# NIO和IO的区别

## 概述

|  |  |
| --- | --- |
| IO | NIO |
| 面向流 | 面向缓冲区 |
| 阻塞IO | 非阻塞IO |
| 无选择器 | 有选择器 |

## 面向流与面向缓冲

1. IO 是面向流的，NIO 是面向缓冲区的。
2. **Java IO 面向流意味着每次从流中读一个或多个字节，直至读取所有字节，它们没有被缓存在任何地方。**此外，它不能前后移动流中的数据。如果需要前后移动从流中读取的数据，需要先将它缓存到一个缓冲区。
3. **Java NIO 的缓冲导向方法是数据读取到一个它稍后处理的缓冲区，需要时可在缓冲区中前后移动。**这就增加了处理过程中的灵活性。但是，还需要检查是否该缓冲区中包含所有您需要处理的数据。而且，需确保当更多的数据读入缓冲区时，不要覆盖缓冲区里尚未处理的数据。

## 阻塞与非阻塞 IO

1. Java IO 的各种流是阻塞的。这意味着，当一个线程调用 read() 或 write() 时，该线程被阻塞，直到有一些数据被读取，或数据完全写入。该线程在此期间不能再干任何事情了。
2. Java NIO 的非阻塞模式，使一个线程从某通道发送请求读取数据，但是它仅能得到目前可用的数据，如果目前没有数据可用时，就什么都不会获取。而不是保持线程阻塞，所以直至数据变的可以读取之前，该线程可以继续做其他的事情。线程通常将非阻塞 IO 的空闲时间用于在其它通道上执行 IO 操作，所以一个单独的线程现在可以管理多个输入和输出通道（channel）。

## 选择器

**Java NIO 的选择器允许一个单独的线程来监视多个输入通道，**你可以注册多个通道使用一个选择器，然后使用一个单独的线程来 “选择” 通道：这些通道里已经有可以处理的输入，或者选择已准备写入的通道。