Java中的IO主要分为3种：BIO（同步阻塞）、NIO（同步非阻塞）和AIO（异步非阻塞）。

## BIO

在JDK 1.4以前，使用Java建立网络连接时，只能采用BIO方式，在服务器端启动一个Server Socket，然后使用accept等待客户端请求，对于每一个请求，使用一个线程来进行 处理用户请求。线程的大部分时间都在等待请求的到来和IO操作，利用率很低。而且线程的开销比较大，数量有限，因此服务器同时能处理的连接数也很低。

## NIO

BIO模式中，是“一个Socket一个线程”；而在NIO中则是使用单个或少量的线程来轮询Socket，当发现Socket上有请求时，才为请求分配线程。因此是“一个请求一个线程”。

具体实现就是把Socket通过Channel注册到Selector，使用一个线程在Selector中轮询，发现Channel有读写的事件，就可以分配给其他线程来处理（通常使用线程池）。

## AIO

从JDK7开始支持AIO模式。通过AsynchronousServerSocketChannel中注册事件回调函数来处理业务逻辑。当IO操作完成以后，回调函数会被调用。如果传入AsynchronousChannelGroup，可以绑定线程池来处理事件。

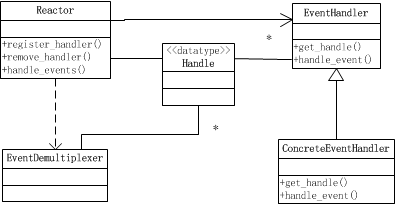
关于JDK的实现，Windows平台基于IOCP实现AIO，Linux只有eppoll模拟实现了AIO。

## Reactor & Proactor

一般情况下，I/O复用机制需要事件分享器（event demultiplexor）。事件分享器的作用，即将那些读写事件源分发给各读写事件的处理者，就像送快递的在楼下喊：谁的什么东西送了，快来拿吧。开发人员在开始的时候需要在分享器那里注册感兴趣的事件，并提供相应的处理者（event handlers），或者是回调函数；事件分享器在适当的时候会将请求的事件分发给这些handler或者回调函数。

涉及到事件分享器的两种模式称为：Reactor and Proactor。Reactor模式是基于同步I/O的，而Proactor模式是和异步I/O相关的。在Reactor模式中，事件分离者等待某个事件或者可应用某个操作的状态发生（比如文件描述符可读写，或者是socket可读写），事件分离者就把这个事件传给事先注册的事件处理函数或者回调函数，由后者来做实际的读写操作。

### Reactor模式



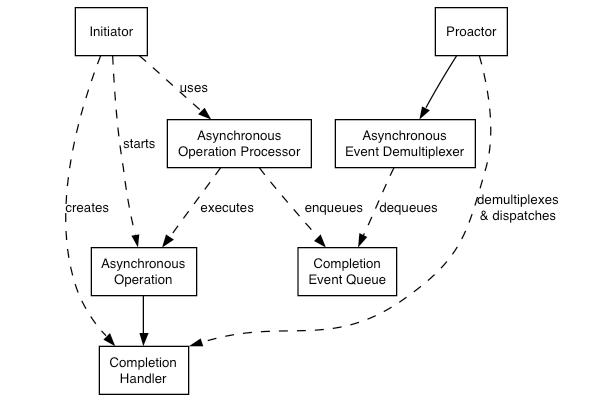
Reactor类图如上所示，Reactor模式又叫反应器或反应堆，即实现注册描述符（Handler）及事件的处理器（EventHandler），当有事件发生的时候，事件多路分发器（Event Demultiplexer）做出反应，调用事件具体的处理函数（ConcreteEventHandler::handle\_event()）。

Reactor模式的典型启动过程如下：

1. 创建Reactor
2. 注册事件处理器( Reactor::register\_handler() )
3. 调用事件多路分发器进入无限事件循环（Reactor::handle\_evnets）
4. 当操作系统通知某描述符状态就绪时，事件多路分发器找出并调用此描述符注册的事件处理器。

Reactor模式已经被广泛使用，著名的开源事件库libevent、libev、libuv都是使用Reactor模式。

### Proactor模式



Proactor模式的类图如上所示，Proactor模式又叫前摄器或主动器模式。它用于实现异步I/O模型，运行流程如下：

1. Initiator主动调用Asynchronous Opertaion Processor发起异步I/O操作
2. 记录异步操作的参数和函数地址放入完成事件队列（Completion Event Queue）中
3. Poractor循环检测异步事件是否完成。如果完成则从完成事件队列中去除回调函数完成回调。
4. Boost库中的asio就使用了Proactor模式，其底层的异步I/O由操作系统提供，而异步事件的分发还是由epoll / kequeue / select等实现。

在Proactor模式中，事件处理者直接发起（或者代由事件分离者发起）一个异步读写操作（相当于请求），而实际的工作是由操作系统来完成的。发起时，需要提供的参数包括用于存放读到数据的缓存区，读的数据大小，或者用于存放外发数据的缓存区，以及这个请求完成后的回调函数等信息。事件分离者得知了这个请求，它默默等待这个请求的完成，然后转发完成事件给相应的事件处理者或者回调。

## Reactor vs Proactor

* 两者主要区别：

### Reactor实现同步I/O多路分发，Proactor实现异步I/O分发。

如果只是处理网络I/O，单线程的Reactor尚可处理，但如果涉及到文件I/O，但线程的Reactor可能被文件I/O阻塞而导致其他事件无法被分发。所以涉及到文件I/O最好还是使用Preactor模式，或者用多线程模拟实现异步I/O的方式。

### Reactor模式注册的是文件描述符的就绪事件，而Proactor模式注册的是完成事件。

Reactor模式有事件发生的时候要判断是读事件还是写事件，然后再调用系统调用( read / write等 )将数据从内核中拷贝到用户数据区继续其他业务处理。

而Proactor模式一般使用的是操作系统的异步I/O接口，发起异步调用（用户提供数据缓冲区）之后操作系统将在内核态完成I/O并拷贝数据到用户提供的缓冲区中，完成事件到达之后，用户只需要实现自己后续的业务处理即可。

### 主动和被动

Reactor模式是一种被动的处理，即有事件发生时被动处理。而Proactor模式则是主动发起异步调用，然后循环检测完成事件。