

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | B/S体系软件设计 |
| 姓 名： | 杨樾 人 |
| 学 院： | 计算机科学与技术学院 |
| 系： |  |
| 专 业： | 软件工程专业 |
| 学 号： | 3160104875 |
| 指导教师： | 胡晓军 |

2019年 5 月 9 日

Table of Contents

[一、引言 3](#_Toc8338067)

[1.1 设计目的 3](#_Toc8338068)

[1.2 设计说明 3](#_Toc8338069)

[二、总体设计 4](#_Toc8338070)

[2.1 功能模块设计 4](#_Toc8338071)

[三、详细设计 5](#_Toc8338072)

[3.1 多线程设计 5](#_Toc8338073)

[3.2request设计 8](#_Toc8338074)

[3.3 response设计 10](#_Toc8338075)

[四、测试与运行 14](#_Toc8338076)

[4.1 程序测试 14](#_Toc8338077)

[4.2 程序运行 14](#_Toc8338078)

[五、心得感想 16](#_Toc8338079)

# 一、引言

本次作业是使用Java的开发一个web服务器。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。同时显示web文件夹的内容，处理post请求，调用php-cgi进行处理。

## 1.1 设计目的

了解HTTP协议，实现一个简单的Web服务器，要求如下。

1、用C/C++、C#、Java这三类编程语言中任一种开发

2、只能基于Socket功能，不能使用现成的HTTP解析类

3、能够让浏览器正确显示web.zip这个文件中的网页

## 1.2 设计说明

本程序采用 Java 程序设计语言，在 Intellij Idea平台下编辑、编译与调试。具体程序由我个人开发而成。使用了java.net.Socket类和java.net.ServerSocke和Process类，工作时间轴如表 1 所示:

**表1 工作时间轴**

|  |  |
| --- | --- |
| 时间 | 完成的工作 |
| 5月1日 | 学习HTTP协议 |
| 5月3日 | request |
| 5月5日 | response |
| 5月7日 | 处理post |
| 5月9日 | 文档撰写 |

