# 1 BIO

ServerSocket、Socket

## 1.1 Server

- 创建一个ServerSocket

- 开始循环监听连接

- 建立连接并获取客户端Socket

- 从客户端Socket获取输入流

- 从输入流读取数据并打印【输出】

## 1.2 Client

- 创建一个客户端Socket

- 从客户端Socket获取输出流

- 创建一个扫描流

- 扫描流循环读取键盘输入的数据，并填充到输出流【输入】

# 2 NIO

Selector、SelectionKey、

ServerSocketChannel、SocketChannel

## 2.1 Server

- 创建一个Selector

- 创建一个ServerSocketChannel

- ServerSocketChannel绑定端口

- ServerSocketChannel设置非阻塞

- ServerSocketChannel注册到Selector，并指定用于监听连接的SelectionKey

- 开始循环监听SelectionKey

- 获取待处理的SelectionKey

- 遍历每一个SelectionKey

- 如果是连接事件：

> - 建立连接并获取客户端SocketChannel

> - 将客户端SocketChannel配置为非阻塞

> - 将客户端SocketChannel注册到Selector，并指定SelectionKey

- 如果是读事件

> - 获取客户端SocketChannel

> - 从客户端SocketChannel读取数据

> - 将数据写入到除了消息发送方的其它SocketChannel

- 移除SelectionKey

## 2.2 Client

- 创建一个客户端SocketChannel

- SocketChannel与远程地址建立连接

- SocketChannel设置为非阻塞

- SocketChannel注册到Selector，并指定用于监听读事件的SelectionKey

- 创建一个扫描流

- 扫描流读取键盘输入并且写入到SocketChannel【输入】

- 创建一个子线程任务：

> - 获取待处理事件的SelectionKey

> - 如果是读事件

> - 从客户端SocketChannel读取数据并打印【输出】

> - 移除SelectionKey

# 3 Netty

NioEventLoopGroup、

ServerBootstrap、Bootstrap、

NioServerSocketChannel、NioSocketChannel、

ChannelPipeline、ChannelHandler、ChannelFuture

## 3.1 Server

- 创建一个bossGroup

- 创建一个workerGroup

- 创建一个ServerBootstrap

- ServerBootstrap配置bossGroup、workerGroup

- ServerBootstrap配置NioServerSocketChannel类型

- ServerBootstrap添加一些通道相关配置

> - SO\_BACKLOG、SO\_KEEPALIVE、TCP\_NODELAY

- ServerBootstrap往ChannelPipeline中添加ChannelHandler

- ServerBootstrap绑定端口并使用ChannelFuture进行监听

- ChannelFuture添加自定义监听器

- 使用ChannelFuture监听关闭通道事件

## 3.2 Client

- 创建一个group

- 创建一个Bootstrap

- Bootstrap配置group

- Bootstrap配置NioSocketChannel类型

- Bootstrap往ChannelPipeline中添加ChannelHandler

- Bootstrap绑定远程地址并使用ChannelFuture进行监听

- 使用ChannelFuture监听关闭通道事件

## 3.3 ChannelHandler

### 3.3.1 具体ChannelHandler

ProtobufEncoder、ProtobufDecoder、

HttpServerCodec、HttpObjectAggregator、ChunkedWriteHandler、WebSocketServerProtocolHandler、

IdleStateHandler、LoggingHandler

### 3.3.2 抽象ChannelHandler

接口或者抽象类：

ChannelInboundHandlerAdapter、ChannelOutboundHandlerAdapter、

StringEncoder、StringDecoder、

MessageToByteEncoder、ByteToMessageDecoder、

SimpleChannelInboundHandler、ReplayingDecoder

方法：

handlerAdded、handlerRemoved、

channelRegistered 、channelUnregistered 、

channelActive、channelInactive、

channelRead、channelReadComplete、

decode、encode、

userEventTriggered、

exceptionCaught

### 3.3.3 ChannelHandler的生命周期

建立连接（添加处理器、注册通道、激活通道）：handlerAdded –> channelRegistered –> channelActive

读取数据（读取通道、通道读取完毕）：channelRead –> channelReadComplete

关闭连接（取消激活通道、取消注册通道、移除处理器）：channelInactive –> channelUnregistered –> handlerRemoved