**计算机网络大作业**

**——软件运行说明文档**

|  |  |
| --- | --- |
| **题目：** | **基于http协议的系统工程多线程web服务器** |

|  |  |
| --- | --- |
| **学生姓名：** | **杨巧 王丁可 蒋佳仪** |
| **学 号：** | **2020141490177 2020141490118**  **2020141490294** |
| **任课教师：** | **宋义忠** |
| **所属班级：** | **计算机网络06班** |
| **学科专业：** | **化学工程学院** |

2022年 6月12日

[一、设计思路及步骤 ————————————————————————————](file:///D:\各科作业\计算机网络与分布式系统\作业\多线程web服务器end.doc#_Toc178570037)3

[1.1. 主线程设计 —————————————————————————————](file:///D:\各科作业\计算机网络与分布式系统\作业\多线程web服务器end.doc#_Toc178570038)3

[1.2.连接处理分线程设计 —————————————————————————](file:///D:\各科作业\计算机网络与分布式系统\作业\多线程web服务器end.doc#_Toc178570042)3

[1.3.处理信息的接收和发送 ————————————————————————](file:///D:\各科作业\计算机网络与分布式系统\作业\多线程web服务器end.doc#_Toc178570042)4

二. 调试过程及实验结果 ——————————————————————————6

2.1. 显示Web页面 ————————————————————————————6

2.2. 运行实例 ————————————————————————————— 7

三. 总结 —————————————————————————————————8

四. 源程序 —————————————————————————————————8

五. 参考文献 ————————————————————————————————11

**一、设计思路及步骤**

**1.1主线程设计**

主线程的设计就是在主线程WebServer 类中实现了服务器端口的侦听，服务器接受一个客户端请求之后创建一个线程实例处理请求，代码如下：

//创建监听端口等待TCP连接请求

public final class WebServer {

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

//创建web服务器的监听端口（套接字），监听6789端口

try (ServerSocket socket = new ServerSocket(6789)) {

while(true) {

//侦听TCP请求

Socket s=socket.accept();

//创建http的request对象，将已建立TCP连接的Socket作为参数传递到构造函数中

HttpRequest http=new HttpRequest(s);

//创建一个Thread对象，将请求对象作为参数传递给Thread的构造函数

Thread thread=new Thread(http);

//启动线程

thread.start();

}

}

}

}

**1.2连接处理分线程设计**：

新建Runnable类实现分线程处理

新建public class SecondThread implements Runnable {

private String name;

public SecondThread(String name) {

this.name=name;

System.out.println(name+"创建成功");

}

@Override

public void run() {

for(int i=0;i<3;i++) {

System.out.println(name+"第"+i+"次运行");

Thread.yield();

}

}

public static void main(String[] args) {

SecondThread r1=new SecondThread("第一个线程");

SecondThread r2=new SecondThread("第二个线程");

Thread t1=new Thread(r1);

Thread t2=new Thread(r2);

System.out.println("开始启动t1.t2进程");

t1.start();

t2.start();

System.out.println("main方法运行完毕");

}

}

在分线程HttpRequest类中实现了HTTP 协议的处理，这个类实现了Runnable 接口，代码如下

final class HttpRequest implements Runnable {

final static String CRLF="\r\n"; //“回车与换行”，用于在http响应报文中

Socket socket;

public HttpRequest(Socket socket)throws Exception {

this.socket=socket;

}

public void run() {

try {

requestprocess();

}catch(Exception e) {

System.out.println(e.getMessage());

}

}

**1.3构建processRequest()方法来处理信息的接收和发送**

作为实现Runnable 接口的主要内容，在run()方法中调用processRequest()方法来处理客户请求内容的接收和服务器返回信息的发送，代码如下：

public void requestprocess() throws Exception{

BufferedReader br=new BufferedReader(new InputStreamReader(socket.getInputStream()));//将从套接字得到的字节输入流，利用转换流，转换成字符流 ，便于之后更好的读取

DataOutputStream dos=new DataOutputStream(socket.getOutputStream());

String requestLine=br.readLine(); //得到请求行

System.out.println(requestLine);

String headerLine;

while((headerLine=br.readLine()).length()!=0) {

System.out.println(headerLine);

}

StringTokenizer stokizer=new StringTokenizer(requestLine); //该类用于分解字符串，默认分隔符有：空格，回车，换行等

stokizer.nextToken();

String fileName="D:\\大二下半学期\\计算机网络\\多线程Web服务器"+stokizer.nextToken(); //在后面寻找文件中，将会在D盘的根目录进行寻找

FileInputStream fis=null;

boolean fileIsExisted=true;

try {

fis=new FileInputStream(fileName); //利用这个类相当于进行了寻找，若没有找到，则会捕获异常。

}catch(FileNotFoundException e1) {

fileIsExisted=false;

}

String statusLine = null; //状态行

String contentTypeLine = null; //Content-type行

Date date=new Date();

