netty 数据分包、组包、粘包处理机制(部分)

LengthFieldBasedFrameDecoder

常用的处理大数据分包传输问题的解决类，先对构造方法LengthFieldBasedFrameDecoder中的参数做以下解释说明“

maxFrameLength：解码的帧的最大长度

lengthFieldOffset ：长度属性的起始位（偏移位），包中存放有整个大数据包长度的字节，这段字节的其实位置

lengthFieldLength：长度属性的长度，即存放整个大数据包长度的字节所占的长度

lengthAdjustmen：长度调节值，在总长被定义为包含包头长度时，修正信息长度。initialBytesToStrip：跳过的字节数，根据需要我们跳过lengthFieldLength个字节，以便接收端直接接受到不含“长度属性”的内容

failFast ：为true，当frame长度超过maxFrameLength时立即报TooLongFrameException异常，为false，读取完整个帧再报异常

下面对各种情况分别描述：

\1. 2 bytes length field at offset 0, do not strip header

lengthFieldOffset = 0

lengthFieldLength = 2

lengthAdjustment = 0

initialBytesToStrip = 0 (= do not strip header)

```
 BEFORE DECODE (14 bytes)         AFTER DECODE (14 bytes)


+--------+----------------+       +--------+----------------+

 | Length | Actual Content |---->| Length | Actual Content |

 | 0x000C | "HELLO, WORLD" |      | 0x000C | "HELLO, WORLD" |

 +--------+----------------+      +--------+----------------+
```

此时数据格式不做任何改变（没有跳过任何字节）

\2. 2 bytes length field at offset 0, strip header

lengthFieldOffset = 0

lengthFieldLength = 2

lengthAdjustment = 0

initialBytesToStrip = 2 (= the length of the Length field)

```
BEFORE DECODE (14 bytes)         AFTER DECODE (12 bytes)

+--------+----------------+      +----------------+

| Length | Actual Content |---->| Actual Content |

| 0x000C | "HELLO, WORLD" |      | "HELLO, WORLD" |

+--------+----------------+      +----------------+
```

此时帧长度为14个字节，但由于前（lengthFieldOffset = 0）两个（lengthFieldLength = 2）字节是表示帧长度的字节，不计入数据，故真实的数据长度为12个字节。

\3. 2 bytes length field at offset 0, do not strip header, the length field represents the length of the whole message

lengthFieldOffset = 0

lengthFieldLength = 2

lengthAdjustment = -2 (= the length of the Length field)

initialBytesToStrip = 0

```
BEFORE DECODE (14 bytes)         AFTER DECODE (14 bytes)

+--------+----------------+     +--------+----------------+

| Length | Actual Content |---->| Length | Actual Content |

| 0x000E | "HELLO, WORLD" |      | 0x000E | "HELLO, WORLD" |

+--------+----------------+     +--------+----------------+
```

此处定义的Length为0x000E共占了两个字节，表示的帧长度为14个字节，前（lengthFieldOffset = 0）两个（lengthFieldLength = 2）字节为Length，由于设置的lengthAdjustment = -2 (= the length of the Length field)，故修正的信息实际长度补2，即解码时往前推2个字节，解码后还是14个字节长度（此种情况是把整个长度封装，一般来讲，我们只封装数据长度）

\4. 3 bytes length field at the end of 5 bytes header, do not strip header

lengthFieldOffset = 2 (= the length of Header 1)

lengthFieldLength = 3

lengthAdjustment = 0

initialBytesToStrip = 0

```
BEFORE DECODE (17 bytes)                AFTER DECODE (17 bytes)

+---------+---------+-------------+    +---------+---------+------------+

| Header 1| Length  |Actual Content|--->| Header 1| Length | Actual Content|

|  0xCAFE | 0x00000C|"HELLO, WORLD"|    |  0xCAFE   |0x00000C| "HELLO, WORLD"|

+---------+---------+-------------+    +----------+--------+-----------+
```

此处lengthFieldOffset = 2，从第3个字节开始表示数据长度，长度占3个字节，真实数据长度为0x00000C 即12个字节，而lengthAdjustment=0，initialBytesToStrip = 0，故解码后的数据与解码前的数据相同。

\4. 3 bytes length field at the beginning of 5 bytes header, do not strip header

lengthFieldOffset = 0

lengthFieldLength = 3

lengthAdjustment = 2 (= the length of Header 1)

initialBytesToStrip = 0

```
BEFORE DECODE (17 bytes)                         AFTER DECODE (17 bytes)

 +----------+----------+----------------+      +----------+----------+----------------+

 |  Length  | Header 1 | Actual Content |----->|  Length  | Header 1 | Actual Content |

 | 0x00000C |  0xCAFE  | "HELLO, WORLD" |      | 0x00000C |  0xCAFE  | "HELLO, WORLD" |

 +----------+----------+----------------+      +----------+----------+----------------+
```

