昵称：Tiny&zzh
园龄：5年2个月
粉丝：50
关注：15
+加关注

< 2017年3月 >

| 日 | 一 | 二 | 三 | 四 | 五 | 六 |
|---|---|---|---|---|---|---|
| 26 | 27 | 28 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 26 | 27 | 28 | 29 | 30 | 31 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |

搜索

找找看
谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

Netty(11)
silverlight(7)
java(7)
Unity3D(7)
中文教程(5)
Netty 4(5)
NIO(4)
Okra(3)
Socket(3)
C#(3)
更多

随笔分类

Java(4)
Netty(7)
Okra(3)
Unity3D(5)

随笔档案

2016年6月 (1)
2016年4月 (4)
2015年6月 (1)
2015年3月 (1)
2015年2月 (1)
2015年1月 (1)
2014年10月 (2)
2014年9月 (2)
2014年8月 (3)
2014年7月 (2)
2014年6月 (1)
2014年4月 (5)
2014年1月 (2)
2013年12月 (4)
2013年11月 (2)
2013年10月 (1)
2013年6月 (2)
2013年4月 (1)
2012年11月 (1)
2012年9月 (2)
2012年5月 (1)
2012年3月 (2)
2012年2月 (1)
2012年1月 (1)
2011年12月 (1)

相册

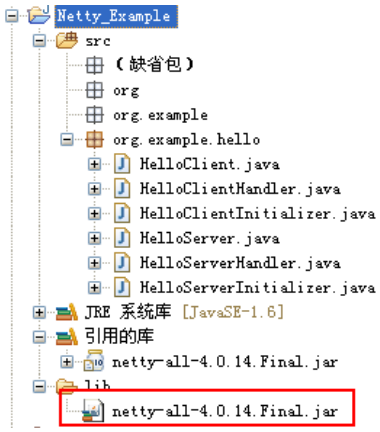
**Netty4.x中文教程系列(二) Hello World !**

在中国程序界。我们都是学着Hello World !慢慢成长起来的。逐渐从一无所知到熟悉精通的。

第二章就从Hello World 开始讲述Netty的中文教程。

首先创建一个Java项目。引入一个Netty 框架的包。这个步骤我在本系列教程的后面就不在重复了。

先上一张我示例的项目工程图给大家看一下:




## 1.下载并为项目添加Netty框架

Netty的包大家可以从Netty官网:http://netty.io/downloads.html 下载


如图所示: Netty提供了三个主要版本的框架包给大家下载。

3.9版本Final 说明这个版本是3.x版本中最新的版本。final意味着功能不再继续添加更新。仅为修改bug等提供继续的更新。

5.x版本由于是开始。不能排除是否稳定运行等问题。加上5.x在4.x的版本上略微修改的。在5.x稳定之前。不推荐大家学习使用。

本教程是基于Netty4.x版本的。

笔者也是从3.6版本，经过了相当痛苦的一段时间才算是真正的过度到4.x版本。


下载之后解压缩。大家可以看到这样一个目录结构。非常的清晰。

第一个文件夹jar是jar包的文件夹。第二个javadoc是API文档。第三个license文件夹是开源的授权文件(可以直接无视)。

javadoc文件夹下面是一个jar包。可以直接解压缩出来。解压缩之后的文件夹就是api文档(以网页的形势展现)。

jar文件夹里面有很多的jar包和一个all-in-one文件夹。都是Netty框架的组成部分。all-in-one里面有两个文件一个是jar包，另一个是对应的source源代码包。这样做的目的是为了给程序员有选择的添加自己所需要的包。

假如读者是初学者的话。推荐直接套用all-in-one里面的jar包。假如你熟悉Netty的话可以根据自己的项目需求添加不同的jar包。

## 2.创建Server 服务端

最新评论

1. Re:Netty4.x中文教程系列(一) 目录及概述
大哥，云盘地址失效了，能再分享一次吗？
--痞子色子
2. Re:[工具分享]JetBrains ReSharper 9.0 正式版和注册码
谢谢博主。也 谢谢 ReSharper。
--InkFx
3. Re:Netty4.x中文教程系列(二) Hello World !
楼主，如果有多个客户端，如何实现在已知客户端ip和port情况下在服务端向客户端发数据呢？
--1115405079
4. Re:Unity3D中uGUI事件系统简述及使用方法总结
总结得挺好
--离落
5. Re:Netty4.x中文教程系列(七)UDP协议
嗯嗯，谢谢楼主！我已经用另外一种方法可以啦，非常感谢您的回复！非常感动！不知楼主是否使用过Netty实现P2P呢？
--雷泡泡

