**1 InetAddress类**

Java提供了InetAddress代表IP地址，InetAddress还有两个子类：Inet4Address与Inet6Address

InetAddress没有提供构造类，而是提供了两个静态方法获取InetAddress实例：

GetByName(String host):根据主机名获取InetAddress对象

GetByAddredd(byte[] addr):根据原始的IP地址获取InetAddress对象

**public** **class** nettest {

**public** **static** **void** main(String[] args){

**try**{

InetAddress ip = InetAddress.*getByName*("www.baidu.com");

System.*out*.println("www.baidu.com 是否可达：" + ip.isReachable(10000));

System.*out*.println(ip.getHostAddress());

InetAddress localIp = InetAddress.*getByAddress*(**new** **byte**[]{127,0,0,1});

System.*out*.println(localIp.isReachable(5000));

System.*out*.println(localIp.getCanonicalHostName());

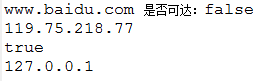
}**catch**(Exception e){

e.printStackTrace();

}

}

}



**2 使用URLDecoder和URLEncoder**

URLDecoder与URLEncoder用于完成普通字符串与application/x-www.form-urlencoded MIME字符串之间的相互转换

http://www.baidu.com/s?wd=**%E7%96%AF%E7%8B%82java%E8%AE%B2%E4%B9%89**&rsv\_bp=0&ch=&tn=baidu&bar=&rsv\_spt=3&ie=utf-8&rsv\_sug3=6&rsv\_sug=0&rsv\_sug4=153&rsv\_sug1=5&inputT=7446

上面的黑体就是application/x-www.form-urlencoded MIME字符串

**try**{

String mimeString = "%B7%E8%BF%F1java%BD%B2%D2%E5";

String keyWord = URLDecoder.*decode*(mimeString, "GBK");

System.*out*.println(keyWord);

String srcString = "疯狂java讲义";

keyWord = URLEncoder.~~encode~~(srcString);

System.*out*.println(keyWord);

}**catch**(Exception e){ }



**3 使用URL和URLConnection**

比较重要的方法：

URLConnection openConnection():返回一个URLConnection对象，它代表了与URL所引用的远程对象的连接

InputStream openStream():打开与此URL的连接，并返回一个用于读取该URL资源的InputStream

下面采用URLConnection做一个http多线程下载：

**下载分配类：**

**public** **class** DownUtil {

**private** **int** threadNum;//线程数

**private** **int** fileSize;//文件大小

**private** String path;//资源下载位置

**private** String filePath;//文件保存位置

**private** DownLoadThread[] downLoadThread;//下载线程

**public** DownUtil(String path, String filePath, **int** threadNum){

**this**.path = path;

**this**.filePath = filePath;

**this**.threadNum = threadNum;

downLoadThread = **new** DownLoadThread[threadNum];

}

**public** **void** DownLoad() **throws** Exception{

URL url = **new** URL(path);

HttpURLConnection httpUrlConnection = (HttpURLConnection)url.openConnection();

httpUrlConnection.setConnectTimeout(5 \* 1000);//设置连接超时

httpUrlConnection.setRequestMethod("GET");//设置请求方式

httpUrlConnection.setRequestProperty("Accept-Language", "zh-CN");

httpUrlConnection.setRequestProperty("Charset", "UTF-8");

httpUrlConnection.setRequestProperty("Connection", "Leep-Alive");

httpUrlConnection.setRequestProperty("Accept", "\*/\*");

fileSize = httpUrlConnection.getContentLength();//获取文件大小

httpUrlConnection.disconnect();//释放连接

**int** currentPartSize = fileSize / threadNum + 1;

RandomAccessFile file = **new** RandomAccessFile(filePath, "rw");//产生一个随机访问的文件

file.setLength(fileSize);

file.close();

**for**(**int** i = 0; i < threadNum; i++){

**int** startPos = i \* currentPartSize;

RandomAccessFile currentPart = **new** RandomAccessFile(filePath, "rw");

currentPart.seek(startPos);//定位到了该文件的起始位置

downLoadThread[i] = **new** DownLoadThread(path, startPos, currentPartSize, currentPart);

downLoadThread[i].start();

}

}

**public** **float** GetCompleteRate(){

**int** sumSize = 0;

**for**(**int** i = 0; i < threadNum; i++){

sumSize += downLoadThread[i].length;

