Java中的IO类型

基本概念

同步异步针对的是被调用者B

**同 步**：A调用B，B的处理是同步的，直到B处理完成之后才会通知A。

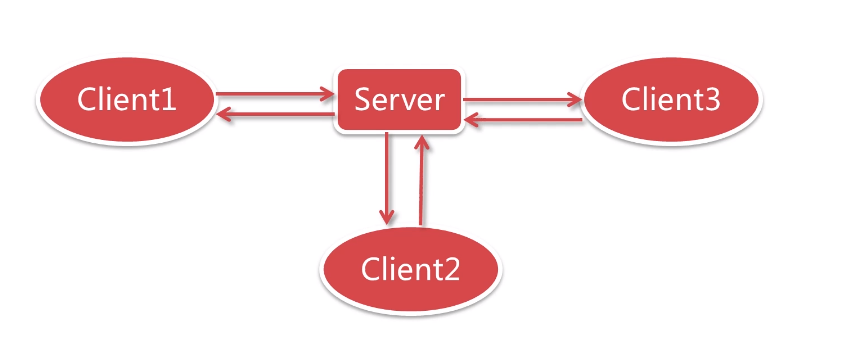
**异 步**：A调用B，B的处理是异步的，B在接到请求后先告诉A我已经接到请求了，然后异步去处理，处理完之后通过回调等方式再通知A。

阻塞非阻塞针对的是调用者A

**阻 塞**：A调用B，A一直等待B的返回，期间处于阻塞状态

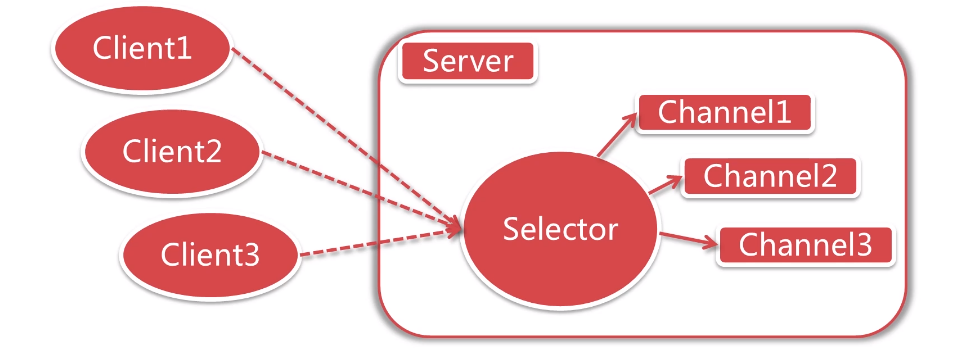
**非阻塞**：A调用B，A调用成功后不用一直等待B返回，可以去做其他任务

Java中的3种IO模型



**BIO** (Blocking I/O) 同步阻塞IO

每有一个客户端连接就需要和Server建立一个线程



**NIO** (New I/O) 同步非阻塞IO，也支持同步阻塞IO

通道 和 缓冲区 是 NIO 中的核心对象，几乎在每一个 I/O 操作中都要使用它们。

**缓冲区**：[Buffer](https://link.zhihu.com/?target=https://links.jianshu.com/go?to=http%3A%2F%2Fdocs.52im.net%2Fextend%2Fdocs%2Fapi%2Fjavase-8u51%2Fjava%2Fnio%2FBuffer.html" \t "https://zhuanlan.zhihu.com/p/_blank) 是一个对象， 它包含一些要写入或者刚读出的数据。 在 NIO 中加入 Buffer 对象，体现了新库与原 I/O 的一个重要区别。在面向流的 I/O 中，数据直接写入或者将数据直接读到 Stream 对象中。

在 NIO 库中，所有数据都是用缓冲区处理的。在读取数据时，它是直接读到缓冲区中的。在写入数据时，它是写入到缓冲区中的。任何时候访问 NIO 中的数据，都是将它放到缓冲区中。

缓冲区实质上是一个数组。通常它是一个字节数组，但是也可以使用其他种类的数组。但是一个缓冲区不 仅仅 是一个数组。缓冲区提供了对数据的结构化访问，而且还可以跟踪系统的读/写进程。

**缓冲区类型**：最常用的缓冲区类型是 [ByteBuffer](https://link.zhihu.com/?target=https://links.jianshu.com/go?to=http%3A%2F%2Fdocs.52im.net%2Fextend%2Fdocs%2Fapi%2Fjavase-8u51%2Fjava%2Fnio%2FByteBuffer.html" \t "https://zhuanlan.zhihu.com/p/_blank)。一个 ByteBuffer 可以在其底层字节数组上进行 get/set 操作(即字节的获取和设置)。

