JBoss Marshalling

笔记本：	<Inbox>		
创建时间：	12/13/2018 10:42	更新时间：	12/13/2018 10:50
作者：	wangqinlinmail@163.com		
标签：	Netty		

官网：http://jbossmarshalling.jboss.org/
jar包：http://jbossmarshalling.jboss.org/downloads

Current Releases: 1.3.x

Name	Version	Size	Release date	License	Download
JBoss Marshalling API	1.3.0.CR9	226 kB	2011-04-27	LGPL	jboss-marshalling-1.3.0.CR9.jar Downloads: 0
JBoss Marshalling API Sources	1.3.0.CR9	171 kB	2011-04-27	LGPL	jboss-marshalling-1.3.0.CR9-sources.jar Downloads: 0
JBoss Marshalling River Protocol	1.3.0.CR9	79 kB	2011-04-27	LGPL	jboss-marshalling-river-1.3.0.CR9.jar Downloads: 0
JBoss Marshalling River Protocol Sources	1.3.0.CR9	45 kB	2011-04-27	LGPL	jboss-marshalling-river-1.3.0.CR9-sources.jar Downloads: 0
JBoss Marshalling Serial Protocol	1.3.0.CR9	68 kB	2011-04-27	LGPL	jboss-marshalling-serial-1.3.0.CR9.jar Downloads: 0
JBoss Marshalling Serial Protocol Sources	1.3.0.CR9	33 kB	2011-04-27	LGPL	jboss-marshalling-serial-1.3.0.CR9-sources.jar Downloads: 0
JBoss Marshalling OSGi Bundle	1.3.0.CR9	376 kB	2011-04-27	LGPL	jboss-marshalling-osgi-1.3.0.CR9.jar Downloads: 0
JBoss Marshalling OSGi Bundle Sources	1.3.0.CR9	171 kB	2011-04-27	LGPL	jboss-marshalling-osgi-1.3.0.CR9-sources.jar Downloads: 0

代码实现：
定义SubscribeReq.proto（查看ProtoBuf文档）
定义SubscribeResp.proto（查看ProtoBuf文档）
SubReqServer实现：

```
importio.netty.bootstrap.ServerBootstrap;
importio.netty.channel.ChannelFuture;
importio.netty.channel.ChannelInitializer;
importio.netty.channel.ChannelOption;
importio.netty.channel.nio.NioEventLoopGroup;
importio.netty.channel.socket.SocketChannel;
importio.netty.channel.socket.nio.NioServerSocketChannel;
importio.netty.handler.logging.LogLevel;
importio.netty.handler.logging.LoggingHandler;
public class SubReqServer {
     public void bind(int port) throws InterruptedException {
          NioEventLoopGroup bossGroup = new NioEventLoopGroup();
          NioEventLoopGroup 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 ch) throws Exception {
                                  ch.pipeline().addLast(MarshallingCodeFactory.buildMarshallingDecoder());
                                  ch.pipeline().addLast(MarshallingCodeFactory.buildMarshallingEncoder());
                                  ch.pipeline().addLast(new SubReqServerHandler());
                             }
                        });
              ChannelFuture f = b.bind(port).sync();
              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) {
                try {
                    port = Integer.valueOf(args[0]);
                } catch (Exception e) {
                }
            }
            new SubReqServer().bind(port);
        }
}
```

SubReqServerHandler

```
importcom.itheima.netty.protobuf.SubscribeReqProto;
importcom.itheima.netty.protobuf.SubscribeRespProto;
importio.netty.channel.ChannelHandlerContext;
importio.netty.channel.ChannelInboundHandlerAdapter;
public class SubReqServerHandler extends ChannelInboundHandlerAdapter {
    @Override
    public void channelRead(ChannelHandlerContext ctx, Object msg) throws Exception {

        SubscribeReqProto.SubscribeReq req = (SubscribeReqProto.SubscribeReq)msg;
        if("LilinFeng".equalsIgnoreCase(req.getUsername())){
            System.out.println("Service accept client subscribe req :["+
req.toString()+"]");
            ctx.writeAndFlush(Resp(req.getSubReqID()));
        }
    }
    private SubscribeRespProto.SubscribeResp Resp(int subReqID){
        SubscribeRespProto.SubscribeResp.Builder builder =
SubscribeRespProto.SubscribeResp.newBuilder();
        builder.setSubReqID(subReqID);
        builder.setRespCode(0);
        builder.setDesc("Netty book order succeed, 3 day later,sent to the desigated
address");
        return builder.build();
    }
    @Override
    public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause) throws
Exception {
        cause.printStackTrace();
        ctx.close();
    }
}
```

