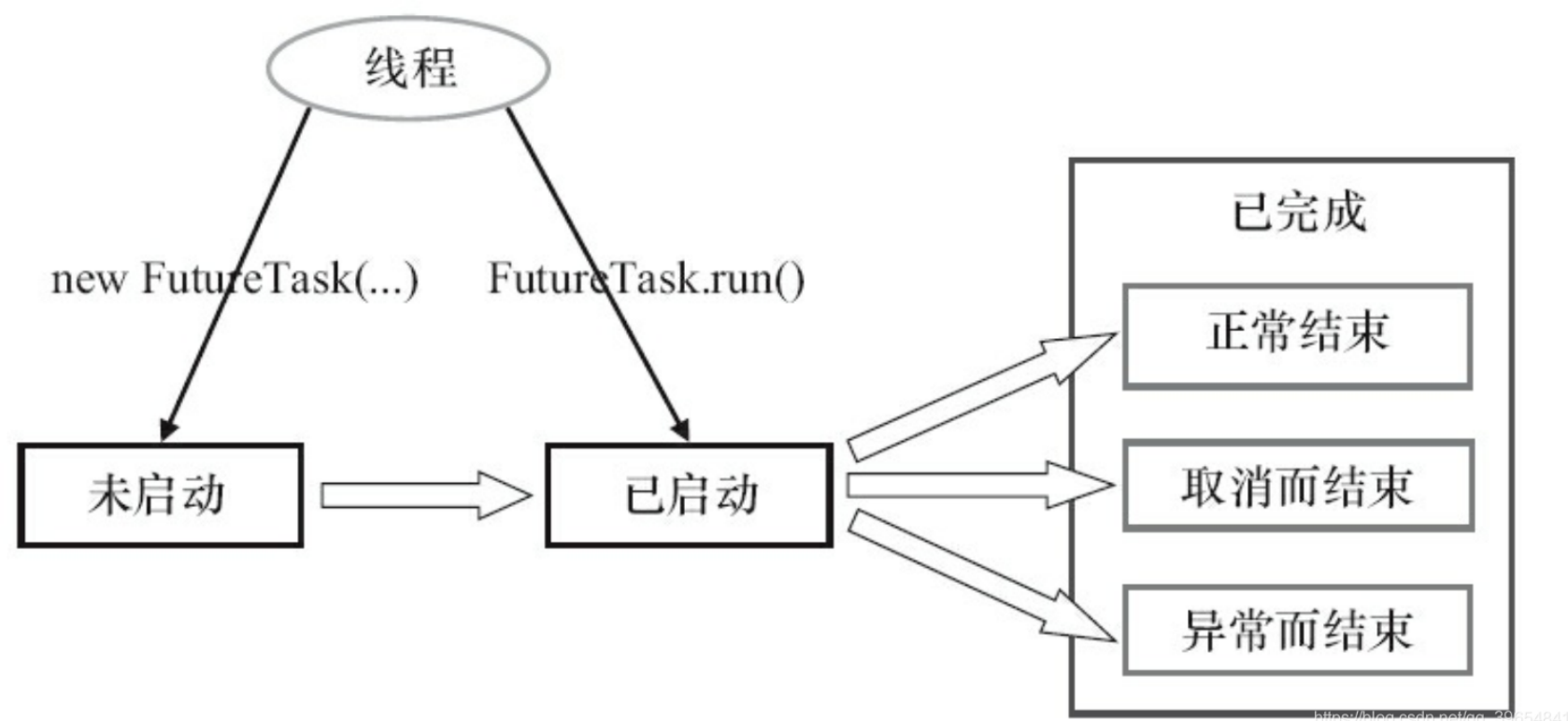
FutureTask状态迁移图

 **SL码路**

关注

26 2 93

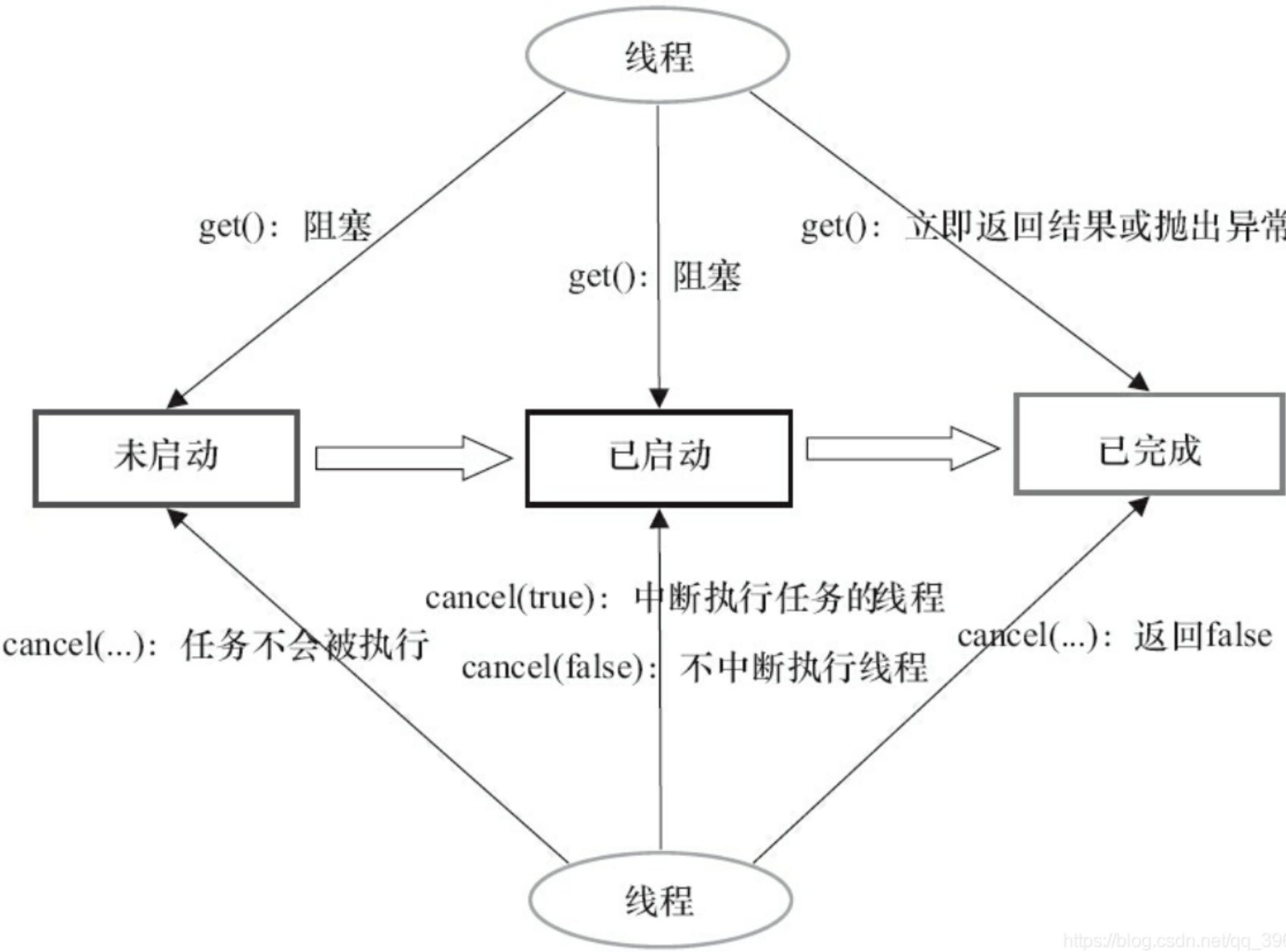
专栏目录



当FutureTask处于未启动或者已启动的状态时，调用FutureTask对象的get方法会将导致调用线程阻塞。当FutureTask处于已完成的状态时，调用FutureTask的get方法会立即放回调用结果或者抛出异常。

