# ChannelPipeline 类

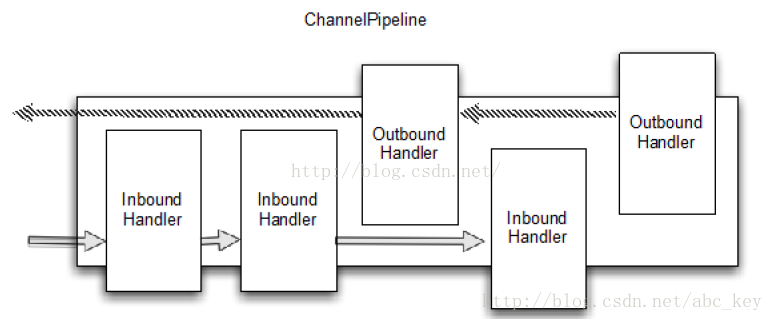
1. 作用

**ChannelPipeline 类是 ChannelHandler 实例对象的列表**，用于处理或截获通道的接收和发送数据。它提供了一种高级的截取过滤模式（类似 serverlet 中的 filter 功能），让用户可以在 ChannelPipeline 中完全控制一个事件以及如何处理 ChannelHandler 与 ChannelPipeline 的交互.

1. 创建

对于每个新的通道 Channel，都会创建一个新的 ChannelPipeline，并将其pipeline 附加到 channel 中。一旦连接，channel 和它之间的联系就会建立起来。通道 Channel 不能附加其他的 pipeline 或与 pipeline 分离开。

1. ChannelPipeline类和具体ChannelHandler实例对象的关系



1. ChannelHandler 通道处理器类与 pipeline 中的 io 处理关系

一个 io 操作可以由一个 ChannelInboundHandler 或 ChannelOutboundHandle 进行处理，并通过调用 ChannelInboundHandler 处理入站 io 或通过

ChannelOutboundHandler 处理出站 IO。

1. ChannelPipeline 是 ChannelHandler 的一个列表，如果一个入站 io 事件发生，这个事件会从第一个开始依次通过每里面的 channelhandler；如果是一个出站 io 事件，则会从最后一个开始依次通过 channelhandler。
2. ChannelPipeline 可以动态添加、删除、替换其中的 ChannelHandler，这样的机制可以提高灵活性。
3. addFirst(...)，添加 ChannelHandler 在 ChannelPipeline 的第一个位置
4. addBefore(...)，在 ChannelPipeline 中指定的 ChannelHandler 名称之前添加 ChannelHandler
5. addAfter(...)，在 ChannelPipeline 中指定的 ChannelHandler 名称之后添加 ChannelHandler
6. addLast(ChannelHandler...)，在 ChannelPipeline 的末尾添加 ChannelHandler
7. remove(...)，删除 ChannelPipeline 中指定的 ChannelHandler
8. replace(...)，替换 ChannelPipeline 中指定的 ChannelHandler

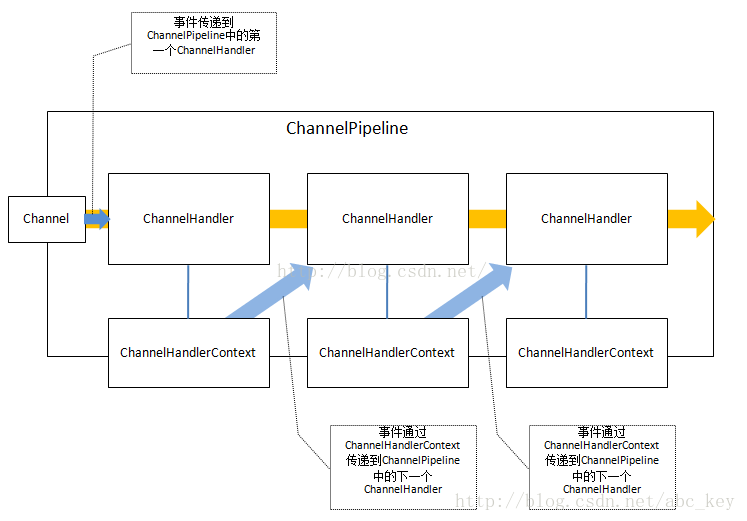
# ChannelHandlerContext类

1. 简介
2. 每个 ChannelHandler 被添加到 ChannelPipeline 中后，都会创建一个

ChannelHandlerContext 类添加到 ChannelHandler 中。

1. 这个context 允许 ChannelHandler 与其他 ChannelHandler 进行交互，这是对同一个 pipeline 而言的。
2. ChannelContext实例

