Java networking 笔记

学而时习之，不亦说乎？

--孔子

本文主要是参考《java networking》这本书，将书中的例子加以实践。涉及JAVA部分，一般都有JAVA例子来实现。涉及TCP协议部分，请参考另外一篇《TCP阅读笔记》

# why java networking?

# Basic network concept

# Basic web concept

# Streams

# Threads

# Looking up internet address

# ULRs and URIs

ProtocolTester 这个例子用生成URL对象的方式，检测协议是否支持；

URLSpliter 解析URL对象的各个组成部分

SourceViewer 下载一个网页

ConentGetter 获取返回的URL对象内容的类型

EncoderTest 展示URLEncoder.encode()方法的用法 encode的意思是有时候url中会有一些敏感字符，比如”~ 等非法非法字符，为了保证这些字符能正常出现在url中，需要对这些字符进行转化。

QueryString 这个类主要是对一个url对象中query部分的内容进行encode，防止在url中出现非法字符

DMoz 这个类主要是通过一个URL类拼装查询的链接，从[http://www.dmoz.org](http://www.dmoz.org/)这个网站获取查询结果

Baidu 这个类和DMoz 类似，只不过是访问baidu，获取搜索结果。结果是乱码，看来中文的网站还是挺麻烦的。

# HTML in swing

# Chap9 Sockets for client

LowPortScanner 通过这个例子，说明Socket对象

DaytimeClient 访问外网的日期服务

TimeClient 访问外网的时间服务

EchoClient

Socket用完后要关闭， 可以彻底关闭，也可以用半关闭的方式

Socket有很多参数

很多例子：

# Chap10 Sockets for server

这章主要是通过一些常用的Server的示例代码，说明SocketServer的用法

10.1

LocalPortScanner 通过初始化ServerSocket对象，来搜索本地的端口；

DayTimeServer 服务端发布日期服务，返回日期信息给客户端

TimeServer 服务端发布时间服务，返回精确的系统个时间给客户端

RandomPort new一个ServerSocket对象的时候，如果不指定端口，会监听一个随机端口

ServerSocket的一些重要参数

server.setSoTimeout(timeout) // ServerSocket对象accept()方法等待客户端连接的超时时间

server.getReuseAddress() //是否允许客户端复用原来的socket通道

server.getReceiveBufferSize() // 服务端接收客户端连接的buffer大小，默认为8192

* 1. socket for server

这个章节主要是几个服务端socket程序的示例。感觉比较实用

ClientTester Server监听某个端口，打印客户端发送的请求.这个程序和之前程序不同之处为,这次接收客户端请求/返回信息给客户端都是新起一个线程来处理.测试这个程序只需要在浏览器中输入<http://localhost:port> 就行。服务端就能打印浏览器的报文

SingleFileHTTPServer 单文件的服务器，main方法在服务端起了一个监听，客户端浏览器访问服务器端的对应端口，服务端会返回一个文件流给客户端。

Redirector HTTP重定向服务。在服务端发起一个监听，接收到来自客户端的重定向请求，比如<http://localhost:9999/getFile> 重定向到http://localhost:9088

服务端接收到请求为： GET /getFile HTTP/1.1 然后服务端根据newSite(http://localhost:9088)和filePath(这里是/getFile)，最后拼装成新的url:

<http://localhost:9088/getFile> 主要的技术细节就是将HTTP请求中的Location属性改为新地址

JHTTP web server 这个Server和之前的SingleFileHTTPServer类似，分为2部分：

1. JHTTP 服务器主进程。主要内容是各种constructor，然后初始化一个线程池。
2. RequestProcessor 用线程池处理HTTP请求

# Secure socket

# Non-blocking I/O

## An Example Client

ChargenClient.java

通过SocketChannel的方式，获取服务端的内容。第一次用到了NIO的API

## An Example Server

ChargenServer.java

以上这两个两个例子介绍了non-blocking I/O的用法：

## Buffers

Java NIO 引入了新的概念，就是Buffer/channel，这个和我们之前经常使用的java stream有点类似，和java stream不同的是:

1. Buffer/channel是block为单位，stream以byte为单位；
2. Buffer/channel支持读写；

Buffer有以下这些概念：

1. Position

Buffer将要读写的位置；

1. Capacity

Buffer的容量；

1. Limit

Buffer可供读写的限制范围，超过了这个范围，就没法读写；

1. Mark

客户端对buffer做的位置标记；

Buffer提供的其他方法：

clear() 清空buffer position->0 limit->capacity

rewind() 重置 position->0 limit->limit

flip() limit->current position position->0

remaining() number of elements between the current postion and the limit

### Creating buffers

### Filling and draining

### Bulk methods

### Data conversion

IntgenServer.java

### View Buffers

IntgenClient.java

### Compacting buffers

EchoServer.java

### Duplicating buffers

NonblockingSingleFielHTTPServer.java

### Slicing Buffers

### Making and resetting

### Object methods

## Channels

### SocketChannel

### ServerSocketChannel

### The channel class

## Readiness Selection

### The Selector class

### The SelectionKey class

# 13 UDP Datagrams and Sockets

尝试学习一下UDP编程

## The UDP Protocol

## The DatagramPacket Class

## The Datagram**S**ocket Class

|  |
| --- |
| UDPDiscardClient.java  UDPDiscardServer.java |

这两个代码是一对，client发送消息，server接收消息。看起来和TCP差不多。

## Some Useful Applications

|  |
| --- |
| UDPPoke.java  UDPTimeClient.java  UDPServer.java  UDPEchoClient.java  SenderThread.java |

## Datagram Channel

|  |
| --- |
| UDPDiscardServerWithChannels.java  UDPEchoServerWithChannels.java  UDPEchoClientWithChannels.java |

# Multicast Sockets

## 代码说明

|  |
| --- |
| MulticastSender.java  MulticastSniffer.java |

这两组代码，说明了Multicast的用法。上一章的几个Demo，都是unicast模式的，就是点对点模式的。这两个代码演示的是群发模式。我们启动多个MulticastSniffer代码，然后启动MulticastSender，每个MulticastSniffer进程都会收到消息。

其中需要特别说明的是，这两个代码，一个客户端一个服务端，分别加入了一个组，地址为："all-systems.mcast.net"。这个地址比较特殊，代表我们局域网(子网)的一个组播地址。一旦服务端网这个组发送了一个消息，局域网中所有客户端都会收到消息。

## 总结

Multicast显然是一个很不错的特性。可以用于(tomcat)集群中，节点之间的组播通信。虽然还有很多问题需要了解，比如Sniffer节点多了是否会影响到消息接收的时效性。子网的范围究竟是多大？如果tomcat集群的节点不在同一个节点，集群内部的通信是否会受到影响？等等。需要再深入研究。但是目前了解的知识，已经给tomcat distribute session manager打好了基础。

## What is a Multicast Socket?

## Working with Multicast Sockets

## Two Simple Examples

|  |
| --- |
| MulticastSniffer.java  MulticastSender.java |

# URL Connections

# Protocol Handlers

# Content Handlers

# Remote Method Invocation

# The Java Mail API