#### 1、什么是Executor框架？

线程池就是线程的集合，线程池集中管理线程，以实现线程的重用，降低资源消耗，提高响应速度等。线程用于执行异步任务，单个的线程既是工作单元也是执行机制，从JDK 1.5开始，为了把工作单元与执行机制分离开，Executor框架诞生了，他是一个用于统一创建与运行的接口。Executor框架实现的就是线程池的功能。

#### 2、Executor框架结构图解

框架包含3大部分：

1. 任务。也就是工作单元，包括被执行任务需要实现的接口：Runnable接口或者Callable接口；
2. 任务的执行。也就是把任务分派给多个线程的执行机制，包括Executor接口及继承自Executor接口的ExecutorService接口。
3. 异步计算的结果。包括Future接口及实现了Future接口的FutureTask类。

框架成员关系图：



使用示意图：



#### 3、使用步骤

1. 创建Runnable并重写run()方法或者Callable对象并重写call()方法：

public class CallableTest implements Callable<String> {

@Override  
 public String call() throws Exception {  
 try {  
 String a = "return String";  
 return a;  
 } catch (Exception e) {  
 e.printStackTrace();  
 return "exception";  
 }  
 }  
}

1. 创建Executor接口的实现类ThreadPoolExecutor类或者ScheduledThreadPoolExecutor类的对象，然后调用其execute()方法或者submit()方法把工作任务添加到线程中，如果有返回值则返回Future对象。其中Callable对象有返回值，因此使用submit()方法；而Runnable可以使用execute()方法，此外还可以使用submit（）方法，只要使用callable（Runnable task）或者callable(Runnable task,  Object result)方法把Runnable对象包装起来就可以，使用callable（Runnable task）方法返回的null，使用callable(Runnable task,  Object result)方法返回result。

ThreadPoolExecutor threadPoolExecutor = new ThreadPoolExecutor(5, 10, 100,

TimeUnit.MILLISECONDS, new ArrayBlockingQueue<Runnable>(5));  
Future<String> future = threadPoolExecutor.submit(new CallableTest());

1. 调用Future对象的get()方法后的返回值，或者调用Future对象的cancel()方法取消当前线程的执行。最后关闭线程池。

try {

System.*out*.println(future.get());  
} catch (Exception e) {  
 e.printStackTrace();  
} finally {  
 threadPoolExecutor.shutdown();  
}

#### 3、Executor框架成员：ThreadPoolExecutor实现类、ScheduledThreadPoolExecutor实现类、Future接口、Runnable和Callable接口、Executors工厂类



1. ThreadPoolExecutor实现类：[点击打开链接](https://blog.csdn.net/tongdanping/article/details/79625109)
2. ScheduledThreadPoolExecutor实现类：[点击打开链接](https://blog.csdn.net/tongdanping/article/details/79627491)
3. Future接口/FutureTask实现类：[点击打开链接](https://blog.csdn.net/tongdanping/article/details/79630637)
4. Runnable和Callable接口：用于实现线程要执行的工作单元。
5. Executors工厂类：提供了常见配置线程池的方法，因为ThreadPoolExecutor的参数众多且意义重大，为了避免配置出错，才有了Executors工厂类。