Java NIO系列教程（二） Channel

原文链接 作者：Jakob Jenkov 译者：airu 校对：丁一

I/O流表示一个输入来源或者输出目标。
流可以表示很多不同类型的来源或者目的，包括磁盘文件、设备和内存数组
流是一系列数据。
输入流、输出流

Java NIO的通道类似流，但又有些不同：

通道的读写是双向的

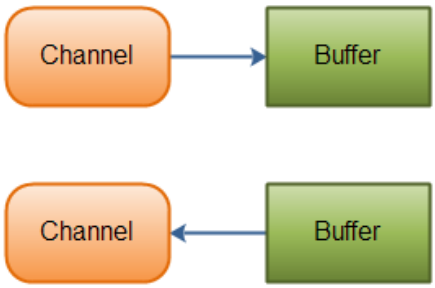
既可以从通道中读取数据，又可以写数据到通道。但流的读写通常是单向的。

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。

通道只与缓冲区打交道

正如上面所说，从通道读取数据到缓冲区，从缓冲区写入数据到通道。如下图所示：



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。

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。

基本的 Channel 示例

下面是一个使用FileChannel读取数据到Buffer中的示例：

```
01 RandomAccessFile aFile = new RandomAccessFile("data/nio-data.txt", "rw");
02 FileChannel inChannel = aFile.getChannel();
03
04 ByteBuffer buf = ByteBuffer.allocate(48);
05
06 int bytesRead = inChannel.read(buf);
07 while (bytesRead != -1) {
```

```
08
09 System.out.println("Read " + bytesRead);
10 buf.flip();
11
12 while(buf.hasRemaining()){
13 System.out.print((char) buf.get());
14 }
15
16 buf.clear();
17 bytesRead = inChannel.read(buf);
18 }
19 aFile.close();
```

注意 buf.flip() 的调用，首先读取数据到Buffer，然后反转Buffer,接着再从Buffer中读取数据。下一节会深入讲解Buffer的更多细节。

原创文章，转载请注明： 转载自并发编程网 – ifeve.com本文链接地址: Java NIO系列教程（二） Channel

56

About Latest Posts

Airu

爱生活， 爱编码，更爱玩。