笔记总链接：http://bbs.itheima.com/thread-200600-1-1.html

Chapter 9 网络编程

了解客户端和服务器端原理

最常见的客户端：浏览器，IE/chrome。

最常见的服务端：服务器，Tomcat。

为了了解其原理：

1. 自定义服务端

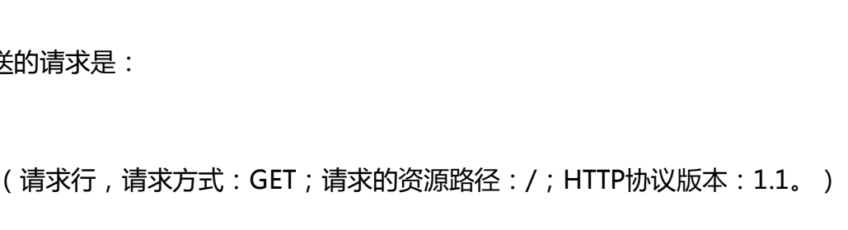
使用已有的客户端IE，了解一下客户端向服务端发了什么请求。

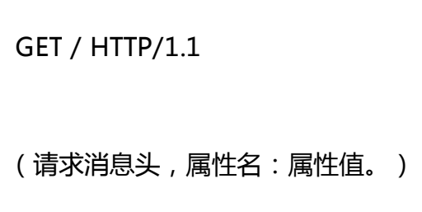
代码：

```java
import java.net.ServerSocket;
import java.net.Socket;
import java.io.InputStream;
import java.io.PrintWriter;
import java.io.IOException;

public class MyTomcat
{
        public static void main(String[] args) throws IOException {

                ServerSocket ss = new ServerSocket(9090);

                Socket s = ss.accept();
                System.out.println(s.getInetAddress().getHostAddress() + "......connected");

                InputStream in = s.getInputStream();

                byte[] buf = new byte[1024];

                int len = in.read(buf);

                String text = new String(buf,0,len);

                System.out.println(text);

                //给客户端一个反馈信息。
                PrintWriter out = new PrintWriter(s.getOutputStream(),true);

                out.println("<font color='red' size='7'>欢迎光临</font>");

                s.close();
                ss.close();
        }
}
```

复制代码

运行结果：





```
D:\code\day26>javac MyTomcat.java

D:\code\day26>java MyTomcat
0:0:0:0:0:0:0:1......connected
GET / HTTP/1.1
Host: localhost:9090
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0
.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/43.0.2357.81 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: zh-CN,zh;q=0.8
```

发送的请求是：

（请求行，请求方式：GET；请求的资源路径：/；HTTP协议版本：1.1。）

GET / HTTP/1.1

（请求消息头，属性名：属性值。）

Host: localhost:9090

Connection: keep-alive

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.81 Safari/537.36

Accept-Encoding: gzip, deflate, sdch

Accept-Language: zh-CN,zh;q=0.8

P.S.

