Java第五阶段\_day01\_Nio案例

**案例一：使用ByteBuﬀer**

public static void main(String[] args) {  
    //1创建缓冲区  
    ByteBuffer buffer=ByteBuffer.allocate(1024);  
    //2向缓冲区中添加内容buffer.put("helloworld".getBytes());  
    //3切换为读模式  
    buffer.flip();  
    //4获取单个字节  
    //buffer.get();  
    //5获取多个字节  
    byte[] data=new byte[buffer.limit()]; buffer.get(data);  
    System.out.println(new String(data));  
    //6清空缓冲区buffer.clear();  
}

**案例二：使用capacity,position和limit**

public class Demo2 {

public static void main(String[] args) {

//1创建直接缓冲区

ByteBuffer buf=ByteBuffer.allocateDirect(1024);

System.out.println("---------写之前----------");

System.out.println("capacity:"+buf.capacity());

System.out.println("limit:"+buf.limit());

System.out.println("position:"+buf.position());

//2写入数据

buf.put("abcdefg".getBytes());

System.out.println("---------写之后----------");

System.out.println("capacity:"+buf.capacity());

System.out.println("limit:"+buf.limit());

System.out.println("position:"+buf.position());

//3切换为读取模式

buf.flip();

System.out.println("---------切换为读取模式----------");

System.out.println("capacity:"+buf.capacity());

System.out.println("limit:"+buf.limit());

System.out.println("position:"+buf.position());

byte[] data=new byte[3];

buf.get(data);

System.out.println("---------读取之后----------");

System.out.println(new String(data,0,3));

System.out.println("capacity:"+buf.capacity());

System.out.println("limit:"+buf.limit());

System.out.println("position:"+buf.position());

buf.mark();

buf.get(data);

System.out.println("----------mark()----------");

System.out.println("---------再次读取之后----------");

System.out.println(new String(data,0,3));

System.out.println("capacity:"+buf.capacity());

System.out.println("limit:"+buf.limit());

System.out.println("position:"+buf.position());

buf.reset();

buf.get(data);

System.out.println("------reset() 重置------------");

System.out.println(new String(data,0,3));

System.out.println("capacity:"+buf.capacity());

System.out.println("limit:"+buf.limit());

System.out.println("position:"+buf.position());

buf.clear();

System.out.println("---------clear()清空--------");

System.out.println("capacity:"+buf.capacity());

System.out.println("limit:"+buf.limit());

System.out.println("position:"+buf.position());

}

}

**案例三：写入文本文件**

//1创建FileOutputStream  
FileOutputStream fos=new FileOutputStream("d:\\out.txt");  
//2获取通道  
FileChannel outChannel = fos.getChannel();  
//3创建缓冲区  
ByteBuffer buffer=ByteBuffer.allocate(1024);  
//4向缓冲区中放入数据  
buffer.put("hello world".getBytes());  
//5切换为读模式buffer.flip();  
//写入  
outChannel.write(buffer);  
//6关闭outChannel.close();  
System.out.println("写入完毕");

**案例四：读取文本文件**

//1创建FileInputStream  
FileInputStream fis=new FileInputStream("d:\\out.txt");  
//2创建通道  
FileChannel inChannel = fis.getChannel();  
//3创建缓冲区  
ByteBuffer buffer=ByteBuffer.allocate(1024);   
int len=inChannel.read(buffer);  
//4处理数据buffer.flip();  
String data=new  String(buffer.array(),0,len);   
System.out.println(data);  
​  
//5关闭inChannel.close();

**案例五：复制文件**

       FileInputStream fis=new FileInputStream("d:\\a.txt");  
 FileOutputStream fos=new FileOutputStream("d:\\b.txt");  
 FileChannel sourceCh=fis.getChannel();   
 FileChannel destCh=fos.getChannel();   
 destCh.transferFrom(sourceCh,0,sourceCh.size());   
 sourceCh.close();   
 destCh.close();

**案例六：ServerSocketChannel、SocketChannel实现阻塞式网络编程**

实现代码

public class TcpServer {  
    public static void main(String[] args) throws Exception{  
    //1创建ServerSocketChannel  
    ServerSocketChannel ssc=ServerSocketChannel.open();  
    //2绑定地址  
    ssc.bind(new InetSocketAddress(1234));  
    //3监听  
    System.out.println("服务器已启动");  
    SocketChannel sc=ssc.accept();  
​  
    //4创建缓冲区  
    ByteBuffer buffer=ByteBuffer.allocate(1024);   
​  
    int len=sc.read(buffer);  
    buffer.flip();   
​  
    String string=new String(buffer.array(),0, len);  
    System.out.println(string);  
    //5关闭  
    sc.close();  
   }  
}

public class TcpClient {  
    public static void main(String[] args) throws Exception {  
    //1创建SocketChannel  
    SocketChannel sc=SocketChannel.open(new InetSocketAddress("127.0.0.1", 1234));  
​  
    //2创建缓冲区  
    ByteBuffer buffer=ByteBuffer.allocate(1024);  
    buffer.put("你还好吗?".getBytes()); buffer.flip();  
​  
    //3发送数据sc.write(buffer);  
    //4关闭  
    sc.close();  
   }  
}

**案例七：实现非阻塞式网络通信**

public class ChatServer{  
    public static void main(String[] args) throws Exception {  
    //1创建ServerSocketChannel  
    ServerSocketChannel ssc=ServerSocketChannel.open();  
    //2设置为非阻塞式  
    ssc.configureBlocking(false);  
    //3绑定地址  
    ssc.bind(new InetSocketAddress("127.0.0.1",9999));  
    //4创建选择器  
    Selector selector=Selector.open();  
    //5注册选择器ssc.register(selector,SelectionKey.OP\_ACCEPT); while(selector.select()>0){  
    Iterator<SelectionKey> it = selector.selectedKeys().iterator(); while(it.hasNext()){  
    SelectionKey selectionKey = it.next();  
    //有客户端连接过来了   
    if(selectionKey.isAcceptable()){  
    SocketChannel socketChannel=ssc.accept(); socketChannel.configureBlocking(false);//设置为非阻塞式编程 socketChannel.register(selector, SelectionKey.OP\_READ);//把客户端socketChannel注册到selctor  
   }else if(selectionKey.isReadable()){  
    //获取SocketChannel  
    SocketChannel channel = (SocketChannel) selectionKey.channel();  
    //创建缓冲区  
    ByteBuffer buffer = ByteBuffer.allocate(1024);   
    int len = 0;  
    while((len = channel.read(buffer)) > 0 ){   
    buffer.flip();  
    System.out.println(new String(buffer.array(), 0, len)); buffer.clear();  
   }  
    if(len==-1){//客户端已经退出  
    channel.close();  
   }  
   }  
    it.remove();  
   }  
   }  
 }  
}

public class ChatClient{  
    public static void main(String[] args) throws Exception{  
    //1创建SocketChannel  
    SocketChannel sc=SocketChannel.open(new InetSocketAddress("127.0.0.1", 9999));  
    //2设置为非阻塞式sc.configureBlocking(false);  
    //3创建缓冲区  
    ByteBuffer buffer=ByteBuffer.allocate(1024); Scanner input=new Scanner(System.in); while(input.hasNext()){  
    String s=input.nextLine(); buffer.put(s.getBytes()); buffer.flip(); sc.write(buffer); buffer.clear();  
    if(s.equals("886")){ break;  
   }  
   }  
    //4关闭sc.close();  
   }  
}