



Java NIO 与 IO之间的区别

2018年3月17日 09:01:05 标签：NIO / IO / java / 阻塞与非阻塞

6700

概述

Java NIO提供了与标准IO不同的IO工作方式：

- Channels and Buffers（通道和缓冲区）：标准的IO基于字节流和字符流进行操作的，而NIO是基于通道（Channel）和缓冲区（Buffer）进行操作，数据总是从通道读取到缓冲区中，或者从缓冲区写入到通道中。
- Asynchronous IO（异步IO）：Java NIO可以让你异步的使用IO，例如：当线程从通道读取数据到缓冲区时，线程还是可以去做其他事情。当数据被写入到缓冲区时，线程可以继续处理它。从缓冲区写入通道也类似。
- Selectors（选择器）：Java NIO引入了选择器的概念，选择器用于监听多个通道的事件（比如：连接打开，数据到达）。因此，单个的线程可以监听多个数据通道。

使用场景

NIO

- 优势在于一个线程管理多个通道；但是数据的处理将会变得复杂；
- 如果需要管理同时打开的成千上万个连接，这些连接每次只是发送少量的数据，采用这种；

传统的IO

- 适用于一个线程管理一个通道的情况；因为其中的流数据的读取是阻塞的；
- 如果需要管理同时打开不太多的连接，这些连接会发送大量的数据；

NIO vs IO区别

NIO vs IO之间的理念上面的区别（NIO将阻塞交给了后台线程执行）

- IO是面向流的，NIO是面向缓冲区的
 - 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；
 - NIO则能前后移动流中的数据，因为是面向缓冲区的
- IO流是阻塞的，NIO流是不阻塞的
 - 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
 - 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。NIO可让您只使用一个（或几个）单线程管理多个通道（网络连接或文件），但付出的代价是解析数据可能会比从一个阻塞流中读取数据更复杂。
 - 非阻塞写也是如此。一个线程请求写入一些数据到某通道，但不需要等待它完全写入，这个线程同时可以去做别的事情。
- 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。这种选择机制，使得一个单独的线程很容易来管理多个通道。

Java NIO 由以下几个核心部分组成：

- Channels
- Buffers
- Selectors

基本上，所有的 IO 在NIO 中从一个Channel 开始。Channel 有点象流。数据可以从Channel读到Buffer中，也可以从Buffer 写到Channel中。这里有个图示：

Channel



evan_man

关注

原创

52

粉丝

63

喜欢

0



等级：博客 4

访问量：17.6

积分：1952

排名：2.37万

广告

他的最新文章

求职那些事儿

JVM、DVM(Dalvik VM)和ART虚比

Android的视图绘制与事件分发流层)

Java中的IO技术使用总结

DialogFragment的使用与底层绘

文章分类

Java技术

Java源码

JVM

Android技术

编程思想

网络编程

展开

文章存档

2016年10月

2016年9月

2016年8月

2016年7月

2016年6月

2016年5月

展开

他的热门文章


Android之三种Menu的使用与分 27084

EventBus的使用与深入学习 16826


RecyclerView的使用与深入分析 15481

RxJava的使用与深入学习 14374


Channel的实现：（涵盖了UDP 和 TCP 网络IO，以及文件IO）

- FileChannel
- DatagramChannel
- SocketChannel
-  SocketChannel

读数据：

-  int Read = inChannel.read(buf);

写数

-  int Written = inChannel.write(buf);

还有部分应用，如配置Channel为阻塞或者非阻塞模式，以及如何注册到Selector上面去，参考Selector部分；

Buffer

Buffer实现：（ byte、 char、 short、 int、 long、 float、 double ）

- ByteBuffer
- CharBuffer
- DoubleBuffer
- FloatBuffer
- IntBuffer
- LongBuffer
- ShortBuffer

Buffer使用

读数据

- flip()方法
 - 将Buffer从写模式切换到读模式
 - 调用flip()方法会将position设回0，并将limit设置成之前position的值。
 - buf.flip();
- (char) buf.get()
 - 读取数据
- Buffer.rewind()
 - 将position设回0，所以你可以重读Buffer中的所有数据
 - limit保持不变，仍然表示能从Buffer中读取多少个元素（ byte、 char等）
- Buffer.mark()方法，可以标记Buffer中的一个特定position。之后可以通过调用
- Buffer.reset()方法，恢复到Buffer.mark()标记时的position
- 一旦读完了所有的数据，就需要清空缓冲区，让它可以再次被写入。
- clear()方法会：
 - 清空整个缓冲区。
 - position将被设回0，limit被设置成 capacity的值
- compact()方法：
 - 只会清除已经读过的数据；任何未读的数据都被移到缓冲区的起始处，新写入的数据将放到缓冲区未读数据的后面。
 - 将position设到最后一个未读元素正后面，limit被设置成 capacity的值