}

**return** (**float**)sumSize / fileSize;

}

}

**下载线程类：**

**public** **class** DownLoadThread **extends** Thread{

**private** String path;

**private** **int** startPos;

**private** **int** currentPartSize;

**private** RandomAccessFile currentPart;

**public** **int** length;//该线程已经下载字节数

**public** DownLoadThread(String path, **int** startPos, **int** currentPartSize, RandomAccessFile currentPart){

**this**.path = path;

**this**.startPos = startPos;

**this**.currentPartSize = currentPartSize;

**this**.currentPart = currentPart;

}

@Override

**public** **void** run() {

**try**{

URL url = **new** URL(path);

HttpURLConnection httpUrlConnection = (HttpURLConnection)url.openConnection();

httpUrlConnection.setConnectTimeout(5 \* 1000);

httpUrlConnection.setRequestMethod("GET");

httpUrlConnection.setRequestProperty("Accept\_language", "zh\_CN");

httpUrlConnection.setRequestProperty("Charset", "UTF-8");

httpUrlConnection.setRequestProperty("Accept", "\*/\*");

InputStream inputStream = httpUrlConnection.getInputStream();

inputStream.skip(**this**.startPos);//跳过startPos个字节，表明该线程只下载自己负责的部分文件

**byte**[] buffer = **new** **byte**[1024];

**int** hasRead = 0;

**while**(length < currentPartSize && (hasRead = inputStream.read(buffer)) != -1){

currentPart.write(buffer, 0, hasRead);

length += hasRead;

}

currentPart.close();

inputStream.close();

}**catch**(Exception e){

e.printStackTrace();

}

}

}

**主函数调用类：**

**public** **class** multithreaddownload {

**public** **static** **void** main(String[] args){

**final** DownUtil downUtil = **new** DownUtil("http://www.youku.com", "E:\\load\\download.txt", 5);

**try**{

downUtil.DownLoad();

**new** Thread(){

@Override

**public** **void** run(){

**while**(downUtil.GetCompleteRate() <= 1.0f){

System.*out*.println("已完成：" + (**int**)(downUtil.GetCompleteRate() \* 100) + " % 100");

**try**{

Thread.*sleep*(1);

}**catch**(Exception e){

e.printStackTrace();

}

}

}

}.start();

}**catch**(Exception e){

e.printStackTrace();

}

}

}

如果要实现多线程的断点下载，则需要增加一个额外的配置文件，其实所有的断点下载工具都会在下载开始时生成两个文件：一个是与网络资源具有相同大小的空文件，一个是配置文件，该配置文件分别为记录每个线程已经下载到哪个字节，当网络断开后再次开始下载时，每个线程根据配置文件里的记录位置向后下载即可。

**4 基于TCP的网络编程**

服务器端：

Java中能接受其他通信实体连接请求的是ServerSocket， 该方法将返回一个与客户端Socket对应的Socket：

Socket accept：如果接受到一个客户端Socket的连接请求，该方法返回一个与客户端Socket对应的Socket

创建Socket：

ServerSocket(int port)：用指定的端口port来创建一个ServerSocket，该端口为一个有效端口，0-65535

ServerSocket(int port, int backlog):增加一个用来改变连接队列长度的参数

ServerSocket(int port, int backlog, InetAddress localAddr):在机器存在多个IP地址情况下，允许通过localAddr参数来指定将ServerSocket绑定到指定的IP地址

当客户端与服务器端产生了对应的Socket后，就可以通过这两个Socket进行通信了，Socket提供了如下方法来获取输入流和输出流：

InputStream getInputStream():返回该Socket对象对应的输入流，让程序通过该输入流从Socket中取出数据

OutputStream getOutputStream():返回该Socket对应对应的输出流，让程序通过该输出流向Socket中输出数据

**一个简单的服务器与客户端的通信范例：**

**public** **class** netServer {

**public** **static** **void** main(String[] args){

**try**{

ServerSocket serverSocket = **new** ServerSocket(3000);

**while**(**true**){

Socket socket = serverSocket.accept();

PrintStream ps = **new** PrintStream(socket.getOutputStream());

ps.println("您收到了来自服务器的祝福");

ps.close();

socket.close();

}

}**catch**(Exception e){

}

}

}

**public** **class** netClient {

**public** **static** **void** main(String[] args){

**try**{

Socket socket = **new** Socket("127.0.0.1", 3000);

BufferedReader bufferedReader = **new** BufferedReader(**new** InputStreamReader(socket.getInputStream()));

String line = bufferedReader.readLine();

System.*out*.println("来自服务器的数据：" + line);

bufferedReader.close();

socket.close();

}**catch**(Exception e){

e.printStackTrace();

}

}

}

运行结果：