1、消息头中属性名及属性值的具体含义，初学者不用追究，在JavaWeb课程中将会深入讲解。

2、HTTP是一个客户端和服务端请求和应答的标准，客户端按照HTTP的标准发送数据到服务端，服务端按照HTTP的标准解析收到的数据。很多软件都内置了此标准。

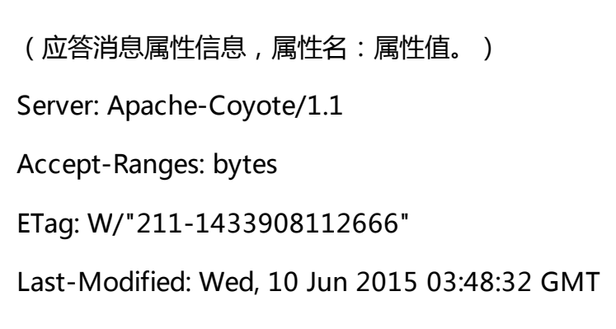
2. 模拟一个浏览器获取信息

代码：

```java
import java.net.Socket;
import java.io.PrintWriter;
import java.io.InputStream;
import java.io.IOException;

public class MyBrowser
{
        public static void main(String[] args) throws IOException {

                Socket s = new Socket("192.168.1.100",8080);

                //模拟浏览器，向tomcat服务端发送符合http协议的请求消息。
                PrintWriter out = new PrintWriter(s.getOutputStream(),true);
                out.println("GET /myweb/1.html HTTP/1.1");
                out.println("Accept: */*");
                out.println("Host: 192.168.1.100:8080");
                out.println("Connection: close");
                out.println();
                out.println();

                InputStream in = s.getInputStream();

                byte[] buf = new byte[1024];
                int len = in.read(buf);

                String str = new String(buf,0,len);
                System.out.println(str);

                s.close();
        }
}
```

复制代码

运行结果：



```
D:\code\day26>java MyBrowser
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: W/"211-1433908112666"
Last-Modified: Wed, 10 Jun 2015 03:48:32 GMT
Content-Type: text/html
Content-Length: 211
Date: Wed, 10 Jun 2015 03:52:16 GMT
Connection: close

<html>
                        <head>
                                                <title>这是我的网页</title>
                        </head>
                        <body>
                                                <h1>欢迎光临</h1>
                                                <font size='5' color='red'>这是
一个tomcat服务器中的资源，是一个html网页。</html>
                        </body>
</html>
```

HTTP服务端发回的应答消息：

（应答行，HTTP的协议版本：1.1；应答状态码：200；应答状态描述信息：OK。）

HTTP/1.1 200 OK

（应答消息属性信息，属性名：属性值。）

Server: Apache-Coyote/1.1

Accept-Ranges: bytes

ETag: W/"211-1433908112666"

Last-Modified: Wed, 10 Jun 2015 03:48:32 GMT

Content-Type: text/html

Content-Length: 211

Date: Wed, 10 Jun 2015 03:52:16 GMT

Connection: close

P.S.

1、应答行中属性名及属性值的具体含义，初学者不用追究，在JavaWeb课程中将会深入讲解。

URL&URLConnection

URI：统一资源标示符。

URL：统一资源定位符，也就是说根据URL能够定位到网络上的某个资源，它是指向互联网“资源”的指针。

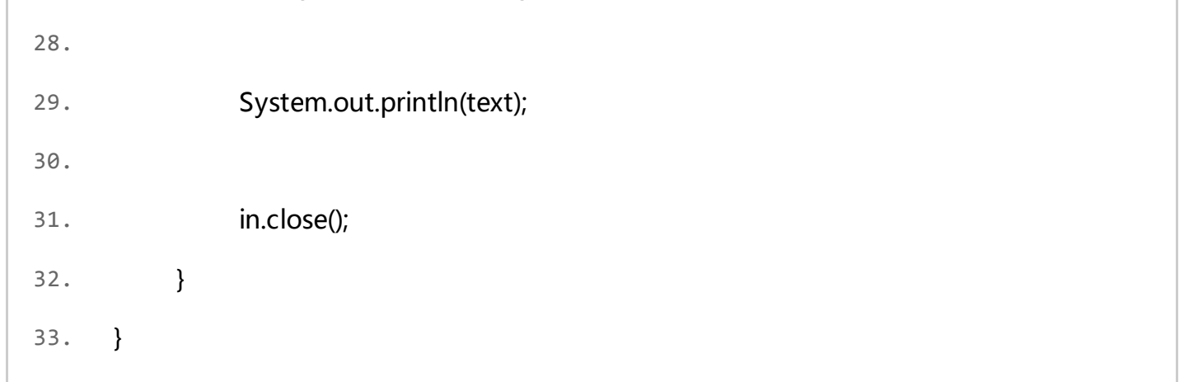
每个URL都是URI，但不一定每个URI都是URL。这是因为URI还包括一个子类，即统一资源名称（URN），它命名资源但不指定如何定位资源。

