# 《计算机网络与分布式系统》课程设计

**姓名：孙熙哲 学号：2019141660118**

1. **运行环境配置和说明文档**

运行环境配置:

操作系统：Windows 10，

jdk版本：1.8，

maven版本：apache-maven-3.3.9

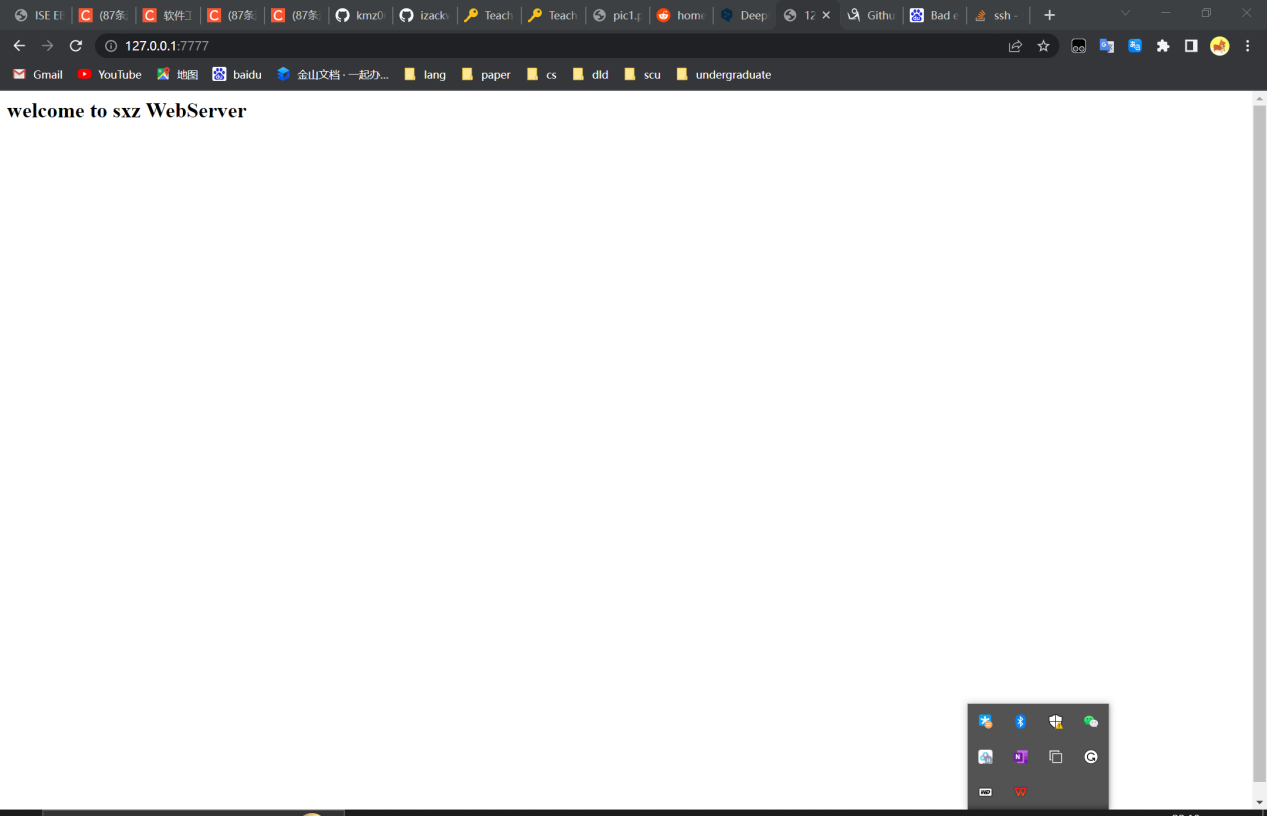
开发工具：IDEA 2019

运行浏览器：Google Chrome, FireFox, Edge

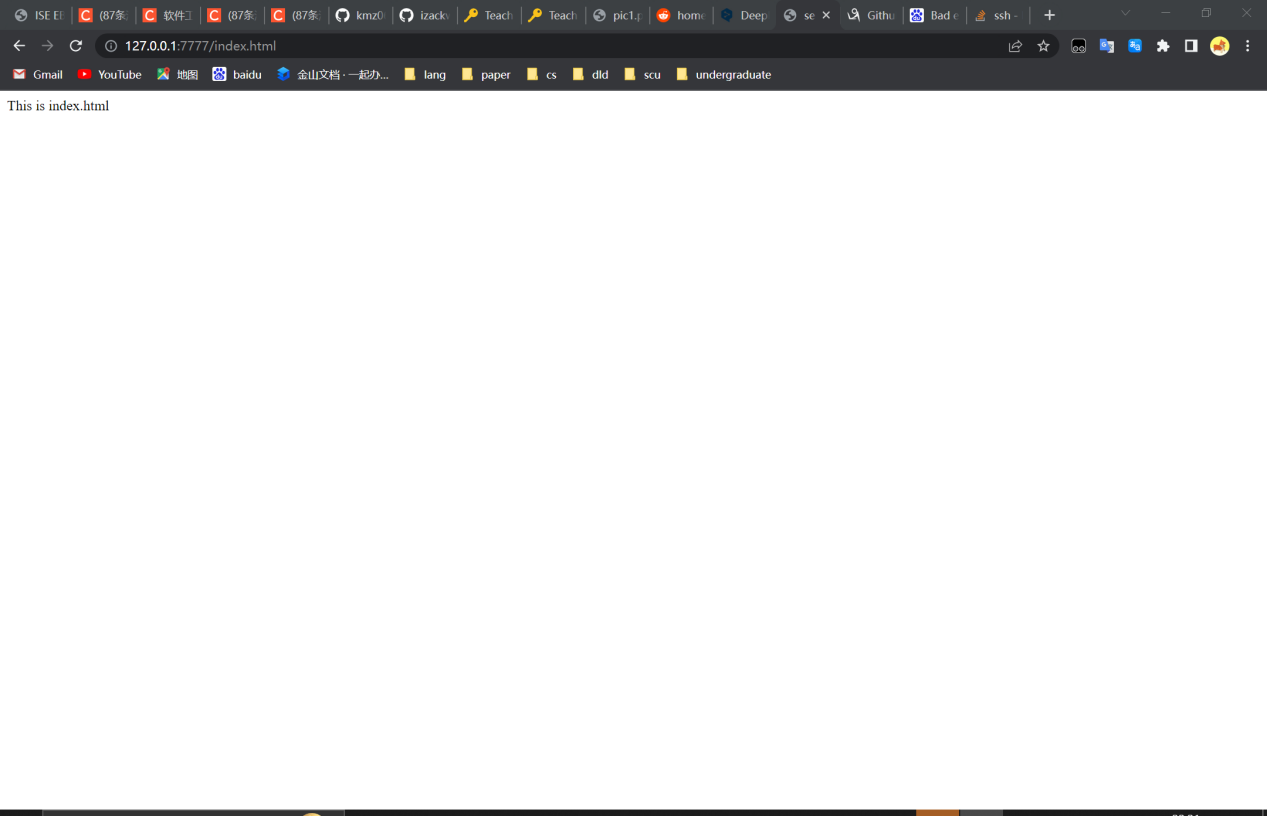
1. **软件运行说明文档（附运行截图）**

运行说明：

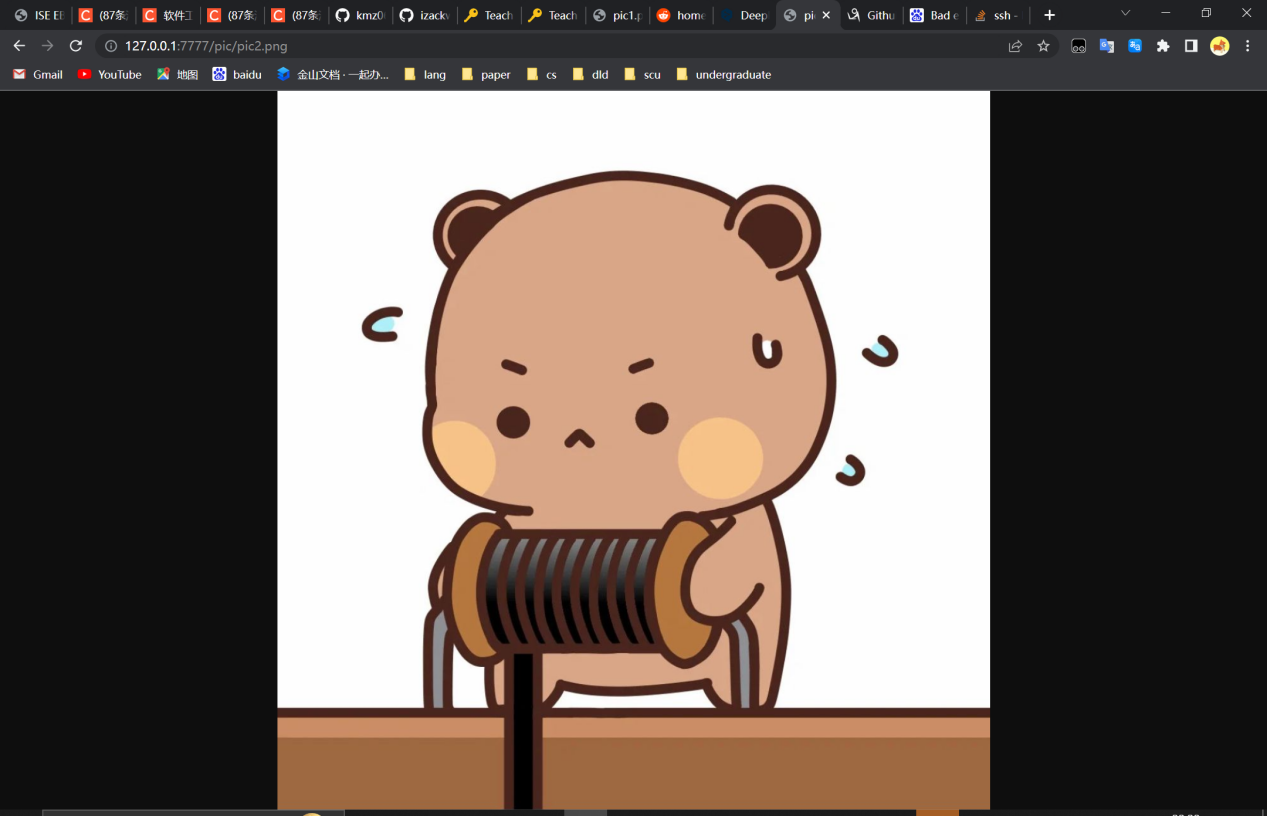
1. 在程序运行后浏览器地址栏内输入127.0.0.1:7777，展示的欢迎页面如下图所示



1. 在地址栏内端口号后输入想要访问的资源的url，如index.html



1. 还可以访问其他在D:\\res目录下的资源



1. **课程设计的完整的可运行的代码。**

Server.java

package server;  
  
