

```
非贷款,0元入学,不1万就业不给1分钱学费,我们已干四年了!
       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。
```

为了了解其原理:

最常见的服务端:服务器, Tomcat。

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:

11.

01. import java.net.ServerSocket; 02. import java.net.Socket; 03. import java.io.InputStream; import java.io.PrintWriter;

05. import java.io.IOException;

06. public class MyTomcat 07.

08. 09. public static void main(String[] args) throws IOException { 10.

13. Socket s = ss.accept(); System.out.println(s.getInetAddress().getHostAddress() + 14. ".....connected"); 15. InputStream in = s.getInputStream(); 17. byte[] buf = new byte[1024]; 18. 19 int len = in.read(buf); 20. 21. 22. String text = new String(buf,0,len); 23. System.out.println(text); 25. //给客户端一个反馈信息。 26. 27.

ServerSocket ss = new ServerSocket(9090);

PrintWriter out = new PrintWriter(s.getOutputStream(),true); 28. out.println("欢迎光临"); 29. 30. 31. s.close(); 32. ss.close(); 33. } 复制代码 运行结果: localhost:9090 localhost:9090

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

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: 01. import java.net.Socket;

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

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

out.println("GET /myweb/1.html HTTP/1.1");

out.println("Host: 192.168.1.100:8080");

out.println("Connection: close");

out.println("Accept: */*");

//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);

<title>这是我的网页</title>

Gecko) Chrome/43.0.2357.81 Safari/537.36

import java.io.PrintWriter; import java.io.InputStream;

import java.io.IOException;

public class MyBrowser

D:\code\day26>java MyBrowser

Server: Apache-Coyote/1.1 Accept-Ranges: bytes

Content-Type: text/html Content-Length: 211

Connection: close

(html)

源"的指针。

示例:

21.

22.

23.

24.

25.

26.

27.

28.

29.

30.

31.

32.

33.

}

运行结果:

(URN),它命名资源但不指定如何定位资源。

ETag: W/"211-1433908112666"

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

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

<head>

HTTP/1.1 200 OK

02.

03. 04.

05. 06.

07.

08. 09.

10. 11.

12.

13.

14. 15.

16. 17.

18. out.println(); out.println(); 19. 20. 21. InputStream in = s.getInputStream(); 22. 23. byte[] buf = new byte[1024]; int len = in.read(buf); 24. 25. String str = new String(buf,0,len); 26. System.out.println(str); 27. 28. 29. s.close(); 30. } 31. } 复制代码 运行结果:

</head> <body> <hi>欢迎光临</hi>
这是 个tomcat服务器中的资源,是一个htm1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

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

import java.net.URL; 01. import java.net.MalformedURLException; 02. import java.io.InputStream; 03. import java.net.URLConnection; 04. import java.io.IOException; 05. 06. public class URLDemo 08. public static void main(String[] args) throws MalformedURLException, IOException { 09. 10. String str_url = "http://192.168.1.100:8080/myweb/1.html?name=lisi"; 11. 12. URL url = new URL(str_url); 13. 14. System.out.println("getProtocol: " + url.getProtocol()); 15. System.out.println("getHost: " + url.getHost()); 16. System.out.println("getPort: " + url.getPort()); 17. System.out.println("getFile: " + url.getFile()); 18. System.out.println("getPath: " + url.getPath()); 19. System.out.println("getQuery: " + url.getQuery()); 20.

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

url.openConnection().getInputStream();

int len = in.read(buf);

System.out.println(text);

<head>

</head>

(body)

<title>这是我的网页</title>

in.close();

D:\code\day26>javac URLDemo.java

getFile: /myweb/1.html?name=lisi

D:∖code∖day26>javac URLDemo.java

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。

开发成本相对低,维护更为简单。

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

import java.io.IOException; 02. import java.net.ServerSocket;

import java.io.InputStream;

06. import java.io.FileOutputStream; import java.io.OutputStream;

import java.io.IOException;

public class UploadPicServer

import java.io.FileNotFoundException;

FileNotFoundException,IOException {

public static void main(String[] args) throws

//创建tcp的socket服务端。

03. import java.net.Socket;

import java.io.File;

D:∖code∖day26>java URLDemo

nl?name=lisi ext/html

常见网络结构

特点:

好处:

特点:

缺点:

TCP服务端

01.

05.

07.

08.

09. 10. 11.

12.

13.

14.

15.

14. 15.

16.

17. 18.

19. 20.

21.

22. 23. 24.

25. 26.

27.

28.

29.

30.

31. 32.

33. 34.

35. 36.

37.

38.

39. 40.

41. 42.

43. 44.

运行结果:

文件(F) 组织 ▼

01.

03.

04.

05. 06.

07. 08. 09.

11.

12. 13. 14.

16.

17. 18.

19.

20. 21. 22.

24.

25.

26.

27.

UploadServerDemo.java

import java.io.IOException; import java.net.ServerSocket;

import java.io.FileOutputStream;

import java.io.FileNotFoundException;

FileNotFoundException,IOException {

public static void main(String[] args) throws

import java.io.OutputStream;

import java.io.IOException;

public class UploadPicServer

import java.net.Socket; import java.io.InputStream;

import java.io.File;

01.

03.

04. 05.

06.

07. 08.

09. 10.