阅读排行榜

1. Netty4.x中文教程系列(二) Hello World !(15455)
2. Netty4.x中文教程系列(一) 目录及概述(15314)
3. Unity3D中uGUI事件系统简述及使用方法总结(9892)
4. Netty4.x中文教程系列(四) ChannelHandler(7550)
5. 【推荐】《Netty in action》书籍(7494)

评论排行榜

1. Netty4.x中文教程系列(二) Hello World !(9)
2. Netty4.x中文教程系列(七)UDP协议(5)
3. Netty4.x中文教程系列(一) 目录及概述(4)
4. 【推荐】《Netty in action》书籍(4)
5. [工具分享]JetBrains ReSharper 9.0 正式版和注册码(3)

推荐排行榜

1. [工具分享]JetBrains ReSharper 9.0 正式版和注册码(3)
2. JAVA笔记-如何将百万级数据高效的导出到Excel表单(2)
3. Netty4.x中文教程系列(五)编解码器Codec(2)
4. 【推荐】《Netty in action》书籍(1)
5. JAVA数据库连接池的革命 -- 从BoneCP到HikariCP(1)

### 2.1创建一个 HelloServer

```
package org.example.hello;


import io.netty.bootstrap.ServerBootstrap;
import io.netty.channel.ChannelFuture;
import io.netty.channel.EventLoopGroup;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.nio.NioServerSocketChannel;

public class HelloServer {

    /**
     * 服务端监听的端口地址
     */
    private static final int portNumber = 7878;

    public static void main(String[] args) throws InterruptedException {
        EventLoopGroup bossGroup= new NioEventLoopGroup();
        EventLoopGroup workerGroup= new NioEventLoopGroup();
        try {
            ServerBootstrap b= new ServerBootstrap();
            b.group(bossGroup, workerGroup);
            b.channel(NioServerSocketChannel.class);
            b.childHandler(new HelloServerInitializer());

            // 服务器绑定端口监听
            ChannelFuture f =b.bind(portNumber).sync();
            // 监听服务器关闭监听
            f.channel().closeFuture().sync();

            // 可以简写为
            /* b.bind(portNumber).sync().channel().closeFuture().sync();*/
        } finally {
            bossGroup.shutdownGracefully();
            workerGroup.shutdownGracefully();
        }
    }
}
```

EventLoopGroup 是在4.x版本中提出来的一个新概念。用于channel的管理。服务端需要两个。和3.x版本一样，一个是boss线程一个是worker线程。

```
b.childHandler(new HelloServerInitializer());        //用于添加相关的Handler
```

服务端简单的代码，真的没有办法在精简了感觉。就是一个绑定端口操作。

### 2.2创建和实现HelloServerInitializer

在HelloServer中的HelloServerInitializer在这里实现。

首先我们需要明确我们到底是要做什么的。很简单。HelloWorld!。我们希望实现一个能够像服务端发送文字的功能。服务端假如可以最好还能返回点消息给客户端，然客户端去显示。

需求简单。那我们下面就准备开始实现。

DelimiterBasedFrameDecoder Netty在官方网站上提供的示例显示 有这么一个解码器可以简单的消息分割。

其次 在decoder里面我们找到了String解码编码器。着都是官网提供给我们的。

```
1 package org.example.hello;
2
3 import io.netty.channel.ChannelInitializer;
4 import io.netty.channel.ChannelPipeline;
5 import io.netty.channel.socket.SocketChannel;
6 import io.netty.handler.codec.DelimiterBasedFrameDecoder;
7 import io.netty.handler.codec.Delimiters;
8 import io.netty.handler.codec.string.StringDecoder;
9 import io.netty.handler.codec.string.StringEncoder;
10
11 public class HelloServerInitializer extends ChannelInitializer<SocketChannel>{
12
```

```
15          ChannelPipeline pipeline =ch.pipeline();
16
17          // 以("\n")为结尾分割的 解码器
18          pipeline.addLast("framer",new DelimiterBasedFrameDecoder(8192,
Delimiters.lineDelimiter()));
19
20          // 字符串解码 和 编码
21          pipeline.addLast("decoder",new StringDecoder());
22          pipeline.addLast("encoder",new StringEncoder());
23
24          // 自己的逻辑Handler
25          pipeline.addLast("handler",new HelloServerHandler());
26      }
27  }
```

