Java 核心技术(进阶)

第五章 Java多线程和并发编程

第九节 Java定时任务执行

华东师范大学 陈良育

定时任务(1)

- Thread/Executor/Fork-Join 多线程
  - 立刻执行
  - 框架调度
- 定时执行
  - 固定某一个时间点运行
  - 以某一个周期

定时任务(2)

- 简单定时器机制
  - 设置计划任务，也就是在指定的时间开始执行某一个任务。
  - TimerTask 封装任务
  - Timer类 定时器
- 参看例子

定时任务(3)

- Executor＋定时器机制
- ScheduledExecutorService
  - 定时任务
  - 周期任务
- 参看例子

定时任务(4)

- Quartz
  - Quartz是一个较为完善的任务调度框架
  - 解决程序中Timer零散管理的问题
  - 功能更加强大
    - Timer执行周期任务，如果中间某一次有异常，整个任务终止执行
    - Quartz执行周期任务，如果中间某一次有异常，不影响下次任务执行
    - ……
- 参看例子Quartz

总结

- 了解定时任务的作用
- 根据业务特点，使用合适的定时任务器

代码(1) TimerTest.java


```
public class TimerTest {
    public static void main(String[] args) throws InterruptedException {
            MyTask task = new MyTask();
            Timer timer = new Timer();

            System.out.println("当前时间："+new Date().toLocaleString());
            //当前时间1秒后，每2秒执行一次
            timer.schedule(task, 1000, 2000);

            Thread.sleep(10000);
            task.cancel();   //取消当前的任务

            System.out.println("==================================");

            Calendar now = Calendar.getInstance();
            now.set(Calendar.SECOND,now.get(Calendar.SECOND)+3);
            Date runDate = now.getTime();
            MyTask2 task2 = new MyTask2();
            timer.scheduleAtFixedRate(task2,runDate,3000); //固定速率


            Thread.sleep(20000);
            timer.cancel();   //取消定时器
    }
}
```

代码(2) MyTask.java & MyTask2.java


```
class MyTask extends TimerTask {
    public void run() {
        System.out.println("运行了！时间为：" + new Date());
    }
}

class MyTask2 extends TimerTask {
    public void run() {
        System.out.println("运行了！时间为：" + new Date());
        try {
            Thread.sleep(4000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

代码(3) ScheduledExecutorTest.java


```
public class ScheduledExecutorTest {

    public static void main(String[] a) throws Exception
    {
        //executeAtFixTime();
        //executeFixedRate();  //3s
        executeFixedDelay();  //4s
    }

    public static void executeAtFixTime() throws Exception {
        ScheduledExecutorService executor = Executors.newScheduledThreadPool(1);
        executor.schedule(
                new MyTask(),
                1,
                TimeUnit.SECONDS);

        Thread.sleep(20000);
        executor.shutdown();
    }
```

代码(4) ScheduledExecutorTest.java


```
/**
 * 周期任务 固定速率 是以上一个任务开始的时间计时，period时间过去后，检测上一个任务是否执行完毕，
 * 如果上一个任务执行完毕，则当前任务立即执行，如果上一个任务没有执行完毕，则需要等上一个任务执行完毕后立即执行。
 * @throws Exception
 */
public static void executeFixedRate() throws Exception {
    ScheduledExecutorService executor = Executors.newScheduledThreadPool(1);
    executor.scheduleAtFixedRate(
            new MyTask(),
            1,
            3000,
            TimeUnit.MILLISECONDS);

    Thread.sleep(20000);
    executor.shutdown();
}
```

代码(5) ScheduledExecutorTest.java


```
    /**
     * 周期任务 固定延时 是以上一个任务结束时开始计时，period时间过去后，立即执行。
     * @throws Exception
     */
    public static void executeFixedDelay() throws Exception {
        ScheduledExecutorService executor = Executors.newScheduledThreadPool(1);
        executor.scheduleWithFixedDelay(
                new MyTask(),
                1,
                3000,
                TimeUnit.MILLISECONDS);

        Thread.sleep(20000);
        executor.shutdown();
    }
}
```

代码(6) MyTask.java


```
class MyTask implements Runnable {
    public void run() {
        System.out.println("时间为：" + new Date());
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        //System.out.println("时间为：" + new Date());
    }
}
```

代码(7) QuartzTest.java


```
public class QuartzTest {

    public static void main(String[] args) {
        try {
            //创建scheduler
            Scheduler scheduler = StdSchedulerFactory.getDefaultScheduler();

            //定义一个Trigger
            Trigger trigger = newTrigger().withIdentity("trigger1", "group1") //定义name/group
                    .startNow()//一旦加入scheduler，立即生效
                    .withSchedule(simpleSchedule() //使用SimpleTrigger
                            .withIntervalInSeconds(2) //每隔2秒执行一次
                            .repeatForever()) //一直执行
                    .build();
```

代码(8) QuartzTest.java


```
            //定义一个JobDetail
            JobDetail job = newJob(HelloJob.class) //定义Job类为HelloQuartz类
                    .withIdentity("job1", "group1") //定义name/group
                    .usingJobData("name", "quartz") //定义属性
                    .build();

            //加入这个调度
            scheduler.scheduleJob(job, trigger);

            //启动
            scheduler.start();

            //运行一段时间后关闭
            Thread.sleep(10000);
            scheduler.shutdown(true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

代码(9) HelloJob.java


```
public class HelloJob implements Job {
    public void execute(JobExecutionContext context) throws JobExecutionException {
        JobDetail detail = context.getJobDetail();
        String name = detail.getJobDataMap().getString("name");
        System.out.println("hello from " + name + " at " + new Date());
    }
}
```

谢 谢！