# 二、总体设计

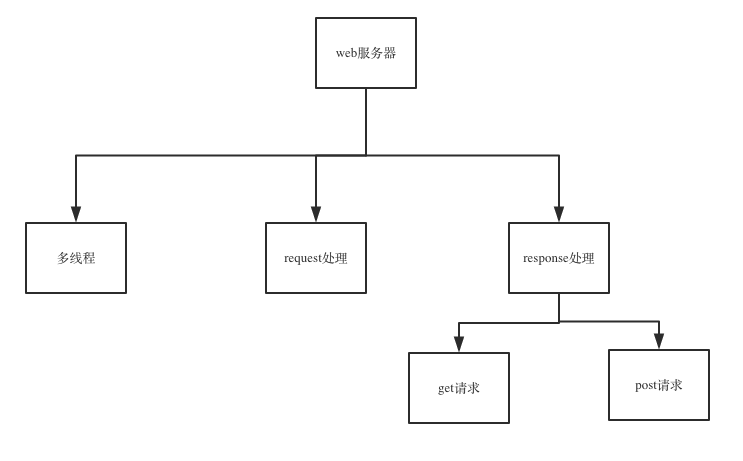
## 2.1 功能模块设计

本程序主要完成了以下功能：

1. GET请求

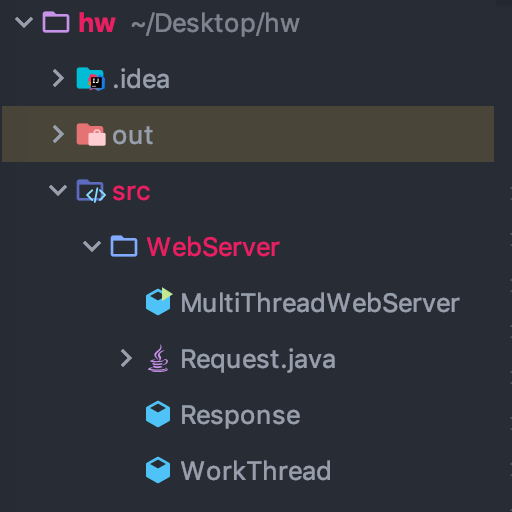
2. POST请求

3. 多线程



# 三、详细设计

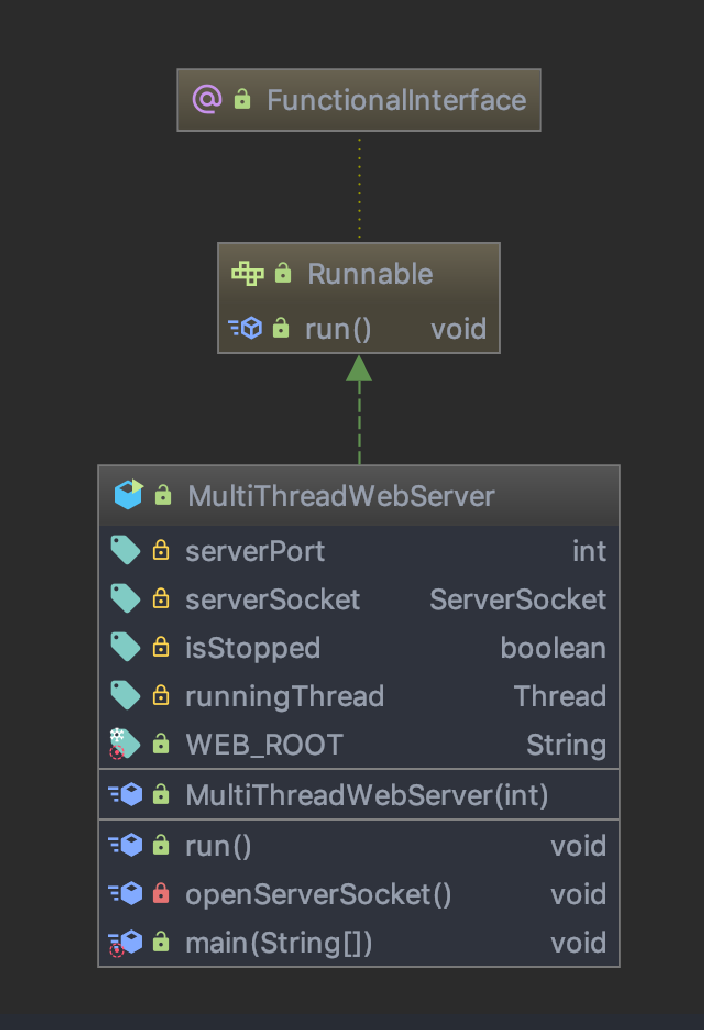
在设计过程中，主要分为三部分，多线程处理，request处理，response处理。以下将逐一进行说明。



