NIO

1. JAVA NIO简介

NIO与原来的IO有同样的作用和目的，但是使用方式完全不同，NIO支持面向缓存区的、基于通道的IO操作。NIO将以更加高效的方式进行文件的读写操作。

1. java NIO 与 IO的主要区别

|  |  |
| --- | --- |
| IO | NIO |
| 面向流 | 面向缓存区 |
| 阻塞IO（Blocking IO） | 非阻塞IO（Non Blocking IO） |
| （无） | 选择器 |
|  |  |

**简而言之**

Channel 负责传输

Buffer 负责存储

1. 缓存区（Buffer）
   1. 依据数据类型不同（boolean除外），提供了相应的缓冲区

注：上述缓存区的管理方式几乎一致，通过allocate()获取缓存区

* + 1. ByteBuffer
    2. CharBuffer
    3. ShortBuffer
    4. IntBuffer
    5. LongBuffer
    6. FloatBuffer
    7. DoubleBuffer
  1. 缓存区存取数据的核心方法
     1. put() 写
     2. get() 读
     3. flip() 切换模式
  2. 缓存区的四个核心属性
     1. private int mark = -1; //标记，表示当前position的位置。可以通过reset()恢复到mark的位置
     2. private int position = 0; //位置，表示缓存区正在操作数据的位置。
     3. private int limit; //界限，表示缓存区可以操作数据的大小。（limit后数据不能进行读写）
     4. private int capacity; //容量，表示缓存区的最大存储数据的容量。一旦声明不能改变。
     5. 0 <= mark <= position <= limit <= capacity
  3. 直接缓存区和非直接缓存区
     1. 非直接缓存区：allocate()方法分配缓存区，将缓存区建立在JVM的内存中
     2. 直接缓存区：allocateDirect()方法分配直接缓存区，将缓存区建立在物理内存中，可以提高效率
  4. demo

|  |
| --- |
| **package** com.learn.buf;  **import** java.nio.ByteBuffer;  */\*\*  \* NIO  \* 缓存区  \*  \* 1.依据数据类型不同（boolean除外），提供了相应的缓冲区  \* ByteBuffer  \* CharBuffer  \* ShortBuffer  \* IntBuffer  \* LongBuffer  \* FloatBuffer  \* DoubleBuffer  \* 注：上述缓存区的管理方式几乎一致，通过allocate()获取缓存区  \*  \* 2.缓存区存取数据的两个核心方法  \* put() 写  \* get() 读  \* flip() 切换模式  \*  \* 3.缓存区的四个核心属性  \*     private int mark = -1; //标记，表示当前position的位置。可以通过reset()恢复到mark的位置  \*     private int position = 0; //位置，表示缓存区正在操作数据的位置。  \*     private int limit;   //界限，表示缓存区可以操作数据的大小。（limit后数据不能进行读写）  \*     private int capacity; //容量，表示缓存区的最大存储数据的容量。一旦声明不能改变。  \*  \*     0 <= mark <= position <= limit <= capacity  \*  \*/* **public class** TestBuffer ***{*public static void** main***(***String***[]*** args***) {***ByteBuffer buffer = ByteBuffer.*allocate****(***1024***)***;          String src = **"abcqwe"**;          buffer.put***(***src.getBytes***())***;         System.***out***.println***(*"------put------"*)***;         System.***out***.println***(***buffer.position***())***;         System.***out***.println***(***buffer.limit***())***;         System.***out***.println***(***buffer.capacity***())***;          buffer.flip***()***;          System.***out***.println***(*"------flip------"*)***;         System.***out***.println***(***buffer.position***())***;         System.***out***.println***(***buffer.limit***())***;         System.***out***.println***(***buffer.capacity***())***;          **byte*[]*** dst = **new byte*[***buffer.limit***()]***;         buffer.get***(***dst***)***;          System.***out***.println***(*"---------get--------"*)***;         System.***out***.println***(*new** String***(***dst***))***;         System.***out***.println***(***buffer.position***())***;         System.***out***.println***(***buffer.limit***())***;         System.***out***.println***(***buffer.capacity***())***;       ***} }*** |

1. 通道（Channel）
   1. FileChannel
   2. SocketChannel
   3. SockeServerChannel
   4. DatagramChannel
2. 使用NIO完成网络通信的三个核心
   1. 通道Channel
      1. 继承关系