import data.MyData;  
  
import java.net.ServerSocket;  
import java.net.Socket;  
  
public class Server implements Runnable {  
  
 public static int *PORT*;//监听端口  
  
 public Server(int port) {  
 this.*PORT* = port;  
 }  
  
  
 public void run() {//重写run方法  
 //主程序  
 ServerSocket serverSocket = null;//声明ServerSocket对象  
 try {  
 serverSocket = new ServerSocket(*PORT*);//创建serverSocket对象  
  
 System.*out*.println("开始监听 " + *PORT*);  
 while (MyData.*isRun*) {  
 //在允许服务器运行的情况下监听端口  
 Socket socket = serverSocket.accept();//开始监听  
 System.*out*.println("收到请求 ");  
 //将socket交给RequestProcess处理  
 RequstProcess requstProcess = new RequstProcess(socket);  
 requstProcess.start();  
 }  
 //退出循环 server停止运行  
 //关闭ServerSocket  
 serverSocket.close();  
 serverSocket = null;  
 } catch (Exception e) {  
 e.printStackTrace();  
 throw new RuntimeException("端口" + *PORT* + "监听失败 " + e.getMessage());  
 }  
 }  
  
 public static void main(String[] args) {  
 Server server = new Server(7777);//创建线程类对象  
 new Thread(server).start();//Thread启用线程  
 }  
}