当然也有其他类型：

[ByteBuffer](https://link.zhihu.com/?target=https://links.jianshu.com/go?to=http%3A%2F%2Fdocs.52im.net%2Fextend%2Fdocs%2Fapi%2Fjavase-8u51%2Fjava%2Fnio%2FByteBuffer.html" \t "https://zhuanlan.zhihu.com/p/_blank)  
 [CharBuffer](https://link.zhihu.com/?target=https://links.jianshu.com/go?to=http%3A%2F%2Fdocs.52im.net%2Fextend%2Fdocs%2Fapi%2Fjavase-8u51%2Fjava%2Fnio%2FCharBuffer.html" \t "https://zhuanlan.zhihu.com/p/_blank)  
 [ShortBuffer](https://link.zhihu.com/?target=https://links.jianshu.com/go?to=http%3A%2F%2Fdocs.52im.net%2Fextend%2Fdocs%2Fapi%2Fjavase-8u51%2Fjava%2Fnio%2FShortBuffer.html" \t "https://zhuanlan.zhihu.com/p/_blank)  
 [IntBuffer](https://link.zhihu.com/?target=https://links.jianshu.com/go?to=http%3A%2F%2Fdocs.52im.net%2Fextend%2Fdocs%2Fapi%2Fjavase-8u51%2Fjava%2Fnio%2FIntBuffer.html" \t "https://zhuanlan.zhihu.com/p/_blank)  
 [LongBuffer](https://link.zhihu.com/?target=https://links.jianshu.com/go?to=http%3A%2F%2Fdocs.52im.net%2Fextend%2Fdocs%2Fapi%2Fjavase-8u51%2Fjava%2Fnio%2FLongBuffer.html" \t "https://zhuanlan.zhihu.com/p/_blank)  
 [FloatBuffer](https://link.zhihu.com/?target=https://links.jianshu.com/go?to=http%3A%2F%2Fdocs.52im.net%2Fextend%2Fdocs%2Fapi%2Fjavase-8u51%2Fjava%2Fnio%2FFloatBuffer.html" \t "https://zhuanlan.zhihu.com/p/_blank)  
 [DoubleBuffer](https://link.zhihu.com/?target=https://links.jianshu.com/go?to=http%3A%2F%2Fdocs.52im.net%2Fextend%2Fdocs%2Fapi%2Fjavase-8u51%2Fjava%2Fnio%2FDoubleBuffer.html" \t "https://zhuanlan.zhihu.com/p/_blank)

每一个 Buffer 类都是 Buffer 接口的一个实例。 除了 ByteBuffer，每一个 Buffer 类都有完全一样的操作，只是它们所处理的数据类型不一样。因为大多数标准 I/O 操作都使用 ByteBuffer，所以它具有所有共享的缓冲区操作以及一些特有的操作。

**通道**：Channel是一个对象，可以通过它读取和写入数据。拿 NIO 与原来的 I/O 做个比较，通道就像是流。

所有数据都通过 Buffer 对象来处理。不会将字节直接写入通道中，相反，您是将数据写入包含一个或者多个字节的缓冲区。同样，您不会直接从通道中读取字节，而是将数据从通道读入缓冲区，再从缓冲区获取这个字节。

**通道类型**：通道与流的不同之处在于通道是双向的。而流只是在一个方向上移动(一个流必须是 [InputStream](https://link.zhihu.com/?target=https://links.jianshu.com/go?to=http%3A%2F%2Fdocs.52im.net%2Fextend%2Fdocs%2Fapi%2Fjavase-8u51%2Fjava%2Fio%2FInputStream.html" \t "https://zhuanlan.zhihu.com/p/_blank)或者 [OutputStream](https://link.zhihu.com/?target=https://links.jianshu.com/go?to=http%3A%2F%2Fdocs.52im.net%2Fextend%2Fdocs%2Fapi%2Fjavase-8u51%2Fjava%2Fio%2FOutputStream.html" \t "https://zhuanlan.zhihu.com/p/_blank) 的子类)， 而 通道 可以用于读、写或者同时用于读写。

因为它们是双向的，所以通道可以比流更好地反映底层操作系统的真实情况。特别是在 UNIX 模型中，底层操作系统通道是双向的。

**AIO** (Asynchronous I/O) 异步非阻塞IO