java.nio.channels接口

    |--SelectableChannel

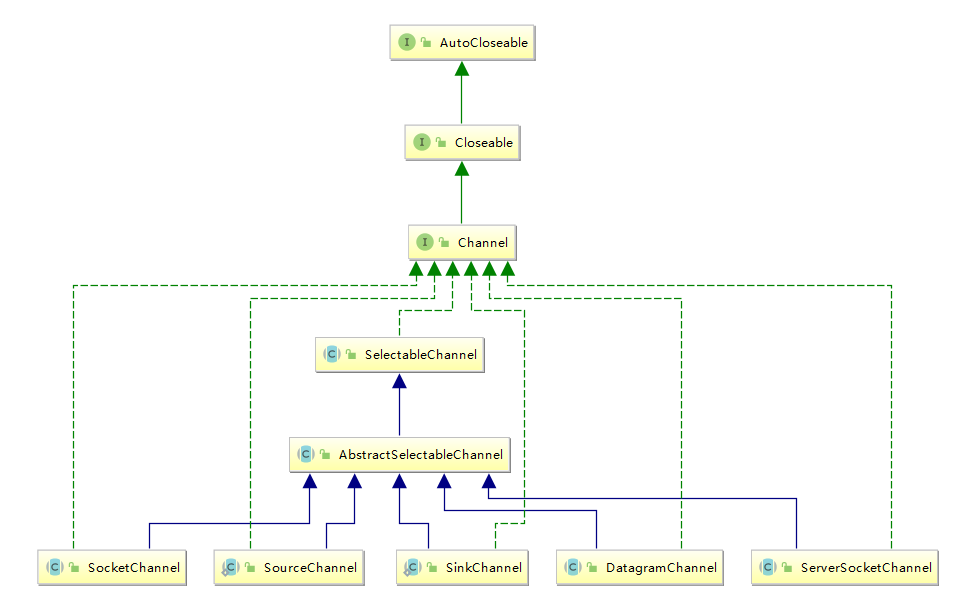
       |--SocketChannel

       |--ServerSocketChannel

       |--DatagramChannel

       |--Pipe.SinkChannel

       |--Pipe.SourceChannel



* 1. 缓存区Buffer
     1. 负责数据的存储
  2. 选择器Selector
     1. 是SelectableChannel的多路复用器，用于监控SelectableChannel的状态

1. NIO的非阻塞式网络通信
   1. NIO的阻塞实现
      1. 客户端

|  |
| --- |
| **package** com.learn;  **import** java.io.IOException; **import** java.net.InetSocketAddress; **import** java.nio.ByteBuffer; **import** java.nio.channels.FileChannel; **import** java.nio.channels.SocketChannel; **import** java.nio.file.Paths; **import** java.nio.file.StandardOpenOption;  */\*\*  \* 客户端  \*/* **public class** Client ***{*public static void** main***(***String***[]*** args***) {***SocketChannel sChannel = **null**;         FileChannel inChannel = **null**;         **try *{****//1.获取通道*sChannel = SocketChannel.*open****(*new** InetSocketAddress***(*"127.0.0.1"**, 9898***))***;              *//要获取本地文件缓存*inChannel = FileChannel.*open****(***Paths.*get****(*"d:/1.zip"*)***, StandardOpenOption.***READ)***;              *//2.分配指定大小的缓冲区*ByteBuffer buffer = ByteBuffer.*allocate****(***1024***)***;              *//3.读取本地文件，并发送到服务端***while *(***inChannel.read***(***buffer***)*** != -1***) {***buffer.flip***()***;                 sChannel.write***(***buffer***)***;                 buffer.clear***()***;             ***}***sChannel.shutdownOutput***()***;              *//4.接收服务端的反馈***while *(***sChannel.read***(***buffer***)*** != -1***) {***buffer.flip***()***;                 System.***out***.println***(*new** String***(***buffer.array***()))***;                 buffer.clear***()***;             ***}          }* catch *(***IOException e***) {***e.printStackTrace***()***;         ***}* finally *{*try *{***sChannel.close***()***;                 inChannel.close***()***;             ***}* catch *(***IOException e***) {***e.printStackTrace***()***;             ***}         }       } }*** |

* + 1. 服务端

|  |
| --- |
| **package** com.learn;  **import** java.io.IOException; **import** java.net.InetSocketAddress; **import** java.nio.ByteBuffer; **import** java.nio.channels.FileChannel; **import** java.nio.channels.ServerSocketChannel; **import** java.nio.channels.SocketChannel; **import** java.nio.file.Paths;  **import static** java.nio.file.StandardOpenOption.***CREATE***; **import static** java.nio.file.StandardOpenOption.***WRITE***;  */\*\*  \* 服务端  \*/* **public class** Server ***{*public static void** main***(***String***[]*** args***) {***ServerSocketChannel ssChannel = **null**;         FileChannel outChannel = **null**;         SocketChannel sChannel = **null**;         **try *{***ssChannel = ServerSocketChannel.*open****()***;             ssChannel.bind***(*new** InetSocketAddress***(*"127.0.0.1"**, 9898***))***;             outChannel = FileChannel.*open****(***Paths.*get****(*"d:/2.zip"*)***, ***WRITE***, ***CREATE)***;              *//接收客户端的数据*sChannel = ssChannel.accept***()***;              ByteBuffer buffer = ByteBuffer.*allocate****(***1024***)***;             **while *(***sChannel.read***(***buffer***)*** != -1***) {***buffer.flip***()***;                 outChannel.write***(***buffer***)***;                 buffer.clear***()***;             ***}****//给客户端返回数据*buffer.put***(*"数据接收完成"**.getBytes***())***;             buffer.flip***()***;             sChannel.write***(***buffer***)***;           ***}* catch *(***IOException e***) {***e.printStackTrace***()***;         ***}* finally *{*try *{***ssChannel.close***()***;                 outChannel.close***()***;                 sChannel.close***()***;             ***}* catch *(***IOException e***) {***e.printStackTrace***()***;             ***}         }     } }*** |