11. 12. 13. try{

☆ 收藏夹

复制代码

1. C/S client/server

2. B/S browser/server

D:\code\day26>java URLDemo

getPath: /myweb/1.html getQuery: name=lisi

}

复制代码

getProtocol: http getHost: 192.168.1.100

getPort: 8080

(html>

byte[] buf = new byte[1024];

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

<hi>欢迎光临</hi>
这是 个tomcat服务器中的资源,是一个htm1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: 01. import java.net.URL; 02. import java.net.MalformedURLException; 03. import java.io.InputStream; import java.net.URLConnection; 04. import java.io.IOException; 05. 06. public class URLDemo 07. 08. 09. public static void main(String[] args) throws MalformedURLException,IOException { 10. 11. String str_url = "http://192.168.1.100:8080/myweb/1.html?name=lisi"; 12. URL url = new URL(str_url); 13. 14. 15. //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。 16. URLConnection conn = url.openConnection(); 17. 18. System.out.println(conn); 19. //由于URLConnection对象已经把响应头给解析了,所以,可以通过 20. URLConnection对象获取响应头某属性名对应的属性值。 21. String value = conn.getHeaderField("Content-Type"); 22. System.out.println(value); 23. 24. } 25. } 复制代码 运行结果:

sun.net.www.protocol.http.HttpURLConnection:http://192.168.1.100:8080/myweb/1.ht

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。

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。

ServerSocket ss = new ServerSocket(10006); 16. 17. //获取客户端。 18. 19. Socket s = ss.accept(); 20. 21. String ip = s.getInetAddress().getHostAddress(); System.out.println(ip + ".....connected"); 22. 23. //读取客户端发来的数据。 24. InputStream in = s.getInputStream(); 26. //将读取到的数据存储到一个文件中。 27. File dir = new File("c:\\pic"); 28. 29. if(!dir.exists()){ 30. dir.mkdirs(); 31. } 32. File file = new File(dir,ip + ".bmp"); 33. FileOutputStream fos = new FileOutputStream(file); 34. 35. byte[] buf = new byte[1024]; 36. int len = 0; 37. 38. while((len = in.read(buf)) != -1){ 40. fos.write(buf,0,len); 41. } 42. //获取socket输出流,将上传成功字样发给客户端。 43. OutputStream out = s.getOutputStream(); 44. 45. out.write("上传成功".getBytes()); 47. fos.close(); 48. 49. s.close(); ss.close(); 50. 51. } 52. 复制代码 TCP客户端 import java.net.Socket; 01. 02. import java.io.FileInputStream; 03. import java.io.OutputStream; import java.io.InputStream; 04. import java.net.UnknownHostException; 05. 06. import java.io.IOException; 07. import java.io.FileNotFoundException; 08. 09. public class UploadPicClient 10. public static void main(String[] args) throws 11. UnknownHostException,IOException,FileNotFoundException{ 12. //1. 创建客户端socket。 13.

Socket s = new Socket("192.168.1.100",10006);

//3. 获取socket输出流,将读到图片数据发送给服务端。

OutputStream out = s.getOutputStream();

while((len = fis.read(buf)) != -1){

out.write(buf,0,len);

InputStream in = s.getInputStream();

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

byte[] bufIn = new byte[1024];

int lenIn = in.read(buf);

System.out.println(text);

D:\code\day26>javac UploadPicServer.java

D:\code\day26>javac UploadPicClient.java

放映幻灯片

新建文件夹

44

D:\code\day26>java UploadPicServer

D:\code\day26>java UploadPicClient

192.168.1.100.....connected

▶ 计算机 ▶ 本地磁盘 (C:) ▶ pic

工具(T) 帮助(H)

共享 ▼

上传成功

编辑(E) 查看(V)

包含到库中 ▼

fis.close();

s.close();

//告诉服务端说:这边的数据发送完毕。让服务端停止读取。

FileInputStream fis = new FileInputStream("c:\\0.bmp");

//2. 读取客户端要上传的图片文件。

byte[] buf = new byte[1024];

int len = 0;

s.shutdownOutput();

//读取服务端发回的内容。

}

E 三 桌面 最近访问的位置 192.168.1.100.b mp 练习:TCP协议服务端多线程技术 代码: UploadTask.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");

InputStream in = s.getInputStream();

File dir = new File("c:\\pic");

dir.mkdirs();

if(!dir.exists()){

28. File file = new File(dir,ip + ".bmp"); 29. 30. //如果文件已经存在于服务端。 while(file.exists()){ 32. file = new File(dir,ip + "(" + (++count) + 33. ").bmp"); 34. } 35. 36. FileOutputStream fos = new FileOutputStream(file); 37. byte[] buf = new byte[1024]; 38. 39. int len = 0; 41. while((len = in.read(buf)) != -1){ 42. fos.write(buf,0,len); 43. 44. } 45. OutputStream out = s.getOutputStream(); 46. 47. out.write("上传成功".getBytes()); 49. fos.close(); 50. 51. s.close(); }catch(Exception e){ 52. e.printStackTrace(); 53. 54. } 55. } 56. } 复制代码

14. 15. ServerSocket ss = new ServerSocket(10006); 17. while(true){ Socket s = ss.accept(); 18. 19. new Thread(new UploadTask(s)).start(); 20. } 21. 22. //ss.close(); 23. 24. } 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~ to Nie Of DE TAX DAT TAX DAE TAX DAE TAX DE TAX DE TAX DE TAX DAE TAX DAE ~爱上海,爱黑马~