```
            @Override
                    protected void initChannel(SocketChannel socketChannel)
throws Exception {
                        socketChannel.pipeline().addLast(new
EchoServerHandler());
                    }
                });
        System.out.println("Echo 服务器启动中...");
        try {
            //绑定端口，同步等待
            ChannelFuture channelFuture =
serverBootstrap.bind(port).sync();
            channelFuture.channel().closeFuture().sync();
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }finally {
            //优雅退出
            workerGroup.shutdownGracefully();
            frontGroup.shutdownGracefully();
        }
    }

    public static void main(String[] args) {
        int port = 8080;
        if (args.length > 0) {
            port = Integer.parseInt(args[0]);
        }
        new EchoServer(port).run();
    }
}


```

```
package com.huahuo.netty.echo.server;

import io.netty.buffer.ByteBuf;
import io.netty.buffer.Unpooled;
import io.netty.channel.ChannelHandlerContext;
import io.netty.channel.ChannelInboundHandlerAdapter;
import io.netty.util.CharsetUtil;

public class EchoServerHandler extends ChannelInboundHandlerAdapter
```

```
{

    @Override
    public void channelRead(ChannelHandlerContext ctx, Object msg)
throws Exception {
        ByteBuf data = (ByteBuf) msg;
        System.out.println("服务端接收到数据
：  "+data.toString(CharsetUtil.UTF_8));

        // 创建新的 ByteBuf,
 ctx.writeAndFlush(Unpooled.copiedBuffer("hello fireworks",
CharsetUtil.UTF_8));
    }

    @Override
    public void channelReadComplete(ChannelHandlerContext ctx) throws
Exception {
        System.out.println("client read complete");
    }

    @Override
    public void exceptionCaught(ChannelHandlerContext ctx, Throwable
cause) throws Exception {
        cause.printStackTrace();
        ctx.close();
    }
}

```

Echo服务流程分析
----

1. EventLoop 和 EventLoopGroup

这就是相当于线程和线程组的关系。

2. BootStrap

就是启动类，会自带一些配置

3. Channel

就是一个到实体的连接的桥梁，生命周期，去配置生命周期会执行的函数

4. ChannelHandler和ChannelPipline

就是用来配置Channel的处理类，pipline就是流水线

EventLoop和EventLoopGroup

EventLoop相当于一个线程，1个EventLoop可以服务多个Channel，1个Channel只有一个EventLoop，可以有多个EventLoop来优化性能，也就是EventLoopGroup。 EventLoopGroup负责分配EventLoop到新创建Channel。里面包含很多EventLoop。EventLoop会维护一个Selector模型，默认创建的线程数量是核数*2。

BootStrap

* group：把两个线程池绑定进来，可以传入单个线程池，如果传入单个线程池，而且这个线程池数量指定为1，那就是单线程的reactor模型。如果不为1那就是多线程模型，传入两个线程池那就是主从线程模型了。

* channel：用来设置IO通道的类型,一般设置为NIO
* option：用来设置TCP连接的一些参数

+ SO_BACKLOG:存放已完成三次握手的请求的等待队列的长度

– 半连接队列 tcp_max_syn_backlog
– 全连接队列 net.core.somaxconn
+ TCP_NODELAY：如果要求实时性高，那就设置成true，他就关闭Nagle算法，就不会延迟发送了。
* handler：用来设置处理器的

Channel

Channel：相当于一个通道

ChannelHandler：负责Channel的逻辑处理

ChannelPipeline：负责管理ChannelHandler的有序容器

Channel状态出现变化就会触发对应的事件

原文链接: https://juejin.cn/post/7385784332445040651