java实现一个简单的Web服务器 - 博客频道 - CSDN

Web服务器也称为超文本传输协议服务器，使用http与其客户端进行通信，基于java的web服务器会使用两个重要的类，

java.net.Socket类和java.net.ServerSocket类，并基于发送http消息进行通信。

这个简单的Web服务器会有以下三个类：

*HttpServer

*Request

*Response

应用程序的入口在HttpServer类中，main()方法创建一个HttpServer实例，然后调用其await()方法，顾名思义，await()方法会在指定端口

上等待HTTP请求，对其进行处理，然后发送响应信息回客户端，在接收到关闭命令前，它会保持等待状态。

该应用程序仅发送位于指定目录的静态资源的请求，如html文件和图像，它也可以将传入到的http请求字节流显示到控制台，但是，它并不发送

任何头信息到浏览器，如日期或者cookies等。

下面为这几个类的源码：

Request：

[java]view plaincopy

```
1. package cn.com.server;
2. import java.io.InputStream;
3. publicclass Request {
4. private InputStream input;
5. private String uri;
6. public Request(InputStream input){
7. this.input=input;
8.  }
9. publicvoid parse(){
10. //Read a set of characters from the socket
11.     StringBuffer request=new StringBuffer(2048);
12. int i;
13. byte[] buffer=newbyte[2048];
14. try {
15.       i=input.read(buffer);
16.     } catch (Exception e) {
17.       e.printStackTrace();
18.       i=-1;
19.     }
20. for(int j=0;j
21.       request.append((char)buffer[j]);
22.     }
23.     System.out.print(request.toString());
24.     uri=parseUri(request.toString());
25.  }
26. public String parseUri(String requestString){
```

```
27. int index1,index2;
28.         index1=requestString.indexOf(" ");
29. if(index1!=-1){
30.             index2=requestString.indexOf(" ",index1+1);
31. if(index2>index1){
32. return requestString.substring(index1+1,index2);
33.             }
34.         }
35. returnnull;
36.     }
37. public String getUri(){
38. returnthis.uri;
39.     }
40. }
```

Request类表示一个HTTP请求，可以传递InputStream对象来创建Request对象，可以调用InputStream对象中的read()方法来读取HTTP请求
的原始数据。
上述源码中的parse()方法用于解析Http请求的原始数据，parse()方法会调用私有方法parseUrI()来解析HTTP请求的URI,除此之外，并没有
做太多的工作，parseUri()方法将URI存储在变量uri中，调用公共方法getUri()会返回请求的uri。

Response:

[java]view plaincopy

```
1. "font-size:10px;">package cn.com.server;
2. import java.io.File;
3. import java.io.FileInputStream;
4. import java.io.IOException;
5. import java.io.OutputStream;
6. /**
7.  * HTTP Response = Status-Line
8.  *      *(( general-header | response-header | entity-header ) CRLF)
9.  *      CRLF
10.  *      [message-body]
11.  *      Status-Line=Http-Version SP Status-Code SP Reason-Phrase CRLF
12.  *
13.  */
14. publicclass Response {
15. privatestaticfinalint BUFFER_SIZE=1024;
16.     Request request;
17.     OutputStream output;
18. public Response(OutputStream output){
19. this.output=output;
20.     }
21. publicvoid setRequest(Request request){
22. this.request=request;
23.     }
```

```
24. publicvoid sendStaticResource()throws IOException{
25. byte[] bytes=newbyte[BUFFER_SIZE];
26.         FileInputStream fis=null;
27. try {
28.             File file=new File(HttpServer.WEB_ROOT,request.getUri());
29. if(file.exists()){
30.                 fis=new FileInputStream(file);
31. int ch=fis.read(bytes,0,BUFFER_SIZE);
32. while(ch!=-1){
33.                     output.write(bytes, 0, BUFFER_SIZE);
34.                     ch=fis.read(bytes, 0, BUFFER_SIZE);
35.                 }
36.             }else{
37. //file not found
38.                 String errorMessage="HTTP/1.1 404 File Not Found\r\n"+
39. "Content-Type:text/html\r\n"+
40. "Content-Length:23\r\n"+
41. "\r\n"+
42. "
```

File Not Found

″;

```
43.                 output.write(errorMessage.getBytes());
44.             }
45.         } catch (Exception e) {
46.             System.out.println(e.toString());
47.         }finally{
48. if(fis!=null){
49.                 fis.close();
50.             }
51.         }
52.     }
53. }"font-size:24px;">
54.
```

Response对象在HttpServer类的await()方法中通过传入套接字中获取的OutputStream来创建。
Response类有两个公共方法：setRequest()和sendStaticResource()，setRequest()方法会接收一个Request对象为参数，sendStaticResource()
方法用于发送一个静态资源到浏览器，如Html文件。

HttpServer：

[java]view plaincopy

```
1. package cn.com.server;
2. import java.io.File;
3. import java.io.InputStream;
4. import java.io.OutputStream;
```

```
5. import java.net.InetAddress;
6. import java.net.ServerSocket;
7. import java.net.Socket;
8. publicclass HttpServer {
9. /**
10.     * WEB_ROOT is the directory where our html and other files reside.
11.     * For this package,WEB_ROOT is the "webroot" directory under the
12.     * working directory.
13.     * the working directory is the location in the file system
14.     * from where the java command was invoke.
15.     */
16. publicstaticfinal String WEB_ROOT=System.getProperty("user.dir")+File.separator+"webroot";
17. privatestaticfinal String SHUTDOWN_COMMAND="/SHUTDOWN";
18. privateboolean shutdown=false;
19. publicstaticvoid main(String[] args) {
20.       HttpServer server=new HttpServer();
21.       server.await();
22.    }
23. publicvoid await(){
24.       ServerSocket serverSocket=null;
25. int port=8080;
26. try {
27.          serverSocket=new ServerSocket(port,1,InetAddress.getByName("127.0.0.1"));
28.       } catch (Exception e) {
29.          e.printStackTrace();
30.          System.exit(0);
31.       }
32. while(!shutdown){
33.          Socket socket=null;
34.          InputStream input=null;
35.          OutputStream output=null;
36. try {
37.             socket=serverSocket.accept();
38.             input=socket.getInputStream();
39.             output=socket.getOutputStream();
40. //create Request object and parse
41.             Request request=new Request(input);
42.             request.parse();
43. //create Response object
44.             Response response=new Response(output);
45.             response.setRequest(request);
46.             response.sendStaticResource();
47.          } catch (Exception e) {
48.             e.printStackTrace();
49. continue;
50.          }
51.       }
```

```
52.    }
53. }
```

这个类表示一个Web服务器，这个Web服务器可以处理对指定目录的静态资源的请求，该目录包括由公有静态变量final WEB_ROOT指明的目录及其所有子目录。

现在在webroot中创建一个html页面，命名为index.html,源码如下：

[html]view plaincopy

```
1. >
2. <html>
3. <head>
4. <metacharset="UTF-8">
5. <title>Insert title heretitle>
6. head>
7. <body>
8. <h1>Hello World!h1>
9. body>
10. html>
```

现在启动该WEB服务器，并请求index.html静态页面。

所对应的控制台的输出：

```
HttpServer [Java Application] C:\Program Files\Java\jre1.8.0_40\bin\javaw.exe (2015年4月1日 上午8:10:53)
GET /index.html HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.1500.72 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: zh-CN,zh;q=0.8
Cookie: CNZZDATA155540=cnzz_eid%3D750120078-1426392136-http%253A%252F%252Flocalhost%253A8080%252F%26ntime%3D1426507017
```


如此，一个简单的http服务器便完成了。