- NETTY

精尽 Netty 源码解析 —— Codec 之 ByteToMessageCodec

1. 概述

本文，我们来分享 ByteToMessageCodec 部分的内容。

在网络通信中，编解码是成对出现的，所以 Netty 提供了 ByteToMessageCodec 类，支持 Encoder 和 Decoder 两个功能。

2. ByteToMessageCodec

`io.netty.handler.codec.ByteToMessageCodec` ，继承 ChannelDuplexHandler 类，通过**组合** MessageToByteEncoder 和 ByteToMessageDecoder 的功能，从而实现编解码的 Codec **抽象类**。

2.1 构造方法

```
public abstract class ByteToMessageCodec<I> extends ChannelDuplexHandler {

    /**
     * 类型匹配器
     */
    private final TypeParameterMatcher outboundMsgMatcher;
```



```
ncoder;

der = new ByteToMessageDecoder() {

        @Override
        public void decode(ChannelHandlerContext ctx, ByteBuf in, List<Object> out) throws Exception {
            ByteToMessageCodec.this.decode(ctx, in, out);
        }

        @Override
        protected void decodeLast(ChannelHandlerContext ctx, ByteBuf in, List<Object> out) throws Exce
            ByteToMessageCodec.this.decodeLast(ctx, in, out);
        }

    };
```

```
    protected ByteToMessageCodec() {
        this(true);
    }

    protected ByteToMessageCodec(Class<? extends I> outboundMessageType) {
        this(outboundMessageType, true);
    }

    protected ByteToMessageCodec(boolean preferDirect) {
        // 禁止共享
        ensureNotSharable();
        // <1> 获得 matcher
        outboundMsgMatcher = TypeParameterMatcher.find(this, ByteToMessageCodec.class, "I");
        // 创建 Encoder 对象
        encoder = new Encoder(preferDirect);
    }

    protected ByteToMessageCodec(Class<? extends I> outboundMessageType, boolean preferDirect) {
        // 禁止共享
        ensureNotSharable();
        // <2> 获得 matcher
        outboundMsgMatcher = TypeParameterMatcher.get(outboundMessageType);
        // 创建 Encoder 对象
        encoder = new Encoder(preferDirect);
    }

    // ... 省略其他无关代码

    private final class Encoder extends MessageToByteEncoder<I> {

        Encoder(boolean preferDirect) {
            super(preferDirect);
        }
```

文章目录


```
e(Object msg) throws Exception {
acceptOutboundMessage(msg);



rContext ctx, I msg, ByteBuf out) throws Exception {
ctx, msg, out);
        }

    }

}
```

- MessageToByteEncoder 部分
  - `encoder` 属性，Encoder 对象，继承自 MessageToByteEncoder 抽象类。只能在构造方法中创建，因为他依赖构造方法的 `preferDirect` 方法参数，所以不能像 `decoder` 直接使用属性赋值。
  - `outboundMsgMatcher` 属性，类型匹配器，在 `<1>` / `<2>` 处初始化。

- ByteToMessageDecoder 部分
  - `decoder` 属性，Decoder 对象，继承自 ByteToMessageDecoder 抽象类。

2.2 ChannelHandler 相关方法

```
@Override
public void channelRead(ChannelHandlerContext ctx, Object msg) throws Exception {
    decoder.channelRead(ctx, msg);
}

@Override
public void write(ChannelHandlerContext ctx, Object msg, ChannelPromise promise) throws Exception {
    encoder.write(ctx, msg, promise);
}

@Override
public void channelReadComplete(ChannelHandlerContext ctx) throws Exception {
    decoder.channelReadComplete(ctx);
}

@Override
public void channelInactive(ChannelHandlerContext ctx) throws Exception {
    decoder.channelInactive(ctx);
}

@Override
public void handlerAdded(ChannelHandlerContext ctx) throws Exception {
    try {
        decoder.handlerAdded(ctx);
    } finally {
        encoder.handlerAdded(ctx);
    }
}
```

文章目录


```
ntext ctx) throws Exception {
```

2.3 MessageToByteEncoder 相关方法

```Java
/**
 * @see ByteToMessageDecoder#decode(ChannelHandlerContext, ByteBuf, List)
 */
protected abstract void decode(ChannelHandlerContext ctx, ByteBuf in, List<Object> out) throws Excepti

/**
 * @see ByteToMessageDecoder#decodeLast(ChannelHandlerContext, ByteBuf, List)
```

```
     */
    protected void decodeLast(ChannelHandlerContext ctx, ByteBuf in, List<Object> out) throws Exception {
        if (in.isReadable()) {
            // Only call decode() if there is something left in the buffer to decode.
            // See https://github.com/netty/netty/issues/4386
            decode(ctx, in, out);
        }
    }
```

2.4 MessageToByteEncoder 相关方法

```
    /**
     * Returns {@code true} if and only if the specified message can be encoded by this codec.
     *
     * @param msg the message
     */
    public boolean acceptOutboundMessage(Object msg) throws Exception {
        return outboundMsgMatcher.match(msg);
    }

    /**
     * @see MessageToByteEncoder#encode(ChannelHandlerContext, Object, ByteBuf)
     */
    protected abstract void encode(ChannelHandlerContext ctx, I msg, ByteBuf out) throws Exception;
```

666. 彩蛋

小小的水文一篇，嘿嘿。

文章目录