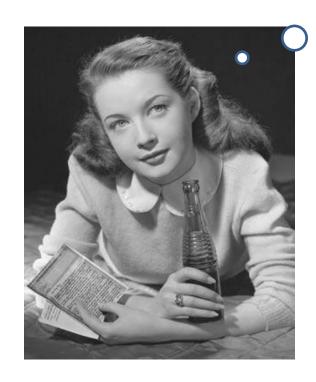


深入浅出Netty

雷腾 L.T - leiteng@taobao.com 搜索算法与技术 – 搜索应用团队

Netty是什么?

Netty 提供异步的、事件驱动的 网络应用程序框架和工具,用 以快速开发高性能、高可靠性 的网络服务器和客户端程序





Netty的架构

Core

Transport Services

Socket & Datagram HTTP Tunnel

In-VM Pipe

Protocol Support

HTTP & WebSocket	SSL · StartTLS	Google Protobuf		
zlib/gzip Compression	Large File Transfer	RTSP		
Legacy Text · Binary Protocols with Unit Testability				

Core

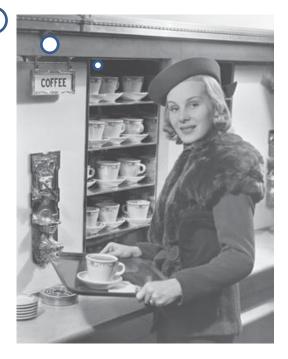
Extensible Event Model

Universal Communication API

Zero-Copy-Capable Rich Byte Buffer

Netty的特性

Netty 有些什 么特性呢?





Netty的特性

▶设计

- 统一的API, 适用于不同的协议(阻塞和非阻塞)
- 基于灵活、可扩展的事件驱动模型
- 高度可定制的线程模型
- 可靠的无连接数据Socket支持(UDP)

★ 性能

- 更好的吞吐量,低延迟
- 更省资源
- 尽量减少不必要的内存拷贝



Netty的特性

★ 安全

- 完整的SSL/TLS和STARTTLS的支持
- 能在Applet与谷歌Android的限制环境运行良好

★ 健壮性

- 不再因过快、过慢或超负载连接导致 OutOfMemoryError
- 不再有在高速网络环境下NIO读写频率不一致的问题

▶易用

- 完善的Java doc,用户指南和样例
- 简洁简单
- 仅依赖于JDK1.5



深入浅出Netty

But how to use it?



HelloWorldServer

```
ServerBootstrap bootstrap = new ServerBootstrap(new
  NioServerSocketChannelFactory(
  Executors.newCachedThreadPool(),
  Executors.newCachedThreadPool()));
bootstrap.setPipelineFactory(new ChannelPipelineFactory() {
  public ChannelPipeline getPipeline() {
    ChannelPipeline pipeline = Channels.pipeline();
    pipeline.addLast("decoder", new StringDecoder());
    pipeline.addLast("encoder", new StringEncoder());
    pipeline.addLast("handler", new HelloWorldServerHandler());
    return pipeline;
});
bootstrap.bind(new InetSocketAddress(8080));
```

HelloWorldServerHandler

```
public void channelConnected(ChannelHandlerContext ctx,
  ChannelStateEvent e) throws Exception {
  e.getChannel().write("Hello, World");
public void exceptionCaught(ChannelHandlerContext ctx,
  ExceptionEvent e) {
  logger.log(Level.WARNING, "Unexpected exception from
  downstream.", e.getCause());
  e.getChannel().close();
```

HelloWorldClient

```
ClientBootstrap bootstrap = new ClientBootstrap(new NioClientSocketChannelFactory( Executors.newCachedThreadPool(), Executors.newCachedThreadPool()));
```

```
bootstrap.setPipelineFactory(new ChannelPipelineFactory() {
   public ChannelPipeline getPipeline() {
      ChannelPipeline pipeline = pipeline();
      pipeline.addLast("decoder", new StringDecoder());
      pipeline.addLast("encoder", new StringEncoder());
      pipeline.addLast("handler", new HelloWorldClientHandler());
      return pipeline;
});
ChannelFuture future = bootstrap.connect(new InetSocketAddress("localhost", 8080));
future.getChannel().getCloseFuture().awaitUninterruptibly();
bootstrap.releaseExternalResources();
```

HelloWorldClientHandler

```
public void messageReceived(ChannelHandlerContext ctx,
  MessageEvent e) {
   String message = (String) e.getMessage();
   System.out.println(message);
   e.getChannel().close();
public void exceptionCaught(ChannelHandlerContext ctx,
  ExceptionEvent e) {
   logger.log(Level.WARNING, "Unexpected exception from
      downstream.", e.getCause());
   e.getChannel().close();
```

深入浅出Netty

刨根问底的勇气?





