# 面向对象程序设计大作业

# ——源码阅读: Netty

# 宋嘉程 2018K8009937001

# 1.Netty的简介及主要功能

### 1.1简介

来自Netty官方醒目"头版"的介绍:

Netty is *an asynchronous event-driven network application framework* for rapid development of maintainable high performance protocol servers & clients.

Netty是 一个异步事件驱动的网络应用程序框架,用于快速开发可维护的高性能协议服务器和客户端。

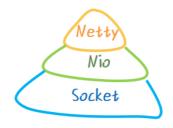
#### 1.2 Netty的特点

既然是网络应用程序框架,那首先就要先看一看传统的NIO有什么主要问题:

- NIO的类库和API繁杂,学习成本高,你需要熟练掌握Selector、ServerSocketChannel、SocketChannel、ByteBuffer等。
- 需要熟悉Java多线程编程。这是因为NIO编程涉及到Reactor模式,你必须对多线程和网络编程非常 熟悉,才能写出高质量的NIO程序。
- ......

### 而相比较而言, Netty的优点有很多:

- API使用简单,学习成本低。
- 性能高,对比其他主流的NIO框架,Netty的性能最优。
- 功能强大,内置了多种解码编码器,支持多种协议。
- 社区活跃,广泛应用......



图一: Netty与NIO, Socket的关系

#### 1.3 Netty的应用

在 Java 领域 Netty 运用非常地广泛,Tomcat、Dubbo、RocketMQ、Zookeeper、Spark、Flink、ElasticSearch 等等这些中间件的网络通讯框架都是基于 Netty 去实现的。

# 2.Netty包含的主要模块

#### 2.1 逻辑结构

	Transport Services	Protocol Support			
	Socket & Datagram	HTTP & WebSocket	SSL · StartTLS	Google Protobuf	
	HTTP Tunnel	zlib/gzip Compression	Large File Transfer	RTSP	
	In-VM Pipe	Legacy Text · Binary Protocols with Unit Testability			
	Extensible Event Model				
Core	Universal Communication API				Core
	Zero-Copy-Capable Rich Byte Buffer				

图二: Netty的逻辑架构图

如上图所示,是Netty官网上给出的Netty架构模型。从图中,我们可以清晰地看出 Netty 结构一共分为三个模块:

### • Core 核心层

Core 核心层是 Netty 最精华的内容,它提供了底层网络通信的通用抽象和实现,包括可扩展的事件模型、通用的通信 API、支持零拷贝的 ByteBuf 等。

### • Protocol Support 协议支持层

协议支持层基本上覆盖了主流协议的编解码实现,如 HTTP、SSL、Protobuf、压缩、大文件传输、WebSocket、文本、二进制等主流协议,此外 Netty 还支持自定义应用层协议。Netty 丰富的协议支持降低了用户的开发成本,基于 Netty 我们可以快速开发 HTTP、WebSocket 等服务。

# • Transport Service 传输服务层

传输服务层提供了网络传输能力的定义和实现方法。它支持 Socket、HTTP 隧道、虚拟机管道等传输方式。Netty 对 TCP、UDP 等数据传输做了抽象和封装,用户可以更聚焦在业务逻辑实现上,而不必关系底层数据传输的细节。

Netty 的模块设计具备较高的**通用性和可扩展性**,它不仅是一个优秀的网络框架,还可以作为网络编程的工具箱。Netty 的设计理念非常优雅,值得我们学习借鉴。

#### 2.2 Netty的源码结构

#### **Transport Services**

#### **Protocol Support**

io.netty.transport
io.netty.transport.epoll
io.netty.transport.kqueue
io.netty.transport.unix.common
io.netty.transport.sctp
io.netty.transport.rxtx
io.netty.transport.udt

io.netty.codec
io.netty.codec.dns
io.netty.codec.haproxy
io.netty.codec.http
io.netty.codec.http2
io.netty.codec.memcache
io.netty.codec.mqtt
io.netty.codec.redis
io.netty.codec.smtp
io.netty.codec.socks
io.netty.codec.stomp
io.netty.codec.xml

