线程池 Executors

如果并发的线程数量很多，并且每个线程都是执行一个时间很短的任务就结束了，这样频繁创建线程就会大大降低 系统的效率，因为频繁创建线程和销毁线程需要时间. 线程池就是一个容纳多个线程的容器，池中的线程可以反复使用，省去了频繁创建线程对象的操作，节省了大量的时间和资源。

线程池的好处

- 降低资源消耗。
- 提高响应速度。
- 提高线程的可管理性。

Java中的四种线程池 . ExecutorService

1. 缓存线程池

```
/**
 * 缓存线程池.
 * (长度无限制)
 * 执行流程:
 *      1.  判断线程池是否存在空闲线程
 *      2.  存在则使用
 *      3.  不存在,则创建线程 并放入线程池, 然后使用
 */
ExecutorService service = Executors.newCachedThreadPool();
//向线程池中 加入 新的任务
service.execute(new Runnable() {
    @Override
    public void run() {
        System.out.println("线程的名称:"+Thread.currentThread().getName());
    }
});
service.execute(new Runnable() {
    @Override
    public void run() {
        System.out.println("线程的名称:"+Thread.currentThread().getName());
    }
});
service.execute(new Runnable() {
    @Override
    public void run() {
        System.out.println("线程的名称:"+Thread.currentThread().getName());
    }
});
```

2. 定长线程池

```
/**
 * 定长线程池.
 * (长度是指定的数值)
 * 执行流程:
```

```
     *      1.  判断线程池是否存在空闲线程
     *      2.  存在则使用
     *      3.  不存在空闲线程,且线程池未满的情况下,则创建线程 并放入线程池, 然后使用
     *      4.  不存在空闲线程,且线程池已满的情况下,则等待线程池存在空闲线程
     */
    ExecutorService service = Executors.newFixedThreadPool(2);
    service.execute(new Runnable() {
        @Override
        public void run() {
            System.out.println("线程的名称:"+Thread.currentThread().getName());
        }
    });
    service.execute(new Runnable() {
        @Override
        public void run() {
            System.out.println("线程的名称:"+Thread.currentThread().getName());
        }
    });
```

3. 单线程线程池

效果与定长线程池 创建时传入数值1 效果一致.

```
    /**
     * 单线程线程池.
     * 执行流程:
     *      1.  判断线程池 的那个线程 是否空闲
     *      2.  空闲则使用
     *      4.  不空闲,则等待 池中的单个线程空闲后 使用
     */
    ExecutorService service = Executors.newSingleThreadExecutor();
    service.execute(new Runnable() {
        @Override
        public void run() {
            System.out.println("线程的名称:"+Thread.currentThread().getName());
        }
    });
    service.execute(new Runnable() {
        @Override
        public void run() {
            System.out.println("线程的名称:"+Thread.currentThread().getName());
        }
    });
```

4. 周期性任务定长线程池

```
public static void main(String[] args) {
    /**
     * 周期任务 定长线程池.
     * 执行流程:
     *      1.  判断线程池是否存在空闲线程
     *      2.  存在则使用
     *      3.  不存在空闲线程,且线程池未满的情况下,则创建线程 并放入线程池, 然后使用
     *      4.  不存在空闲线程,且线程池已满的情况下,则等待线程池存在空闲线程
     *
     * 周期性任务执行时:
     *      定时执行, 当某个时机触发时, 自动执行某任务 .
```

```
     */

    ScheduledExecutorService service = Executors.newScheduledThreadPool(2);

    /**
     * 定时执行
     *  参数1.    runnable类型的任务
     *  参数2.    时长数字
     *  参数3.    时长数字的单位
     */
    /*service.schedule(new Runnable() {

        @Override
        public void run() {
            System.out.println("俩人相视一笑~ 嘿嘿嘿");
        }
    },5,TimeUnit.SECONDS);
    */
    /**
     * 周期执行
     *  参数1.    runnable类型的任务
     *  参数2.    时长数字(延迟执行的时长)
     *  参数3.   周期时长(每次执行的间隔时间)
     *  参数4.    时长数字的单位
     */
    service.scheduleAtFixedRate(new Runnable() {

        @Override
        public void run() {
            System.out.println("俩人相视一笑~ 嘿嘿嘿");
        }
    },5,2,TimeUnit.SECONDS);

}
```