- org.jboss.netty.bootstrap
 - ▶ Bootstrap: ChannelFactory, ChannelPipeline, ChannelPipelineFactory
 - 対 初始化channel的辅助类
 - ★ 为具体的子类提供公共数据结构
 - ServerBootstrap: bind()
 - ★ 创建服务器端channel的辅助类
 - ★ 接收connection请求
 - ClientBootstrap : connect()
 - ◎ 创建客户端channel的辅助类
 - ★ 发起connection请求
 - ConnectionlessBootstrap : connect() , bind()
 - 🔖 创建无连接传输channel的辅助类(UDP)
 - ★ 包括Client 和Server



org.jboss.netty.buffer

取代nio中的java.nio.ByteBuffer,相比ByteBuffer

- ◆ 可以根据需要自定义buffer type
- ▶ 内置混合的buffer type, 以实现zero-copy
- ☀ 提供类似StringBuffer的动态dynamic buffer
- ★ 不需要调用flip方法
- ▶ 更快的性能

推荐使用ChannelBuffers的静态工厂创建ChannelBuffer



- ✓ org.jboss.netty.channel.socket
 TCP, UDP接口,继承了核心的channel API

- ✓ org.jboss.netty.channel.socket.http

 基于http的客户端和相应的server端的实现,工作在有防火墙的情况。



- ◇ org.jboss.netty.handler.stream
 异步写入大数据,不会产生outOfMemory也不会花费很多内存

- 🔷 org.jboss.netty.handler.codec.base64 Base64 编码
- ★ org.jboss.netty.handler.codec.compression 压缩格式
- ★ org.jboss.netty.handler.codec.embedder 嵌入模式下编码和解码
- 🛸 org.jboss.netty.handler.codec.frame 评估流的数据的排列和内容
- 🔷 org.jboss.netty.handler.codec.http.websocket websocket编码解码
- ★ org.jboss.netty.handler.codec.http http的编码解码以及类型信息
- 🛸 org.jboss.netty.handler.codec.oneone 对象到对象编码解码
- 💉 org.jboss.netty.handler.codec.protobuf Protocol Buffers的编码解码
- 💉 org.jboss.netty.handler.codec.replay 在阻塞io中实现非阻塞解码
- ★ org.jboss.netty.handler.codec.rtsp RTSP的编码解码
- ★ org.jboss.netty.handler.codec.serialization 序列化对象到bytebuffer实现
- 🔷 org.jboss.netty.handler.codec.string 字符串编码解码,继承oneone

- ★ org.jboss.netty.logging 根据不同的log framework 实现的类
- org.jboss.netty.util.internal
 netty内部util类,不被外部使用



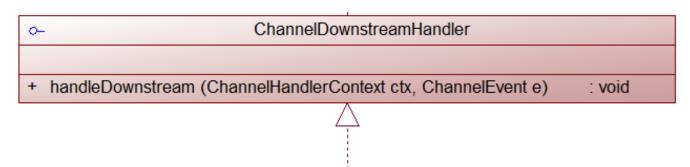
Netty事件驱动模型

0-	ChannelUpstreamHandler	
+ handleUpstream (Ch	annelHandlerContext ctx, ChannelEvent e)	: void
	Ą	