当FutureTask处于未启动状态时，调用FutureTask对象的cancel方法将导致线程永远不会被执行；当FutureTask处于已启动状态时，调用FutureTask对象cancel(true)方法将以中断执行此任务的线程的方式来试图停止此任务；当FutureTask处于已启动状态时，调用FutureTask对象cancel(false)方法将不会对正在进行的任务产生任何影响；当FutureTask处于已完成状态时，调用FutureTask对象cancel方法将返回false；

FutureTask的get和cancel的执行示意图



FutureTask使用

可以把FutureTask交给Executor执行；也可以通ExecutorService.submit (...) 方法返回一个FutureTask，然后执行FutureTask.get()方法或FutureTask.cancel (...) 方法。除此以外，还可以单独使用FutureTask。

当一个线程需要等待另一个线程把某个任务执行完后它才能继续执行，此时可以使用FutureTask。假设有多个线程执行若干任务，每个任务最多只能被执行一次。当多个线程试图同时执行同一个任务时，只允许一个线程执行任务，其他线程需要等待这个任务执行完后才能继续执行。下面是对应的示例代码。

```
1 private final ConcurrentMap<Object, Future<String>> taskCache = new ConcurrentHashMap<>();
2
3 private String executionTask(final String taskName)throws ExecutionException, InterruptedException {
4     while (true) {
5         Future<String> future =
```

SL码路

关注

26

2

93





专栏目录

```
6         if (future == null) {
7             Callable<String> task = () -> taskName;
8             FutureTask<String> futureTask = new FutureTask<>(task);
9             future = taskCache.putIfAbsent(taskName, futureTask); // 1.3
10            if (future == null) {
11                future = futureTask;
12                futureTask.run(); // 1.4执行任务
13            }
14        }
15        try {
16            return future.get(); // 1.5,
17        } catch (CancellationException e) {
18            taskCache.remove(taskName, future);
19        }
20    }
21 }
```



