# 开发环境搭建

## 准备

1. 建立一个Maven项目
2. 导入netty依赖

## 服务器实例

1. Discard服务器，忽略一切消息
2. Time服务器，发送消息后立即关闭连接

具体见文件

# 二、Netty 中流数据的传输处理问题

## SocketBuffer 的警告

1.1 流传输的本质

在如 TCP/IP的以流为基础传输数据中，数据被接收后，被保存在一个 socket 接收缓冲区中。不幸的是，这个以流为基础的缓冲区 buffer 不是一个包 packet 的队列，而是一个字节 byte 队列。

1.2 TCP/IP 中的拆包和粘包

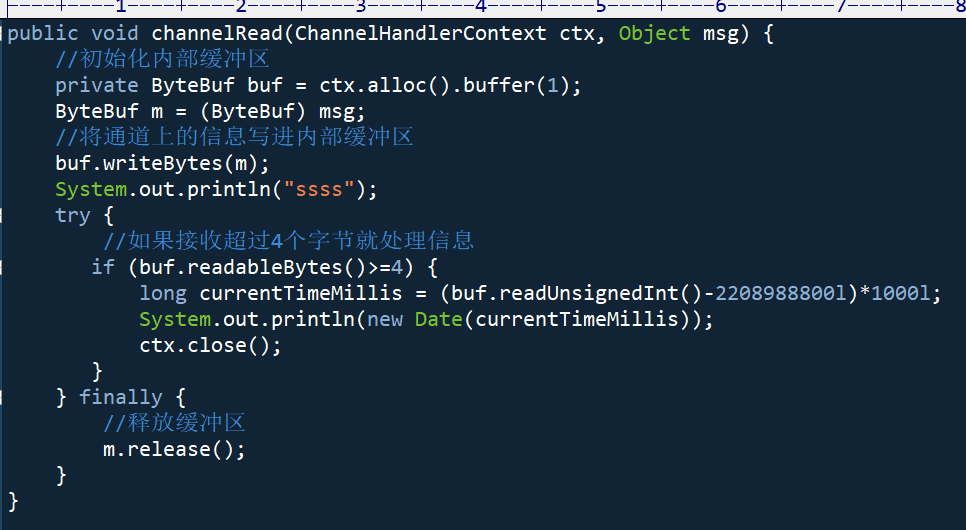
这意味着，即使你发送两个消息 message 作为 2 个独立的包，操作以系统不会把他们作为两个消息 message，而是仅仅当做一堆字节。因此，无法保证你读到的数据就是对方写的数据，这就是TCP/IP中常见的拆包、粘包问题。

## 创建内部标识缓冲区

* 1. 问题

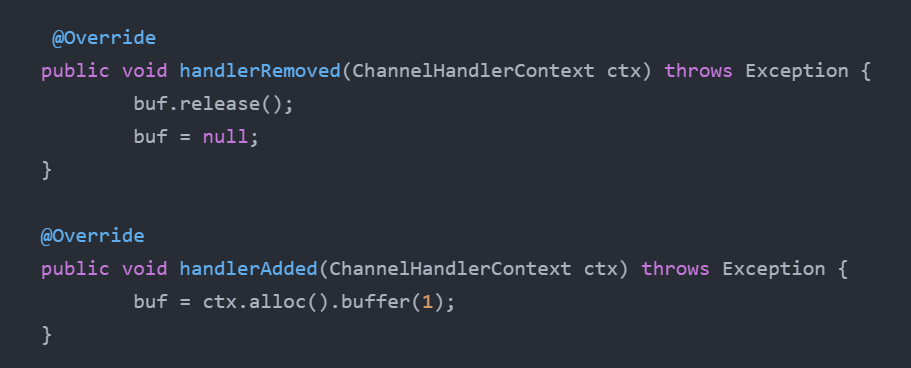
一个 32 位的整形数据是一个非常小的数据，创建一个内部积累的缓冲区，并且等待直到 4 个字节都被接收到这个内部的缓冲区中。

* 1. 解决



* 1. 生命周期方法

ChannelHandler 有两个生命周期监听方法：handlerAdded 和 handlerRemoved 方法，你可以完成任意初始化任务，只要它不会被阻塞很长的时间。