## 3.1 多线程设计

。

登陆功能中的类如下：



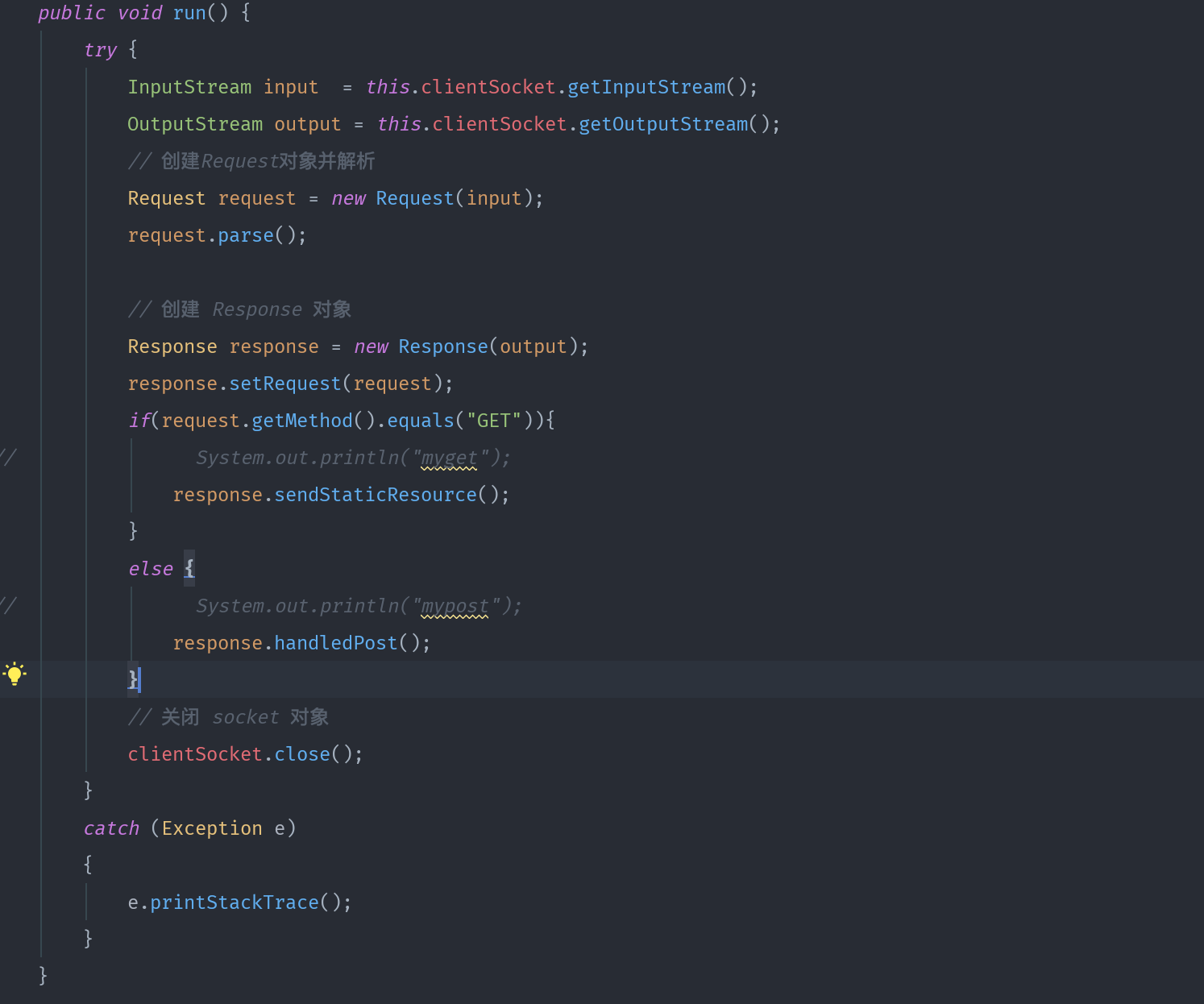
serverPost是服务器端口，默认8080。serverSocket是监听socket连接的变量。run()方法实现了多线程。

函数如下：



当客户端连接服务器成功后，会新开一个线程进行处理。

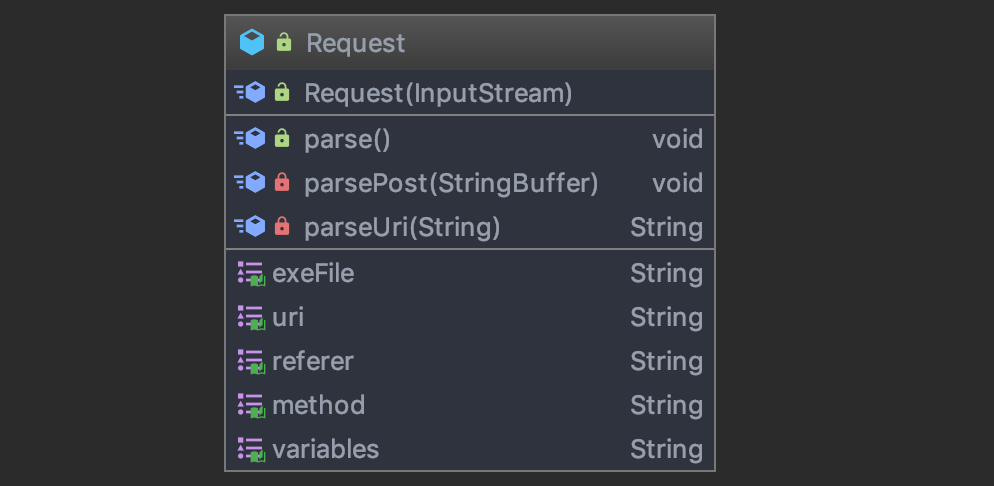
以下是workthread中对该客户端的消息的处理：



首先通过request类中的方法，对HTTP报文中的内容进行解析，判定其请求方法。

## 3.2request设计

request设计如下：



主要变量如下：

*public* InputStream input;  
*private* String uri;  
*private* String method;  
*public* PostMethodInfo postInfo;

