# 什么是netty

Netty是由JBOSS提供的一个java开源网络通讯框架。Netty可以提供异步的，非阻塞的，事件驱动的网络应用程序框架和工具，非常适合用来快速开发高性能、高可靠性协议的网络服务器和客户端程序。

# Netty中的常用概念

## Bootstrap

Bootstrap可以说是netty客户端启动引导类。通过这个类的实例对象，可以设置相关启动参数和配置。

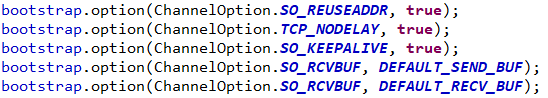
1、设置传输流channel模式



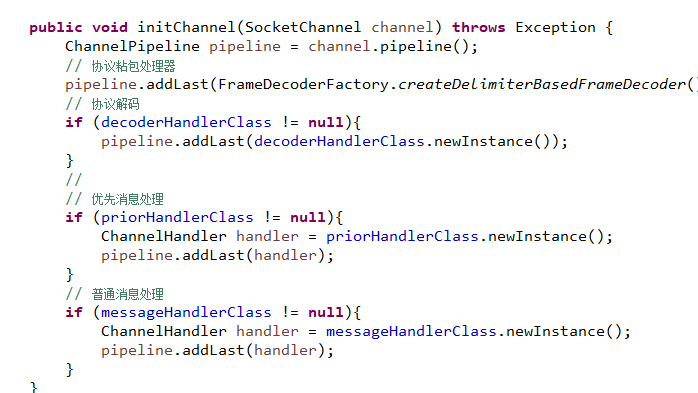
2、设置工作线程组



3、设置socket相关参数选项

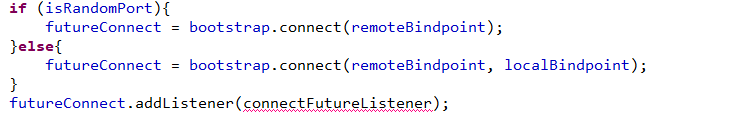


4、设置相关channelhandler的初始化器，包括拆包器、解码器、消息处理器等





5、连接服务端



## ChannelOption

主要用于配置socket相关参数：

1、ChannelOption.SO\_BACKLOG

ChannelOption.SO\_BACKLOG对应的是tcp/ip协议listen函数中的backlog参数，函数listen(int socketfd,int backlog)用来初始化服务端可连接队列，服务端处理客户端连接请求是顺序处理的，所以同一时间只能处理一个客户端连接，多个客户端来的时候，服务端将不能处理的客户端连接请求放在队列中等待处理，backlog参数指定了队列的大小

2、ChannelOption.SO\_REUSEADDR

ChanneOption.SO\_REUSEADDR对应于套接字选项中的SO\_REUSEADDR，这个参数表示允许重复使用本地地址和端口，比如，某个服务器进程占用了TCP的80端口进行监听，此时再次监听该端口就会返回错误，使用该参数就可以解决问题，该参数允许共用该端口，这个在服务器程序中比较常使用，比如某个进程非正常退出，该程序占用的端口可能要被占用一段时间才能允许其他进程使用，而且程序死掉以后，内核一需要一定的时间才能够释放此端口，不设置SO\_REUSEADDR就无法正常使用该端口。

3、ChannelOption.SO\_KEEPALIVE

Channeloption.SO\_KEEPALIVE参数对应于套接字选项中的SO\_KEEPALIVE，该参数用于设置TCP连接，当设置该选项以后，连接会测试链接的状态，这个选项用于可能长时间没有数据交流的连接。当设置该选项以后，如果在两小时内没有数据的通信时，TCP会自动发送一个活动探测数据报文。

4、ChannelOption.SO\_SNDBUF和ChannelOption.SO\_RCVBUF

ChannelOption.SO\_SNDBUF参数对应于套接字选项中的SO\_SNDBUF，ChannelOption.SO\_RCVBUF参数对应于套接字选项中的SO\_RCVBUF这两个参数用于操作接收缓冲区和发送缓冲区的大小，接收缓冲区用于保存网络协议站内收到的数据，直到应用程序读取成功，发送缓冲区用于保存发送数据，直到发送成功。

