6.Netty 初认识 -- 粘包拆包 --DelimiterBasedFrameDecoder, 固定分隔符（$$$---$$$）示例二

作者：289306290

原文链接：https://ld246.com/article/1583130983017

来源网站：链滴

EchoServer.java

```
package club.wujingjian.com.wujingjian.netty.delimiter.server;

import io.netty.bootstrap.ServerBootstrap;
import io.netty.buffer.ByteBuf;
import io.netty.buffer.Unpooled;
import io.netty.channel.ChannelFuture;
import io.netty.channel.ChannelInitializer;
import io.netty.channel.ChannelOption;
import io.netty.channel.EventLoopGroup;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.SocketChannel;
import io.netty.channel.socket.nio.NioServerSocketChannel;
import io.netty.handler.codec.DelimiterBasedFrameDecoder;
import io.netty.handler.codec.string.StringDecoder;
import io.netty.handler.logging.LogLevel;
import io.netty.handler.logging.LoggingHandler;

public class EchoServer {

    public void bind (int port) throws  Exception {
        //配置服务端的NIO线程组
        EventLoopGroup bossGroup = new NioEventLoopGroup();
        EventLoopGroup workerGroup = new NioEventLoopGroup();

        try {
            ServerBootstrap b = new ServerBootstrap();
            b.group(bossGroup, workerGroup)
                    .channel(NioServerSocketChannel.class)
                    .option(ChannelOption.SO_BACKLOG, 100)
                    .handler(new LoggingHandler(LogLevel.INFO))
                    .childHandler(new ChannelInitializer<SocketChannel>() {
                        @Override
                        protected void initChannel(SocketChannel socketChannel) throws Exception {
                            //创建分隔符缓冲对象ByteBuf,本例中使用"$_"作为分隔符
                            ByteBuf delimiter = Unpooled.copiedBuffer("$_".getBytes());
                            //第一个参数1024表示单条消息的最大长度,当达到该长度后仍然没有查找到分隔符
就抛出TooLongFrameException，防止由于异常码流确实分隔符导致的内存溢出
                            //这是Netty解码器的可靠性保护,第二个参数就是分隔符缓冲对象
                            socketChannel.pipeline().addLast(new DelimiterBasedFrameDecoder(1024, d
limiter));
                            socketChannel.pipeline().addLast(new StringDecoder());
                            socketChannel.pipeline().addLast(new EchoServerHandler());
                        }
                    });

            //绑定端口,同步等待成功
            ChannelFuture f = b.bind(port).sync();
            //等待服务端监听端口关闭
            f.channel().closeFuture().sync();
        } finally {
            bossGroup.shutdownGracefully();
            workerGroup.shutdownGracefully();
```

```
        }
    }

    public static void main(String[] args) throws Exception {
        int port = 8080;
        if (args != null && args.length > 0) {
            port = Integer.parseInt(args[0]);
        }
        new EchoServer().bind(port);

    }
}
```

EchoServerHandler.java

```
package club.wujingjian.com.wujingjian.netty.delimiter.server;

import io.netty.buffer.ByteBuf;
import io.netty.buffer.Unpooled;
import io.netty.channel.ChannelHandler;
import io.netty.channel.ChannelHandlerAdapter;
import io.netty.channel.ChannelHandlerContext;

@ChannelHandler.Sharable
public class EchoServerHandler extends ChannelHandlerAdapter {

    int counter = 0;

    @Override
    public void channelRead(ChannelHandlerContext ctx, Object msg) throws Exception {
        String body = (String) msg;
        System.out.println("This is " + ++counter + " times receive client : [" + body
                + "]");
        body += "$_"; //由于我们设置DelimiterBasedFrameDecoder过滤掉了分隔符,所以,返回给客
端时需要在请求消息尾部拼接分隔符"$_",
        //最后创建ByteBuf,将原始消息重新返回给客户端
        ByteBuf echo = Unpooled.copiedBuffer(body.getBytes());
        ctx.writeAndFlush(echo);
    }

    @Override
    public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause) throws Excepti
n {
        cause.printStackTrace();;
        ctx.close();
    }
}
```

EchoClient.java

```
package club.wujingjian.com.wujingjian.netty.delimiter.client;
```

```
import io.netty.bootstrap.Bootstrap;
import io.netty.buffer.Unpooled;
import io.netty.channel.ChannelFuture;
import io.netty.channel.ChannelInitializer;
import io.netty.channel.ChannelOption;
import io.netty.channel.EventLoopGroup;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.SocketChannel;
import io.netty.channel.socket.nio.NioServerSocketChannel;
import io.netty.channel.socket.nio.NioSocketChannel;
import io.netty.handler.codec.DelimiterBasedFrameDecoder;
import io.netty.handler.codec.string.StringDecoder;

public class EchoClient {

    public void connect(int port, String host) throws  Exception {
        //配置客户端NIO线程组
        EventLoopGroup group = new NioEventLoopGroup();
        try {
            Bootstrap b = new Bootstrap();
            b.group(group).channel(NioSocketChannel.class)
                    .option(ChannelOption.TCP_NODELAY, true)
                    .handler(new ChannelInitializer<SocketChannel>() {
                        @Override
                        protected void initChannel(SocketChannel socketChannel) throws Exception {
                            socketChannel.pipeline().addLast(new DelimiterBasedFrameDecoder(1024, U
pooled.copiedBuffer("$-".getBytes())));
                            socketChannel.pipeline().addLast(new StringDecoder());
                            socketChannel.pipeline().addLast(new EchoClientHandler());
                        }
                    });
            //发起异步连接操作
            ChannelFuture f = b.connect(host, port).sync();

            //等待客户端链路关闭
            f.channel().closeFuture().sync();
        }finally {
            group.shutdownGracefully();
        }
    }

    public static void main(String[] args) throws Exception {
        int port = 8080;
        if (args != null && args.length > 0) {
            port = Integer.parseInt(args[0]);
        }

        new EchoClient().connect(port,"127.0.0.1");
    }
}
```

EchoClientHandler.java

```
package club.wujingjian.com.wujingjian.netty.delimiter.client;

import io.netty.buffer.Unpooled;
import io.netty.channel.ChannelHandlerAdapter;
import io.netty.channel.ChannelHandlerContext;

public class EchoClientHandler extends ChannelHandlerAdapter {
    private int counter ;

    static final String ECHO_REQ = "Hi,Welcome to Netty.$_";

    public EchoClientHandler(){}

    @Override
    public void channelActive(ChannelHandlerContext ctx) throws Exception {
        for (int i = 0; i < 10; i++) {
            ctx.writeAndFlush(Unpooled.copiedBuffer(ECHO_REQ.getBytes()));
        }
    }

    @Override
    public void channelRead(ChannelHandlerContext ctx, Object msg) throws Exception {
        System.out.println("This is " + ++counter + " times receive server :[" + msg + "]");
    }

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

    @Override
    public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause) throws Excepti
n {
        cause.printStackTrace();
        ctx.close();
    }
}
```

允许结果服务端:

客户端没有输出内容