示例：

```java
import java.net.URL;
import java.net.MalformedURLException;
import java.io.InputStream;
import java.net.URLConnection;
import java.io.IOException;

public class URLDemo
{
        public static void main(String[] args) throws MalformedURLException,IOException {

                String str_url = "http://192.168.1.100:8080/myweb/1.html?name=lisi";

                URL url = new URL(str_url);

                System.out.println("getProtocol：" + url.getProtocol());
                System.out.println("getHost：" + url.getHost());
                System.out.println("getPort：" + url.getPort());
                System.out.println("getFile：" + url.getFile());
                System.out.println("getPath：" + url.getPath());
                System.out.println("getQuery：" + url.getQuery());

                InputStream in = url.openStream();//相当于
url.openConnection().getInputStream();

                byte[] buf = new byte[1024];
                int len = in.read(buf);

                String text = new String(buf,0,len);

                System.out.println(text);

                in.close();
        }
}
```

复制代码

运行结果：



```
D:\code\day26>javac URLDemo.java

D:\code\day26>java URLDemo
getProtocol: http
getHost: 192.168.1.100
getPort: 8080
getFile: /myweb/1.html?name=lisi
getPath: /myweb/1.html
getQuery: name=lisi
<html>
                        <head>
                                                <title>这是我的网页</title>
                        </head>
                        <body>
                                                <h1>欢迎光临</h1>
                                                <font size='5' color='red'>这是
一个tomcat服务器中的资源，是一个html网页。</html>
                        </body>
</html>
```

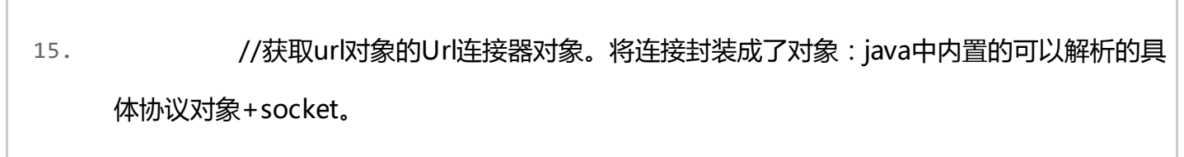
之所以运行结果中响应头不见了，只能看到主体数据的原因在于：URLConnection对象已经把响应头给解析了。

代码：

```java
import java.net.URL;
import java.net.MalformedURLException;
import java.io.InputStream;
import java.net.URLConnection;
import java.io.IOException;

public class URLDemo
{
        public static void main(String[] args) throws MalformedURLException,IOException {

                String str_url = "http://192.168.1.100:8080/myweb/1.html?name=lisi";

                URL url = new URL(str_url);

                //获取url对象的Url连接器对象。将连接封装成了对象：java中内置的可以解析的具体协议对象+socket。
                URLConnection conn = url.openConnection();

                System.out.println(conn);

                //由于URLConnection对象已经把响应头给解析了，所以，可以通过URLConnection对象获取响应头某属性名对应的属性值。
                String value = conn.getHeaderField("Content-Type");

                System.out.println(value);
        }
}
```

复制代码

运行结果：



```
D:\code\day26>javac URLDemo.java

D:\code\day26>java URLDemo
sun.net.www.protocol.http.HttpURLConnection:http://192.168.1.100:8080/myweb/1.ht
ml?name=lisi
text/html
```

常见网络结构

1. C/S client/server
   - 特点：
     - 该结构的软件，客户端和服务端都需要编写。
     - 开发成本较高，维护较为麻烦。
   - 好处：
     - 客户端在本地可以分担一部分任务。例如，杀毒软件直接对本机文件进行杀毒。

2. B/S browser/server
   - 特点：
     - 该结构的软件，只开发服务器端，不开发客户端，因为客户端直接由浏览器取代。
     - 开发成本相对低，维护更为简单。
   - 缺点：
     - 所有运算都要在服务端完成。