IO 事件通过 contxt 将事件传入到 pipeline 中的下一个 channelhandler 中



# ChannelHandler 以及子类

1. 简介
2. Netty 中有 3 个实现了 ChannelHandler 接口的类，
3. 其中 2 个是接口（ChannelInboundHandler 用来处理入站数据也就是接收数据、ChannelOutboundHandler 用来处理出站数据也就是写数据），
4. 一个是抽象类 ChannelHandlerAdapter 类。
5. 常用方法

ChannelHandler 提供了在它的生命周期内添加或从 ChannelPipeline 中删除的方法：

1. handlerAdded:ChannelHandler 添加到实际上下文中准备处理事件。
2. handlerRemoved：将 ChannelHandler 从实际上下文中删除，不再处理事件。
3. exceptionCaught：处理跑出的异常。

这三个方法都需要传递 ChannelHandlerContext 参数对象，每个 ChannelHandler 被添加到 Channelpipeline 时会自动创建 ChannelHandlerContext。

1. Netty 还提供了一个实现了 ChannelHandler 的抽象类：ChannelHandlerAdapter 类。他实现了父类的所有方法，基本上就是**传递事件到 pipeline 中的下一个 ChannelHandler 直到结束。**

# ChannelInboundHandler 类

1. 常用方法
2. channelRegistered:ChannelHandlerContext 的 Channel 被注册到 EventLoop 中。
3. channelUnregistered：ChannelHandlerContext 的 channel 从 eventloop 中注销。
4. channelActive 方法：ChannelHandlerContext 的 channel 已被激活。
5. channelInactive 方法：ChannelHandlerContext 的 channel 结束生命周期。
6. channelRead 方法：从当前 Channel 的对端读取消息。
7. channelReadComplete 方法：消息读取完毕有执行。
8. userEventTriggered 方法：一个用户事件被触发。
9. channelWritabilityChanned 方法：改变通道的可写状态，可以使用 Channel.isWritable 检查。
10. exceptionCaught，重写父类 ChannelHandler 的方法，处理异常.
11. ChannelnboundHandlerAdapter类

实现了 ChannelInboundHandler 接口并继承 ChannelHandlerAdapter 的类

1. 实现了 ChannelInboundHandler接口中所有的方法，作用就是处理消息并将消息转发到 ChannelPipeline 中的下一个 channelhandler。
2. ChannelInboundHandlerAdapter 的 channelRead 方法处理完消息后不会自动释放消息，要手动释放。
3. 若想自动释放收到的消息，可以使用 SimpleChannelInboundHandler<I>.
4. 使用ChannelInboundHandler、ChannelInboundHandlerAdapter、

SimpleChannelInboundhandler 这三个来处理接收消息，使用哪一个取决于需求；

1. 使用 SimpleChannelInboundHandler 处理消息，

使用 ChannelInboundHandlerAdapter 处理其他的 “入站” 事件或状态改变。

1. ChannelInitializer类

ChannelInitializer 用来初始化 ChannelHandler，将自定义的各种 ChannelHandler 添加到 ChannelPipeline 中。



# ChannelOutboundHandler类

1. 常用方法（一般用来处理出站信息）
2. bind 方法：channel 绑定本地方法。
3. connect 方法：Channel 连接操作。
4. disconnect 方法：Channel 断开连接。
5. close 方法：关闭 Channel。
6. deregister 方法：注销 Channel 方法
7. read 方法：读取方法，实际是截获 ChannelHandlerContext.read
8. write 方法：写操作，实际是通过 ChannelPipeline 写消息，Channel.flush 方法刷新到实际通道中
9. flush 方法：刷新消息到通道。
10. 一旦请求停止从 ChannelPipeline 转发参数，则必须得到通知，是因为它所有重要的方法才去 ChannelPromise 机制。
11. 要记得释放并通知 ChannelPromise，若 ChannelPromise 没有被通知可能会导致其中一个 ChannelFutureListener 不被通知去处理一个消息。