此处由于修正的字节数是2，且initialBytesToStrip = 0，故整个数据的解码数据保持不变

总字节数是17，开始的三个字节表示字节长度：12，修正的字节是2,（即从第三个字节开始，再加两个开始是真正的数据，其中跳过的字节数是0）

5. 2 bytes length field at offset 1 in the middle of 4 bytes header, strip the first header field and the length field

lengthFieldOffset = 1 (= the length of HDR1)

lengthFieldLength = 2

lengthAdjustment = 1 (= the length of HDR2)

initialBytesToStrip = 3 (= the length of HDR1 + LEN)

```
BEFORE DECODE (16 bytes)                       AFTER DECODE (13 bytes)

 +------+--------+------+----------------+      +------+----------------+

 | HDR1 | Length | HDR2 | Actual Content |----->| HDR2 | Actual Content |

 | 0xCA | 0x000C | 0xFE | "HELLO, WORLD" |      | 0xFE | "HELLO, WORLD" |

 +------+--------+------+----------------+      +------+----------------+
```

从第2个字节开始解码，取两个字节作为帧长度，为12个字节，然后，修正一个字节，从第5个字节到最后表示帧数据，解码时，由于initialBytesToStrip=3，表示跳过前三个字节（去掉），故从第四个字节开始解析，解析出来后，如右图所示。

6. 2 bytes length field at offset 1 in the middle of 4 bytes header, strip the first header field and the length field, the length field represents the length of the whole message

lengthFieldOffset = 1

lengthFieldLength = 2

lengthAdjustment = -3 (= the length of HDR1 + LEN, negative)

initialBytesToStrip = 3

```
BEFORE DECODE (16 bytes)                       AFTER DECODE (13 bytes)

 +------+--------+------+----------------+      +------+----------------+

 | HDR1 | Length | HDR2 | Actual Content |----->| HDR2 | Actual Content |

 | 0xCA | 0x0010 | 0xFE | "HELLO, WORLD" |      | 0xFE | "HELLO, WORLD" |

 +------+--------+------+----------------+      +------+----------------+
```

从第二个字节开始，取两个字节作为帧长度，为16个字节，然后补3个字节，故往前找三个字节，从HDP1开始解码，而又因为initialBytesToStrip=3，解码时忽略掉前三个字节，故从第四个字节开始解析，解析结果如右图所示。

总结：一般来讲，当lengthAdjustment 为负数时，Length表示的是整个帧的长度，当lengthAdjustment为正数或0时，表示真实数据长度。

(6) LengthFieldPrepender

编码类，自动将

```
+----------------+

| "HELLO, WORLD" |

+----------------+
```

格式的数据转换成

```
+--------+----------------+

+ 0x000C | "HELLO, WORLD" |

 +--------+----------------+
```

格式的数据，

如果lengthIncludesLengthFieldLength设置为true,则编码为

```
+--------+----------------+

\+ 0x000E | "HELLO, WORLD" |

+--------+----------------+
```

格式的数据

应用场景：自定义pipelineFactory类: MyPipelineFactory implements ChannelPipelineFactory

中

pipeline.addLast("frameEncode", new LengthFieldPrepender(4, false));

(7) TooLongFrameException

定义的数据包超过预定义大小异常类

(8) CorruptedFrameException

定义的数据包损坏异常类

3. frame包应用demo

解决分包问题，通常配置MyPipelineFactory中设置，示例如下：

```
public class MyPipelineFactory implements ChannelPipelineFactory {**



    @Override

    public ChannelPipeline getPipeline() throws Exception {**

       ChannelPipeline pipeline = Channels.pipeline();**

       pipeline.addLast("decoder", new LengthFieldBasedFrameDecoder(Integer.MAX_VALUE, 0, 4, 0,
4));

        pipeline.addLast("encoder", new LengthFieldPrepender(4, false));**

       pipeline.addLast("handler", new MyHandler());**

       return pipeline;**

    }



}
```

在客户端设置

```
pipeline.addLast("encoder", new LengthFieldPrepender(4, false));**

       pipeline.addLast("handler", new MyHandler());**
```

前四个字节表示真实的发送的数据长度**Length，编码时会自动加上；

在服务器端设置

```
pipeline.addLast("decoder", new LengthFieldBasedFrameDecoder(Integer.MAX_VALUE, 0, 4, 0, 4));
```

真实数据最大字节数为**Integer.MAX_VALUE，解码时自动去掉前面四个字节