

昵称: Tiny&zzh
园龄: 5年2个月
粉丝: 50
关注: 15
[+加关注](#)

<2017年3月>

日	一	二	三	四	五	六
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

搜索

找找看

谷歌搜索

常用链接

[我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

我的标签

[Netty\(11\)](#)
[silverlight\(7\)](#)
[java\(7\)](#)
[Unity3D\(7\)](#)
[中文教程\(5\)](#)
[Netty 4\(5\)](#)
[NIO\(4\)](#)
[Okra\(3\)](#)
[Socket\(3\)](#)
[C#\(3\)](#)
[更多](#)

随笔分类

[Java\(4\)](#)
[Netty\(7\)](#)
[Okra\(3\)](#)
[Unity3D\(5\)](#)

随笔档案

[2016年6月 \(1\)](#)
[2016年4月 \(4\)](#)
[2015年6月 \(1\)](#)
[2015年3月 \(1\)](#)
[2015年2月 \(1\)](#)
[2015年1月 \(1\)](#)
[2014年10月 \(2\)](#)
[2014年9月 \(2\)](#)
[2014年8月 \(3\)](#)
[2014年7月 \(2\)](#)
[2014年6月 \(1\)](#)
[2014年4月 \(5\)](#)
[2014年1月 \(2\)](#)
[2013年12月 \(4\)](#)
[2013年11月 \(2\)](#)
[2013年10月 \(1\)](#)
[2013年6月 \(2\)](#)
[2013年4月 \(1\)](#)
[2012年11月 \(1\)](#)
[2012年9月 \(2\)](#)
[2012年5月 \(1\)](#)
[2012年3月 \(2\)](#)
[2012年2月 \(1\)](#)
[2012年1月 \(1\)](#)
[2011年12月 \(1\)](#)

相册

Netty4.x中文教程系列(五)编解码器Codec

上一篇文章详细解释了ChannelHandler的相关构架设计，版本和设计逻辑变更等等。

这篇文章主要在于讲述Handler里面的Codec，也就是相关的编解码器。原本想把编解码器写在上一篇文章里面的。后来想想Netty里面的编解码器太多了。想要一次写完比较困难。于是重新开了一篇文章来专门写这个。

1. Hello World！实例中的使用

在这里先讲一下我们第一篇文章里面的实例使用到编解码器。

1.1 DelimiterBasedFrameDecoder解码器

DelimiterBasedFrameDecoder 顾名思义我们可以理解为基于分隔符的帧解码器。参数有两个，一个是最大帧长度，另外一个定义分隔符。

在Delimiters中提供给我们两种分隔符。一种是"0x00-NUL"分隔符。另外一种就是实例中使用的"\r\n"或"\n"分隔符。

```
public DelimiterBasedFrameDecoder(
    int maxFrameLength, boolean stripDelimiter, boolean failFast, ByteBuf... delimiters) {
    validateMaxFrameLength(maxFrameLength);
    if (delimiters == null) {
        throw new NullPointerException("delimiters");
    }
    if (delimiters.length == 0) {
        throw new IllegalArgumentException("empty delimiters");
    }

    if (isLineBased(delimiters) && !isSubclass()) {
        lineBasedDecoder = new LineBasedFrameDecoder(maxFrameLength, stripDelimiter, failFast);
        this.delimiters = null;
    } else {
        this.delimiters = new ByteBuf[delimiters.length];
        for (int i = 0; i < delimiters.length; i++) {
            ByteBuf d = delimiters[i];
            validateDelimiter(d);
            this.delimiters[i] = d.slice(d.readerIndex(), d.readableBytes());
        }
        lineBasedDecoder = null;
    }
    this.maxFrameLength = maxFrameLength;
    this.stripDelimiter = stripDelimiter;
    this.failFast = failFast;
}
```

在构造函数中我们可以看出，当分隔符是"\n"的时候，框架默认解码器为基于行的帧解码器（LineBasedFrameDecoder）。否则按照可读比特长度进行帧解码。

1.2 StringDecoder 字符串解码器 和 编码器

解码器:将比特流转换为默认编码的字符串。默认编码为UTF-8。当然开发者可以通过设置字符编码参数来设置字符编码。编码器:将字符串转换为Byte[]

2. Netty中Handler详述

在Netty的类库的handler目录可以看出它的基本结构(下图):