	SimpleChannelUpstreamHandler			
- logger : InternalLogger = InternalLoggerFactory.getInstance(SimpleChannelUpstreamHandler.class.getName())				
+ < <constructor>></constructor>	SimpleChannelUpstreamHandler ()			
+ < <implement>></implement>	handleUpstream (ChannelHandlerContext ctx, ChannelEvent e)	: void		
+	messageReceived (ChannelHandlerContext ctx, MessageEvent e)	: void		
+	exceptionCaught (ChannelHandlerContext ctx, ExceptionEvent e)	: void		
+	channelOpen (ChannelHandlerContext ctx, ChannelStateEvent e)	: void		
+	channelBound (ChannelHandlerContext ctx, ChannelStateEvent e)	: void		
+	channelConnected (ChannelHandlerContext ctx, ChannelStateEvent e)	: void		
+	channelInterestChanged (ChannelHandlerContext ctx, ChannelStateEvent e)	: void		
+	channelDisconnected (ChannelHandlerContext ctx, ChannelStateEvent e)	: void		
+	channelUnbound (ChannelHandlerContext ctx, ChannelStateEvent e)	: void		
+	channelClosed (ChannelHandlerContext ctx, ChannelStateEvent e)	: void		
+	writeComplete (ChannelHandlerContext ctx, WriteCompletionEvent e)	: void		
+	childChannelOpen (ChannelHandlerContext ctx, ChildChannelStateEvent e)	: void		
+	childChannelClosed (ChannelHandlerContext ctx, ChildChannelStateEvent e)	: void		



Netty事件驱动模型



SimpleChannelDownstreamHandler				
	0: 1 01			
+ < <constructor>></constructor>	SimpleChannelDownstreamHandler ()			
+ < <implement>></implement>	handleDownstream (ChannelHandlerContext ctx, ChannelEvent e)	: void		
+	writeRequested (ChannelHandlerContext ctx, MessageEvent e)	: void		
+	bindRequested (ChannelHandlerContext ctx, ChannelStateEvent e)	: void		
+	connectRequested (ChannelHandlerContext ctx, ChannelStateEvent e)	: void		
+	setInterestOpsRequested (ChannelHandlerContext ctx, ChannelStateEvent e)	: void		
+	disconnectRequested (ChannelHandlerContext ctx, ChannelStateEvent e)	: void		
+	unbindRequested (ChannelHandlerContext ctx, ChannelStateEvent e)	: void		
+	closeRequested (ChannelHandlerContext ctx, ChannelStateEvent e)	: void		



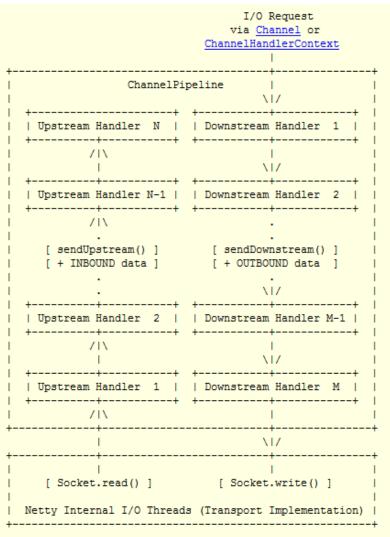
Netty Pipeline 流处理

- ▶ Upstream 接收请求
- ▶ Downstream 发送请求

```
ChannelPipeline p = Channels.pipeline();
p.addLast("1", new UpstreamHandlerA());
p.addLast("2", new UpstreamHandlerB());
p.addLast("3", new DownstreamHandlerA());
p.addLast("4", new DownstreamHandlerB());
p.addLast("5", new UpstreamHandlerX());
```

Upstream: 1 → 2 → 5 顺序处理

Downstream: 4 → 3 逆序处理





Netty 通用通信API

- connect
- bind
- write
- close
- disconnect
- unbind
- isOpen
- isBound
- isConnected
- isReadable
- isWrita ble