写数据

- buf.put(127);

Buffer的三个属性

- capacity：含义与模式无关；Buffer的一个固定的大小值；Buffer满了需要将其清空才能再写；
 - ByteBuffer.allocate(48)；该buffer的capacity为48byte
 - CharBuffer.allocate(1024);该buffer的capacity为1024个char
- position：含义取决于Buffer处在读模式还是写模式（ 初始值为0，写或者读操作的当前位置）

Retrofit的使用与深入学习（上）

 8251

Java NIO 与 IO之间的区别

 6635

Retrofit的使用与深入学习（下）

 6001

OkHttp深入学习（一）——初探

 5553

DialogFragment的使用与底层绘制

 5451


ToolBar概述


 5232

联系我们



请扫描二维码联系

 webmaster@c

 400-660-0108

 QQ客服  客

关于 招聘 广告服务 

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

- 写数据时，初始的position值为0；其值最大可为capacity-1
- 将Buffer从写模式切换到读模式，position会被重置为0
- limit：含义取决于Buffer处在读模式还是写模式（写limit=capacity；读limit等于最多可以读取到的数据）
 - 写模式下，limit等于Buffer的capacity
 - 切换Buffer到读模式时，limit表示你最多能读到多少数据；

Selector

概述

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就不会阻塞，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。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。

使用

1. 创建：Selector selector = Selector.open();
 2. 注册通道：
 1. channel.configureBlocking(false);
 1. //与Selector一起使用时，Channel必须处于非阻塞模式
 2. //这意味着不能将FileChannel与Selector一起使用，因为FileChannel不能切换到非阻塞模式(而套接字通道都可以)
 2. SelectionKey key = channel.register(selector, SelectionKey.OP_READ);
 1. //第二个参数表明Selector监听Channel时对什么事件感兴趣
 2. //SelectionKey.OP_CONNECT SelectionKey.OP_ACCEPT SelectionKey.OP_READ SelectionKey.OP_WRITE
 3. //可以用或操作符将多个兴趣组合一起
 3. SelectionKey
 1. 包含了interest集合、ready集合、Channel、Selector、附加的对象（可选）
 2. int interestSet = key.interestOps(); 可以进行类似interestSet & SelectionKey.OP_CONNECT的判断
- 使用：
1. select()：阻塞到至少有一个通道在你注册的事件上就绪了
 2. selectNow()：不会阻塞，不管什么通道就绪都立刻返回
 3. selectedKeys()：访问“已选择键集（selected key set）”中的就绪通道
 4. close()：使用完selector需要其close()方法会关闭该Selector，且使注册到该Selector上的所有SelectionKey实例无效

```

1. Set selectedKeys = selector.selectedKeys();
2. Iterator keyIterator = selectedKeys.iterator();
3. while(keyIterator.hasNext()) {
4.     SelectionKey key = keyIterator.next();
5.     if(key.isAcceptable()) {
6.         // a connection was accepted by a ServerSocketChannel.
7.     } else if (key.isConnectable()) {
8.         // a connection was established with a remote server.
9.     } else if (key.isReadable()) {
10.        // a channel is ready for reading
11.    } else if (key.isWritable()) {
12.        // a channel is ready for writing
13.    }
14.    keyIterator.remove(); //注意这里必须手动remove: 表明该selectkey我已经处理过了;
15. }

```

Java测试关键代码

```
1. RandomAccessFile aFile = new RandomAccessFile("data/nio-data.txt", "rw");
```

```
2. FileChannel inChannel = aFile.getChannel(); //从一个InputStream outputStream中获取channel
3.
4. //create buffer with capacity of 48 bytes
5. ByteBuffer buf = ByteBuffer.allocate(48);
6.
7. int bytesRead = inChannel.read(buf); //read into buffer.
8. while (bytesRead != -1) {
9.
10.    buf.flip(); //make buffer ready for read
11.
12.    while (buf.hasRemaining()){
13.        System.out.print((char) buf.get()); // read 1 byte at a time
14.    }
15.
16.    buf.clear(); //make buffer ready for writing
17.    bytesRead = inChannel.read(buf);
18. }
19. aFile.close();
```

