Callable<V>、Future<V>、RunnableFuture<V>接口及FutureTask<V>类

# Callable<V>接口

## package

public interface **Callable<V>**

Callable<V>接口存在于**java.util.concurrent**包(**并发包**)中，有两个子接口：

All Known Subinterfaces:DocumentationTool.DocumentationTask, JavaCompiler.CompilationTask

JDK中没有**Callable<V>**接口的实现类。

@FunctionalInterface

public interface Callable<V>

Type Parameters: V - **the result type** of method call

Functional Interface:This is a functional interface and can therefore be used as the assignment target **for a lambda expression or method reference.**

## 功能

**A task that returns a result** and may throw an exception. Implementors define a single method with no arguments called **call**.(无参的call方法)

The **Callable** interface is similar to **Runnable**, in that both are designed for classes whose instances are potentially executed **by another thread**. A **Runnable**, however, does not return a result and cannot throw a checked exception.

Callable接口与Runnable接口相似，因为它们都是为了线程执行任务而设计的。但是Runnable接口中的**run()方法**不能返回值，而Callable接口中的**call()方法**可以返回指定的结果。**Runnable接口不能返回值，也不能抛出检查时异常**。

**The Executors class** contains utility methods to convert from other common forms to **Callable** classes.

## Callable接口方法：call()方法

**V call() throws Exception**

Computes a result, or throws an exception if unable to do so.

**Returns**: computed result

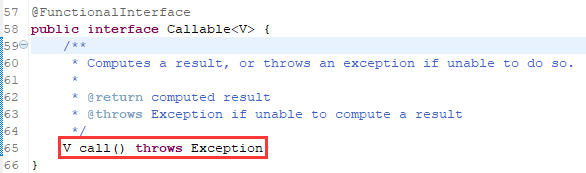
**Throws**: Exception - if unable to compute a result

**Runnable接口的run方法：**

**void run()** ： When an object implementing interface Runnable is used to create a thread, starting the thread causes the object's run method to be called in that separately executing thread.

## Callable<V>接口源码

Callable<V>接口只有一个无参数的可以返回参数的可抛出检查异常的**call()方法**。



# Callable<V>与Runnable接口的区别

## 相同点：二者十分相似，都是@FunctionalInterface功能接口，都是为了线程任务而设计的，都只定义了一个方法，Callable定义了call()方法，Runnable定义了run()方法；

都可以通过FutureTask获取任务的执行状态。

## 不同点：

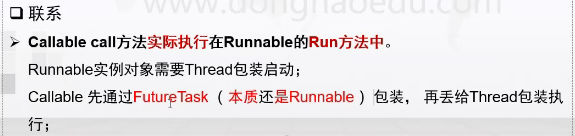
简单说：call方法有返回值且可以抛出异常；而run方法没有返回值也不能抛出异常。

Callable<V>定义的无参数的call()方法，可以返回结果，返回值是泛型，创建的时候传递进去，执行结束后返回，且该方法可以抛出检查时异常Checked Exception；而Runnable接口中定义的是无参数的run()方法，不能返回结果，也不能抛出检查时异常。

## 原理分析

执行Callable任务，需要先通过FutureTask进行包装，由于FutureTask实现了Runnable接口，所以FutureTask本质上还是Runnable，将包装了Callable的FutureTask传递给Thread包装执行。

Callable的call方法实际上执行在Runnable的run方法中。



## 创建线程的第3种方法

### 第1种:创建Target实现Runnable的run方法，传入Thread执行。

Runnable target = **new** Runnable() {  
 **public void** run() {  
 System.***out***.println(**" Hello Runnable! "**);  
 }  
};  
**new** Thread(target).start();

### 第2种：继承Thread类，覆写run方法

**new** Thread(){  
 **public void** run(){  
 System.***out***.println(**" 继承Thread创建线程"**);  
 }  
}.start();

### 第3种：与第1种类似，创建Task实现Callable接口的call方法，利用FutureTask包装Task，然后传入**Thread**执行，如下：

Callable task = **new** Callable() {  
 **public** Object call() **throws** Exception {  
 **return "Hello Callable!"**;  
 }  
};  
*//包装转换成****Runnable***FutureTask<String> futureTask = **new** FutureTask<String>(task);  
**new** Thread(futureTask).start();  
System.***out***.println(**"futureTask = "** + futureTask.get());

