## JavaNIO概述

Javanio由channels buffers和selectors组成。

基本上，所有的 IO 在NIO 中都从一个Channel 开始。Channel 有点像流。 数据可以从Channel读到Buffer中，也可以从Buffer 写到Channel中。

Channel和Buffer有好几种类型。下面是JAVA NIO中的一些主要Channel的实现：

* FileChannel
* DatagramChannel
* SocketChannel
* ServerSocketChannel

这些通道涵盖了UDP 和 TCP 网络IO，以及文件IO

以下是Java NIO里关键的Buffer实现：

* ByteBuffer
* CharBuffer
* DoubleBuffer
* FloatBuffer
* IntBuffer
* LongBuffer
* ShortBuffer

这些Buffer覆盖了你能通过IO发送的基本数据类型：byte, short, int, long, float, double 和 char。

Java NIO 还有个 MappedByteBuffer，用于表示内存映射文件。

Selector允许单线程处理多个 Channel。如果你的应用打开了多个连接（通道），但每个连接的流量都很低，使用Selector就会很方便。例如，在一个聊天服务器中。



要使用Selector，得向Selector注册Channel，然后调用它的select()方法。这个方法会一直阻塞到某个注册的通道有事件就绪。一旦这个方法返回，线程就可以处理这些事件，事件的例子有如新连接进来，数据接收等。

## Channel

Java NIO的通道类似流，但又有些不同：

* 既可以从通道中读取数据，又可以写数据到通道。但流的读写通常是单向的。
* 通道可以异步地读写。
* 通道中的数据总是要先读到一个Buffer，或者总是要从一个Buffer中写入。

**Channel的实现**

这些是Java NIO中最重要的通道的实现：

* FileChannel
* DatagramChannel
* SocketChannel
* ServerSocketChannel

FileChannel 从文件中读写数据。（抽象类）

DatagramChannel 能通过UDP读写网络中的数据。

SocketChannel 能通过TCP读写网络中的数据。

ServerSocketChannel可以监听新进来的TCP连接，像Web服务器那样。对每一个新进来的连接都会创建一个SocketChannel。

## Buffer

Java NIO中的Buffer用于和NIO通道进行交互。如你所知，数据是从通道读入缓冲区，从缓冲区写入到通道中的。

缓冲区本质上是一块可以写入数据，然后可以从中读取数据的内存。这块内存被包装成NIO Buffer对象，并提供了一组方法，用来方便的访问该块内存。

**Buffer的基本用法**

使用Buffer读写数据一般遵循以下四个步骤：

1. 写入数据到Buffer
2. 调用flip()方法
3. 从Buffer中读取数据
4. 调用clear()方法或者compact()方法

当向buffer写入数据时，buffer会记录下写了多少数据。一旦要读取数据，需要通过flip()方法将Buffer从写模式切换到读模式。在读模式下，可以读取之前写入到buffer的所有数据。