上面的三个解码和编码都是系统。

另外我们自己的Handler怎么办呢。在最后我们添加一个自己的Handler用于写自己的处理逻辑。

### 2.3 增加自己的逻辑HelloServerHandler

自己的Handler我们这里先去继承extends官网推荐的SimpleChannelInboundHandler<C> 。在这里C，由于我们需求里面发送的是字符串。这里的C改写为String。

```
 1 package org.example.hello;
 2
 3 import java.net.InetAddress;
 4
 5 import io.netty.channel.ChannelHandlerContext;
 6 import io.netty.channel.SimpleChannelInboundHandler;
 7
 8 public class HelloServerHandler extends SimpleChannelInboundHandler<String>{
 9
10     @Override
11     protected void channelRead0(ChannelHandlerContext ctx, String msg)throws
Exception {
12         // 收到消息直接打印输出
13         System.out.println(ctx.channel().remoteAddress() + " Say : " msg);
14
15         // 返回客户端消息 - 我已经接收到了你的消息
16         ctx.writeAndFlush("Received your message !\n");
17     }
18
19     /*
20      *
21      * 覆盖 channelActive 方法 在channel被启用的时候触发 (在建立连接的时候)
22      *
23      * channelActive 和 channelInActive 在后面的内容中讲述，这里先不做详细的描述
24      * */
25     @Override
26     public void channelActive(ChannelHandlerContext ctx)throws Exception {
27
28         System.out.println("RamoteAddress : " + ctx.channel().remoteAddress() + "
active !");
29
30         ctx.writeAndFlush( "Welcome to " +
InetAddress.getLocalHost().getHostName() + " service!\n");
31
32         super.channelActive(ctx);
33     }
34 }
```

在channelHandlerContent自带一个writeAndFlush方法。方法的作用是写入Buffer并刷入。

注意:在**3.x**版本中此处有很大区别。在**3.x**版本中**write()**方法是自动**flush**的。在**4.x**版本的前面几个版本也是一样的。但是在**4.0.9**之后修改为**WriteAndFlush**。普通的**write**方法将不会发送消息。需要手动在**write**之后**flush()**一次

这里channeActive的意思是当连接活跃(建立)的时候触发.输出消息源的远程地址。并返回欢迎消息。

channelRead0 在这里的作用是类似于3.x版本的messageReceived()。可以当做是每一次收到消息是触发。

我们在这里的代码是返回客户端一个字符串"Received your message !".

## 2.Client客户端

类似于服务端的代码。我们不做特别详细的解释。

直接上示例代码:

```
 1 package org.example.hello;
 2
 3 import io.netty.bootstrap.Bootstrap;
 4 import io.netty.channel.Channel;
 5 import io.netty.channel.EventLoopGroup;
 6 import io.netty.channel.nio.NioEventLoopGroup;
 7 import io.netty.channel.socket.nio.NioSocketChannel;
 8
 9 import java.io.BufferedReader;
10 import java.io.IOException;
11 import java.io.InputStreamReader;
12
13 public class HelloClient {
14
15     public static String host = "127.0.0.1";
16     public static int port = 7878;
17
18     /**
19      * @param args
20      * @throws InterruptedException
21      * @throws IOException
22      */
23     public static void main(String[] args) throws InterruptedException,
IOException {
24         EventLoopGroup group =new NioEventLoopGroup();
25         try {
26             Bootstrap b =new Bootstrap();
27             b.group(group)
28             .channel(NioSocketChannel.class)
29             .handler(new HelloClientInitializer());
30
31             // 连接服务端
32             Channel ch =b.connect(host, port).sync().channel();
33
34             // 控制台输入
35             BufferedReader in =new BufferedReader(new
InputStreamReader(System.in));
36             for (;;) {
37                 String line =in.readLine();
38                 if (line == null) {
39                     continue;
40                 }
41                 /*
42                  * 向服务端发送在控制台输入的文本 并用"\r\n"结尾
43                  * 之所以用\r\n结尾 是因为我们在handler中添加了
DelimiterBasedFrameDecoder 帧解码。
44                  * 这个解码器是一个根据\n符号位分隔符的解码器。所以每条消息的最后必须加上\n
否则无法识别和解码
45                  * */
46                 ch.writeAndFlush(line + "\r\n");
47             }
48         } finally {
49             // The connection is closed automatically on shutdown.
50             group.shutdownGracefully();
51         }
52     }
53 }
```