文件通道

```
1. RandomAccessFile aFile = new RandomAccessFile("data/nio-data.txt", "rw");
2. FileChannel inChannel = aFile.getChannel();
```

读数据

```
1. ByteBuffer buf = ByteBuffer.allocate(48);
2. int bytesRead = inChannel.read(buf);
```

写数据

```
1. String newData = "New String to write to file..." + System.currentTimeMillis();
2. ByteBuffer buf = ByteBuffer.allocate(48);
3. buf.clear();
4. buf.put(newData.getBytes());
5. buf.flip();
6. while (buf.hasRemaining()) {
7.     channel.write(buf);
8. }
```

Socket 通道

```
1. SocketChannel socketChannel = SocketChannel.open();
2. socketChannel.connect(new InetSocketAddress("http://jenkov.com", 80));
```

读数据

```
1. ByteBuffer buf = ByteBuffer.allocate(48);
2. int bytesRead = socketChannel.read(buf);
```

写数据

```
1. String newData = "New String to write to file..." + System.currentTimeMillis();
2. ByteBuffer buf = ByteBuffer.allocate(48);
3. buf.clear();
4. buf.put(newData.getBytes());
5. buf.flip();
6. while (buf.hasRemaining()) {
7.     socketChannel.write(buf);
8. }
```

ServerSocket 通道

```
1. ServerSocketChannel serverSocketChannel = ServerSocketChannel.open();
```

```
2. serverSocketChannel.socket().bind(new InetSocketAddress(9999));

3. while(true){

4.     SocketChannel socketChannel =

5.         serverSocketChannel.accept();

6.     //do something with socketChannel...

7. }
```



Datagram 通道(channel的读写操作与前面的有差异)

```
1. DatagramChannel channel = DatagramChannel.open();

2. channel.socket().bind(new InetSocketAddress(9999));
```

读数据

```
1. ByteBuffer buf = ByteBuffer.allocate(48);

2. buf.clear();

3. channel.receive(buf);

4. //receive()方法会将接收到的数据包内容复制到指定的Buffer。如果Buffer容不下收到的数据，多出的数据将被丢弃。
```

写数据

```
1. String newData = "New String to write to file..." + System.currentTimeMillis();

2. ByteBuffer buf = ByteBuffer.allocate(48);

3. buf.clear();

4. buf.put(newData.getBytes());

5. buf.flip();

6. int bytesSent = channel.send(buf, new InetSocketAddress("jenkov.com", 80));
```



严禁讨论涉及中国之军/政相关话题，违者会被禁言、封号！



z15732621582 2017-12-29 10:41

回复 2楼

楼主分享的很好



z15732621582 2017-12-29 10:39

回复 1楼

感谢楼主分享

NIO与IO的区别



u010031673 2016年06月24日 20:35 3801

JAVA NIO vs IO 当我们学习了Java NIO和IO后，我们很快就会思考一个问题： 什么时候应该使用IO，什么时候我应该使用NIO 在下文中我会尝试用例子阐述java NIO 和IO...

NIO与传统IO的区别



zhouhl_cn 2011年06月26日 02:19 44585

传统的socket IO中，需要为每个连接创建一个线程，当并发的连接数量非常巨大时，线程所占用的栈内存和CPU线程切换的开销将非常巨大。使用NIO，不再需要为每个线程创建单独的线程，可以用一个含有有限...

NIO与IO的区别


 xyls12345

2014年07月29日 11:00

 10144

分类： java 2-04-23 09:54 1994人阅读 评论(0) 收藏 举报 nio是new io的简称，从jdk1.4就被引入了。现在的jdk已经到了1.6了，

Java中NIO与NIO的区别和使用场景

 xujiangdong1992

2017年06月19日 15:12

 483

这几天主 了NIO，因为之前对IO使用的也比较多，所以有一个简单的对比，并且把学习的成果记录下来。 java.NIO包 里包括三个基本的组件 | buffer：因为NI...

