---------------------------------------------------

计划课程安排：

FUNDAMENTAL(7天)         该阶段已结束

OOP(11天)           该阶段已结束

JAVASE01(10天) 该阶段已结束

JAVASE02(10天) ←当前为该阶段第9天

WEBBASIC(7天)

JQUERY(2天)

DATABASE(6天)

JDBC(3天)

SERVLETJSP(12天)

SPRINGMYBATIS01(8天)

AJAX(2天)

SPRINGMYBATIS02(12天)

LINUX(2天)

NGINX(2天)

REDIS(2天)

T-STORE(5天)

预计合计：101天

预计剩余：64天

注：上述信息随时可能发生更改，仅供参考。

---------------------------------------------------

**2019年4月22日**

**Java就业班-第46天**

**8:30实训室2-早自习**

薪王已篡火，我从初始火炉回来了！

那么今天的晨讲是什么呢？——方法调用

从现阶段的情况来说内容太过于基础，5分钟就讲完了。

然后被班主任喷了。

还不知道今天讲什么，但据说这周会开始进入web前端的内容。

然后这周也是51假期前的9天课连上，我特么肝爆！

**9:00实训室2-正课**

开始上课。

首先聊一聊什么是post请求。

将login.html里的get改为post保存后再打开服务器登录页面测试登陆的结果为：

<http://127.0.0.1:8088/myweb/login>（地址栏没有了”?”及后面的输入信息）

↑

正确的登录处理页面url应该如此。

**9:10实训室2-正课**

新建一个项目——WebServer\_v14

README.txt----------------

本版本主要工作：支持post请求提交form表单数据

当页面表单含有用户隐私信息，或者上传附件这样的操作时，提交形式应当使用post。

post会将表单数据放在请求的消息正文部分传递。

如果只是传递用户输入的数据（不含有附件）时，浏览器会同时在这次请求的消息头中包含两个头：

Content-Length:说明消息正文的长度（单位字节）

Content-Type:application/x-www-from-urlencoded

当Contetn-Type的值为上述，那么就说明这个消息正文的内容

是form表单提交的用户数据，是一个字符串，就是原get请求中url的"?"右侧内容。

实现：

1.以登录为例，修改登录页面login.html的form表单的提交形式为post，即：method="post"

2.完成HttpRequest中解析消息正文的方法parseContent()。

2.1-首先判断请求方式是否为post，如果是才进行解析消息正文的操作。

2.2-通过消息头中Content-Length获取到消息正文的长度，

并通过输入流读取这些字节回来，以一个字节数组形式保存。

因此我们要在HttpRequest中定义一个属性，是一个byte[]，

用来保存正文数据。

2.3-再根据Content-Type的值判定正文内容，并对正文数据进行相应转换，

这里暂时只是判断是否为form表单数据，

即：只看值是否为：application/x-www-from-urlencoded

2.4-将正文数据转换为字符串（字符集ISO8859-1)，转换出来的内容应当与

原get请求中url里"?"右侧内容一致。例如：

username=Kamui\_sato&password=123456%……

2.5-将转换的字符串进行解析，并存入到属性paramenter这个Map中来保存

用户提交的所有参数。

以上操作完毕后，Servlet又可以通过request获取到用户提交的参数并处理业务了。

--------------------------

功能实现：

package com.webserver.http;

import java.io.IOException;

import java.io.InputStream;

import java.net.Socket;

import java.util.HashMap;

import java.util.Map;

import com.webserver.core.EmptyRequestException;

/\*\*

\* 请求对象 该类的每一个实例用于表示浏览器发送过来的一个请求内容， 每个请求由三部分构成（请求行，消息头，消息正文）

\*

\* @author Administrator

\*

\*/

public class HttpRequest {

/\*

\* 请求行相关信息

\*/

// 请求方式

private String method;

// 请求资源的抽象路径

private String url;

// 请求时用的协议版本

private String protocol;

// 保存url中"?"左侧的请求部分

private String requestURI;

// 保存url中"?"右侧的参数部分

private String queryString;

// 保存解析出来的每一个参数（key:参数名 value：参数值）

private Map<String, String> parameters = new HashMap<>();

/\*

\* 消息头相关信息

\*/

/\*

\* key:消息头名 value:消息头对应的值

\*/

private Map<String, String> headers = new HashMap<String, String>();

/\*

\* 消息正文相关信息

\*/

// 消息正文的数据

private byte[] data;

/\*

\* 与连接相关的属性

\*/

private Socket socket;

private InputStream in;

