- **线程池如何创建?**

ThreadExecutor 4中构造函数 7个参数

- **线程池如何接收并执行一个或者多个任务?**

任务封装在一个Worker类

```
private final class Worker extends AbstractQueuedSynchronizer implements Runnable {
    Worker(Runnable firstTask) {
            setState(-1); // inhibit interrupts until runWorker
            this.firstTask = firstTask;
            this.thread = getThreadFactory().newThread(this);
    }

    public void run() {
            runWorker(this);
    }
}
```

通过构造函数可以知道，创建Worker对象就会创建一个线程，但是线程传递的参数是this，即该Worker对象，会不会感到很奇怪？！为什么不直接把firstTask传递给新创建的线程呢？！这是一个很关键的点，如果如我们所愿，firstTask任务直接传递给新线程，那么当firstTask执行完之后，该线程如何获取队列中的任务呢？或许你有更好的方法，但这里采用下面这种思想：把去等待队列中获取任务的过程封装成一个Runnable任务(就是该Worker对象)，新创建的线程启动后，就会执行该Runnable任务，该Runnable任务执行完firstTask任务后，就会不断的从等待队列中获取任务，直到等待队列为空，该线程才会销毁

- **线程池中的线程如何创建？何时创建？存活到何时?**

创建流程 workCnt、coreThreadNum、MaxThreadNum

线程池内的存活时间也跟任务数有关；如果线程一直处于工作状态，那么一直存活；如果有池内线程数大于核心线程数且有空闲线程，那么空闲线程等待keepAliveTime之后进行销毁；如果池内线程数小于核心线程数且有空闲线程，线程不会销毁，除非设置了allowCoreThreadTimeOut属性(true/false)，其值默认为false，核心线程不会销毁，若设置为true，则空闲核心线程维持keepAliveTime时间后销毁。

- **线程池中的线程间是如何调度的？即调度机制是什么?**

据线程池的状态以及当前线程数(workerCount)判断是否需要新增一个worker，还是直接放入等待队列，还是执行拒绝策略。

- **线程池如何存放多余任务?**

队列

Direct handoffs 等待队列(workQueue)

Unbounded queues.使用无界队列(例如没有预定义容量的LinkedBlockingQueue)

Bounded queues.当使用有限的maximumPoolSize时，有界队列(例如ArrayBlockingQueue)

- **线程池如何销毁？何时销毁？**

线程池状态

运行(RUNNING)：该状态下的线程池接收新任务并处理队列中的任务；线程池创建完毕就处于该状态，也就是正常状态；

关机(SHUTDOWN)：线程池不接受新任务，但处理队列中的任务；线程池调用shutdown()之后的池状态；

停止(STOP)：线程池不接受新任务，也不处理队列中的任务，并中断正在执行的任务；线程池调用shutdownNow()之后的池状态；

清理(TIDYING)：线程池所有任务已经终止，workCount(当前线程数)为0；过渡到清理状态的线程将运行terminated()钩子方法；

终止(TERMINATED)：terminated()方法结束后的线程池状态；

线程池的五种状态，从侧面也反应出线程池的生命周期，线程池某刻的状态用一个AtomicInteger变量ctl来表示，变量ctl可以说明线程池的两个属性：工作线程数(workerCount)和运行状态(runState)；线程池一旦创建则workerCount默认为0，runState为RUNNING。

参考 https://blog.csdn.net/nobody_1/article/details/98305882

技术架构 -实现原理

分布式xxl-job 实现原理

rabbit mq 原理