其中postInfo是为了存储post请求中的参数。

parse方法中，会对HTTP报文首行进行解析，判断其是GET请求还是POST请求。

*public void* parse() {  
  
 StringBuffer request = *new* StringBuffer(2048);  
 *int* i;  
 *byte*[] buffer = *new byte*[2048];  
 *try* {  
 i = input.read(buffer);  
 } *catch* (IOException e) {  
 e.printStackTrace();  
 i = -1;  
 }  
 *for* (*int* j = 0; j < i; j++) {  
 request.append((*char*) buffer[j]);  
 }  
 *if* (request.substring(0,3).equals("GET")){  
 *this*.method = "GET";  
 }  
 *else* {  
 *this*.method = "POST";  
 parsePost(request);  
*// System.out.println(request);* }  
 uri = parseUri(request.toString());  
  
 }

parseUri方法是解析要处理的文件

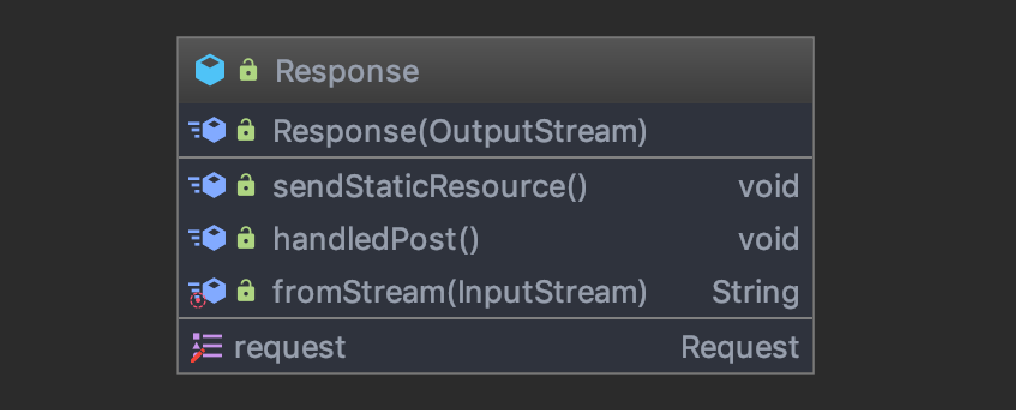
*private* String parseUri(String requestString) {  
 *int* index1, index2;  
 String s = "";  
 index1 = requestString.indexOf(' ');  
 *if* (index1 != -1) {  
 index2 = requestString.indexOf(' ', index1 + 1);  
 *if* (index2 > index1)  
  
 s = requestString.substring(index1 + 1, index2).replace("#", "");  
 *if* (s.equals("/")){  
 *return* "index.html";  
 }  
 *else return* s;  
 }  
 *return* "index.html";  
*// return null;* }

parsePost是拿到post请求的body正文：

*private void* parsePost(StringBuffer str){  
 System.out.println(str);  
 String [] arr = str.toString().split("\\s+");  
 postInfo.exefile = arr[1];  
 arr = str.toString().split("\r\n\r\n");  
  
 *if*(arr.length > 1){  
 postInfo.variables = arr[1];  
 }  
 *int* a = str.indexOf("Referer: ")+9;  
 String temp = str.substring(a);  
 arr = temp.split("\r\n");  
 postInfo.referer = arr[0];  
}

## 3.3 response设计

response类如下，主要完成文件的写回和处理post请求时调用php-cgi功能。



sendStaticResource是传回服务器上的文件：

*public void* sendStaticResource() *throws* IOException {  
 *byte*[] bytes = *new byte*[BUFFER\_SIZE];  
 FileInputStream fis = *null*;  
 *try* {  
 *//将web文件写入到OutputStream字节流中* File file = *new* File(MultiThreadWebServer.WEB\_ROOT, request.getUri());  
*// System.out.println(request.getUri());  
  
 if* (file.exists()) {  
 output.write("HTTP/1.0 200 OK".getBytes());  
 output.write("MIME\_version:1.0".getBytes());  
 output.write("Content\_Type:text/html".getBytes());  
 *int* len = (*int*) file.length();  
 output.write(("Content\_Length:" + len).getBytes());  
 output.write("\n\n".getBytes());*//报文头和信息之间要空一行  
  
 int* byteread = 0;  
 InputStream in = *new* FileInputStream(file);  
  