5、ChannelOption.SO\_LINGER

ChannelOption.SO\_LINGER参数对应于套接字选项中的SO\_LINGER,Linux内核默认的处理方式是当用户调用close（）方法的时候，函数返回，在可能的情况下，尽量发送数据，不一定保证会发生剩余的数据，造成了数据的不确定性，使用SO\_LINGER可以阻塞close()的调用时间，直到数据完全发送

6、ChannelOption.TCP\_NODELAY

ChannelOption.TCP\_NODELAY参数对应于套接字选项中的TCP\_NODELAY,该参数的使用与Nagle算法有关Nagle算法是将小的数据包组装为更大的帧然后进行发送，而不是输入一次发送一次,因此在数据包不足的时候会等待其他数据的到了，组装成大的数据包进行发送，虽然该方式有效提高网络的有效负载，但是却造成了延时，而该参数的作用就是禁止使用Nagle算法，使用于小数据即时传输，于TCP\_NODELAY相对应的是TCP\_CORK，该选项是需要等到发送的数据量最大的时候，一次性发送数据，适用于文件传输。

## NioEventLoopGroup

是用来处理I/O操作的多线程事件循环器，包括两大类，一类是用于接收（acceptor）客户端发送过来的连接的主线程组（bossGroup），一类是用于处理（reactor&handler）已经接收的连接的工作线程组（workerGroup）。

# Netty和传统socket比对

## 并发高

Netty是一款基于NIO（Nonblocking I/O，非阻塞IO）开发的网络通信框架，对比于BIO（Blocking I/O，阻塞IO），他的并发性能得到了很大提高。

在BIO中，等待客户端发数据这个过程是阻塞的，这样就造成了一个线程只能处理一个请求的情况，而机器能支持的最大线程数是有限的，这就是为什么BIO不能支持高并发的原因。而NIO中，当一个Socket建立好之后，Thread并不会阻塞去接受这个Socket，而是将这个请求交给Selector，Selector会不断的去遍历所有的Socket，一旦有一个Socket建立完成，他会通知Thread，然后Thread处理完数据再返回给客户端——这个过程是阻塞的，这样就能让一个Thread处理更多的请求了。

## 传输快

Netty的传输快其实也是依赖了NIO的一个特性——零拷贝。我们知道，Java的内存有堆内存、栈内存和字符串常量池等等，其中堆内存是占用内存空间最大的一块，也是Java对象存放的地方，一般我们的数据如果需要从IO读取到堆内存，中间需要经过Socket缓冲区，也就是说一个数据会被拷贝两次才能到达他的的终点，如果数据量大，就会造成不必要的资源浪费。

Netty针对这种情况，使用了NIO中的另一大特性——零拷贝，当他需要接收数据的时候，他会在堆内存之外开辟一块内存，数据就直接从IO读到了那块内存中去，在netty里面通过ByteBuf可以直接对这些数据进行直接操作，从而加快了传输速度。

Netty支持的zero-copy有2种：

1、包装FileChannel.tranferTo方法实现zero-copy

 Netty中通过在FileRegion中包装了NIO的FileChannel.transferTo()方法实现了零拷贝。

 FileRegion是一个接口，默认的实现类是：DefaultFileRegion

2、Netty的CompositeChannelBuffer类允许我们将多段数据合并为一整段虚拟数据供用户使用，而过程中不需要对数据进行拷贝操作Netty允许我们将多段数据合并为一整段虚拟数据供用户使用，而过程中不需要对数据进行拷贝操作。

## 封装好，使用简单

Netty提供了统一的API编程接口，只要通过Bootstrap引导类实例对象设置相关参数并实现所需要的channelhandler接口，最后调用connect或者bind方法进行端口监听和客户端连接。只需实现相关接口并调整几行代码就可以启动。同时netty还提供了相应的拆包器和解码器，可以结合实际情况使用，加快代码开发。