* 1. NIO的非阻塞实现
     1. 客户端

|  |
| --- |
| **package** com.learn;  **import** java.io.IOException; **import** java.net.InetSocketAddress; **import** java.nio.ByteBuffer; **import** java.nio.channels.SocketChannel; **import** java.util.Date; **import** java.util.Scanner;  */\*\*  \* Hello world!  \*/* **public class** Client ***{*public static void** main***(***String***[]*** args***) {***SocketChannel sChannel = **null**;         **try *{****//1.获取通道*sChannel = SocketChannel.*open****(*new** InetSocketAddress***(*"127.0.0.1"**, 9898***))***;             *//2.切换非阻塞模式*sChannel.configureBlocking***(*false*)***;             *//3.分配指定大小的缓存区*ByteBuffer buffer = ByteBuffer.*allocate****(***1024***)***;             *//4.发送数据到服务器*Scanner scanner = **new** Scanner***(***System.***in)***;             **while *(***scanner.hasNext***()) {*final** String next = scanner.next***()***;                  buffer.put***((*new** Date***()***.toString***()*** + **"\n"** + next***)***.getBytes***())***;                 buffer.flip***()***;                 sChannel.write***(***buffer***)***;                 buffer.clear***()***;             ***}           }* catch *(***IOException e***) {***e.printStackTrace***()***;         ***}* finally *{*try *{***sChannel.close***()***;             ***}* catch *(***IOException e***) {***e.printStackTrace***()***;             ***}         }       } }*** |

* + 1. 服务端

|  |
| --- |
| **package** com.learn;  **import** java.io.IOException; **import** java.net.InetSocketAddress; **import** java.nio.ByteBuffer; **import** java.nio.channels.SelectionKey; **import** java.nio.channels.Selector; **import** java.nio.channels.ServerSocketChannel; **import** java.nio.channels.SocketChannel; **import** java.util.Iterator;  */\*\*  \* Hello world!  \*/* **public class** Server ***{*public static void** main***(***String***[]*** args***) {***ServerSocketChannel ssChannel = **null**;         **try *{****//1.获取通道*ssChannel = ServerSocketChannel.*open****()***;              *//2.切换非阻塞模式*ssChannel.configureBlocking***(*false*)***;              *//3.绑定连接*ssChannel.bind***(*new** InetSocketAddress***(*"127.0.0.1"**, 9898***))***;              *//4.获取选择器*Selector selector = Selector.*open****()***;              *//5.将通道注入到选择器上，并且指定“监听接收事件”*ssChannel.register***(***selector, SelectionKey.***OP\_ACCEPT)***;              *//6.轮询获取选择器上已经“准备就绪”的事件***while *(***selector.select***()*** > 0***) {****//7.***final** Iterator***<***SelectionKey***>***iterator = selector.selectedKeys***()***.iterator***()***;                 **while *(***iterator.hasNext***()) {****//8.获取准备“就绪”的事件***final** SelectionKey selectionKey = iterator.next***()***;                      *//9.判断具体是什么事件***if *(***selectionKey.isAcceptable***()) {****//10.若准备就绪，获取客户端连接***final** SocketChannel sChannel = ssChannel.accept***()***;                         *//11.设置非阻塞模式*sChannel.configureBlocking***(*false*)***;                         *//12.将该通道注册到选择器上*sChannel.register***(***selector, SelectionKey.***OP\_READ)***;                     ***}* else if *(***selectionKey.isReadable***()) {****//13.获取当前选择器上“读就绪”状态的通道*SocketChannel sChannel = ***(***SocketChannel***)*** selectionKey.channel***()***;                         *//14.读取数据*ByteBuffer buffer = ByteBuffer.*allocate****(***1024***)***;                          **while *(***sChannel.read***(***buffer***)*** != -1***) {***buffer.flip***()***;                             System.***out***.println***(*new** String***(***buffer.array***()))***;                             buffer.clear***()***;                         ***}                      }                 }             }          }* catch *(***IOException e***) {***e.printStackTrace***()***;         ***}* finally *{*try *{***ssChannel.close***()***;             ***}* catch *(***IOException e***) {***e.printStackTrace***()***;             ***}         }     } }*** |

1. 管道
2. java NIO 2 (Path,Paths与Files)