下面的是HelloClientInitializer代码貌似是和服务端的完全一样。我没注意看。其实编码和解码是相对的。多以服务端和客户端都是解码和编码。才能通信。

```
 1 package org.example.hello;
 2
 3 import io.netty.channel.ChannelInitializer;
 4 import io.netty.channel.ChannelPipeline;
 5 import io.netty.channel.socket.SocketChannel;
 6 import io.netty.handler.codec.DelimiterBasedFrameDecoder;
 7 import io.netty.handler.codec.Delimiters;
 8 import io.netty.handler.codec.string.StringDecoder;
 9 import io.netty.handler.codec.string.StringEncoder;
10
11 public class HelloClientInitializer extends ChannelInitializer<SocketChannel>{
12
13     @Override
14     protected void initChannel(SocketChannel ch) throws Exception {
15         ChannelPipeline pipeline =ch.pipeline();
16
17         /*
18          * 这个地方的 必须和服务端对应上。否则无法正常解码和编码
19          *
20          * 解码和编码 我将会在下一张为大家详细的讲解。再次暂时不做详细的描述
21          *
22          * */
23         pipeline.addLast("framer",new DelimiterBasedFrameDecoder(8192,
Delimiters.lineDelimiter()));
24         pipeline.addLast("decoder",new StringDecoder());
25         pipeline.addLast("encoder",new StringEncoder());
26
27         // 客户端的逻辑
28         pipeline.addLast("handler",new HelloClientHandler());
29     }
30 }
```

HellClientHandler：

```
 1 package org.example.hello;
 2
 3 import io.netty.channel.ChannelHandlerContext;
 4 import io.netty.channel.SimpleChannelInboundHandler;
 5
 6 public class HelloClientHandler extends SimpleChannelInboundHandler<String>{
 7
 8     @Override
 9     protected void channelRead0(ChannelHandlerContext ctx, String msg)throws
Exception {
10
11         System.out.println("Server say : " +msg);
12     }
13
14     @Override
15     public void channelActive(ChannelHandlerContext ctx)throws Exception {
16         System.out.println("Client active ");
17         super.channelActive(ctx);
18     }
19
20     @Override
21     public void channelInactive(ChannelHandlerContext ctx)throws Exception {
22         System.out.println("Client close ");
23         super.channelInactive(ctx);
24     }
25 }
```

本教程的示例源代码:http://pan.baidu.com/s/1hABzK#dir

大家可以再我的百度云盘里面找到。

下面上几张成果图:

客户端在连接建立是输出了Client active 信息，并收到服务端返回的Welcome消息。

输入Hello World ! 回车发送消息。服务端响应返回消息已接受。

```
Server say : Welcome to PC-201302271653 service!
 Hello World !
Server say : Received your message !
```

1.客户端控制台截图

```
2013-12-27 8:58:30 io.netty.util.internal.Pl
信息: Your platform does not provide complet
RamoteAddress : /127.0.0.1:3672 active !
/127.0.0.1:3672 Say :  Hello World !
```

2.服务端控制台截图

**作者**：TinyZ
出处：http://www.cnblogs.com/zou90512/
关于作者：努力学习，天天向上。不断探索学习，提升自身价值。记录经验分享。
本文版权归作者和博客园共有，欢迎转载，但未经作者同意必须保留此段声明，且在文章页面明显位置给出原文链接
如有问题，可以通过 zou90512@126.com 联系我，非常感谢。
笔者网店: http://aoleitaisen.taobao.com. 欢迎广大读者围观

分类: Netty

标签: Netty, 中文教程, java, NIO, Netty 4, Hello World

好文要顶 关注我 收藏该文

Tiny&zzh
关注 - 15
粉丝 - 50
+加关注

1 0

« 上一篇：Netty4.x中文教程系列(一) 目录及概述
» 下一篇：Netty4.x中文教程系列(三) Hello World ！ 详解

posted @ 2013-12-26 18:14 Tiny&zzh 阅读(15455) 评论(9) 编辑 收藏

评论列表

#1楼 2014-04-21 21:32 当象爱上猫

请问博主，为什么服务端发数据用\n结尾，而客户端必须是\r\n呢？

支持(0) 反对(0)

#2楼[楼主 ] 2014-04-23 09:30 Tiny&zzh

@ 当象爱上猫
\r是为了在控制台输出的时候能够换行显示。否则会堆在一起
\n是帧解码器的分隔符

支持(1) 反对(0)

