# 第一章 初识netty

* netty是一个提供了易于使用的API的客户端/服务器框架
* 并发高-NIO(非阻塞IO)
* 传输快-零拷贝

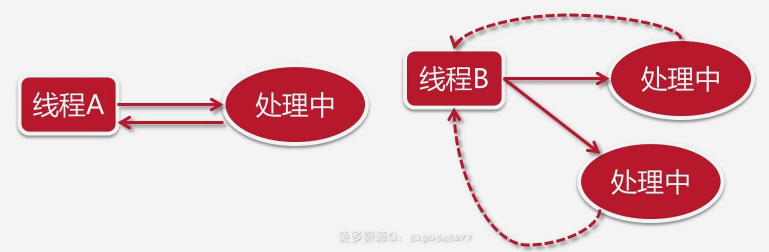
## 阻塞与非阻塞

* 线程访问资源，该资源是否准备就绪的一种处理方式



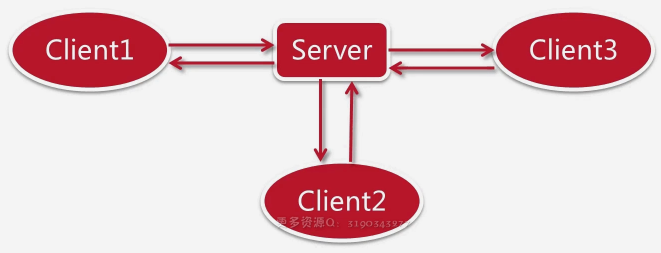
## 同步和异步

* 同步和异步是指访问数据的一种机制



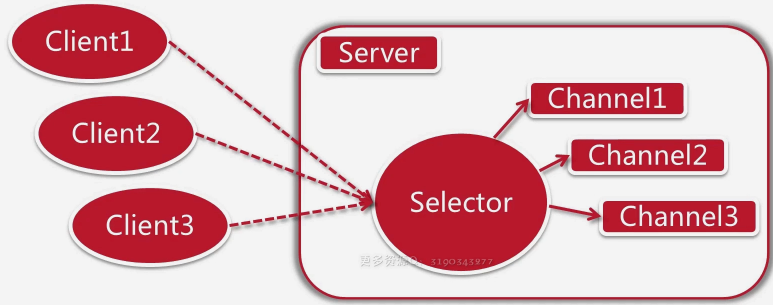
## BIO(同步阻塞IO)

* 同步阻塞IO，Block IO，并发能力低，耗时久，依赖网速和带宽



## NIO(同步非阻塞IO)

* 同步非阻塞IO，New IO（Non-Block IO）



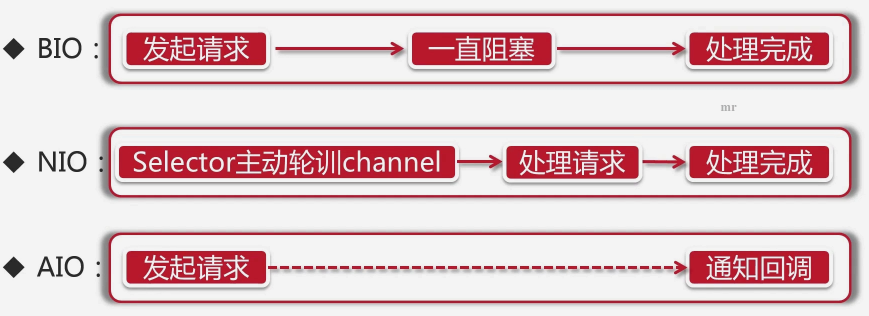
## AIO

* 异步非阻塞IO(在某个地方外做自己的事，等到别人主动通知了，再去做之前的事)
* 异步阻塞IO(在某个地方干等着，需要别人主动通知，很少用)

## 生活实例

* BIO：去餐馆吃饭，都坐满了，此时就一直光等着，主动观测那一桌空了，一旦有空闲桌，就立马去占座位
* NIO：餐馆都坐满了，此时先去外面坐着玩手机或者做别的事，然后时不时主动去观察有没有空桌子，如果有就去占
* 异步阻塞：在餐馆外面等着，啥都不做，等到有空桌子了，老板来通知了，再去占位
* 异步非阻塞(AIO)：在餐馆外面玩手机或者其他事，等到老板说有空桌子了，通知去占位