0-		Channel	
+ OP_NONE	: int	= 0	
+ OP_READ	: int	= 1	
+ OP_WRITE	: int	= 4	
+ OP_READ_WRITE	: int	= OP_READ OP_WRITE	
+ getld ()			: Integer
+ getFactory ()			: ChannelFactory
+ getParent ()			: Channel
+ getConfig ()			: ChannelConfig
+ getPipeline ()			: ChannelPipeline
+ isOpen ()			: boolean
+ isBound ()			: boolean
+ isConnected ()			: boolean
+ getLocalAddress ()			: SocketAddress
+ getRemoteAddress	()		: SocketAddress
+ write (Object messa	ge)		: ChannelFuture
+ write (Object messa	ge, So	ocketAddress remoteAddress)	: ChannelFuture
+ bind (SocketAddress			: ChannelFuture
+ connect (SocketAdd	ress r	emoteAddress)	: ChannelFuture
+ disconnect ()			: ChannelFuture
+ unbind ()			: ChannelFuture
+ close ()			: ChannelFuture
+ getCloseFuture ()			: ChannelFuture
+ getInterestOps ()			: int
+ isReadable ()			: boolean
+ isWritable ()			boolean
+ setInterestOps (int in	terest	Ops)	: ChannelFuture
+ setReadable (boolea			: ChannelFuture



Netty Zero-Copy-Capable Buffer

▶ 序列访问索引

```
+-----+
| discardable bytes | readable bytes | writable bytes |
+-----+
| 0 <= readerIndex <= writerIndex <= capacity
```

- get & set : not modify the readerIndex or writerIndex
- read & write: modify the readerIndex or writerIndex

Netty Zero-Copy-Capable Buffer

- **★** ChannelBuffer
 - 定义接口
- ChannelBuffers
 - 静态工厂
 - 隐藏具体类型

5	Nedlary ()	ChannelBufferFactory
•	order ()	ByteOrder
	isDirect () readerindex ()	boolean
٠	readerindex (int readerindex)	void int
	maderinde (i) maderinde (ii) maderinde (ii) maderinde (iii) ma	void void int int
	settrdex (nt readerindex, int writerindex) readableBytes ()	int
•	witableBytes() readable()	int boolean
	unitable 0	hoolean
•	deer() mariReaderIndex() mariReaderIndex() mariWeterIndex()	void void void
	reseReaderIndex ()	void
	nanvittenidex() restVitteridex()	void void
:	discard ReadBytes () ensure Vitteb in Rutes (int. vitteb in Rutes)	void void byte
•	getByte (ntindex)	byte
•	getShort (int index)	short short
•	getUn signed Short (int index)	int
	getUnagnedNedium (int index)	int int long
•	getint (int index) pettin (ignedict (int index)	int long
	getLong (int index)	long
	get/nar (int index) get/loat (int index)	long char foat double void void void
•	getDouble (nt index)	double
	getBytes (int index, ChannelBuffer dst, int length)	void
•	getBytes (int index, ChannelBuffer dst, int dstindex, int length) oetflytes (int index, byte dstf)	: void : void
•	ma Sind and control of a contro	void void void void void int
•	getBytes (int index, OutputSteam out, int length)	void
•	getBytes (int index, GatheringByteChannel out, int length) setByte (int index, int value)	int
•	setShort (int index, int value)	void
•	settledium (int index, int value) settlet (int index, int value)	void
•	setLong (int index, long value)	void
	setFloat (intindex, first value)	void void void void void void void void
•	setDouble (nt index, double value) setBytes int index. ChannelBuffer set)	void
	setBytes (int index, ChannelBuffer sic, int length)	void
•	exibytes (int index, Unanneiguiter sic, int scindex, int length) exibytes (int index, byte sic[])	void
•	setBytes (int index, byte src(), int srcindex, int length) setBytes (int index, ByteBytes src)	
•	mtBytes (int index, inputStream in, int length)	void int int
	setBytes (int index, ScatteringByteChannel in, int length) setZero (int index, int length)	ont void
•	setZers (oi nee, rel heggs) setZers (oi nee, rel heggs) medible ()	void byte short short int int
	readShot ()	short
•	readUnsignedShort ()	int
	readUndignedMedium ()	int
•	readint () readint () reading enedint ()	
•	readLong ()	long long
•	readChar() readFloat()	foat
•	readDouble ()	double
	readBytes (Charnel Bu flerindexFinder indexFinder)	ChannelBuffer
•	readSlice (int length) readSlice (Channel BufferindesFinder indesFinder)	ChannelBuffer ChannelBuffer ChannelBuffer void void void
•	readBytes (Channel Buffer dst)	void
•	readDytes (Channel Buffer dst, int length) readBytes (Channel Buffer dst, int dslindex, int length)	void
•	readBytes (byte ds(j)) madBytes (byte ds(j)) (c) definder (c) (e) (e) (b)	void void void void void int
	readBytes (ByteBuffer dist)	void
•	readBytes (OutputStream out, int length)	void
	dipOyles (int length)	void
•	supplyes (unannerouned fidex Finder Index Index) write Dyte (intivalue)	void
	writeShort (int value)	void
	maked studies () maked by a depart of () maked by a de	void void void void
•	witeLong (ong value) witeChar (int value)	void
•	writeFloat (float value)	word
•	writeBytes (ChannelBuffer arc)	void
ĺ	writeBytes (ChannelBuffer at), int length)	void void void void void
•	writeDytes (byte srdj.)	void
١	writeDytes (byte srd), int sraindex, int length) writeDytes (byteSuffer src)	void
	writeBytes (inputStreem in, int length)	void void void int int
٠	writeBytes (ScatteringByteChannel in, int length) writeZero (int length)	
•	indexOf (nt fornindex, int tolindex, byte value)	int
•	indexor ant tramingex, int tolindex, Channel BullferindexFinder indexFinder) bytexBefore (byte value)	int
١	byte-Before (Channel BufferindexFinder indexf inder)	int
	bytesBefore (int length, ChannelBu flerindexFinder indexFinder)	int
١	bytestlefore (int index, int length, byte value) bytestlefore (int index, int length, Change Buttertouteuf index indext index)	int int int
•	copy ()	ChannelBuffer
	copy (int index, int length) slice ()	ChannelBuffer ChannelBuffer ChannelBuffer ChannelBuffer
ŝ	6 of 0 control of the	ChannelBuffer
	duplicate () toByteBuffer ()	ChannelBuffer ByteBuffer
i	toByteBuffer (int index, int length)	ChannelBuffer ChanselBuffer ByteBuffer ByteBuffer ByteBuffer ByteBuffer[ByteBuffer[boolean byte[int String String String String
í	todytebu fiels () toByteBu fiels (int index, int length)	ByteBuffer[]
į	hasArray ()	boolean
	arrayOffset ()	int
١	toString (Charset charset)	String
	tooting snt index, int rength, Charset charset) toSting (String charset)iame)	String String
i	toSting (Sting charactione, ChannelButterhotexFinder terminato#inder)	String
í	to Stong (nt Index, int length, String charact Name. Channel Buffel ndexFinder term in a torFinder)	String String
á	hashCode ()	int boolean
		000000