String dates="Date: "+date.toString()+CRLF; //Date行

String entityBody = null; //实体部分

if(fileIsExisted) {

statusLine="HTTP/1.1 200 OK"+CRLF;

if(fileName.endsWith(".html") || fileName.endsWith(".htm") || fileName.endsWith(".txt")) {

contentTypeLine="Content-type: text/html;charset=utf-8"+CRLF;

}

else if(fileName.endsWith(".jpg")) {

contentTypeLine="Content-type: image/jpg;charset=utf-8"+CRLF;

}

else if(fileName.endsWith(".gif")) {

contentTypeLine="Content-type: image/gif;charset=utf-8"+CRLF;

}

else {

contentTypeLine="Content-type: Unknown;charset=utf-8"+CRLF;

}

dos.writeBytes(statusLine);

dos.writeBytes(contentTypeLine);

dos.writeBytes(dates);

dos.writeBytes(CRLF);

while(fis.available()>0) {

byte by[]=new byte[1024];

fis.read(by);

dos.write(by);

}//将我本地的文件内容写入到套接字的输出流中，也就是将我的文件传给浏览器

}else {

statusLine="HTTP/1.1 404 NotFound"+CRLF;

contentTypeLine = "Content-type: text/html;charset=utf-8"+CRLF;

entityBody ="<html><title>Not found</title><h1>404 NotFound</h1></html>";

dos.writeBytes(statusLine);

dos.writeBytes(contentTypeLine);

dos.writeBytes(dates);

dos.writeBytes(CRLF);

dos.writeBytes(entityBody);

}

fis.close();

dos.close();

br.close();

socket.close();

}

}

**二、调试过程及实验结果**

**2.1显示Web页面**

**<html>**

**<head>**

**<meta charset = "UTF-8">**

**</head>**

**<body>**

**<p>欢迎访问</p>**

**<img src="1.jpg" width="200" />**

**</body>**

**</html>**

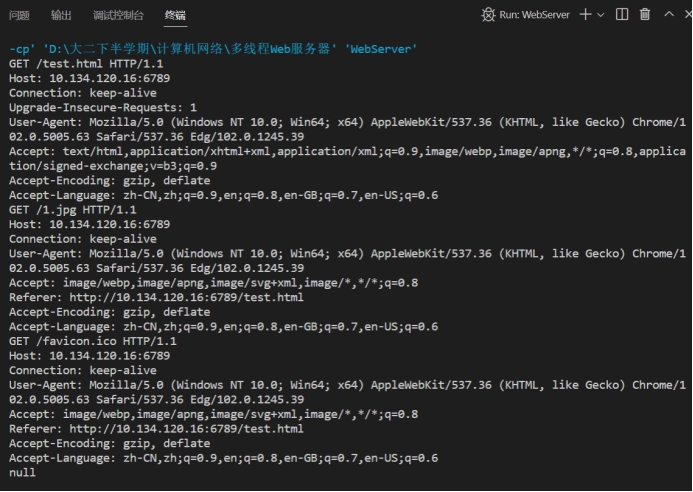
**2.2运行实例**

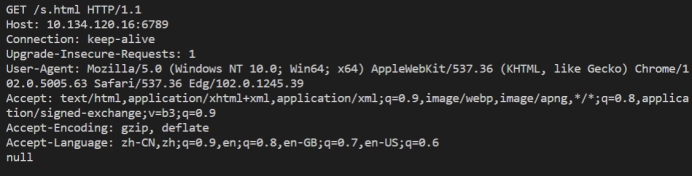
将编译后的webserver.class、requestHandler.class和index.html文件至于网络主机的同一目录中.

首先运行服务器程序,服务器运行后显示端口信息”httpServer runing on port 6789”,

然后在浏览器地址栏中输入http://localhost:6789/index.html，就可以正确的显示网页。







**三、总结**

Java中实现线程的方式有两种，一是生成Thread类的子类，并定义该子类自己的run方法，线程的操作在方法run中实现。但我们定义的类一般是其他类的子类，而Java又不允许多重继承，因此我们实现线程的方法是实现Runnable接口。通过覆盖Runnable接口中的run方法实现该线程的功能。

**四、源程序**

import java.net.ServerSocket;

import java.net.Socket;

import java.io.BufferedReader;

import java.io.DataOutputStream;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.InputStreamReader;

import java.util.Date;

import java.util.StringTokenizer;

//创建监听端口等待TCP连接请求

public final class WebServer {

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

//创建web服务器的监听端口（套接字），监听6789端口

try (ServerSocket socket = new ServerSocket(6789)) {

while(true) {

//侦听TCP请求

Socket s=socket.accept();

//创建http的request对象，将已建立TCP连接的Socket作为参数传递到构造函数中

HttpRequest http=new HttpRequest(s);