SubReqClient实现：

```
importio.netty.bootstrap.Bootstrap;
importio.netty.channel.ChannelFuture;
importio.netty.channel.ChannelInitializer;
importio.netty.channel.ChannelOption;
importio.netty.channel.nio.NioEventLoopGroup;
importio.netty.channel.socket.SocketChannel;
importio.netty.channel.socket.nio.NioSocketChannel;
importio.netty.handler.codec.protobuf.ProtobufDecoder;
importio.netty.handler.codec.protobuf.ProtobufEncoder;
```

```
importio.netty.handler.codec.protobuf.ProtobufVarint32FrameDecoder;
importio.netty.handler.codec.protobuf.ProtobufVarint32LengthFieldPrepender;
public class SubReqClient {

     public void connect(int port,String host) throws InterruptedException {
          NioEventLoopGroup group = new NioEventLoopGroup();
          try {
              Bootstrap b = new Bootstrap();
              b.group(group).channel(NioSocketChannel.class).option(ChannelOption.TCP_NODELAY,
true)
                        .option(ChannelOption.CONNECT_TIMEOUT_MILLIS, 3000)
                        .handler(new ChannelInitializer<SocketChannel>() {
                             @Override
                             protected void initChannel(SocketChannel ch) throws Exception {
                                  ch.pipeline().addLast(new ProtobufVarint32FrameDecoder());
                                  ch.pipeline().addLast(new
ProtobufDecoder(SubscribeRespProto.SubscribeResp.getDefaultInstance()));
                                  ch.pipeline().addLast(new
ProtobufVarint32LengthFieldPrepender());
                                  ch.pipeline().addLast(new ProtobufEncoder());
                                  ch.pipeline().addLast(new SubReqClientHandle());
                             }
                        });
              ChannelFuture f = b.connect(host, port).sync();
              f.channel().closeFuture().sync();
          } finally {
              group.shutdownGracefully();
          }
     }
     public static void main(String[] args) throws Exception {
          int port = 8080;
          if (args != null && args.length > 0) {
              try {
                   port = Integer.valueOf(args[0]);
              } catch (Exception e) {
              }
          }
          new SubReqClient().connect(8080,"127.0.0.1");
     }
}
```

SubReqClientHandle

```
import java.util.ArrayList;
import java.util.List;
importio.netty.channel.ChannelHandlerContext;
importio.netty.channel.ChannelInboundHandlerAdapter;
public class SubReqClientHandle extends ChannelInboundHandlerAdapter {
     public SubReqClientHandle() {
     }
     @Override
     public void channelActive(ChannelHandlerContext ctx) throws Exception {
          for (int i = 0; i < 10; i++) {
              ctx.write(subReq(i));
          }
          ctx.flush();
     }
     private static SubscribeReqProto.SubscribeReq subReq(int i) {
          SubscribeReqProto.SubscribeReq.Builder builder =
SubscribeReqProto.SubscribeReq.newBuilder();
          builder.setSubReqID(i);
          builder.setUsername("Lilinfeng");
```

```
            builder.setProductName("Netty book");
            List<String> address = new ArrayList<String>();
            address.add("beijing");
            address.add("shanghai");
            address.add("shengzheng");
            builder.addAllAddress(address);
            return builder.build();
        }
        @Override
        public void channelRead(ChannelHandlerContext ctx, Object msg) throws Exception {
            System.out.println("Receive server response :[" + msg + "]");
        }
        @Override
        public void channelReadComplete(ChannelHandlerContext ctx) throws Exception {
            ctx.flush();
        }
        @Override
        public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause) throws
Exception {
            cause.printStackTrace();
            ctx.close();
        }
}
```