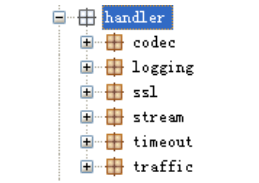
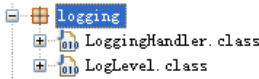


图2.1 handler包结构

整个包由6个主要部分组成，笔者将由简入繁，慢慢想读者解释每个包的含义和用法。(若有不正确之处，希望大家能给予指点)

2.1 Logging 日志



用于Netty中的日志输出。

2.1.1 loggingHandler

最新评论

1. Re:Netty4.x中文教程系列(一) 目录及概述
大哥，云盘地址失效了，能再分享一次吗？

--痞子色子

2. Re:[工具分享]JetBrains ReSharper 9.0 正式版和注册码
谢谢博主。也 谢谢 ReSharper。

--lnkFx

3. Re:Netty4.x中文教程系列(二) Hello World !
楼主，如果有多个客户端，如何实现在已知客户端ip和port情况下在服务端向客户端发送数据呢？

--1115405079

4. Re:Unity3D中uGUI事件系统简述及使用
方法总结
总结得挺好

--离落

5. Re:Netty4.x中文教程系列(七)UDP协议
嗯嗯，谢谢楼主！我已经用另外一种方法可以啦，非常感谢您的回复！非常感动！不知楼主是否使用过Netty实现P2P呢？

--雷泡泡

阅读排行榜

1. Netty4.x中文教程系列(二) Hello World !(15461)
2. Netty4.x中文教程系列(一) 目录及概述 (15319)
3. Unity3D中uGUI事件系统简述及使用
方法总结(9894)
4. Netty4.x中文教程系列(四) ChannelHandler(7550)
5. 【推荐】《Netty in action》书籍(7495)

评论排行榜

1. Netty4.x中文教程系列(二) Hello World !(9)
2. Netty4.x中文教程系列(七)UDP协议(5)
3. Netty4.x中文教程系列(一) 目录及概述 (4)
4. 【推荐】《Netty in action》书籍(4)
5. [工具分享]JetBrains ReSharper 9.0 正式版和注册码(3)

推荐排行榜

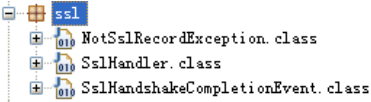
1. [工具分享]JetBrains ReSharper 9.0 正式版和注册码(3)
2. JAVA笔记-如何将百万级数据高效的导出到Excel表单(2)
3. Netty4.x中文教程系列(五)编解码器Codec(2)
4. 【推荐】《Netty in action》书籍(1)
5. JAVA数据库连接池的革命 -- 从BoneCP到HikariCP(1)

部事件的ChannelHandler，默认是记录全部DEBUG级别以上的事件。它的功能是记录全部事件，包含Inbound和Outbound的，之所以选择了继承ChannelDuplexHandler，是由于ChannelDuplexHandler继承ChannelInboundHandlerAdapter实现ChannelOutboundHandler。所以相当于Netty框架内的全部通信相关的事件都会得到处理。

2.1.2LogLevel

在这里作者定义了5个级别的log。TRACE，DEBUG，INFO，WARN，ERROR。

2.2 Ssl



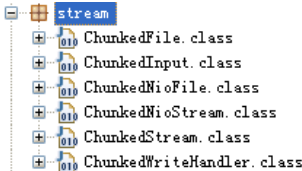
用于SSL协议解析和编码。

2.2.1 SslHandler

熟悉了解过Http的朋友应该是知道ssl的。SSL 的英文全称是“Secure Sockets Layer”，中文名为“安全套接层协议层”，它是网景（Netscape）公司提出的基于WEB应用的安全协议。SSL 协议指定了一种在应用程序协议（如HTTP、Telenet、NMTP和FTP等）和TCP/IP协议之间提供数据安全性分层的机制，它为TCP/IP连接提供数据加密、服务器认证、消息完整性以及可选的客户机认证。现在的相当一部分网站都有SSL加密。而SslHandler则是Netty提供的Ssl解码编码处理。

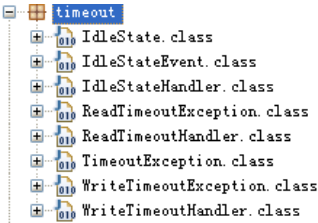
2.2.2 NotSslRecordException 和 SslHandshakeCompletionEvent 抛出异常和处理完成触发事件。