RequestProcess.java

package server;  
  
import data.MyData;  
  
import java.io.\*;  
import java.net.Socket;  
  
*/\*\*  
 \** ***@descripstion*** *请求  
 \*/*public class RequstProcess extends Thread {  
 private Socket socket;//利用constructor初始化  
  
 public RequstProcess(Socket socket) {  
 this.socket = socket;  
 }  
  
 public void run() {  
 //从socket取出输出流 然后从输入流取出数据  
 InputStream inputStream = null;//字节输入转换为缓冲字符输入流  
 InputStreamReader inputStreamReader = null;//转换流  
 BufferedReader bufferedReader = null;//字符缓冲流  
  
 //声明输出流 输出流指向客户端  
 OutputStream outputStream = null;  
 PrintWriter printWriter = null;  
  
 try {  
 outputStream = socket.getOutputStream();//从socket取出输出流  
 printWriter = new PrintWriter(outputStream);  
 inputStream = socket.getInputStream();  
 inputStreamReader = new InputStreamReader(inputStream);//转换成字符流再包装  
 bufferedReader = new BufferedReader(inputStreamReader);//字符缓冲流  
  
 String line = null;  
 Integer lineCount = 1;  
 String requestPath = "";//储存请求路径  
 String host = "";  
 while ((line = bufferedReader.readLine()) != null) {  
 System.*out*.println(line);//解析请求行  
 if (lineCount == 1) {  
 String[] infos = line.split(" ");  
 if (infos != null || infos.length > 2) {  
 requestPath = infos[1];  
 } else {  
 throw new RuntimeException("请求行解析失败 " + line);  
 }  
 } else {  
 String[] infos = line.split(":");  
 if (infos != null || infos.length == 2) {  
 if (infos[0].equals("Host")) {  
 host = infos[1];  
 }  
 }  
 }  
 lineCount++;  
 if (line.equals("")) {//读到空行就结束 http是长连接 读不到文件末尾  
 break;  
 }  
 }  
 //输出请求  
 if (!requestPath.equals("")) {  
 System.*out*.println("处理请求:http://" + host + requestPath);  
 //根据url响应请求  
 if (requestPath.equals("/")) {  
 //无请求资源名称  
 printWriter.println("HTTP/1.1 200 OK");//输出响应行  
 printWriter.println("Content-Type: text/html;charset=utf-8");  
 printWriter.println("\r\n");  
 printWriter.println("<h2>welcome to sxz WebServer</h2>");  
 printWriter.flush();  
 System.*out*.println("响应欢迎页");  
 } else {  
 //取出后缀  
 String postfix = requestPath.substring(requestPath.lastIndexOf(".") + 1);  
 requestPath = requestPath.substring(1);//去掉开头的/  
 //判断是在根目录还是子目录下  
 if (requestPath.contains("/")) {//子目录  
 File file = new File(MyData.*resourcePath* + requestPath);  
 if (file.exists() && file.isFile()) {  
 response200(outputStream, file.getAbsoluteFile().getPath(), postfix);  
 } else {  
 response404(outputStream);  
 }  
 } else {//根目录  
 //判断资源是否存在  
 //获取根目录下文件名称  
 File root = new File(MyData.*resourcePath*);  
 if (root.isDirectory()) {  
 File[] list = root.listFiles();  
 System.*out*.println(list[0].getName());  
 boolean isExist = false;  
 for (File file : list) {  
 if (file.isFile() && file.getName().equals(requestPath)) {  
 isExist = true;  
 break;  
 }  
 }  
 if (isExist) {//文件存在  
 response200(outputStream, MyData.*resourcePath* + requestPath, postfix);  
 } else {  
 response404(outputStream);  
 }  
 } else {  
 throw new RuntimeException("静态资源不存在:" + MyData.*resourcePath*);  
 }  
 }  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 throw new RuntimeException(e);  
 } finally {  
 try {  
 if (inputStream != null) {  
 inputStream.close();  
 }  
 if (bufferedReader != null) {  
 bufferedReader.close();  
 }  
 if (printWriter != null) {  
 printWriter.close();  
 }  
 if (outputStream != null) {  
 outputStream.close();  
 }  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
 }  
  