/\*\*

\* 初始化HttpRequest对象 初始化的过程就是解析请求的过程。 当前HttpRequest对象就表示浏览器发送过来的 这个请求内容了。

\*/

public HttpRequest(Socket socket) {

System.out.println("HttpRequest:开始解析请求……");

try {

this.socket = socket;

this.in = socket.getInputStream();

/\*

\* 解析请求的三步： 1.解析请求行 2.解析消息头 3.解析消息正文

\*/

parseRequestLine();

parseHeaders();

parseContent();

} catch (Exception e) {

e.printStackTrace();

}

System.out.println("HttpRequest:解析请求完毕！");

}

/\*\*

\* 解析请求行

\*

\* @throws EmptyRequestException

\*/

private void parseRequestLine() throws EmptyRequestException {

System.out.println("开始解析请求行……");

/\*

\* 1.通过输入流读取第一行字符串（请求行内容） 2.将请求行内容按照空格拆分为三部分

\* 3.将三部分内容设置到对应属性上（method,url,protoco）

\*/

try {

String line = readLine();

System.out.println("开始解析请求行：" + line);

/\*

\* 判断是否为空请求

\*/

if ("".equals(line)) {

throw new EmptyRequestException();

}

/\*

\* 后期循环接受客户端连接后，下面代码可能会出现 数组下标越界，这是由于空请求引起的，后面会解决

\*/

String data[] = line.split("\\s");

this.method = data[0];

this.url = data[1];// 这里会出现下标越界

this.protocol = data[2];

System.out.println("method:" + method);

System.out.println("url:" + url);

System.out.println("protocol:" + protocol);

// 进一步解析抽象路径部分

parseURL();

} catch (EmptyRequestException e) {

System.out.println("空请求……");

} catch (Exception e) {

e.printStackTrace();

}

System.out.println("解析请求行完毕！");

}

private void parseURL() {

System.out.println("开始解析抽象路径……");

/\*

\* 一个请求的请求行大众抽象路径部分有两种情况： 1.不含有参数的，如： /myweb/index.html 2.含有参数的，如：

\* /myweb/reg?username=xxx&password=xxx&... 因此我们对抽象路径进一步解析： 1.首先判断url的值是否含有"？"

\* 1.1若不含有"?"则直接将url的值赋给属性requestURI,当前方法结束。 1.2若含有"?"执行步骤2

\*

\* 2.将url按照"?"拆分为两部分，第一部分应当是请求部分，赋值给属性requestURI. 第二部分应当是参数部分，赋值给属性queryString

\*

\* 3.进一步解析queryString,将其按照"&"拆分为若干个参数， 每一个参数再按照"="拆分为参数名与参数值。

\* 并将参数名作为key,参数值作为value保存到属性parameters这个Map中完成解析工作。

\*/

if (url.contains("?") == false) {

requestURI = url;

} else {

String data[] = url.split("\\?");

requestURI = data[0];

if (data.length > 1) {

queryString = data[1];

putParameter(queryString);

}

}

System.out.println("requestURI:" + requestURI);

System.out.println("queryString:" + queryString);

System.out.println("parameters:" + parameters);

// 进一步解析

System.out.println("解析抽象路径完毕！");

}

/\*\*

\* 解析消息头

\*/

private void parseHeaders() {

System.out.println("开始解析消息头……");

/\*

\* 1.循环调用readLine方法读取每一个消息头 2.将消息头按照：“拆分，并将消息头的名字作为key，

\* 消息头的值作为value保存到属性headers这个Map中” 3.如果调用readLine方法返回的是一个空字符串，

\* 则说明了本次单独读取到了CRLF，那么就可以停止解析消息头了。

\*/

try {

String line = null;

while (true) {

line = readLine();

if ("".equals(line)) {

break;

}

String data[] = line.split(": ");

headers.put(data[0], data[1]);

}

} catch (Exception e) {

e.printStackTrace();

}

System.out.println("消息头：" + headers);

System.out.println("解析消息头完毕！");

}

/\*\*

\* 解析消息正文

\*/

private void parseContent() {

System.out.println("开始解析消息正文……");

try {

if("post".equals(this.method.toLowerCase())) {

String type = headers.get("Content-Type");

if("application/x-www-form-urlencoded".equals(type)) {

//获取消息正文

int length = Integer.parseInt(headers.get("Content-Length"));

System.out.println("消息正文长度为：" + length);

data = new byte[length];

//读取消息正文数据

in.read(data);

String text = new String(data,"ISO8859-1");

System.out.println("消息正文为：" + text);

putParameter(text);

}

}

}

catch (IOException e) {

e.printStackTrace();

}

System.out.println("解析消息正文完毕！");

