

## pipeline server read 请求后的 handler调用流程

笔记本: wangle

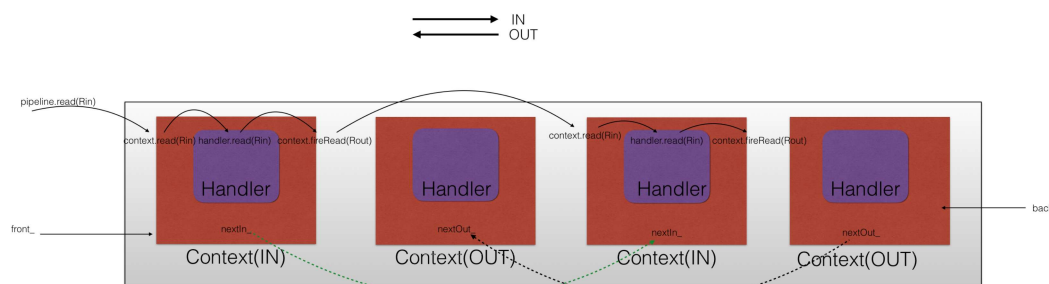
创建时间: 2019/12/4 9:18

更新时间: 2019/12/8 17:12

作者: 1969726254@qq.com

URL: <https://www.zhihu.com/question/19593742>

```
EchoHandler::read EchoServer.cpp:37
wangle::ContextImpl<EchoHandler>::read HandlerContext-inl.h:281
wangle::ContextImpl<wangle::StringCodec>::fireRead HandlerContext-inl.h:183
wangle::StringCodec::read StringCodec.h:37
wangle::ContextImpl<wangle::StringCodec>::read HandlerContext-inl.h:281
wangle::InboundContextImpl<wangle::LineBasedFrameDecoder>::fireRead HandlerContext-inl.h:352
wangle::ByteToMessageDecoder<std::unique_ptr<folly::IOBuf, std::default_delete<folly::IOBuf> > >::read ByteToMessageDecoder.h:73
wangle::InboundContextImpl<wangle::LineBasedFrameDecoder>::read HandlerContext-inl.h:401
wangle::ContextImpl<wangle::AsyncSocketHandler>::fireRead HandlerContext-inl.h:183
wangle::AsyncSocketHandler::readDataAvailable AsyncSocketHandler.h:154
folly::AsyncSocket::handleRead AsyncSocket.cpp:2023
folly::AsyncSocket::ioReady AsyncSocket.cpp:1812
folly::AsyncSocket::ioHandler::handlerReady AsyncSocket.h:1052
folly::EventHandler::libeventCallback EventHandler.cpp:161
```



一个节点运行过程为:

上一个节点 调用 `HandlerContext::fire_read`, `fire_read`函数中 调用 下一个节点的 (`InboundLink` 也是下一个context) `read`函数。  
`context`的`read` 函数 调用 `context`中`handler` 的`read`函数, `handler`的`read`函数 调用这个节点`context`的 `fire_read`函数, 执行下一个节点的操作。  
这样整个调用就串起来了。

`ContextImpl`就是最终的Context实现, 也就是要被添加到Pipeline中 (比如使用`addBack`) 的容器 (`ctxs_`, `inCtxs_`, `outCtxs_`) 的最终Context, 在最后的`finalize`方法中还会进一步将容器中的Context组装成`front_`和`back_`单向链表。