 */\*\*  
 \** ***@param*** *outputStream  
 \** ***@param*** *filePath  
 \** ***@param*** *postfix  
 \*/* private void response200(OutputStream outputStream, String filePath, String postfix) {  
 PrintWriter printWriter = null;  
 //准备输入流 获取磁盘上的文件  
 InputStream inputStream = null;  
 InputStreamReader inputStreamReader = null;  
 BufferedReader bufferedReader = null;  
  
 try {  
 printWriter = new PrintWriter((outputStream));  
 if (postfix.equals("jpg") || postfix.equals("png") || postfix.equals("gif")) {  
 outputStream.write("HTTP/1.1 200 OK\r\n".getBytes());//输出响应行  
 outputStream.write(("Content-Type:image/" + postfix + "\r\n").getBytes());  
 outputStream.write("\r\n".getBytes());//输出空行表示响应头结束  
 //利用字节输入流读取文件内容  
 inputStream = new FileInputStream(filePath);  
 int len = -1;  
 byte[] buff = new byte[1024];  
 while ((len = inputStream.read(buff)) != -1) {  
 outputStream.write(buff, 0, len);  
 outputStream.flush();  
 }  
 } else if (postfix.equals("html") || postfix.equals("js") || postfix.equals("css") || postfix.equals("json")) {  
 outputStream.write("HTTP/1.1 200 OK\r\n".getBytes());//输出响应行  
 if (postfix.equals("html")) {  
 outputStream.write(("Content-Type:text/html;charset=utf-8\r\n").getBytes());  
 } else if (postfix.equals("js")) {  
 outputStream.write(("Content-Type:application/x-javascript\r\n").getBytes());  
 } else if (postfix.equals("css")) {  
 outputStream.write(("Content-Type:text/css\r\n").getBytes());  
 } else if (postfix.equals("json")) {  
 outputStream.write(("Content-Type:application/json;charset=utf-8\r\n").getBytes());  
 }  
 outputStream.write("\r\n".getBytes());//输出空行表示响应头结束  
 FileInputStream fileInputStream = new FileInputStream(filePath);  
 InputStreamReader reader = new InputStreamReader(fileInputStream);  
 BufferedReader bufferedReader1 = new BufferedReader(reader);  
 String line = null;  
 while ((line = bufferedReader1.readLine()) != null) {  
 printWriter.println(line);  
 printWriter.flush();  
 }  
 } else {  
 response404(outputStream);  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 if (printWriter != null) {  
 printWriter.close();  
 }  
 } catch (Exception ex) {  
 ex.printStackTrace();  
 }  
 }  
 }  
  
 */\*\*  
 \* 响应404  
 \*  
 \** ***@param*** *outputStream  
 \*/* private void response404(OutputStream outputStream) {  
 PrintWriter printWriter = null;  
 try {  
 printWriter = new PrintWriter((outputStream));  
 printWriter.println("HTTP/1.1 404");//输出响应行  
 printWriter.println("Content-Type: text/html;charset=utf-8");  
 printWriter.println("\r\n");  
 printWriter.println("<h2>资源不存在</h2>");  
 printWriter.flush();  
 System.*out*.println("404页");  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 if (printWriter != null) {  
 printWriter.close();  
 }  
 } catch (Exception ex) {  
 ex.printStackTrace();  
 }  
 }  
 }  
}

MyData.java

package data;  
  
public class MyData {  
 public static boolean *isRun* =true;  
 public static boolean *isPush* =true;  
 public static String *resourcePath*="D:/res/";  
}