 *while* ((byteread = in.read(bytes)) != -1) {  
 output.write(bytes, 0, byteread);  
 }  
 output.flush();  
  
 } *else* {  
 *// file not found* System.out.println("not found");  
 String errorMessage = "HTTP/1.1 404 File Not Found\r\n" + "Content-Type: text/html\r\n"  
 + "Content-Length: 23\r\n" + "\r\n" + "<h1>File Not Found</h1>";  
 output.write(errorMessage.getBytes());  
 }  
 } *catch* (Exception e) {  
 System.out.println(e.toString());  
 } *finally* {  
 *if* (fis != *null*)  
 fis.close();  
 }  
 }

handlePost方法是处理Post请求，将Post请求的参数传到php-cgi中，然后读出返回值，再传送到客户端。

*public void* handledPost(){  
 String []env = {  
 "REDIRECT\_STATUS=true",  
 "GATEWAY\_INTERFACE=CGI/1.1",  
 "CONTENT\_TYPE=application/x-www-form-urlencoded; charset=UTF-8",  
 "HTTP\_HOST=127.0.0.1:8080",  
 "HTTP\_USER\_AGENT=Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_14\_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.131 Safari/537.36",  
 "REQUEST\_METHOD=POST",  
 "SCRIPT\_FILENAME="+MultiThreadWebServer.WEB\_ROOT+request.getExeFile(),  
 "CONTENT\_LENGTH="+String.*valueOf*(request.postInfo.variables.length()),  
 };  
*// System.out.println("handlpost");  
 try* {  
 Process p = Runtime.*getRuntime*().exec("php-cgi", env);  
 OutputStream out = p.getOutputStream();  
*// System.out.println(request.postInfo.variables);* out.write(request.postInfo.variables.getBytes());  
 out.flush();  
 InputStream is = p.getInputStream();  
*// System.out.println(fromStream(is));  
  
 byte*[] bytes = *new byte*[2048];  
 *int* readlength = is.read(bytes,0,2048);  
  
 output.write("HTTP/1.0 200 OK".getBytes());  
*// output.write("MIME\_version:1.0".getBytes());* output.write("Content\_Type:text/html".getBytes());  
  
 output.write(("Content\_Length:" + readlength).getBytes());  
 output.write("\n\n".getBytes());*//报文头和信息之间要空一行* output.write(bytes, 66, readlength);  
*// String s = new String(bytes);* output.flush();  
 }*catch* (Exception e){  
 e.printStackTrace();  
 }  
   
 }

env是配置的php-cgi的环境变量。

Process p是调用php-cgi进程，将post的body的loginName和password等参数传进去，然后通过p.getInputStream拿到返回值。再传回到客户端。

