Java N(A)IO - 框架: Netty

Netty是一个高性能、异步事件驱动的NIO框架，提供了对TCP、UDP和文件传输的支持。作为当前最流行的NIO框架，Netty在互联网领域、大数据分布式计算领域、游戏行业、通信行业等获得了广泛的应用，一些业界著名的开源组件也基于Netty构建，比如RPC框架、zookeeper等。

NIO框架

目前流行的NIO框架非常的多。在论坛上、互联网上大家讨论和使用最多的有以下几种:

- 原生JAVA NIO框架:

JAVA NIO通信框架基于多路复用IO原理，我们将详细讲解它的工作原理。

- APACHE MINA 2:

是一个网络应用程序框架，用来帮助用户简单地开发高性能和高可扩展性的网络应用程序。它提供了一个通过Java NIO在不同的传输例如TCP/IP和UDP/IP上抽象的事件驱动的异步API。

- NETTY 4/5:

Netty是由JBOSS提供的一个java开源框架。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。我们将讲解NETTY 4 的工作原理。另外说一句: MINA和NETTY的主要作者是同一人Trustin Lee。

- Grizzly:

Grizzly是一种应用程序框架，专门解决编写成千上万用户访问服务器时候产生的各种问题。使用JAVA NIO作为基础，并隐藏其编程的复杂性。

比较好的基于NIO的开源框架(Netty)

优点

- api简单，开发门槛低
- 功能强大，内置了多种编码、解码功能
- 与其它业界主流的NIO框架对比，netty的综合性能最优
- 社区活跃，使用广泛，经历过很多商业应用项目的考验
- 定制能力强，可以对框架进行灵活的扩展

例子

```
<dependency>
     <groupId>org.jboss.netty</groupId>
     <artifactId>netty</artifactId>
     <version>3.2.5.Final</version>
</dependency>
```

- 服务端。接收客户端请求并将内容打印出来，同时发送一个消息收到回执。

```
public class NettyServer {

    private static int HEADER_LENGTH = 4;

    public void bind(int port) throws Exception {

        ServerBootstrap b = new ServerBootstrap(new
NioServerSocketChannelFactory(Executors.newCachedThreadPool(),

 Executors.newCachedThreadPool()));

        // 构造对应的pipeline
        b.setPipelineFactory(new ChannelPipelineFactory() {

            public ChannelPipeline getPipeline() throws Exception {
                ChannelPipeline pipelines = Channels.pipeline();
                pipelines.addLast(MessageHandler.class.getName(), new MessageHandler());
                return pipelines;
            }
        });
        // 监听端口号
        b.bind(new InetSocketAddress(port));
    }

    // 处理消息
    static class MessageHandler extends SimpleChannelHandler {

        public void messageReceived(ChannelHandlerContext ctx, MessageEvent e) throws Exception
{
            // 接收客户端请求
            ChannelBuffer buffer = (ChannelBuffer) e.getMessage();
            String message = new String(buffer.readBytes(buffer.readableBytes()).array(), "UTF-
8");
            System.out.println("<服务端>收到内容=" + message);

            // 给客户端发送回执
            byte[] body = "服务端已收到".getBytes();
            byte[] header =
ByteBuffer.allocate(HEADER_LENGTH).order(ByteOrder.BIG_ENDIAN).putInt(body.length).array();
            Channels.write(ctx.getChannel(), ChannelBuffers.wrappedBuffer(header, body));
            System.out.println("<服务端>发送回执,time=" + System.currentTimeMillis());

        }
    }

    public static void main(String[] args) {
        try {
            new NettyServer().bind(1088);
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
        ;
    }
}
```

- 客户端。向服务端发送一个请求，然后打印服务端响应的内容。

```
public class NettyClient {

    private final ByteBuffer readHeader  = ByteBuffer.allocate(4).order(ByteOrder.BIG_ENDIAN);
    private final ByteBuffer writeHeader = ByteBuffer.allocate(4).order(ByteOrder.BIG_ENDIAN);
    private SocketChannel    channel;

    public void sendMessage(byte[] body) throws Exception {
        // 创建客户端通道
        channel = SocketChannel.open();
        channel.socket().setSoTimeout(60000);
        channel.connect(new InetSocketAddress(AddressUtils.getHostIp(), 1088));

        // 客户端发请求
        writeWithHeader(channel, body);

        // 接收服务端响应的信息
        readHeader.clear();
        read(channel, readHeader);
        int bodyLen = readHeader.getInt(0);
        ByteBuffer bodyBuf = ByteBuffer.allocate(bodyLen).order(ByteOrder.BIG_ENDIAN);
        read(channel, bodyBuf);
        System.out.println("<客户端>收到响应内容: " + new String(bodyBuf.array(), "UTF-8") + ",长
度:" + bodyLen);
    }

    private void writeWithHeader(SocketChannel channel, byte[] body) throws IOException {
        writeHeader.clear();
        writeHeader.putInt(body.length);
        writeHeader.flip();
        // channel.write(writeHeader);
        channel.write(ByteBuffer.wrap(body));
    }

    private void read(SocketChannel channel, ByteBuffer buffer) throws IOException {
        while (buffer.hasRemaining()) {
            int r = channel.read(buffer);
            if (r == -1) {
                throw new IOException("end of stream when reading header");
            }
        }
    }

    public static void main(String[] args) {
        String body = "客户发的测试请求！";
        try {
            new NettyClient().sendMessage(body.getBytes());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```