http://wiki.jikexueyuan.com/project/java-nio-zh/java-nio-socketchannel.html

在Java NIO体系中，SocketChannel是用于TCP网络连接的套接字接口，相当于Java网络编程中的Socket套接字接口。创建SocketChannel主要有两种方式，如下：

1. 打开一个SocketChannel并连接网络上的一台服务器。
2. 当ServerSocketChannel接收到一个连接请求时，会创建一个SocketChannel。

建立一个SocketChannel连接

打开一个SocketChannel可以这样操作：

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

关闭一个SocketChannel连接

关闭一个SocketChannel只需要调用他的close方法，如下：

```
socketChannel.close();
```

从SocketChannel中读数据

从一个SocketChannel连接中读取数据，可以通过read()方法，如下：

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

首先需要开辟一个Buffer。从SocketChannel中读取的数据将放到Buffer中。

接下来就是调用SocketChannel的read()方法.这个read()会把通道中的数据读到Buffer中。read()方法的返回值是一个int数据，代表此次有多少字节的数据被写入了Buffer中。如果返回的是-1,那么意味着通道内的数据已经读取完毕，到底了（链接关闭）。

向SocketChannel写数据

向SocketChannel中写入数据是通过write()方法，write也需要一个Buffer作为参数。下面看一下具体的示例：

```
String newData = "New String to write to file..." +
System.currentTimeMillis();
ByteBuffer buf = ByteBuffer.allocate(48);
buf.clear();
buf.put(newData.getBytes());
buf.flip();
while(buf.hasRemaining()) {
```

```
    channel.write(buf);
}
```

仔细观察代码，这里我们把write()的调用放在了while循环中。这是因为我们无法保证在write的时候实际写入了多少字节的数据，因此我们通过一个循环操作，不断把Buffer中数据写入到SocketChannel中直到Buffer中的数据全部写入为止。

非阻塞模式

我们可以把SocketChannel设置为non-blocking（非阻塞）模式。这样的话在调用connect(), read(), write()时都是异步的。

connect()

如果我们设置了一个SocketChannel是非阻塞的，那么调用connect()后，方法会在链接建立前就直接返回。为了检查当前链接是否建立成功，我们可以调用finishConnect(),如下：

```
socketChannel.configureBlocking(false);
socketChannel.connect(new
InetSocketAddress("http://jenkov.com", 80));
while(! socketChannel.finishConnect() ){
    //wait, or do something else...
}
```

write()

在非阻塞模式下，调用write()方法不能确保方法返回后写入操作一定得到了执行。因此我们需要把write()调用放到循环内。这和前面在讲write()时是一样的，此处就不在代码演示。

read()

在非阻塞模式下，调用read()方法也不能确保方法返回后，确实读到了数据。因此我们需要自己检查的整型返回值，这个返回值会告诉我们实际读取了多少字节的数据。

Selector结合非阻塞模式

SocketChannel的非阻塞模式可以和Selector很好的协同工作。把一个活多个SocketChannel注册到一个Selector后，我们可以通过Selector指导哪些channels通道是处于可读，可写等等状态的。后续我们会再详细阐述如果联合使用Selector与SocketChannel。