# 四、测试与运行

## 4.1 程序测试

测试环境：Chrome浏览器+安装最新版本php的mac+IntelliJ。

运行端口为127.0.0.1:8080。

注：如果不是用IntelliJ运行本程序，可能会出现web文件夹路径错误的问题，无法得到index.html文件。

在程序代码基本完成后，经过不断的调试与修改，最后测试本次所设计的web服务器可以正常运行，没有出现明显的错误和漏洞。

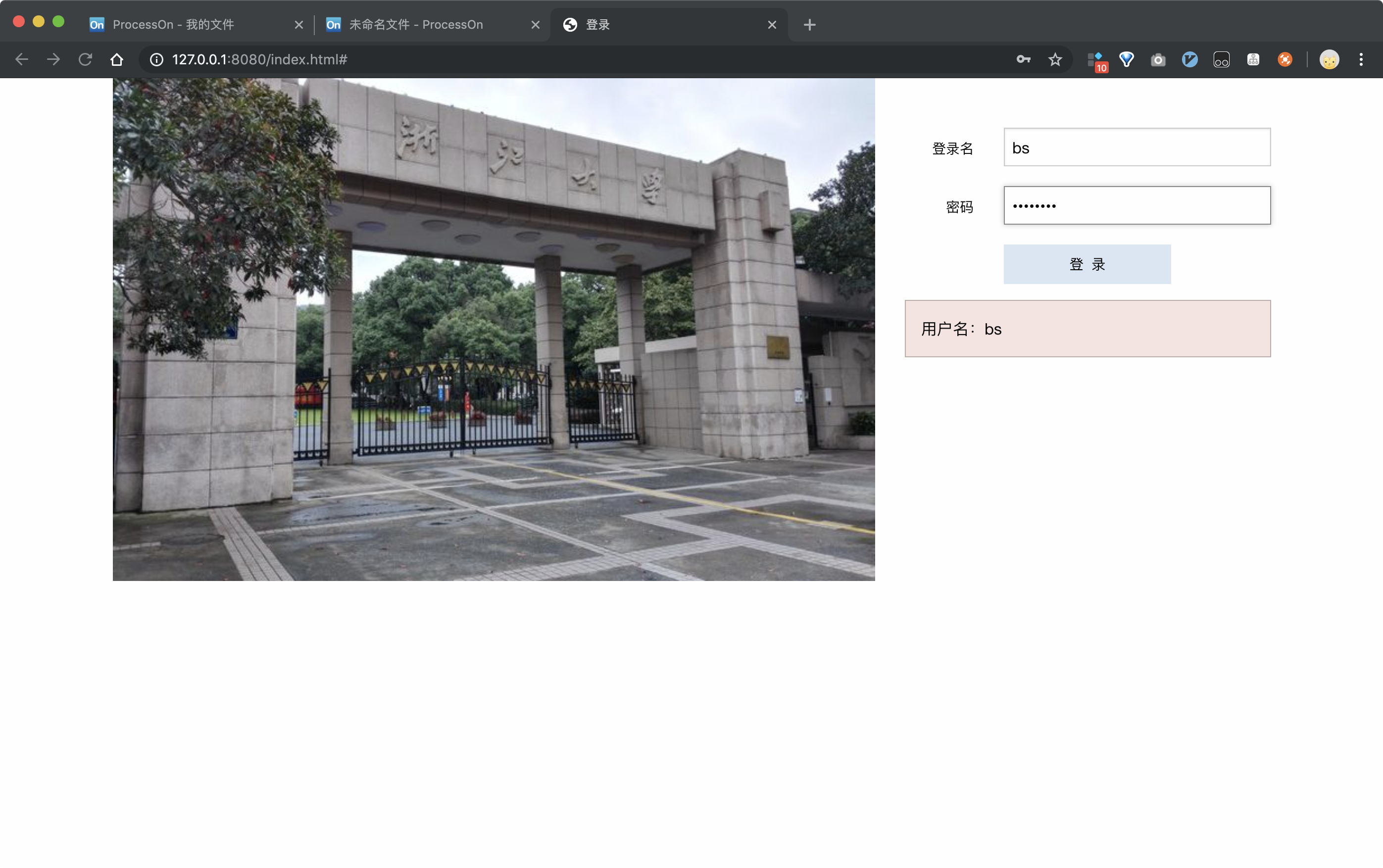
## 4.2 程序运行

**4.2.1 get请求：**



结论：该程序实现了get请求，可以将index.html以及其他js css文件传回客户端。

**4.2.2 post界面：**





结论：该web服务器可以实现作业要求的post。

# 五、心得感想

此次的实验，我对于Java的socket编程有了一定的认识，最重要的是了解了HTTP报文内容和多线程处理机制，我在一开始对多线程是有些畏惧心理的，但是Java的多线程处理起来，比c要容易太多。另外就是在调用php-cgi的时候，环境变量的配置出了很大问题，网上可以供查询的资料比较少，卡了比较长的时间。配置好了之后，发现只能传进去参数，拿不到loginName，报错是loginName这个index not found。由于无法对php-cgi内部进行debug，又仔细检查了一下环境变量，发现忘了添加METHOD=POST这个，所以php-cgi没有返回用户名。