空桌子就是上面的一个channel。



## 常见面时

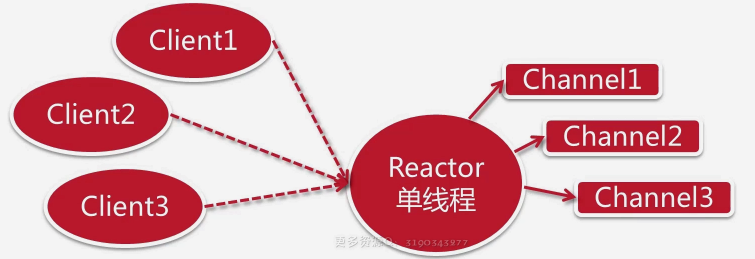
* BIO、NIO、AIO的区别是什么？
* 什么是同步阻塞BIO
* 什么是同步非阻塞NIO
* 什么是异步非阻塞AIO

# 第二章 Netty

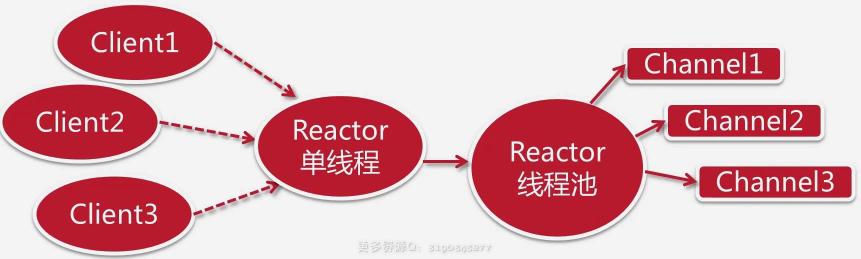
* 概述
* 使用NIO所遇到的问题
* 使用Netty的优点

## Reactor线程模型

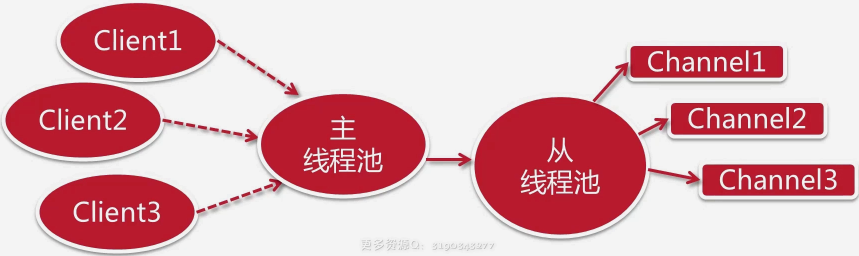
* 单线程模型：所有的IO操作都由同一个NIO线程处理的



* 多线程模型：由一组NIO线程处理IO操作



* 主从线程模型：一组线程池接收请求，一组线程池处理IO

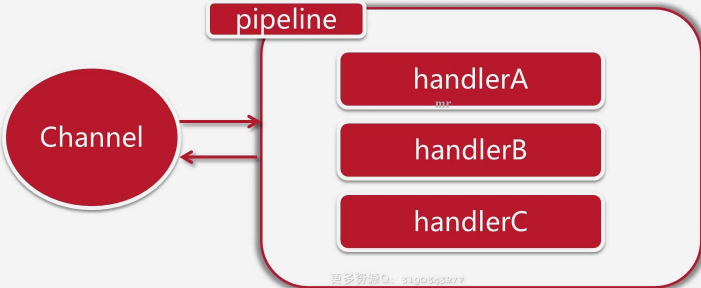


# 第三章 Hello Netty服务器

* 构建一对主从线程组
* 定义服务器启动类
* 为服务器设置Channel
* 设置处理从线程池的助手类初始化器
* 监听启动和关闭服务器

## 设置channel初始化器

* 每一个channel由多个handler共同组成管道（pipeline）



## 实时通信

* Ajax轮训
* Long pull
* Websocket

## Websocket api

* Var socket = new WebSocket("ws://[ip]:[port]”)
* 生命周期:onopen()、onmessage()、onerror()、onclose()
* 主动方法:Socket.send()、Socket.close()