# Future <V>接口

## 继承关系

public interface Future<V> 存在于java.util.concurrent包中。

Type Parameters**:**V - The result type returned by this Future's get method

All Known Subinterfaces:Response<T>, RunnableFuture<V>, RunnableScheduledFuture<V>, ScheduledFuture<V>

All Known Implementing Classes: CompletableFuture, CountedCompleter, **ForkJoinTask**, FutureTask, RecursiveAction, **RecursiveTask**, SwingWorker

## 功能介绍

Future接口用于(代表)异步操作的结果。

(在Netty中有个对应的子接口： **io.netty.channel.ChannelFuture**用于获取异步操作的结果(在Netty中所有的I/O操作都是异步的).)

A **Future** represents the result **of an asynchronous computation**. Methods are provided ①*to check if the computation is complete*, ②*to wait for its completion*, and ③*to retrieve the result of the computation*. The result can only be retrieved using method **get** when the computation has completed, **blocking** if necessary until it is ready. Cancellation is performed by the **cancel** method. Additional methods are provided to determine if the task completed normally or was cancelled. Once a computation has completed, the computation cannot be cancelled. If you would like to use a Future *for the sake of* cancellability but not provide a usable result, you can declare types of the form **Future<?>** and return null as a result of the underlying task.

Future<V>提供了三种功能：

* 1. **能够获取任务执行结果；②能够中断(cancel)任务及判断是否中断；③判断任务是否执行完成**。

**Future接口及其子类就是用于对具体的Runnable或者Callable任务的执行进行取消(cancel)、查询是否完成(isDone)、获取执行结果(get)的三种操作。通过get方法获取执行结果，该方法会阻塞直到任务返回结果。**

Future<V>只是一个接口，无法直接用来创建对象使用，一般情况下需要使用FutureTask<V>，FutureTask<V>目前是Future<V>接口的一个唯一实现类。FutureTask<V>实现了RunnableFuture<V>接口，因此需要学习RunnableFuture<V>接口。

## 方法介绍

### get方法：*to retrieve the result of the computation*

**V get()**

Waits if necessary for the computation to complete, and then retrieves its result.

**V get(long timeout, TimeUnit unit)**

Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

### isDone(): to check if the computation is complete

boolean **isDone**() Returns true if this task completed.

### cancel：取消线程执行

boolean cancel(boolean mayInterruptIfRunning)

Attempts to cancel execution of this task.

### boolean isCancelled()：判断是否取消

Returns true if this task was cancelled before it completed normally.

## 使用示例Sample Usage

Sample Usage (Note that the following classes are all made-up.)

interface **ArchiveSearcher** { String search(String target); }

class App {

ExecutorService executor = ...

ArchiveSearcher searcher = ...

void showSearch(final String target) throws InterruptedException {

Future<String> future = executor.submit(**new Callable<String>() {**

**public String call() {**

**return searcher.search(target);**

**}}**);

displayOtherThings(); // do other things while searching

try {

displayText(future.get()); // use future

} catch (ExecutionException ex) { cleanup(); return; }

}

}

The **FutureTask** class is an implementation of Future that implements Runnable, and so may be executed by an Executor. For example, the above construction with submit could be replaced by:

FutureTask<String> future = new FutureTask<String>(**new Callable<String>() {**

**public String call() {**

**return searcher.search(target);**

**}}**);

executor.execute(future);

***Memory consistency effects***: Actions taken by the asynchronous computation happen-before actions following the corresponding **Future.get()** in another thread.

补充：ExecutorService的submit方法：

**<T> Future<T> submit(Callable<T> task)**

Submits a value-returning task for execution and returns a Future representing the pending results of the task.

**Future<?> submit(Runnable task)**

Submits a Runnable task for execution and returns a Future representing that task.

**<T> Future<T> submit(Runnable task, T result)**

Submits a Runnable task for execution and returns a **Future** representing that task.