SL码路

关注



26



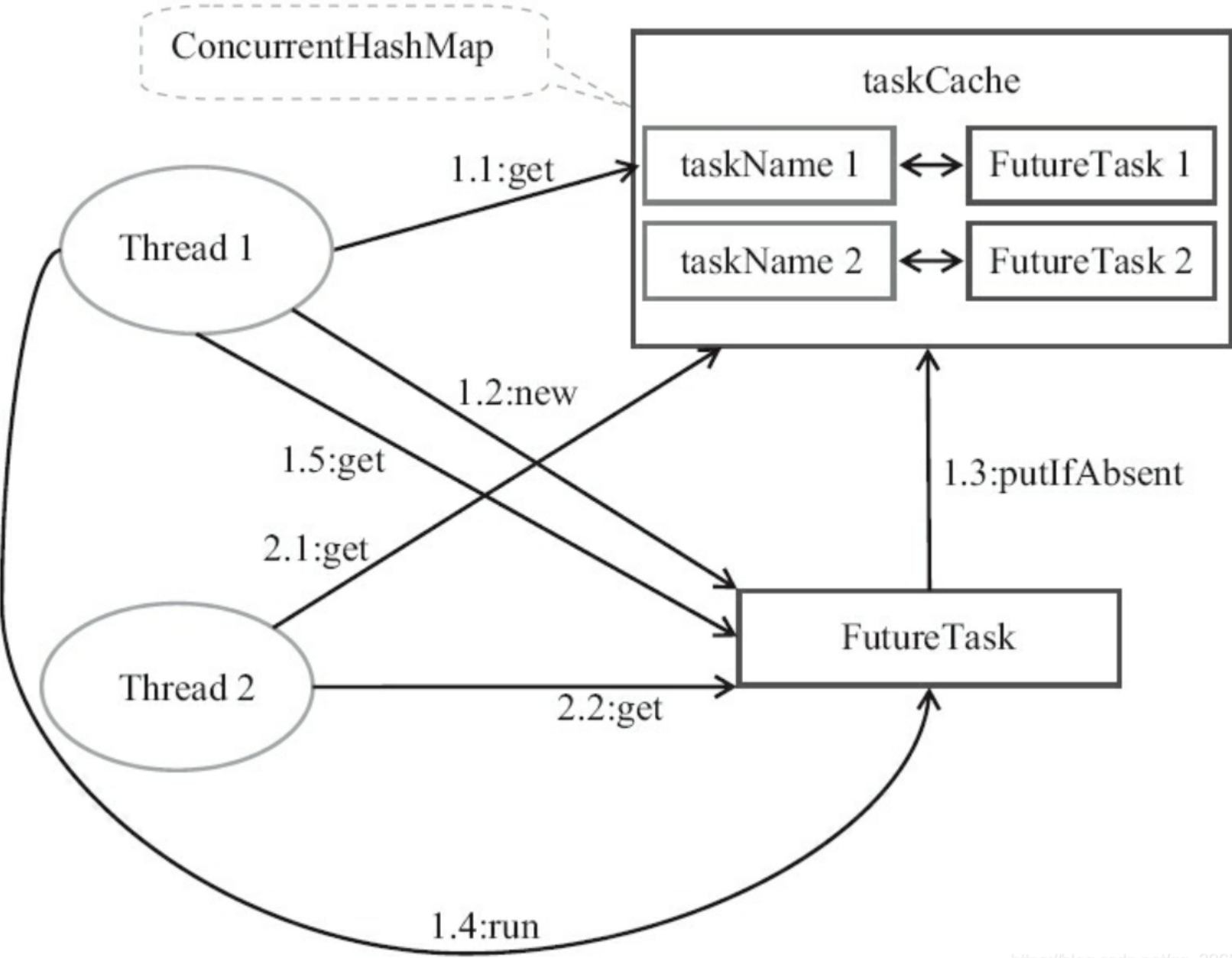
2



93



专栏目录



https://blog.csdn.net/qq_39654841

当两个线程试图同时执行同一个任务时，如果Thread 1执行1.3后Thread 2执行2.1，那么接下来Thread 2将在2.2等待，直到Thread 1执行完1.4后Thread 2才能从2.2（FutureTask.get()）返回

FutureTask实现

jdk1.8的FutureTask有个说明：
修订说明:这与这个类以前依赖AbstractQueuedSynchronizer的版本不同，主要是为了避免在取消竞争期间保留中断状态让用户感到意外。在当前的设计中，Sync控件依赖于通过CAS更新的“state”字段来跟踪完成，以及一个简单的Treiber堆栈来保存等待的线程。
风格注意:与往常一样，我们绕过了使用 AtomicXFieldUpdaters的开销，而是直接使用Unsafe。

Future

 SL码路

关注

 26

 2

 93





专栏目录

FutureTask实现了Future接口，Future接口有5个方法：

1、boolean cancel(boolean mayInterruptIfRunning)

尝试取消当前任务的执行。如果任务已经取消、已经完成或者其他原因不能取消，尝试将失败。如果任务还没有启动就调用了cancel(true)，任务将永远不会被执行。如果任务已经启动，参数mayInterruptIfRunning将决定任务是否应该中断执行该任务的线程，以尝试中断该任务。

如果任务不能被取消，通常是因为它已经正常完成，此时返回false，否则返回true

2、boolean isCancelled()

如果任务在正常结束之前被取消返回true

3、boolean isDone()

正常结束、异常或者被取消导致任务完成，将返回true

4、V get()

等待任务结束，然后获取结果，如果任务在等待过程中被终端将抛出InterruptedException，如果任务被取消将抛出CancellationException，如果任务中执行过程中发生异常将抛出ExecutionException。

5、V get(long timeout, TimeUnit unit)

任务最多在给定时间内完成并返回结果，如果没有在给定时间内完成任务将抛出TimeoutException。

FutureTask状态转换

FutureTask有以下7中状态：

```
private volatile int state;
private static final int NEW = 0;
private static final int COMPLETING = 1;
private static final int NORMAL = 2;
private static final int EXCEPTIONAL = 3;
private static final int CANCELLED = 4;
private static final int INTERRUPTING = 5;
private static final int INTERRUPTED = 6;
```