## 添加默认处理器

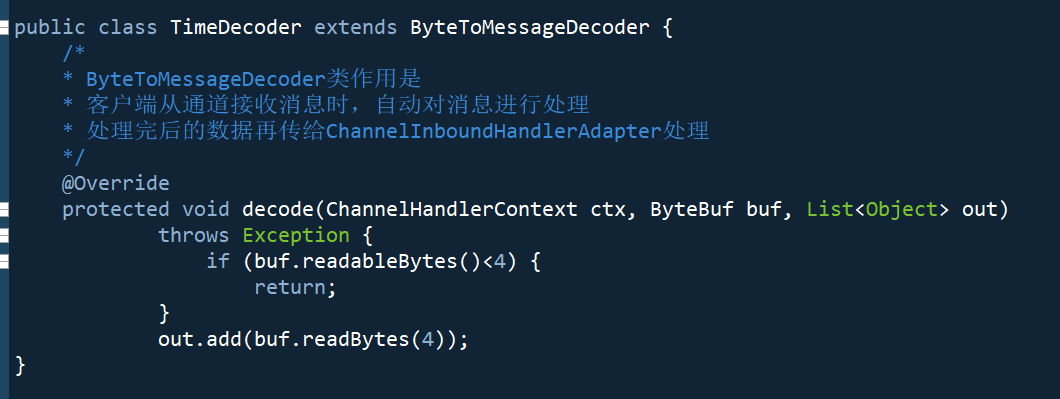
* 1. 客户端处理器TimeClientHandler

添加Netty自动处理器ByteToMessageDecoder（客户端的编码器）

TimeDecoder extends ByteToMessageDecoder



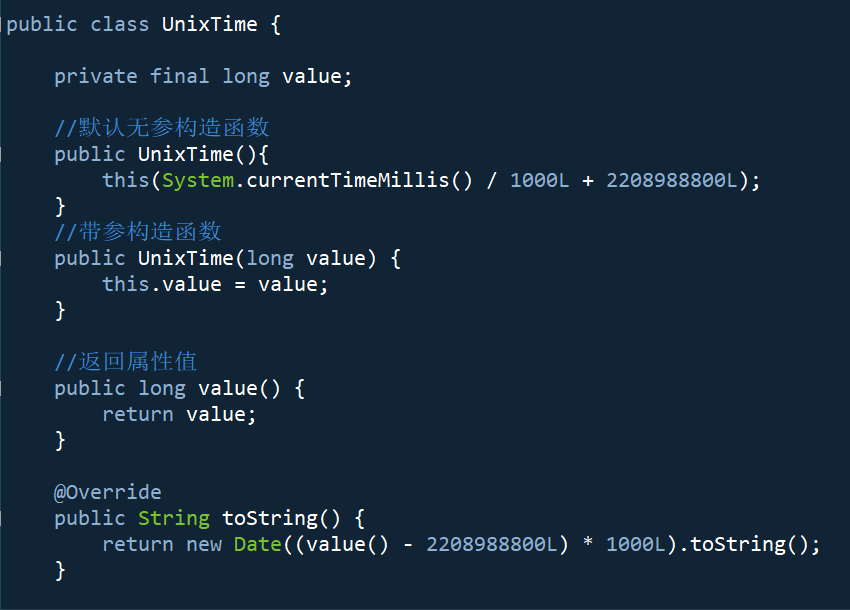
* 1. 编写自定义解码



1. 无论何时，**当新数据接收到时**，ByteToMessageDecoder 会调用一个内部可维护的 decode 方法来处理内部积累的 buffer 缓冲区。
2. decode 方法可以决定当没有足够的数据时，不添加到 out 对象中。当有更多的数据接收到后，ByteToMessageDecoder 会再次调用 decode 方法。
3. 如果 decode 方法添加一个对象到 out 列表对象中，这意味着解码器成功的解码了一个消息。ByteToMessageDecoder 会释放掉累计缓冲区已经读取的部分。需要注意的是，我们没有必要去解码多条 message 消息，因为 ByteToMessageDecoder 会一直调用 decode 方法直到没有数据添加到 out 列表对象中

