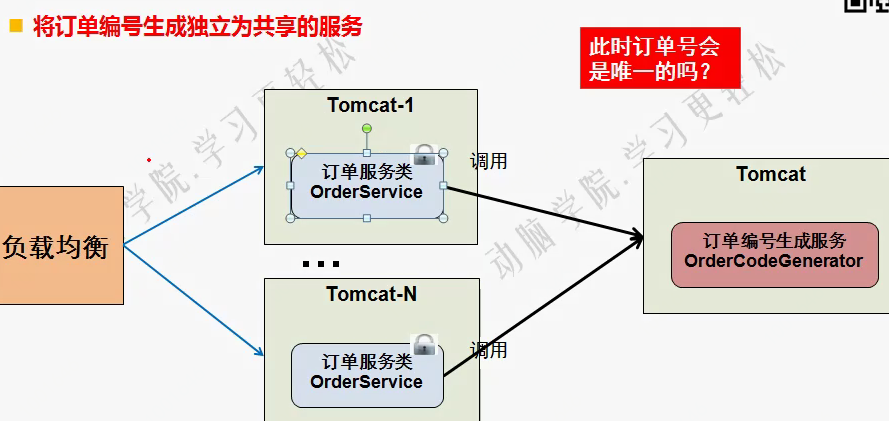
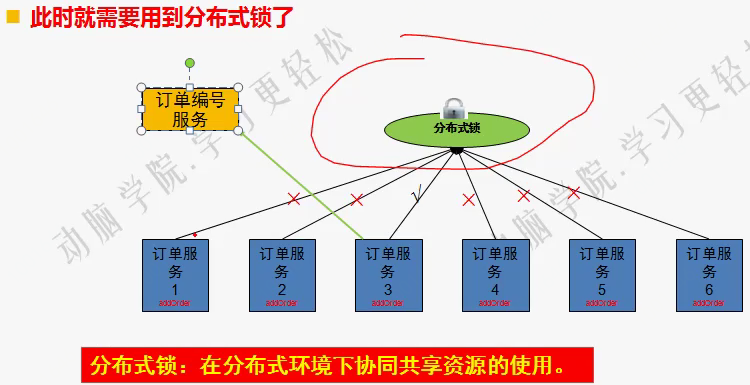
分布式锁笔记