FutureTask任务的运行状态，最初为NEW。运行状态仅在set、setException和cancel方法中转换为终端状态。在完成过程中，状态可能呈现出瞬时值INTERRUPTING(仅在中断运行程序以满足cancel(true)的情况下)或者COMPLETING(在设置结果时)状态时。从这些中间状态到最终状态的转换使用成本更低的有序/延迟写，因为值是统一的，需要进一步修改。

state：表示当前任务的运行状态，FutureTask的所有方法都是围绕state开展的，state声明为volatile，保证了state的可见性，当对state进行修改时所有的线程都会看到。

NEW：表示一个新的任务，初始状态

COMPLETING：当任务被设置结果时，处于COMPLETING状态，这是一个中间状态。

NORMAL：表示任务正常结束。

EXCEPTIONAL：表示任务因异常而结束

CANCELLED：任务还未执行之前就调用了cancel(true)方法，任务处于CANCELLED

INTERRUPTING：当任务调用cancel(true)中断

SL码路

关注

26

2

93






专栏目录

INTERRUPTED：任务调用cancel(true)中断程序时会调用interrupt()方法中断线程运行，任务状态由INTERRUPTING转变为INTERRUPTED


- 1
- 可能的状态过渡：
- 2
- 1、NEW -> COMPLETING -> NORMAL：正常结束
- 3
- 2、NEW -> COMPLETING -> EXCEPTIONAL：异常结束
- 4
- 3、NEW -> CANCELLED：任务被取消
- 5
- 4、NEW -> INTERRUPTING -> INTERRUPTED：任务出现中断

futuretask java_futuretask用法及使用场景介绍 weixin_26911099的博客 225
FutureTask可用于异步获取执行结果或取消执行任务的场景。通过传入Runnable或者Callable的任务给FutureTask，直接调用其run方法或者...


FutureTask.get(timeOut)执行原理浅析 飞羽流觞 1814
使用java多线程解决问题的时候，为了提高效率，我们常常会异步处理一些计算任务并在最后异步的获取计算结果，这个过程的实现离不开...




优质评论可以帮助作者获得更高权重




评论




暂7师师长常乃超： java 并发编程的艺术。 3 月前 回复 ...





SL码路 博主 回复： 对，有参考 3 月前 回复 ...



相关推荐

更多相似内容

FutureTask使用及解析_tangedegushi的博客_futuretask使用 8-20
有时候我们可能会遇到这样的需要,线程执行完后需要返回运行结果,这个时候该怎么做呢?其实这时候就需要用到FutureTask了,先来看个dem...

Java线程池FutureTask实现原理详解 08-28
主要介绍了Java线程池FutureTask实现原理详解，小编觉得还是挺不错的，具有一定借鉴价值，需要的朋友可以参考下

线程池详解（包括Future和FutureTask） yancychas的博客 8988
Java 四种线程池的使用 https://juejin.im/post/59df0c1af265da432f301c8d 1，线程池的作用 线程池作用就是限制系统中执行线程的数量。 ...

三十一、并发编程之FutureTask详解 白夜行悟空 802
聊聊多线程 你真的了解并发吗 多线程和并发 多线程和多进程 线程一定快吗 学习并发的四个阶段 学习目标 适合人群 荐书 熟练掌握API，能...

Java并发--FutureTask详解 吴声子夜歌的博客 1800
FutureTask Future接口和实现Future接口的FutureTask类，代表异步计算的结果。 FutureTask简介 FutureTask除了实现Future接口外，还实...

FutureTask 轻描淡写 21
FutureTask 多线程执行任务时，有比较耗时操作，但又需要其返回结果时，可以使用FutureTask public class FutureTaskDemo { public stati...

Future和FutureTask的区别
先看下Future、FutureTask相关类的关系 Future只是一

 SL码路

关注

 26

 2

 93





专栏目录

- FutureTask 深度解析

FutureTask 深度解析

liulipuo的专栏 3万+
- 如何将java项目打成jar包通过命令行运行

1.选中项目后，右键export 2.选择java下的Runnable JAR file 这个打jar包的方式可以将第三方jar包也打进去 之后next 3.如图：第一...

z_victoria123的博客 2万+
- FutureTask 是什么？

简介 在 Java 中为了编程异步事件，我们使用 Thread 类和 Runnable 接口，它们可以开发并行应用程序。问题是在执行结束时不能返回值...

