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

在Java NIO中，ServerSocketChannel是用于监听TCP链接请求的通道，正如Java网络编程中的ServerSocket一样。
ServerSocketChannel实现类位于java.nio.channels包下面。 下面是一个示例程序：

```
ServerSocketChannel serverSocketChannel =
ServerSocketChannel.open();
serverSocketChannel.socket().bind(new
InetSocketAddress(9999));
while(true) {
  SocketChannel socketChannel = serverSocketChannel.accept();
  //do something with socketChannel...
}
```

打开ServerSocketChannel

打开一个ServerSocketChannel我们需要调用他的open()方法，例如：

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

关闭ServerSocketChannel

关闭一个ServerSocketChannel我们需要调用他的close()方法，例如：

```
serverSocketChannel.close();
```

监听链接

通过调用accept()方法,我们就开始监听端口上的请求连接。当accept()返回时，他会返回一个SocketChannel连接实例,实际上accept()是阻塞操作,他会阻塞带去线程知道返回一个连接;

很多时候我们是不满足于监听一个连接的,因此我们会把accept()的调用放到循环中,就像这样：

```
while(true){
    SocketChannel socketChannel =
serverSocketChannel.accept();
    //do something with socketChannel...
}
```

当然我们可以在循环体内加上合适的中断逻辑，而不是单纯的在while循环中写true，以此来结束循环监听；

非阻塞模式

实际上ServerSocketChannel是可以设置为非阻塞模式的。在非阻塞模式下，调用accept()函数会立刻返回，如果当前没有请求的链接，那么返回值为空null。因此我们需要手动检查返回的SocketChannel是否为空，例如：

```
ServerSocketChannel serverSocketChannel =
ServerSocketChannel.open();
serverSocketChannel.socket().bind(new
InetSocketAddress(9999));
serverSocketChannel.configureBlocking(false);
while(true){
    SocketChannel socketChannel =
serverSocketChannel.accept();
    if(socketChannel != null){
        //do something with socketChannel...
    }
}
```