2.3 Stream 流



用于文件的的传输。将Java里面的File转换为Stream流，然后进行传输。

2.4 Timeout 空闲检测

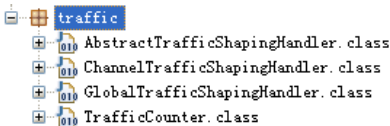


用于Netty框架中空闲超时相关。

IdleState 空闲状态。Netty中的空闲时间包括，读空闲，写空闲和读写空闲3种时间。

顾名思义，读空闲即一段时间内没有接受到消息，写空闲即一段时间内没有发送消息。读写空闲即一段时间内读写都空闲。主要是用于检测空闲状态。并且特定条件下服务端关闭和释放一些资源。

2.5 Traffic 流量统计



用于流量统计。

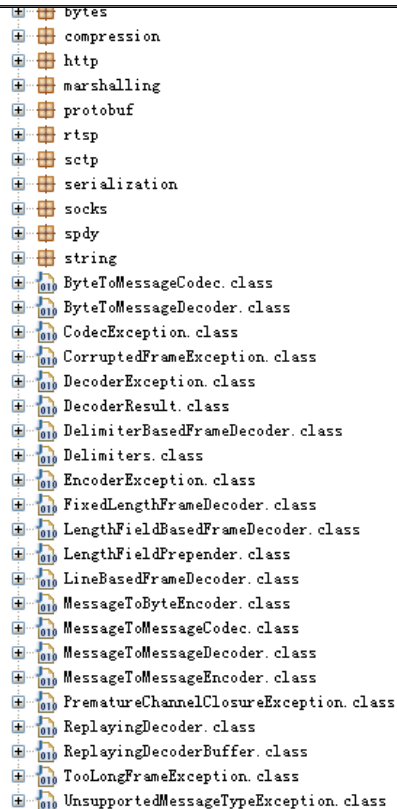
Netty提供了一个流量统计抽象类，一个Channel流量统计Handler和一个全局带宽流量统计Handler。

2.6 Codec 编解码器

这个是目前Netty的重点。也是最核心最复杂的部分。也是笔者认为Netty设计里面最好的一部分。

用于数据的编解码。

编解码器可以理解为一个规范让客户和服务端能够理解和识别字节流所包含的意思。其实编码和解码就是这么简单的事情，没什么复杂的。



The screenshot displays a file explorer view of the Netty 4.x package structure. It shows a hierarchy of sub-packages and classes. The sub-packages include 'bytes', 'compression', 'http', 'marshalling', 'protobuf', 'rtsp', 'sctp', 'serialization', 'socks', 'spdy', and 'string'. Below these, there are numerous classes, many of which are marked with a '010' icon, indicating they are deprecated. These classes include 'ByteToMessageCodec.class', 'ByteToMessageDecoder.class', 'CodecException.class', 'CorruptedFrameException.class', 'DecoderException.class', 'DecoderResult.class', 'DelimiterBasedFrameDecoder.class', 'Delimiters.class', 'EncoderException.class', 'FixedLengthFrameDecoder.class', 'LengthFieldBasedFrameDecoder.class', 'LengthFieldPrepender.class', 'LineBasedFrameDecoder.class', 'MessageToByteEncoder.class', 'MessageToMessageCodec.class', 'MessageToMessageDecoder.class', 'MessageToMessageEncoder.class', 'PrematureChannelClosureException.class', 'ReplayingDecoder.class', 'ReplayingDecoderBuffer.class', 'TooLongFrameException.class', and 'UnsupportedMessageTypeException.class'.

看到codec里面的这么一大堆的包类。读者是不是赶脚着很头晕？ $O(n_n)O$ 哈哈~。作者设计了非常友好的包逻辑结构。方便我们理解源码。

即Codec包中的类优先看，下面的子包都是一些Netty开发者们提供的一些实现。

有没有看到HelloWorld出现的LengthFieldBaseFrameDecoder和FixedLengthFrameDecoder。

这里主要讲的是Netty中最重要的两个编解码器ByteToMessageCodec和MessageToMessageCodec。之所只讲着两个的原因是其他的编解码器都是继承于这两个的。

ByteToMessageCodec: 在Netty4.x版本中允许传递Java中的对象，所以这个编解码起的作用就是讲Byte流转换为对象。而MessageToMessageCodec则是将Object转换为Object。这两个的区分其实很模糊。可能需要大家亲自动手写过之后才会有比较好的感受。