## 创建协议（编码/解码器）

* 1. 制定通信协议（利用POJO对象替代byte[]）

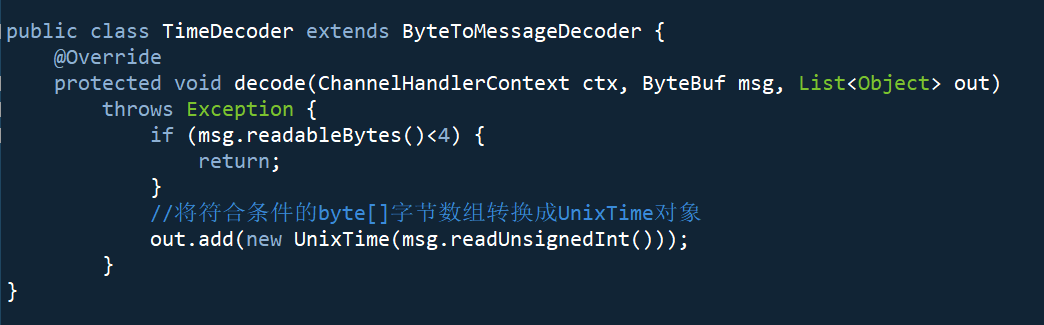


* 1. 编写客户端解码器

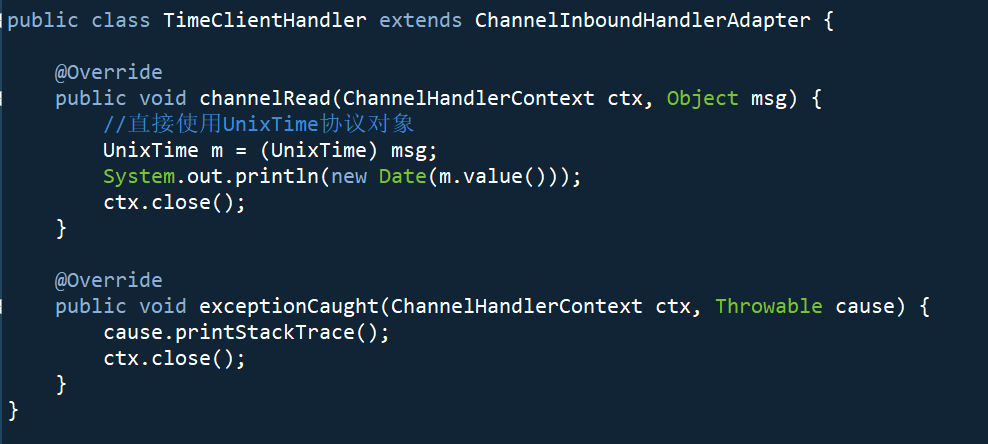
1. 设置客户端添加解码器



1. 解码器解码协议,将通道的byte[]转化成协议对象

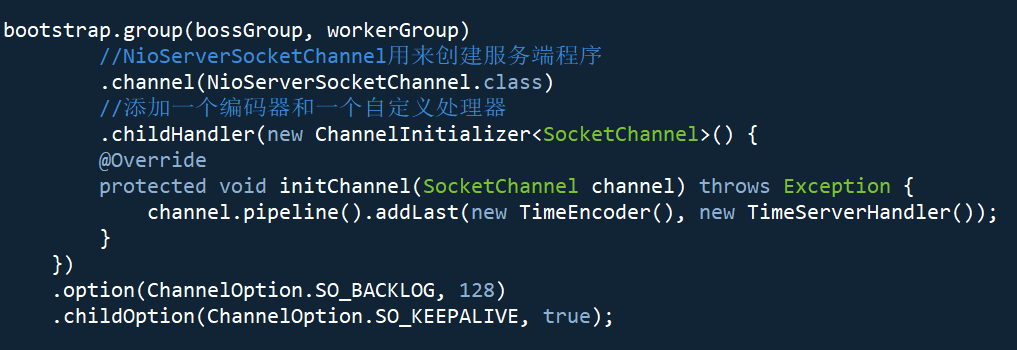


1. 自定义处理器使用协议对象

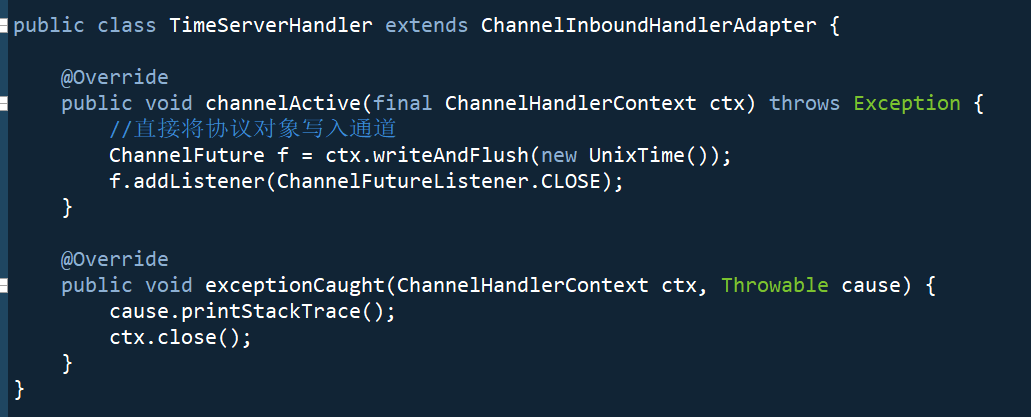


* 1. 编写服务端编码器

1. 设置服务器是添加编码器



1. 服务端通道直接发送协议对象



1. 在发送数据前编码器编码，将协议对象编码成byte[]