## Future<V>接口源码

public interface **Future<V>** {

boolean cancel(boolean mayInterruptIfRunning);

boolean isCancelled();

boolean isDone();

V get() throws InterruptedException, ExecutionException;

V get(long timeout, TimeUnit unit)

throws InterruptedException, ExecutionException, TimeoutException;

}

# RunnableFuture<V>接口

## 继承关系

public interface **RunnableFuture<V>** extends **Runnable**, **Future<V>**

**位于java.util.concurrent包中。**

Type Parameters：V - The result type returned **by this Future's get method**

All Superinterfaces: **Future<V>, Runnable**

All Known Subinterfaces: **RunnableScheduledFuture**<V>

All Known Implementing Classes: **FutureTask**, SwingWorker

Since: 1.6

## 功能介绍

A **Future** that is **Runnable**. Successful execution of the **run** method causes completion of the Future and **allows access to its results**.

## 方法介绍

### 继承于Future<V>的方法

cancel, **get, get,** isCancelled, isDone

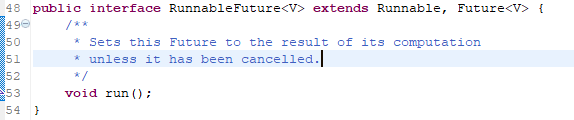
注意：不要搞混了，call方法属于Callable<V>接口的方法。

### 继承于Runnable的方法

**void run()**

Sets this Future to the result of its computation unless it has been cancelled.

## RunnableFuture<V>源码



# FutureTask<V>

## 继承关系

public class **FutureTask<V>** extends Object implements **RunnableFuture<V>**



Type Parameters: V - The result type returned by this **FutureTask**'s get methods

All Implemented Interfaces: **RunnableFuture<V>(继承了Runnable**和 **Future<V>)**

Since: 1.5

## 功能介绍

A **cancellable** asynchronous computation. This class provides a base implementation of Future, with methods to start and cancel a computation, query to see if the computation is complete, and retrieve the result of the computation. The result can only be retrieved when the computation has completed; the get methods will block if the computation has not yet completed. Once the computation has completed, the computation cannot be restarted or cancelled (unless the computation is invoked using runAndReset()).

**A FutureTask can be used to wrap a Callable or Runnable object**. Because **FutureTask** implements Runnable, *a* ***FutureTask*** *can be submitted to an Executor for execution*.

In addition to serving as a **standalone** class, this class provides protected functionality that may be useful when creating customized task classes.

RunnableFuture<V>接口继承了Runnable接口和Future<V>接口，而FutureTask<V>实现了RunnableFuture<V>接口，因此FutureTask<V>既可以作为Runnable被线程执行，又可以作为Future<V>得到Callable的返回值。(所以FutureTask<V>的构造方法的参数可以为Runnable也可以为Callable)

## 构造方法：包裹Callable<V>或Runnable

**FutureTask(Callable<V> callable)**

Creates a **FutureTask** that will, upon running, execute the given Callable.

**FutureTask(Runnable runnable, V result)**

Creates a **FutureTask** that will, upon running, execute the given Runnable, and ***arrange that get will return the given result on successful completion***.

## 一般方法

### 实现了RunnableFuture<V>接口，对该接口的方法进行了实现

void run()

Sets this **Future** to the result of its computation unless it has been cancelled.

**V get()**

**V get(long timeout, TimeUnit unit)**

**boolean isCancelled()**

**boolean isDone()**

**boolean cancel(boolean mayInterruptIfRunning)**

### 还有新的protected方法

protected boolean runAndReset()

Executes the computation without setting its result, and then resets this future to initial state, failing to do so if the computation encounters an exception or is cancelled.

protected void set(V v)

Sets the result of this future to the given value unless this future has already been set or has been cancelled.

protected void setException(Throwable t)

Causes this future to report an ExecutionException with the given throwable as its cause, unless this future has already been set or has been cancelled.

protected void **done**()

Protected method invoked when this task transitions to state isDone (whether normally or via cancellation).

# Fork/Join框架就是基于Future实现的。