面向对象程序设计大作业

——源码阅读：Netty

宋嘉程 2018K8009937001

1.简单回顾

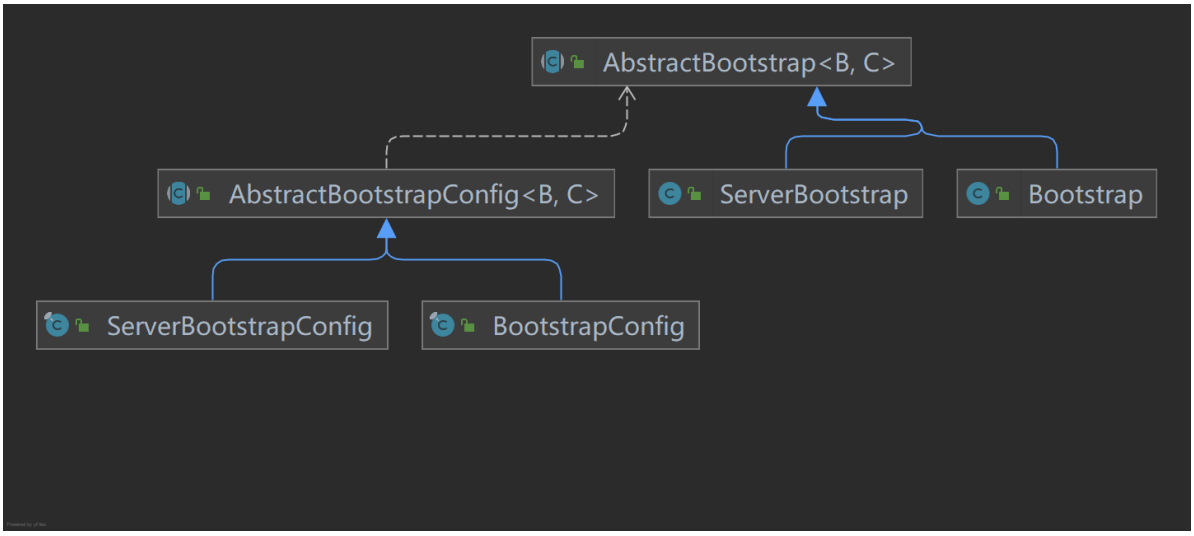
前文主要介绍了netty的概况，特点，应用，并对netty的主要模块和架构进行了解析。下面我们主要将注意力放在对transport模块的分析上。

2.类间关系

2.1 Bootstrap

Bootstrap 是“引导”的意思，它主要负责整个 Netty 程序的启动、初始化、服务器连接等过程，它相当于一条主线，串联了 Netty 的其他核心组件。Bootstrap类是Netty提供的一个便利的工厂类，可以通过它来完成Netty的客户端或服务端的Netty组件的组装，以及Netty程序的初始化和启动执行。Netty的官方解释是，理论上完全可以不用这个Bootstrap类，可以一点点去手动创建通道、完成各种设置和启动注册到EventLoop反应器，然后开始事件的轮询和处理，但是这个过程会非常麻烦。通常情况下，使用这个便利的Bootstrap工具类的效率会高很多。正如netty官方文档中对其的解释”The helper classes with fluent API which enable an easy implementation of typical client side and server side channel initialization.“

下图给出了Bootstrap包中的类间关系：



> 图一：Bootstrap 类间关系

如上图所示，Netty 中的引导器共分为两大类：一个为用于客户端引导的 Bootstrap，另一个为用于服务端引导的 ServerBootStrap，它们都继承自抽象类 AbstractBootstrap，父类AbstractBootstrap提供了通用功能。另外，netty还设计了客户端的BootstrapConfig与服务器端的ServerBootstrapConfig两个类，两者都继承自抽象类AbstractBootstrapConfig。而AbstractBootstrapConfig又依赖于AbstractBootstrap。

其中，AbstractBootstrap是个抽象类，它实现Cloneable接口。另外声明B``C两个泛型。

- B：继承AbstractBootstrap类，用于表示自身的类型。

- C：继承Channel类，表示创建的Channel类型。

另外，对于Bootstrap 和 ServerBootStrap，两者十分相似，都提供通过链式编程风格（method-chaining）初始化一些核心变量，如group、channelFactory、options、attrs、handler等。两者非常重要的区别在于 Bootstrap 可用于连接远端服务器，只绑定一个 EventLoopGroup。而 ServerBootStrap 则用于服务端启动绑定本地端口，会绑定两个 EventLoopGroup，这两个 EventLoopGroup 通常称为 Boss 和 Worker。通俗的讲，这里的 Boss 和 Worker 可以理解为“老板”和“员工”的关系。每个服务器中都会有一个 Boss，也会有一群做事情的 Worker。Boss 会不停地接收新的连接，然后将连接分配给一个个 Worker 处理连接。

有了 Bootstrap 组件，我们可以更加方便地配置和启动 Netty 应用程序，它是整个 Netty 的入口，串接了 Netty 所有核心组件的初始化工作。

2.2 Channel

Channel 的字面意思是“通道”，它是网络通信的载体。Channel提供了基本的 API 用于网络 I/O 操作，如register、bind、connect、read、write、flush 等。Netty 自己实现的 Channel 是以 JDK NIO Channel 为基础的，相比较于 JDK NIO，Netty 的 Channel 提供了更高层次的抽象，同时屏蔽了底层 Socket 的复杂性，赋予了 Channel 更加强大的功能，使得用户再使用 Netty 时基本不需要再与 Java Socket 类直接打交道。

下图给出了Channel包中所涉及到得类与类间关系，可以看到Channel中各种类的关系错综复杂：

> 图二：Channel包中的类与类间关系

整个族群中，AbstractChannel 是最为关键的一个抽象类，从它继承出了AbstractNioChannel、、AbstractEpollChannel、LocalChannel、EmbeddedChannel等类（AbstractOioChannel：@deprecated），每个类代表了不同的协议以及相应的IO模型。除了 TCP 协议以外，Netty 还支持很多其他的连接协议，并且每种协议还有 NIO(异步 IO) 和 OIO(Old-IO，即传统的阻塞 IO) 版本的区别. 不同协议不同的阻塞类型的连接都有不同的 Channel 类型与之对应。下面是一些常用的 Channel 类型：

- NioSocketChannel：代表异步的客户端 TCP Socket 连接
- NioServerSocketChannel：异步的服务器端 TCP Socket 连接
- NioDatagramChannel：异步的 UDP 连接
- NioSctpChannel：异步的客户端 Sctp 连接
- NioSctpServerChannel：异步的 Sctp 服务器端连接

当然 Channel 会有多种状态，如连接建立、连接注册、数据读写、连接销毁等。随着状态的变化，Channel 处于不同的生命周期，每一种状态都会绑定相应的事件回调，下面的表格我列举了 Channel 最常见的状态所对应的事件回调。

事件	说明
channelRegistered	Channel 创建后被注册到 EventLoop 上
channelUnregistered	Channel 创建后未注册或者从 EventLoop 取消注册
channelActive	Channel 处于就绪状态，可以被读写
channelInactive	Channel 处于非就绪状态
channelRead	Channel 可以从远端读取到数据
channelReadComplete	Channel 读取数据完成

3.小结

在本节中，主要对transport模块中的Bootstrap以及Channel做了简要的分析。其中，BootStrap 和 ServerBootStrap 分别负责客户端和服务端的启动，它们是非常强大的辅助工具类；Channel 是网络通信的载体，提供了与底层 Socket 交互的能力。下一节中我们将继续分析netty中高级设计意图。