# Channel接口

Channel 是 Java NIO 的一个基本构造，可以把 Channel 看作是传入（入站）或者传出（出站）数据的载体。因此，它可以被打开或者被关闭，连接或者断开连接。

基本的 I/O 操作（bind()、 connect()、 read()和 write()）依赖于底层网络传输所提  
供的原语。在基于 Java 的网络编程中，其基本的构造是 class Socket。 Netty 的 Channel 接  
口所提供的 API，大大地降低了直接使用 Socket 类的复杂性。此外， Channel 也是拥有许多  
预定义的、专门化实现的广泛类层次结构的根，下面是一个简短的部分清单：  
ν EmbeddedChannel；  
ν LocalServerChannel；  
ν NioDatagramChannel；  
ν NioSctpChannel；  
ν NioSocketChannel。

Netty 的 Channel 实现是线程安全的，因此你可以存储一个到 Channel 的引用，并且每当你需要向远程节点写数据时， 都可以使用它， 即使当时许多线程都在使用它。

# 回调

一个回调其实就是一个方法，一个指向已经被提供给另外一个方法的方法的引用。这使得后者。

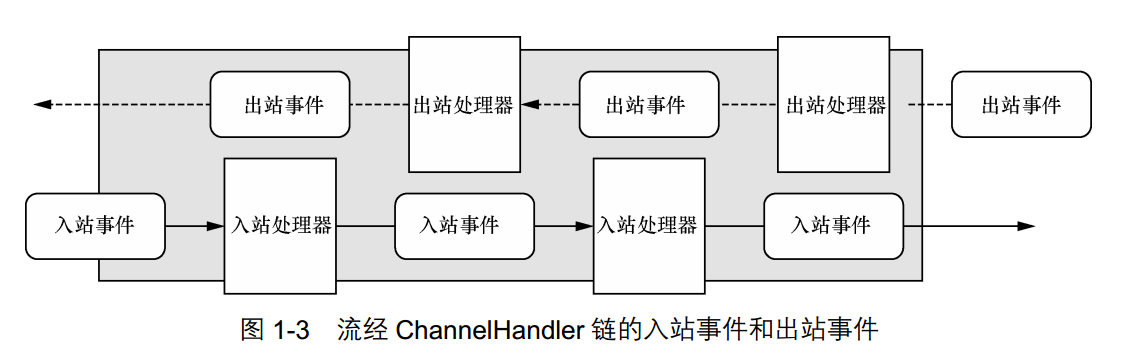
# Future ChannelFuture 接口

Future 提供了另一种在操作完成时通知应用程序的方式。这个对象可以看作是一个异步操作的结果的占位符；它将在未来的某个时刻完成，并提供对其结果的访问。

Netty 提供了ChannelFuture 接口，其 addListener()方法注册了一个 ChannelFutureListener，以  
便在某个操作完成时（无论是否成功）得到通知。

# 事件和 ChannelHandler

每个事件都可以被分发给 ChannelHandler 类中的某个用户实现的方法。



ChannelHandler，是一个借口族的父借口，它的实现负责接收并响应事件通知。在Netty应用程序中，所有的数据处理逻辑都包含在这些核心抽象的实现中。

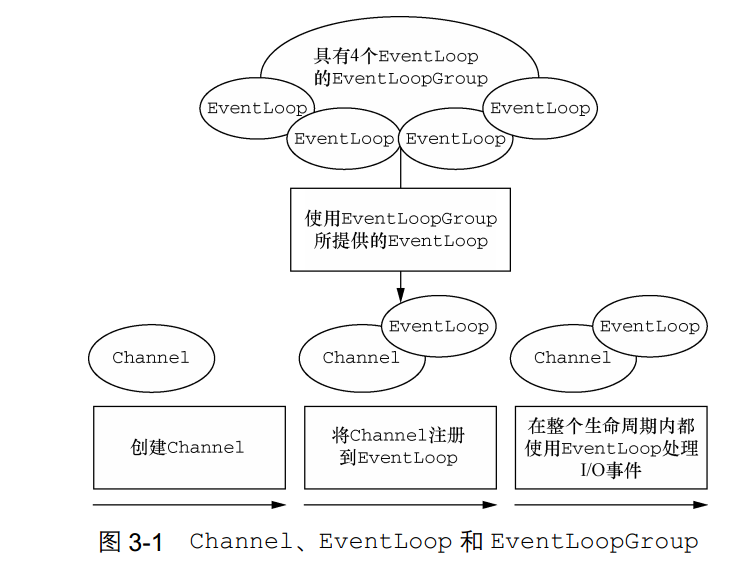
在Echo服务器中，会响应传入的消息，所以它需要实现ChannelInboundHandler接口，用来定义响应入站事件的方法。

# Channel、 EventLoop 和 ChannelFuture

Channel—Socket；  
EventLoop—控制流、多线程处理、并发；  
ChannelFuture—异步通知。

# EventLoop 接口

EventLoop 定义了 Netty 的核心抽象，用于处理连接的生命周期中所发生的事件。



这些关系是：   
ν 一个 EventLoopGroup 包含一个或者多个 EventLoop；  
ν 一个 EventLoop 在它的生命周期内只和一个 Thread 绑定；  
ν 所有由 EventLoop 处理的 I/O 事件都将在它专有的 Thread 上被处理；  
ν 一个 Channel 在它的生命周期内只注册于一个 EventLoop；  
ν 一个 EventLoop 可能会被分配给一个或多个 Channel。  
注意，在这种设计中，一个给定 Channel 的 I/O 操作都是由相同的 Thread 执行的，实际  
上消除了对于同步的需要。

# ChannelHandler 接口

Netty 的主要组件是 ChannelHandler， 它充当了所有处理入站和出站数据的应用程序逻辑的容器。

Netty 以适配器类的形式提供了大量默认的 ChannelHandler 实现为什么需要适配器类  
有一些适配器类可以将编写自定义的 ChannelHandler 所需要的努力降到最低限度，因为它们提  
供了定义在对应接口中的所有方法的默认实现。  
下面这些是编写自定义 ChannelHandler 时经常会用到的适配器类：  
ν ChannelHandlerAdapter  
ν ChannelInboundHandlerAdapter  
ν ChannelOutboundHandlerAdapter  
ν ChannelDuplexHandler

ChannelPipeline 持有所有将应用于入站和出站数据以及事件的 ChannelHandler 实例，这些 ChannelHandler 实现了应用程序用于处理状态变化以及数据处理的逻辑。  
ChannelHandler 的典型用途包括：  
ν 将数据从一种格式转换为另一种格式；  
ν 提供异常的通知；  
ν 提供 Channel 变为活动的或者非活动的通知；  
ν 提供当 Channel 注册到 EventLoop 或者从 EventLoop 注销时的通知；  
ν 提供有关用户自定义事件的通知。

# ChannelPipeline 接口

ChannelPipeline 提供了 ChannelHandler 链的容器，并定义了用于在该链上传播入站和出站事件流的 API。

当 Channel 被创建时，它会被自动地分配到它专属的 ChannelPipeline。ChannelHandler 安装到 ChannelPipeline 中的过程如下所示：  
ν 一个ChannelInitializer的实现被注册到了ServerBootstrap中；  
ν 当 ChannelInitializer.initChannel()方法被调用时， ChannelInitializer将在 ChannelPipeline 中安装一组自定义的 ChannelHandler；  
ν ChannelInitializer 将它自己从 ChannelPipeline 中移除。