包base64: 继承MessageToMessageEncoder<ByteBuf>。是base64编码的一些东西。

包bytes: 继承MessageToMessageEncoder<ByteBuf>。用于字节数组和Netty里面的ByteBuf互相转换

包compression: 用于ByteBuf数据压缩和解压缩的。继承MessageToMessageEncoder<ByteBuf>

包http: 用于HTTP请求相关的。这个包里面就比较复杂了。下面详细讲一下。

1. 包空间: HTTP内容, 请求, 响应等。
2. 包cors: 包名称是(跨域资源分享) Cross Origin Resource Sharing 的简写。用于客户端跨域请求。可以参考 <http://www.w3.org/TR/cors/>
3. 包multipart: POST消息和文件上传相关的一些。只是粗略看了一下。
4. 包websocketx: 针对近年来Html5发展起来的websocket技术的。不过貌似由于Html5标准还未正式的确定。所以这个包里面的内容比较多。编解码器版本也很多。相信以后统一标准之后会简单一些。暂时不推荐吧

包marshalling:

包protoBuf: 用于 [Google Protocol Buffers](#) 编解码

包rtsp: 实时流传输协议 (Real Time Streaming Protocol, RTSP)

包sctp: 流控制传输协议 (Stream Control Transmission Protocol, SCTP)

包serialization: 用于序列化的Java对象的和ByteBuf之间的转换。

包socks: 用于Java Socket通信相关的。支持Socket4a和 Socket 5两个版本

包spdy: SPDY协议是近年来发展的一种协议。主要目的是为了减少网页加载的时间。它是HTTP协议的增强版本。它从某种程度上讲提高了HTTP协议在数据传输时的速度和性能

包string: 用于java里面字符串的编解码

作者：TinyZ

出处：<http://www.cnblogs.com/zou90512/>

关于作者：努力学习，天天向上。不断探索学习，提升自身价值。记录经验分享。

本文版权归作者和博客园共有，欢迎转载，但未经作者同意必须保留此段声明，且在文章页面明显位置给出原文链接

如有问题，可以通过 zou90512@126.com 联系我，非常感谢。

笔者网店: <http://aoleitaisen.taobao.com>. 欢迎广大读者围观

分类: [Netty](#)

标签: [Netty](#), [Netty 4.x](#), [中文教程](#), [Netty 4](#)

好文要顶

关注我

收藏该文



[Tiny&zzh](#)

[关注 - 15](#)

[粉丝 - 50](#)

[+加关注](#)

2

0

« 上一篇: [关于Netty4.x中文教程系列更新进度的说明和道歉](#)

» 下一篇: [关于Java的一些NIO框架的一点想法](#)

posted @ 2014-04-09 14:27 Tiny&zzh 阅读(6517) 评论(2) 编辑 收藏

评论列表

#1楼 2014-04-10 16:10 yaobiao

我想通过netty获取连接的底层socket，现在不知道怎么取，有没有办法呢

[支持\(0\)](#) [反对\(0\)](#)

#2楼[楼主] 2014-04-10 21:01 Tiny&zzh

@ yaobiao

在Netty里面的AbstractNioChannel里面定义了一个变量是ch。在实例化的时候赋值为java里面nio的channel

实现了一个方法javaChannel()获得是获取java的ServerSocket等。

这些全部都被Netty的作者封装起来了。当然。我们自己去继承AbstractNioChannel来实现自己的东西

4.0.17final: 参考Netty框架里面的NioServerSocketChannel (io.netty.channel.socket.nio.NioServerSocketChannel) 你就可以看到相应的实现了

[支持\(0\)](#) [反对\(0\)](#)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#)网站首页。

最新IT新闻:

- 死磕微信！支付宝推“奖励金” 付钱立得/最高999元
- [Linux Kernel 4.11](#)首个候选版本开放下载
- 微软透露Windows 10游戏模式细节：提升低端PC游戏性能
- 今日Google Doodle: 纪念印尼科莫多国家公园成立37周年
- 菜鸟裹裹公布2016包裹里程 看你快递走了多远
- » [更多新闻...](#)

最新知识库文章:

- 垃圾回收原来是这么回事
- 「代码家」的学习过程和学习经验分享
- 写给未来的程序媛
- 高质量的工程代码为什么难写
- 循序渐进地代码重构
- » [更多知识库文章...](#)