ChannelBuffers					
BIG_ENDIAN LITTLE_ENDIAN EMPTY_BUFFER	: ByteOrder				
HEXDUMP_TABLE	: char[] = new char[256 * 4]				
< <stationitializer>></stationitializer>	1.=0.00.00.00.=0.00.00.00.00.00.00.	; void			
		: ChannelBuffer			
		: ChannelBuffer : ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
	dynamicBuffer (int estimatedLength)	: ChannelBuffer			
	dynamicBuffer (Byte Order endianness, int estimatedLength)	: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer : ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
	compositeBuffer (ByteOrder endianness, List <channelbuffer> components)</channelbuffer>	: ChannelBuffer			
		: ChannelBuffer			
	H 500 (4.5 March 1991) 50	: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer : ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
	copiedBuffer (ChannelBuffer buffers)	: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer : ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: ChannelBuffer			
		: String			
		: String : int			
		; boolean			
		; int			
		int			
		: int			
	swap Short (short value)	: short			
		: int			
		: int			
		: long			
		: int			
		: int			
	firstIndexOf (ChannelBufferbuffer, int fromIndex, inttoIndex, ChannelBufferIndexFinder indexFinder) lastIndexOf (ChannelBufferbuffer, int fromIndex, inttoIndex, ChannelBufferIndexFinder indexFinder)				
		: ByteBuffer			
		: String			
< <constructor>></constructor>	ChannelBuffers ()				