//创建一个Thread对象，将请求对象作为参数传递给Thread的构造函数

Thread thread=new Thread(http);

//启动线程

thread.start();

}

}

}

}

final class HttpRequest implements Runnable {

final static String CRLF="\r\n"; //“回车与换行”，用于在http响应报文中

Socket socket;

public HttpRequest(Socket socket)throws Exception {

this.socket=socket;

}

public void run() {

try {

requestprocess();

}catch(Exception e) {

System.out.println(e.getMessage());

}

}

public void requestprocess() throws Exception{

BufferedReader br=new BufferedReader(new InputStreamReader(socket.getInputStream()));//将从套接字得到的字节输入流，利用转换流，转换成字符流 ，便于之后更好的读取

DataOutputStream dos=new DataOutputStream(socket.getOutputStream());

String requestLine=br.readLine(); //得到请求行

System.out.println(requestLine);

String headerLine;

while((headerLine=br.readLine()).length()!=0) {

System.out.println(headerLine);

}

StringTokenizer stokizer=new StringTokenizer(requestLine); //该类用于分解字符串，默认分隔符有：空格，回车，换行等

stokizer.nextToken();

String fileName="D:\\大二下半学期\\计算机网络\\多线程Web服务器"+stokizer.nextToken(); //在后面寻找文件中，将会在D盘的根目录进行寻找

FileInputStream fis=null;

boolean fileIsExisted=true;

try {

fis=new FileInputStream(fileName); //利用这个类相当于进行了寻找，若没有找到，则会捕获异常。

}catch(FileNotFoundException e1) {

fileIsExisted=false;

}

String statusLine = null; //状态行

String contentTypeLine = null; //Content-type行

Date date=new Date();

String dates="Date: "+date.toString()+CRLF; //Date行

String entityBody = null; //实体部分

if(fileIsExisted) {

statusLine="HTTP/1.1 200 OK"+CRLF;

if(fileName.endsWith(".html") || fileName.endsWith(".htm") || fileName.endsWith(".txt")) {

contentTypeLine="Content-type: text/html;charset=utf-8"+CRLF;

}

else if(fileName.endsWith(".jpg")) {

contentTypeLine="Content-type: image/jpg;charset=utf-8"+CRLF;

}

else if(fileName.endsWith(".gif")) {

contentTypeLine="Content-type: image/gif;charset=utf-8"+CRLF;

}

else {

contentTypeLine="Content-type: Unknown;charset=utf-8"+CRLF;

}

dos.writeBytes(statusLine);

dos.writeBytes(contentTypeLine);

dos.writeBytes(dates);

dos.writeBytes(CRLF);

while(fis.available()>0) {

byte by[]=new byte[1024];

fis.read(by);

dos.write(by);

}//将我本地的文件内容写入到套接字的输出流中，也就是将我的文件传给浏览器

}else {

statusLine="HTTP/1.1 404 NotFound"+CRLF;

contentTypeLine = "Content-type: text/html;charset=utf-8"+CRLF;

entityBody ="<html><title>Not found</title><h1>404 NotFound</h1></html>";

dos.writeBytes(statusLine);

dos.writeBytes(contentTypeLine);

dos.writeBytes(dates);

dos.writeBytes(CRLF);

dos.writeBytes(entityBody);

}

fis.close();

dos.close();

br.close();

socket.close();

}

}

**五、参考文献**

《计算机网络：自顶向下方法（第七版）》机械工业出版社

[(54条消息) 多线程Web服务器的搭建与实现\_沉蕴陌鹭的博客-CSDN博客\_多线程web服务器](https://blog.csdn.net/weixin_46188069/article/details/116615469?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522165561985916780357255899%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=165561985916780357255899&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-116615469-null-null.142%5ev17%5epc_search_result_control_group,157%5ev15%5enew_3&utm_term=%E5%A4%9A%E7%BA%BF%E7%A8%8Bweb%E6%9C%8D%E5%8A%A1%E5%99%A8&spm=1018.2226.3001.4187)

[(55条消息) java多线程web服务器\_chongshangyunxiao321的博客-CSDN博客](https://blog.csdn.net/chongshangyunxiao321/article/details/51095149?ops_request_misc=&request_id=&biz_id=102&utm_term=%E5%A4%9A%E7%BA%BF%E7%A8%8Bweb%E6%9C%8D%E5%8A%A1%E5%99%A8&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-4-51095149.nonecase&spm=1018.2226.3001.4187)

[(55条消息) socket编程 java多线程web服务器设计与实现\_Void&Exists的博客-CSDN博客](https://blog.csdn.net/a1004070060/article/details/90112364?ops_request_misc=&request_id=&biz_id=102&utm_term=%E5%A4%9A%E7%BA%BF%E7%A8%8Bweb%E6%9C%8D%E5%8A%A1%E5%99%A8&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-9-90112364.nonecase&spm=1018.2226.3001.4187)