#3楼 2015-05-27 08:47 l-d-s

引用

\r是为了在控制台输出的时候能够换行显示。否则会堆在一起
\n是帧解码器的分隔符

也就是说，如果不是为了显示，这个\r是可以不用添加的。

支持(0) 反对(0)

#4楼 2015-06-02 11:08 博杨

绑定端口的服务端没有编译错误，但是老报这个错
java.lang.NoClassDefFoundError: java/lang/AutoCloseable
at java.lang.ClassLoader.defineClass1(Native Method)
at java.lang.ClassLoader.defineClass(ClassLoader.java:621)
at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:124)
at java.net.URLClassLoader.defineClass(URLClassLoader.java:260)
at java.net.URLClassLoader.access$000(URLClassLoader.java:56)

at java.net.URLClassLoader.findClass(URLClassLoader.java:188)
at java.lang.ClassLoader.loadClass(ClassLoader.java:307)
at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:301)
at java.lang.ClassLoader.loadClass(ClassLoader.java:252)
at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:320)
at java.lang.ClassLoader.defineClass1(Native Method)
at java.lang.ClassLoader.defineClass(ClassLoader.java:621)
at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:124)
at java.net.URLClassLoader.defineClass(URLClassLoader.java:260)
at java.net.URLClassLoader.access$000(URLClassLoader.java:56)
at java.net.URLClassLoader$1.run(URLClassLoader.java:195)
at java.security.AccessController.doPrivileged(Native Method)
at java.net.URLClassLoader.findClass(URLClassLoader.java:188)
at java.lang.ClassLoader.loadClass(ClassLoader.java:307)
at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:301)
at java.lang.ClassLoader.loadClass(ClassLoader.java:252)
at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:320)
Caused by: java.lang.ClassNotFoundException: java.lang.AutoCloseable
at java.net.URLClassLoader$1.run(URLClassLoader.java:200)
at java.security.AccessController.doPrivileged(Native Method)
at java.net.URLClassLoader.findClass(URLClassLoader.java:188)
at java.lang.ClassLoader.loadClass(ClassLoader.java:307)
at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:301)
at java.lang.ClassLoader.loadClass(ClassLoader.java:252)
at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:320)
... 24 more
Exception in thread "main"
不知道你有没有遇到过这个问题

支持(0) 反对(0)

---

#5楼[楼主 ] 2015-06-02 22:31 Tiny&zzh

@ 博杨
这个错误是因为AutoCloseable接口找不到。这个接口是在jdk1.7以上版本才有的。
Netty.io官网上面虽然写着支持4.x版本支持最低jdk1.6版本。但是github上面的需求已经是最新稳定版本依赖jdk1.7了。

这应该算是netty官网的一点疏忽把。推荐关注github的netty项目。项目源码和更新进度，bug解决情况上面都是有追踪的

支持(0) 反对(0)

---

#6楼 2015-06-03 08:52 博杨

@ Tiny&zzh
恩，昨天换了一个jdk就好了

支持(0) 反对(0)

---

#7楼 2016-03-16 14:38 ken_wangxr

客户端消息发出去，服务端没有响应啊，怎么回事？解码器搞错了我

支持(0) 反对(0)

---

#8楼[楼主 ] 2016-03-17 15:50 Tiny&zzh

@ ken_wangxr
1. 检查确认客户端和服务端使用的地址和端口是否一致
2. 检查确认一下客户端是否消息发送出去了。使用的write还是writeAndFlush
2. 检查确认帧解码器是否统一。示例使用的是基于行的解码。发消息后面需要带"\r\n"。

支持(0) 反对(0)

---

#9楼 2016-08-23 00:02 1115405079

楼主，如果有多个客户端，如何实现在已知客户端ip和port情况下在服务端向客户端发数据呢？

支持(0) 反对(0)

---

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

最新IT新闻:
· 死磕微信！支付宝推“奖励金” 付钱立得/最高999元
· Linux Kernel 4.11首个候选版本开放下载
· 微软透露Windows 10游戏模式细节：提升低端PC游戏性能
· 今日Google Doodle：纪念印尼科莫多国家公园成立37周年

最新知识库文章:
· 垃圾回收原来是这么回事
· 「代码家」的学习过程和学习经验分享
· 写给未来的程序媛
· 高质量的工程代码为什么难写
· 循序渐进地代码重构
» 更多知识库文章...

Copyright ©2017 Tiny&zzh