小册总结

> Netty 相关的知识点到这里就告一段落了，最后，我们再用专门一节来对我们这本小册做一下总结回顾。

1. Netty 是什么?

经过我们整个小册的学习，我们可以了解到，Netty 其实可以看做是对 BIO 和 NIO 的封装，并提供良好的 IO 读写相关的 API，另外还提供了非常多的开箱即用的 handler，工具类等等。

2. 服务端和客户端启动

Netty 提供了两大启动辅助类，ServerBootstrap 和 Bootstrap， 他们的启动参数类似，都是分为

1. 配置 IO 类型，配置线程模型。
2. 配置 TCP 参数，attr 属性。
3. 配置 handler。server 端除了配置 handler，还需要配置 childHandler，他是定义每条连接的处理器。

3. ByteBuf

接着，我们又学习了 Netty 对二进制数据的抽象类 ByteBuf，ByteBuf 底层又可以细分为堆内存和堆外内存，它的 API 要比 jdk 提供的 ByteBuffer 要更好用，ByteBuf 所有的操作其实都是基于读指针和写指针来进行操作的，把申请到的一块内存划分为可读区、可写区，另外还提供了自动扩容的功能。

4. 自定义协议拆包与编解码

通常，我们要实现客户端与服务端的通信，需要自定义协议，说白了就是双方商量在字节流里面，对应位置的字节段分别表示什么含义。

我们用的最多的协议呢就是基于长度的协议，一个协议数据包里面包含了一个长度字段，我们在解析的时候，首先第一步就是从字节流里面根据自定义协议截取出一个个数据包，使用的最多的拆包器就是 LengthFieldBasedFrameDecoder，只需要给他配置一些参数，即可实现自动拆包。

拆包之后呢，我们就拿到了代表字节流区段的一个个 ByteBuf，我们的解码器的作用就是把这些个 ByteBuf 变成一个个 java 对象，这样我们后续的 handler 就可以进行相应的逻辑的处理。

5. handler 与 pipeline

Netty 对逻辑处理流的处理其实和 TCP 协议栈的思路非常类似，分为输入和输出，也就是 inBound 和 outBound 类型的 handler，inBound 类 handler 的添加顺序与事件传播的顺序相同，而 outBound 类 handler 的添加顺序与事件传播的顺序相反，这里一定要注意。

无状态的 handler 可以改造为单例模式，但是千万记得要加 @ChannelHandler.Sharable 注解，平行等价的 handler 可以使用压缩的方式减少事件传播路径，调用 ctx.xxx() 而不是 ctx.channel().xxx() 也可以减少事件传播路径，不过要看应用场景。

另外，每个 handler 都有自己的生命周期，Netty 会在 channel 或者 channelHandler 处于不同状态的情况下回调相应的方法，channelHandler 也可以动态添加，特别适用于一次性处理的 handler，用完即删除，干干净净。

6. 耗时操作的处理与统计

对于耗时的操作，不要直接在 NIO 线程里做，比如，不要在 channelRead0() 方法里做一些访问数据库或者网络相关的逻辑，要扔到自定义线程池里面去做，然后要注意这个时候，writeAndFlush() 的执行是异步的，需要通过添加监听回调的方式来判断是否执行完毕，进而进行延时的统计。

关于 Netty 的知识点大概就这么多，如果你读完这小节，觉得已经很轻松，那么恭喜你，Netty 大部分的知识点你已经掌握了，接下来就可以进阶学习了。

另外，如果后续笔者发现本小册有遗漏的知识点，也会陆续补充到本小册中，感谢坚持到最后的小伙伴，一定要反复地把本小册的最终代码亲自多写几遍哦！

7. 总结 & 思考

该总结的总结完了，最后大家可以在留言区留言，谈谈本小册给你工作中可以带来的帮助，或者有哪些地方还需要学习的，不明白的，都可以告诉我，等共性问题多了之后，第二本小册的素材也就有了。

Netty 入门门槛高，其实是因为这方面的资料太少了，并不是因为他有多难，大家其实都可以像搞透 Spring 一样搞透 Netty，搞不透的话呢，最后一小节，我也会再给大家补充点进阶资料，这些资料也都是经过广大网友认证过的，质量还是可以保证的。

如果大家觉得小册还行的话呢，也可以把本小册分享给你的朋友，据说，现在分享可以赚佣金了哦，每一位同学购买，两瓶可乐就到手了呢。

最后，祝大家学习愉快，早日升职加薪，有缘的话，咱们第二本小册再见！