NanguoCoffee

分析Netty工作流程

分析Netty工作流程：

下面以Netty中Echo的例子进行流程跟踪，并简要的

服务器启动->客户端连接-> 服务器处理连接-> 服务器处理客户端数据<-> 客户端处理服务器数据

1：客户端连接：

我们直接看这行代码：

bootstrap.connect(**new** InetSocketAddress(host, port));

通过帮助类ClientBootstrap来连接服务器。

Debug源码进去发现最后是某个Channel类进行connect操作。

而这个Channel是如何来的呢？其实是从前面的 ChannelFactory和ChannelPipelineFactory得到的。

Channel.connect-> AbstractChannel.connect->Channels.connect(…);

Channels是Channel的帮助类，封装一些常用的操作。在封装操作时，基本都是触发事件。

这里发起一个connectd的Downstream的事件。

所有的事件都是丢给ChannelPipeline进行管理，ChannelPipeline使用了责任链模式来将事件传送给注册到Pipeline中的ChannelHandler，由ChannelHandler进行处理。如果遍历了所有的ChannelHandler后则交给ChannelSink进行处理，ChannelSink根据不同的事件进行不同的处理，对于connect事件，ChannelSink发送连接操作后则将该Channel注册到NioWorker中，以后的任何事件都通过NioWorker（封装selector的操作）来进行处理。

客户端连接的流程为：

ClientBootstrap.connect –>Channel.connect->

AbstractChannel.connect->Channels.connect(…)

-> 发送connect事件-> ChannelSink->发起实际的连接操作->将Channel注册给Nioworker

2：服务器启动：

bootstrap.bind(…)-> 触发ServerSocketChannel.*open*()的事件

->捕捉open事件,channel.bind-> Channels.bind(…) -> 发起bind命令-> PipelineSink进行处理-> 使用socket进行bind，等待连接事件。

3: 服务器处理连接：

服务器启动后，NioServerSocketPipelineSink.Boss.run()在监听accept事件-> 捕捉到accept事件 -> 将NioWorker进行注册NioSocketChannel

-> 向java.nio.SocketChannel注册op_read的监听。

4：客户端开始向服务器发送数据：

当客户端连接Server后，就会发起Connected的upstream事件->通过Pipeline进行处理-> SimpleChannelUpstreamHandler.handleUpstream()->

EchoClientHandler.channelConnected()

5:服务器端接收并处理数据

接收数据：

NioWorker.run()->nioworker. processSelectedKeys()->

Nioworker. Read()将从SocketChannel读取的数据封装成

ChannelBuffer ->发送upstream事件:*fireMessageReceived*(channel,buffer) –> 由注册到Pipeline中的Hanlder进行处理:EchoServerHandler. messageReceived(…)

发送数据：

e.getChannel().write(e.getMessage());->Channels.write()->

< 2010

日	一	二
28	29	30
5	6	7
12	13	14
19	20	21
26	27	28
2	3	4

导航
博客园
首页
新随笔
联系
订阅XML
管理

统计
随笔 - 24
文章 - 0
评论 - 0
引用 - 0

公告
昵称：NanguoCoff
园龄：6年8个月
粉丝：4
关注：0
+加关注

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(2)
utf-8(2)
unicode(2)
upload(1)
编码(1)
面向对象(1)
问题排查(1)
NIO(1)
Singleton(1)
spring 事务管理(1)
更多

随笔分类
Database(3)
Java
Java Concurrency(
Java IO(2)
JVM(1)
Others(8)
Spring(1)
编码相关(5)
性能优化(2)

随笔档案
2013年3月 (6)
2011年2月 (1)
2011年1月 (2)
2010年12月 (3)
2010年11月 (12)

阅读排行榜
1. 分析Netty工作流程
2. JAVA 编码方式(4
3. 统一Spring 和 非
(2845)
4. 如何与第三方接口
5. 在Netty中使用Ap
fileupload(1737)

推荐排行榜
1. 分析Netty工作流程

发起downstream事件-> NioServerSocketPipelineSink. handleAcceptedSocket()将向外写的事件放入Channel中,然后通过NioWorker.writeFromUserCode()进行发送。

6:客户端：客户端的流程和服务器端类似。

总结：

1： Netty将操作封装成事件，比如： 发起连接时，产生connect的downstream事件。连接完毕后，产生upstream的connect事件。

2：所有的事件都是放入Pipeline进行传送，传送的过程中可能被注册到pipeline中的Handler进行处理

3：在Pipeline传送完后，都必须都通ChannelSink进行处理。Sink默认处理了琐碎的操作，必须连接、读写等等。

4：Channels:几乎所有的操作都能在这里找到，当然Channels一般是发送事件

5：NioWorker: 处理IO事件的核心类，并承担了分发的责任。

分类: Java IO

标签: netty, NIO

NanguoCoffee
关注 - 0
粉丝 - 4
+加关注

2 0

« 上一篇：支持start,stop,restart的运行脚本
» 下一篇：JVMS3总结之JVM基本结构

posted on 2010-12-10 16:48 NanguoCoffee 阅读(5898) 评论(0) 编辑 收藏

2. 知道为啥HashMa
的次幂? (1)
3. 在Netty中使用Ap
fileupload(1)
4. Java字符相关知识
5. JVMS3总结之JVM