io.netty.handler.proxy

io.netty.buffer io.netty.common io.netty.resolver io.netty.resolver.dns

Core

图三: Netty的源码结构

Netty 源码分为多个模块,模块之间职责划分非常清楚。如同上文逻辑结构中的划分一样,Netty 源码模块的划分也是基本契合的。

## • Core 核心层模块

**netty-common**模块是 Netty 的核心基础包,提供了丰富的工具类,其他模块都需要依赖它。在 common 模块中,常用的包括**通用工具类**和自定义并发包。

在netty-buffer 模块中Netty自己实现了的一个更加完备的ByteBuf 工具类,用于网络通信中的数据载体。由于人性化的 Buffer API 设计,它已经成为 Java ByteBuffer 的完美替代品。ByteBuf 的动态性设计不仅解决了 ByteBuffer 长度固定造成的内存浪费问题,而且更安全地更改了 Buffer 的容量。此外 Netty 针对 ByteBuf 做了很多优化,例如缓存池化、减少数据拷贝的 CompositeByteBuf 等。

**netty-resolver**模块主要提供了一些有关**基础设施**的解析工具,包括 IP Address、Hostname、DNS 等。

#### • Protocal Support 协议支持层模块

**netty-codec**模块主要负责编解码工作,通过编解码实现原始字节数据与业务实体对象之间的相互转化。如下图所示,Netty 支持了大多数业界主流协议的编解码器,如 HTTP、HTTP2、Redis、XML 等,为开发者节省了大量的精力。此外该模块提供了抽象的编解码类ByteToMessageDecoder 和 MessageToByteEncoder,通过继承这两个类我们可以轻松实现自定义的编解码逻辑。

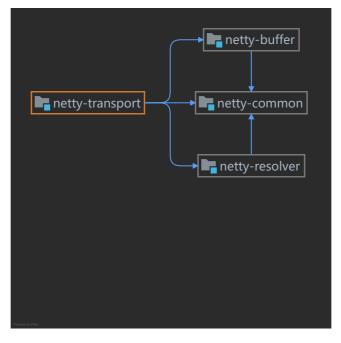
codec [netty-codec]
codec-dns [netty-codec-dns]
codec-haproxy [netty-codec-haproxy]
codec-http [netty-codec-http]
codec-http2 [netty-codec-http2]
codec-memcache [netty-codec-memcache]
codec-mqtt [netty-codec-mqtt]
codec-redis [netty-codec-redis]
codec-smtp [netty-codec-smtp]
codec-socks [netty-codec-socks]
codec-stomp [netty-codec-stomp]
codec-xml [netty-codec-xml]

图四: Netty-codec的部分模块

**netty-handler**模块主要负责数据处理工作。Netty 中关于数据处理的部分,本质上是一串有序 handler 的集合。netty-handler 模块提供了开箱即用的 ChannelHandler 实现类,例如日志、IP 过滤、流量整形等,如果你需要这些功能,仅需在 pipeline 中加入相应的 ChannelHandler 即可。

## • Transport Service 传输服务层模块

netty-transport 模块可以说是 Netty 提供数据**处理和传输的核心模块**。该模块提供了很多非常重要的接口,如 Bootstrap、Channel、ChannelHandler、EventLoop、EventLoopGroup、ChannelPipeline 等。其中 Bootstrap 负责客户端或服务端的启动工作,包括创建、初始化Channel等;EventLoop负责向注册的 Channel 发起 I/O 读写操作;ChannelPipeline 负责ChannelHandler 的有序编排。另外,下图给出了netty-transport模块与其他几个模块的联系。



图五: netty-transport模块与前述几个模块的联系

# 3.小结

前文我们从逻辑架构和源码结构分别对 Netty 的整体架构进行了初步介绍,可见 Netty 的分层架构设计非常合理,实现了各层之间的逻辑解耦,对于开发者来说,只需要扩展业务逻辑即可。

下一步,我目前计划下一步对Transport模块的源码进行阅读和分析,可能会重点关注bootstrap和 channel部分的内容。