练习：TCP协议上传图片客户端和服务端

TCP服务端

```java
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.io.InputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

public class UploadPicServer
{
        public static void main(String[] args) throws
FileNotFoundException,IOException {

                //创建tcp的socket服务端。
                ServerSocket ss = new ServerSocket(10006);

                //获取客户端。
                Socket s = ss.accept();

                String ip = s.getInetAddress().getHostAddress();
                System.out.println(ip + "......connected");

                //读取客户端发来的数据。
                InputStream in = s.getInputStream();

                //将读取到的数据存储到一个文件中。
                File dir = new File("c:\\pic");
                if(!dir.exists()){
                        dir.mkdirs();
                }
                File file = new File(dir,ip + ".bmp");
                FileOutputStream fos = new FileOutputStream(file);

                byte[] buf = new byte[1024];

                int len = 0;

                while((len = in.read(buf)) != -1){
                        fos.write(buf,0,len);
                }

                //获取socket输出流，将上传成功字样发给客户端。
                OutputStream out = s.getOutputStream();

                out.write("上传成功".getBytes());

                fos.close();
                s.close();
                ss.close();
        }
}
```

复制代码

TCP客户端

```java
import java.net.Socket;
import java.io.FileInputStream;
import java.io.OutputStream;
import java.io.InputStream;
import java.net.UnknownHostException;
import java.io.IOException;
import java.io.FileNotFoundException;

public class UploadPicClient
{
        public static void main(String[] args) throws
UnknownHostException,IOException,FileNotFoundException{

                //1. 创建客户端socket。
                Socket s = new Socket("192.168.1.100",10006);

                //2. 读取客户端要上传的图片文件。
                FileInputStream fis = new FileInputStream("c:\\0.bmp");

                //3. 获取socket输出流，将读到图片数据发送给服务端。
                OutputStream out = s.getOutputStream();

                byte[] buf = new byte[1024];

                int len = 0;

                while((len = fis.read(buf)) != -1){
                        out.write(buf,0,len);
                }

                //告诉服务端说：这边的数据发送完毕。让服务端停止读取。
                s.shutdownOutput();

                //读取服务端发回的内容。
                InputStream in = s.getInputStream();
                byte[] bufIn = new byte[1024];

                int lenIn = in.read(buf);
                String text = new String(buf,0,lenIn);
                System.out.println(text);

                fis.close();
                s.close();
        }
}
```

复制代码

运行结果：

```
D:\code\day26>javac UploadPicServer.java

D:\code\day26>java UploadPicServer
192.168.1.100......connected
```

```
D:\code\day26>javac UploadPicClient.java

D:\code\day26>java UploadPicClient
上传成功
```

练习：TCP协议服务端多线程技术

代码：

UploadTask.java

```java
import java.net.Socket;
import java.io.InputStream;
import java.io.FileOutputStream;
import java.io.File;
import java.io.OutputStream;

public class UploadTask implements Runnable
{
        private Socket s;

        public UploadTask(Socket s){
                this.s = s;
        }

        public void run(){

                int count = 0;

                String ip = s.getInetAddress().getHostAddress();
                System.out.println(ip + "......connected");

                try{
                        InputStream in = s.getInputStream();

                        File dir = new File("c:\\pic");
                        if(!dir.exists()){
                                dir.mkdirs();
                        }
                        File file = new File(dir,ip + ".bmp");

                        //如果文件已经存在于服务端。
                        while(file.exists()){
                                file = new File(dir,ip + "(" + (++count) +
").bmp");
                        }

                        FileOutputStream fos = new FileOutputStream(file);

                        byte[] buf = new byte[1024];

                        int len = 0;

                        while((len = in.read(buf)) != -1){
                                fos.write(buf,0,len);
                        }

                        OutputStream out = s.getOutputStream();

                        out.write("上传成功".getBytes());

                        fos.close();
                        s.close();
                }catch(Exception e){
                        e.printStackTrace();
                }
        }
}
```

复制代码

UploadServerDemo.java

```java
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.io.InputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

public class UploadPicServer
{
        public static void main(String[] args) throws
FileNotFoundException,IOException {
```

```

                ServerSocket ss = new ServerSocket(10006);

                while(true){
                        Socket s = ss.accept();
                        new Thread(new UploadTask(s)).start();
                }

                //ss.close();
        }
}
```

复制代码

常见客户端和服务端

最常见的客户端：

浏览器：IE。

最常见的服务端：

服务器：Tomcat。

服务器处理客户端的请求，并且返回响应信息。

P.S.

关于Tomcat的内部机制等知识将会在JavaWeb课程中做为重点讲解，同学们在这里了解即可。

~END~

~爱上海，爱黑马~