1、newFixedThreadPool()方法

可以通过调用Executors类的static newFixedThreadPool()方法获得一个固定线程池。

语法

```
ExecutorService fixedPool = Executors.newFixedThreadPool(2);
Java
```

其中，

- 最多2个线程将处于活动状态。
- 如果提交了两个以上的线程，那么它们将保持在队列中，直到线程可用。
- 如果一个线程由于执行关闭期间的失败而终止，则执行器尚未被调用，则创建一个新线程。
- 线程会一直存在，直到池关闭。

2、newCachedThreadPool()方法

通过调用Executors类的静态newCachedThreadPool()方法可以获得缓存的线程池。

语法

```
ExecutorService executor = Executors.newCachedThreadPool();
Java
```

其中，

- newCachedThreadPool()方法创建一个具有可扩展线程池的执行器。
- 这样的执行者适合于发起许多短命的任务的应用程序。

3、newScheduledThreadPool()方法

通过调用Executors类的static newScheduledThreadPool()方法获得一个调度的线程池。

语法

```
ExecutorService executor = Executors.newScheduledThreadPool(1);
```

4、newSingleThreadExecutor()方法

通过调用Executors类的static newSingleThreadExecutor()方法获得单个线程池。

语法

```
ExecutorService executor = Executors.newSingleThreadExecutor();
```

newSingleThreadExecutor()方法创建一次执行单个任务的执行程序。

5、ThreadPoolExecutor类

java.util.concurrent.ThreadPoolExecutor是一个ExecutorService，可以使用可能的几个池线程来执行每个提交的任务，通常使用Executors工厂方法进行配置。 它还提供了各种实用方法来检查当前线程统计信息并进行控制。

6、ScheduledThreadPoolExecutor类

java.util.concurrent.ScheduledThreadPoolExecutor是ThreadPoolExecutor的子类，并且可以额外地调度在给定延迟之后运行的命令，或定期执行。