Java NIO与IO的详细区别(通俗篇)

 kingmax54212008

2017年04月20日 22:34

 423

内核空间、用户空间、计算机体系结构、计算机组成原理、.....确实有点儿深奥。 我的新书《代码之谜》会有专门的章节讲 解相关知识，现在写个简短的科普文： 就速度来说 CPU > ...

Java的NIO及与IO区别

 u012440687

2016年09月25日 14:18

 758

请参阅：Java NIO 系列教程NIO初窥 Java NIO (New IO) 是从Java 1.4版本开始引入的一个新的IO API，可以替代标准的J ava IO API NIO和IO主要区别 ...

NIO与传统IO代码区别实例

2018年01月24日 15:01

63KB

下载



几种内存泄露检测工具的比较

 bluehawksky

2014年10月07日 01:44

 14353

概述 内存泄漏(memory leak)指由于疏忽或错误造成程序未能释放已经不再使用的内存的情况，在大型的、复杂的应用程序 中，内存泄漏是常见的问题。当以前分配的一片内存不再需要使用或无法访问时， ...

java中IO与NIO的区别与各自的应用场景

 wodeyuer125

2014年09月22日 13:35

 7209

我应该何时使用IO，何时使用NIO呢？在本文中，我会尽量清晰地解析Java NIO和IO的差异、它们的使用场景，以及它们如 何影响您的代码设计。 Java NIO和IO的主要区别 下...

Java中NIO和IO的比较

 weixin_36380516

2017年04月26日 22:24

 1978

NIO是为了弥补IO操作的不足而诞生的，NIO的一些新特性有：非阻塞I/O，选择器，缓冲以及管道。管道（ Channel ），缓 冲（ Buffer ），选择器（ Selector ）是其主要特征。 概念解释： ...

JAVA IO与NIO优劣浅析

 zmx729618

2016年07月08日 11:24

 1756

NIO 设计背后的基石：反应器模式，用于事件多路分离和分派的体系结构模式。 反应器（ Reactor ）：用于事件多路分离和 分派的体系结构模式 通常的，对一个文件描述符指定的文件或设备， ...

treemap原理


 zhangyuan19880606

2016年04月24日 16:59

 2511

右旋：rotateRight() [java] view plain copy print? private void rotateLeft(Entry p) { if...

Java中HashMap和TreeMap的区别深入理解

 cslbupt 2017年03月03日 15:34 2264


首先介绍一下什么是Map。在数组中我们是通过数组下标来对其内容索引的，而在Map中我们通过对象来对对象进行索引，用来索引的对象叫做key，其对应的对象叫做value。这就是我们平时说的键值对。 ...

Java I 与IO

 keda8997110 2014年02月20日 13:31 52133

Java NIO与IO

java中IO和NIO的区别和适用场景

 zhansong_1987 2015年05月20日 18:05 3534


这几天主 了NIO，因为之前对IO使用的也比较多，所以有一个简单的对比，并且把学习的成果记录下来。 java.NIO包包括三个基本的组件 | buffer：因为NIO是基于缓冲的， ...

关于NIO笔记(一)：IO与NIO的区别

 u014206695 2017年02月24日 17:13 299


NIO的简介：Java NIO (New IO)是从java1.4版本开始引入一个新的IO，可以代替标准的Java IO API。 NIO与原来的IO有同样的作用和目的，但是使用的方式完全...

Java NIO 详解（一）

 suifeng3051 2015年09月14日 11:07 22255


NIO即新的输入输出，这个库是在JDK1.4中才引入的。它在标准java代码中提供了高速的面向块的IO操作。 一、基本概念描述1.1 I/O简介I/O即输入输出，是计算机与外界世界的一个借口。IO操作...

nio是什么

 fgstudent 2014年12月17日 09:26 1826


之前使用openfire的时候接触到底层是用nio实现的，对于nio以前也没接触过，基本都是使用传统的io，正好看到一篇文章介绍挺清楚的。 1.nio 是 java New IO 的简称，在 j...

什么是NIO，与传统IO区别

 guan_sen 2017年12月11日 09:35 127

1：什么是NIO nio 是 java New IO 的简称，在 jdk1.4 里提供的新 api 。 Sun 官方标榜的特性如下： - 为所有的原始类型提供 (Buffer) 缓存支持。 -...

websocket 和webservice--跨平台

 qqMBG 2016年12月30日 15:46 4017

websocket 和webservice区别与联系socket和webservice都有跨平台的优点， 但是： socket偏底层，效率高，但是开发成本大。 w...