# 为什么使用分布式锁？



每个Tomcat中紧紧只是自己JDK提供的锁，只能控制自身的并发访问。

**分布式锁**：在**分布式环境下**协同共享资源的使用。



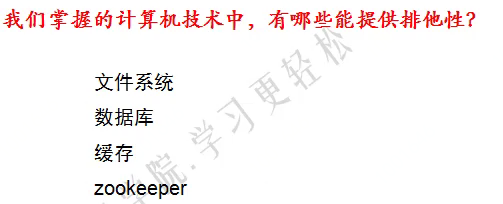
# 分布式锁的实现方式有哪些？

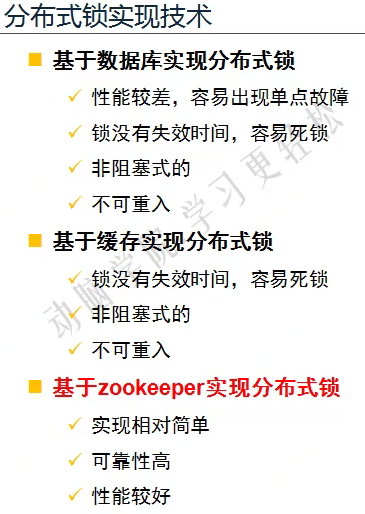
## 锁具有什么特点？

独占性（排他性）：只有一个线程能获取到；

阻塞性：其他未抢到的线程阻塞，直到锁释放出来，再抢。

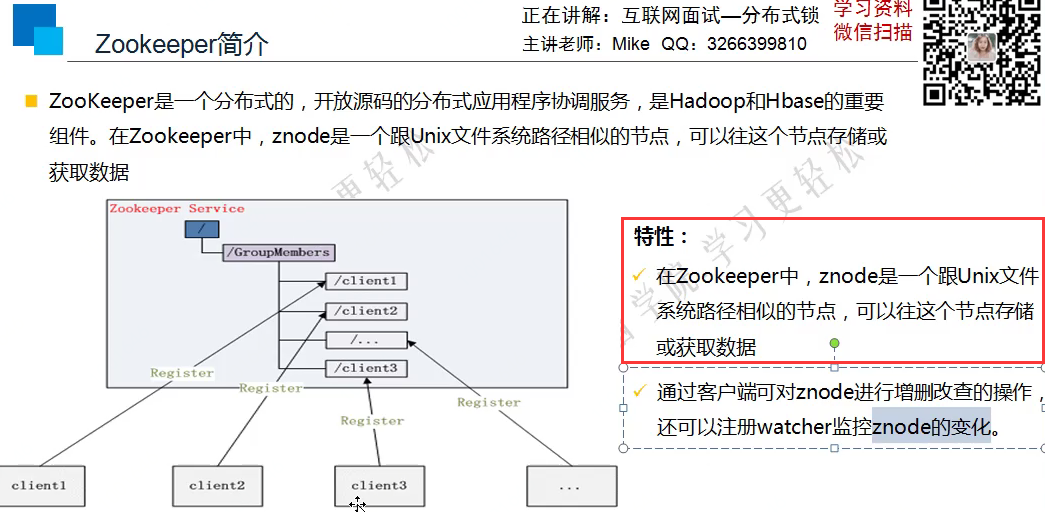
## 排他性





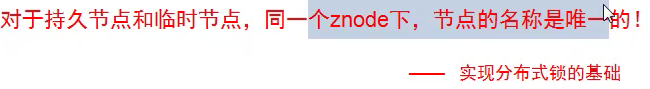
# 基于zookeeper的分布式锁

## Zookeeper简介



## Zookeeper的节点类型





## Zookeeper的典型应用场景



## 利用Zookeeper实现分布式锁

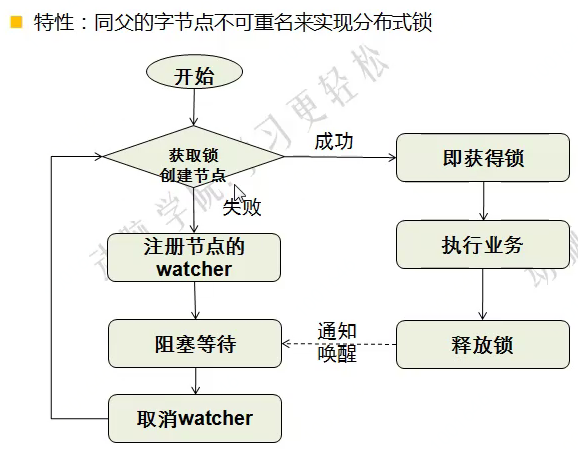
依据特性：**同父的子节点不可重名**；

**利用Zookeeper实现分布式锁原理：**

**依据：同父的子节点不可重名；**

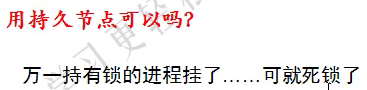
**实现Lock接口，覆写tryLock、lock、unLock即可。**

**创建一个Zookeeper客户端，指定一个节点，在tryLock中进行添加节点操作，若成功能，即获取锁，否则等待。如何等待并被唤醒呢？利用CountDownLatch倒数器锁存器，设置值为1，调用其await方法阻塞；在Zookeeper的Watcher中，收到删除节点通知之后，就调用CountDownLatch的countDown方法，唤醒阻塞的线程。**

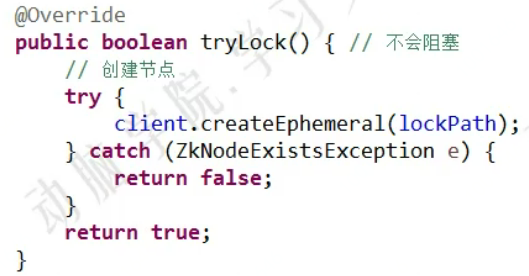


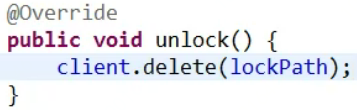
lock与tryLock的区别：

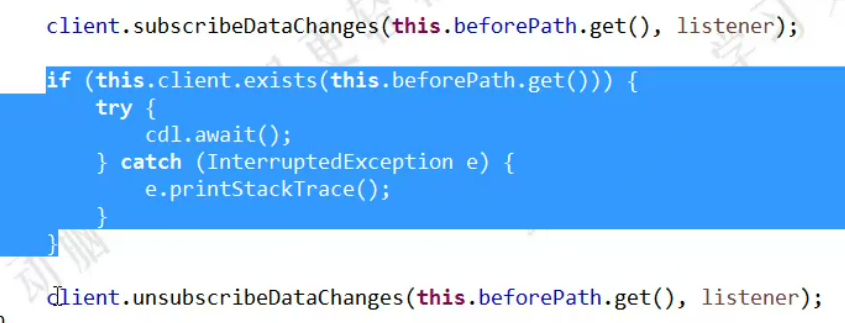
lock会阻塞，tryLock不会阻塞。tryLock能获取到，就获取，否则直接返回。



用**临时节点**。







通过Zookeeper的Watcher唤醒阻塞的线程：



## 惊群效应：大规模集群。

比如10000只鸽子，只投递了一个食物，把10000个鸽子都惊动了。

解决惊群效应的策略：

**利用临时顺序节点实现**。**只会通知后一个节点，不会全部通知**。

**原理**：100号节点只需要注册99号节点的Watcher；2号只需要注册1号的Watcher；

这样1号删除节点，只会通知2号节点。

