Netty In Action中文版 - 第十六章：从EventLoop取消注册和重新注册

标签： java netty nio

2014-09-15 21:26 评论

分类： java (31)

目录(?) [+]

本章介绍

- EventLoop
- 从EventLoop注册和取消注册
- 在Netty中使用旧的Socket和Channel

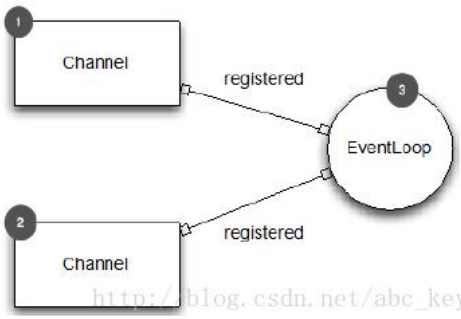
Netty提供了一个简单的方法来连接Socket/Channel，这是在Netty之外创建并转移他们的责任到Netty。这允许
缝方式一步一步迁移到Netty；Netty还允许取消注册的通道来停止处理IO，这可以暂停程序处理并释放资源。

这些功能在某些情况或某种程度上可能不是非常有用，但使用这些特性可以解决一些困难的问题。举个例子，有一
络，其用户增长非常快，系统程序需要处理每秒几千个交互或消息，如果用户持续增长，系统将会处理每秒数以万计的
随着用户数的增长，系统将消耗大量的内存和CPU而导致性能低下；此时最需要做的就是改进他们，并且不要花太多的
况下，系统必须保持功能正常能处理日益增长的数据量，此时，注册/注销事件循环就派上用场了。

通过允许外部Socket/Channel来注册和注销，Netty能够以这样的方式改进旧系统的缺陷，所有的Netty程序都可
式整合到现有系统，本章将重点讲解Netty是如何整合。

16.1 注册和取消注册的Channel和Socket

前面章节讲过，每个通道需要注册到一个EventLoop来处理IO或事件，这是在引导过程中自动完成。下图显示了



上图只是显示了他们关系的一部分，通道关闭时，还需要将注册到

有时不得不处理java.nio.channels.SocketChannel或其他java.ni

们可以使用Netty来包装预先创建的java.nio.channels.Channel，然后

有的东西。下面代码显示了此功能：

```
[java] view plain copy
01. //nio
02. java.nio.channels.SocketChannel mySocket = java.nio.channels.S
03. //netty
04. SocketChannel ch = new NioSocketChannel(mySocket);
```

* 【游戏设计模式】之四 《游戏编程模式》全书内容提炼总结
* 带你开发一款给Apk中自动注入代码工具iodetools(完善篇)


```
EventLoopGroup group = new NioEventLoopGroup();
//register channel
ChannelFuture registerFuture = group.register(ch);
//de-register channel
ChannelFuture deregisterFuture = ch.deregister();
```

Netty也适用于包装OIO，看下面代码：

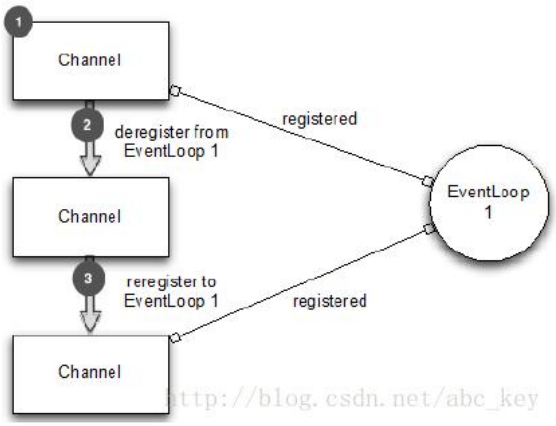
```
[java] view plain copy
//oio
Socket mySocket = new Socket("www.baidu.com", 80);
//netty
SocketChannel ch = new OioSocketChannel(mySocket);
EventLoopGroup group = new OioEventLoopGroup();
//register channel
ChannelFuture registerFuture = group.register(ch);
//de-register channel
ChannelFuture deregisterFuture = ch.deregister();
```

只有2个重点如下：

1. 使用Netty包装已创建的Socket或Channel必须使用与之对应的实现，如Socket是OIO，则使用Netty的OioSock SocketChannel是NIO，则使用NioSocketChannel。
2. EventLoop.register(...)和Channel.deregister(...)都是非阻塞异步的，也就是说它们可能不会理解执行完成，可能 ChannelFuture，我们在需要进一步操作或确认完成操作时可以添加一个ChannelFutureLister或在ChannelFutu 择哪一种方式看实际需求，一般建议使用ChannelFutureLister，应避免阻塞。

16.2 挂起IO处理

在一些情况下可能需要停止一个指定通道的处理操作，比如程序耗尽内存、崩溃、失去一些消息，此时，我们可以清理系统资源，以保持程序稳定继续处理后续消息。若这样做，最好的方式就是从EventLoop取消注册的通道，这可以事件。若需要被取消的通道再次处理事件，则只需要将该通道重新注册到EventLooop即可。看下图：