冯小石 4544
- FutureTask介绍及使用

Future Future是一个接口，它定义了5个方法： boolean cancel(boolean mayInterruptIfRunning); boolean isCancelled(); boolean isDone(); ...

闵浮龙的博客 2万+
- Java多线程之FutureTask的用法及解析

1 FutureTask概念FutureTask一个可取消的异步计算， FutureTask 实现了Future的基本方法，提空 start cancel 操作，可以查询计算是否已...

chenliguan的博客 3万+
- 多线程中Future与FutureTask的区别和联系

线程的创建方式中有两种，一种是实现Runnable接口，另一种是继承Thread，但是这两种方式都有个缺点，那就是在任务执行完成之后无...

LittleCadet 2万+
- Java多线程-FutureTask的get方法阻塞问题

FutureTask类中get方法阻塞的问题： get方法的实现： /** * @throws CancellationException {@inheritDoc} */ public V get() throws Interrupt...

Leetp 1万+
- (十一) J.U.C-FutureTask

FutureTask FutureTask是J.U.C中的类，是一个可删除的异步计算类。这个类提供了Future接口的基本实现，使用相关方法启动和取消计...

weixin_33709364的博客 70
- FutureTask的使用方法及实现原理

FutureTask用法介绍 FutureTask是JDK并发包为Future接口提供的一个实现，代表一个支持取消操作（cancel）的异步计算任务。它实现了...

zhuguang10的博客 1091
- FutureTask的用法及两种常用的使用场景

FutureTask可用于异步获取执行结果或取消执行任务的场景。通过传入Runnable或者Callable的任务给FutureTask，直接调用其run方法或者...

linchunquan的专栏 3万+
- FuttrueTask原理分析

通常一个请求分为请求-处理-返回，如果通过异步线程去完成一个任务，我们通常会选择FutureTask +Submit+ Callable()来实现获取返回值。

行而不辍，未来可期 - King 109



SL码路
码龄4年 暂无认证

44

25

25

245

SL码路

关注

26

2

93

专栏目录

https://blog.csdn.net/qq_39654841/article/details/90631795

8/10

1284

积分

25

粉丝

72

获赞

35

评论

262

收藏

















私信

关注

搜博主文章

🔍

- 热门文章
- springboot全局字符编码设置（解决乱码问题）

👁 108489
- FutureTask详解

👁 45891
- Windows平台及服务器部署安装多个Tomcat服务（详细版）

👁 27899
- springboot项目生成war包并部署到Tomcat服务器

👁 16735
- java 判断一个对象是否为空对象

👁 7961

- 分类专栏
- 

mybatis

6篇
- 

笔记

1篇
- 

spring全家桶

1篇
- 

spring MVC

1篇
- 

设计模式

3篇
- 

他山之石
- ▼

最新评论

springboot全局字符编码设置（解决乱码...

SL码路

关注

26

2

93





专栏目录

SL码路: 对, 有参考

FutureTask详解

暂7师师长常乃超: java 并发编程的艺术, .

使用 hibernate 进行 批量删除

SL码路: 不可以, 谢谢指出, 已经改了

使用 hibernate 进行 批量删除

sunxili520: delete Email where 可以吗 没有 from

您愿意向朋友推荐“博客详情页”吗？



强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章

java 设计模式之模板方法模式

mybatis源码探索之代理封装阶段

mybatis源码探索之SqlSession

| | |
|-----------|-----------|
| 2021年 7篇 | 2020年 10篇 |
| 2019年 13篇 | 2018年 10篇 |
| 2017年 2篇 | |

目录

FutureTask介绍

FutureTask使用

FutureTask实现

Future

FutureTask状态转换



SL码路

关注

26

2

93



专栏目录