Netty数据流分析

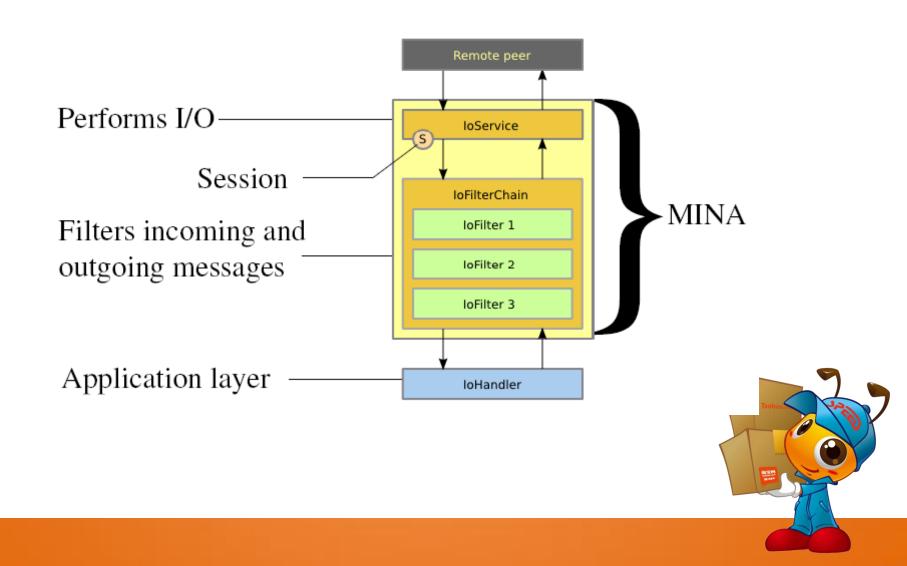
- ▶ 服务器启动
- ▶ 服务器主通道监听
- ★ 服务器子通道开通
- ★ 客户端启动
- ◆ 客户端主通道监听
- ◆ 客户端子通道开通



Netty VS Mina



Mina 架构



Mina 服务器端架构

№ IO Acceptor

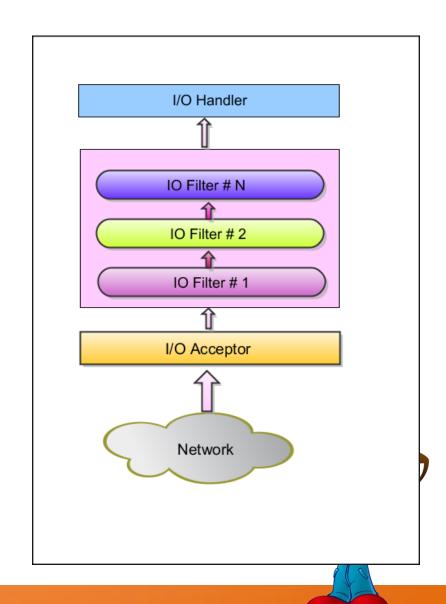
- 实现了loService
- 监听网络请求或数据包
- 新连接,创建session

> IO Filter

- 过滤和传输
- 编码解码

№ IO Handler

- 实现业务逻辑



Mina 客户架构

№ IO Connector

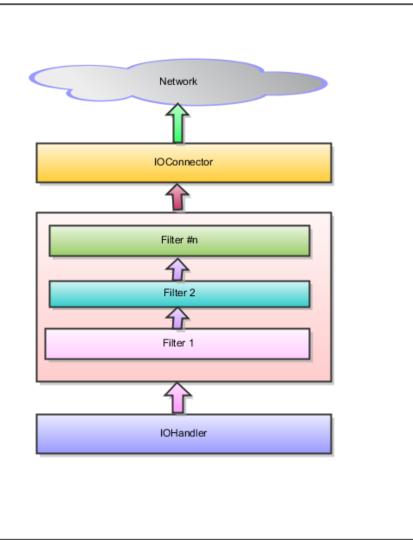
- 实现IO Server
- 连接到Socket
- bind到Server
- 创建session

⋄ IO Filter

- 过滤和传输
- 编码解码

№ IO Handler

- 实现业务逻辑





Netty VS Mina

- ▶ Netty基于Pipeline处理, Mina基于Filter过滤
- ★ Netty的事件驱动模型具有更好的扩展性和易用性
- ▶ Https , SSL , PB , RSTP , Text &Binary等协议支持
- ★ Netty中UDP传输有更好的支持
- 會方测试Netty比Mina性能更好





Q & A

雷腾 L.T

Email: leiteng@taobao.com