看下面代码：

```
[java] view plain copy
EventLoopGroup group = new NioEventLoopGroup();
Bootstrap bootstrap = new Bootstrap();
bootstrap.group(group).channel(NioSocketChannel.class)
        .handler(new SimpleChannelInboundHandler<ByteBuf>() {
            @Override
            protected void channelRead0(ChannelHandlerContext ctx,
                    ByteBuf msg) throws Exception {
                //remove this ChannelHandler and de-register
                ctx.pipeline().remove(this);
                ctx.deregister();
            }
        });
ChannelFuture future = bootstrap.connect(
        new InetSocketAddress("www.baidu.com", 80)).sync();
//....
Channel channel = future.channel();
//re-register channel and add ChannelFutureLister
group.register(channel).addListener(new ChannelFutureListener(
    @Override
    public void operationComplete(ChannelFuture future) throws
        if(future.isSuccess()){
            System.out.println("Channel registered");
        }else{
```

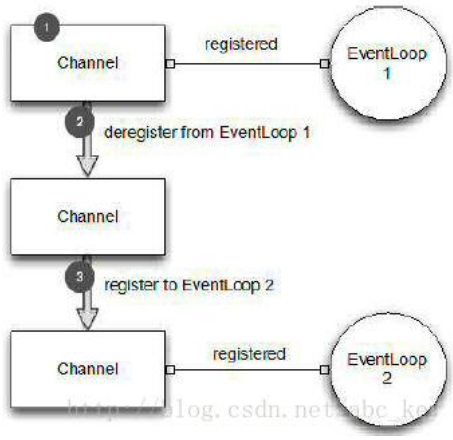
```
            System.out.println("register channel on EventLoop fail");
            future.cause().printStackTrace();
        }
    }
});
```

16.3 迁移通道到另一个事件循环

另一个取消注册和注册一个Channel的用例是将一个活跃的Channel移到另一个EventLoop，有下面一些原因可能

- 当前EventLoop太忙碌，需要将Channel移到一个不是很忙碌的EventLoop；
- 终止EventLoop释放资源同时保持活跃Channel可以继续使用；
- 迁移Channel到一个执行级别较低的非关键业务的EventLoop中。

下图显示迁移Channel到另一个EventLoop：



看下面代码：

[java] view plain copy

```
EventLoopGroup group = new NioEventLoopGroup();
final EventLoopGroup group2 = new NioEventLoopGroup();
Bootstrap b = new Bootstrap();
b.group(group).channel(NioSocketChannel.class)
        .handler(new SimpleChannelInboundHandler<ByteBuf>() {
            @Override
            protected void channelRead0(ChannelHandlerContext ctx,
                    ByteBuf msg) throws Exception {
                // remove this channel handler and de-register
                ctx.pipeline().remove(this);
                ChannelFuture f = ctx.deregister();
                // add ChannelFutureListener
                f.addListener(new ChannelFutureListener() {
                    @Override
                    public void operationComplete(ChannelFuture future)
                            throws Exception {
                        // migrate this handler register to group2
                        group2.register(future.channel());
                    }
                });
            }
        });
ChannelFuture future = b.connect("www.baidu.com", 80);
future.addListener(new ChannelFutureListener() {
    @Override
    public void operationComplete(ChannelFuture future)
            throws Exception {
        if (future.isSuccess()) {
            System.out.println("connection established");
        } else {
            System.out.println("connection attempt failed");
            future.cause().printStackTrace();
        }
    }
});
```


16.4 Summary

至此，netty in action中文版系列博文已完成了，一次不经意的baidu了

系列内容一模一样，呵呵 ，看来netty中文资料的需求还是有一些的。

顶 0　　踩 0

- 上一篇　Netty In Action中文版 - 第十三章：通过UDP广播事件
- 下一篇　Netty In Action中文版 - 第十四章：实现自定义的编码解码器

我的同类文章

java (31)

- Hibernate---在Hibernate中获取数据... 2014-12-18 阅读 220
- Netty In Action中文版 - 第十五章：选... 2014-09-15 阅读 241
- Netty In Action中文版 - 第十三章：通... 2014-09-15 阅读 347
- Netty In Action中文版 - 第十一章：W... 2014-09-15 阅读 397
- Netty In Action中文版 - 第九章：引导... 2014-09-15 阅读 309
- Spring定时任务的几种实现 2014-11-
- Netty In Action中文版 - 第十四章：实... 2014-09-
- Netty In Action中文版 - 第十二章：SP... 2014-09-
- Netty In Action中文版 - 第[illegible] 2014-09-
- Netty In Action中文版 - 第八章：附带... 2014-09-

更多文章

参考知识库

Java SE知识库
18244 关注 | 509 收录

Java EE知识库
12477 关注 | 820 收录

Java 知
20832 关

猜你在找

JavaSE高级篇——（IO流+多线程+…
Cocos2d-x 3.x高级视频教程：Ob…
Java大型企业级项目实战：VOD展…
Java基础核心技术：IO(day15-da…
Jquery入门到精通（各html、css…
Netty in Action中文版基于fina…
Netty In Action中文版 - 第六…
Netty In Action中文版 - 第七…
Netty In Action中文版 - 第十…
Netty In Action中文版 - 第十…

查看评论

暂无评论

发表评论

用 户 名：gavin2lee7
评论内容：

提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目