}

/\*\*

\* 通过对应客户端的输入流读取一行字符串 （以CRLF结尾）

\*

\* @return

\* @throws IOException

\*/

private String readLine() throws IOException {

// 读取一行字符串，以CRLF结尾

StringBuilder builder = new StringBuilder();

// c1表示上次读取到的字符，c2表示本次读取到的字符

int c1 = -1, c2 = -1;

while ((c2 = in.read()) != -1) {

// 是否连续读取到了CR,LF

if (c1 == HttpContext.CR && c2 == HttpContext.LF) {

break;

}

builder.append((char) c2);

c1 = c2;

}

return builder.toString().trim();

}

public String getMethod() {

return method;

}

public String getUrl() {

return url;

}

public String getProtocol() {

return protocol;

}

/\*\*

\* 获取指定名字消息头对应的值

\*

\* @param name

\* @return

\*/

public String getHeaders(String name) {

return headers.get(name);

}

public String getRequestURI() {

return requestURI;

}

public String getQueryString() {

return queryString;

}

/\*\*

\* 根据给定的参数名获取对应的参数值

\*

\* @param name

\* @return

\*/

public String getParameter(String name) {

return this.parameters.get(name);

}

/\*\*

\* 根据给定的参数文本，通过“&”和“=”的分割后，

\* 将值分别保存为parameter集合的键值

\* @param text

\*/

public void putParameter(String text) {

String data[] = text.split("&");

for (String pm : data) {

String kv[] = pm.split("=");

if (kv.length > 1) {

parameters.put(kv[0], kv[1]);

} else {

parameters.put(kv[0], null);

}

}

}

}

**12:00实训室2-午休**

给了一上午的时间来自行实现功能。

没胃口吃饭了，补觉。

**14:38实训室2-正课**

睡的正香被项目经理叫醒了……

又是一个新的需求。

新建一个项目WebServer\_v15

README.txt-------------

本版本主要工作：完成修改密码功能

流程：

1.用户访问修改页面update.html

2.在页面上输入用户名，旧密码，新密码

点击修改按钮

表单提交路径"./update"

3.处理的Servlet名为：UpdateServlet

当用户名或密码不对时提示修改失败:update\_fail.html

修改成功提示:update\_success.html

form提交为post形式提交。

(不再提示实现步骤。)

------------------------

**16:34实训室2-正课**

功能实现：（因相关代码量较多，如有需要请下载up主的Day46--WebServer\_v15源代码）

**17:01实训室2-正课**

讲师讲解完了功能实现的过程，和我写的没有什么区别，所以就没有修改我的源代码。

看样子今天的课就这样了，真的是能混啊。

说是因为课程进度推的太快，导致现在超标了，剩下几天没什么可讲的了，所以就只好放慢进度了。

明天或者后天，作为一个附加课程吧，讲师将讲解如何实现验证码的功能。

**17:10实训室2-下课时间**

下课了。

吃饭之前顺带帮舍友解决了一个小BUG，原因仅仅是因为在update\_fail.html写的url返回地址多了一个“/”。就这个问题我们通篇检查了代码……

**18:37实训室2-晚自习**

晚自习了，今天状态不佳，昨晚传火到比较晚，今天可能不太适合看书，试着做做机构的练习题吧。

**19:32实训室2-晚自习**

（以下贴出的题目均为Up主做错或不知道答案的题……）

**1.有关javadoc命令生成文档叙述正确的是：**

A.如果是一个包命令为javadoc –d docDir nameOfPackage

B.如果是一个包命令为javadoc –d docDir nameOfPackages

C.如果文件在默认包中命令为javadoc –d docDir \*.java

D.如果文件在默认包中命令为javadoc –d docDir \*.javac

答案AC

**2.指出下列程序运行的结果（ ）**

public class Example{

String str=new String("good");

char[]ch={'a','b','c'};

public static void main(String args[]){

Example ex=new Example();

ex.change(ex.str,ex.ch);

System.out.print(ex.str+" and ");

System.out.print(ex.ch);

}

public void change(String str,char ch[]){

str="test ok";

ch[0]='g';

}

}

A.good and abc

B.good and gbc

C.test ok and abc

D.test ok and gbc

答案B

引用类型变量保存的是对象的地址，一个对象可以同时被多个变量所引用，并且操作，一个变量只能引用一个对象，对一个对象进行操作。

**3.以下代码编译运行的结果是（）。**

public class Base{

private void test() {

String aStr = "? One? ";

String bStr = aStr;

aStr.toUpperCase();

aStr.trim();

System.out.println("[" + aStr + "," + bStr + "]");

}

static public void main(String[] a) {

new Base().test();

}

}

A.[ONE,?One?]

B.[? One? ,One ]

C.[ONE,ONE]

D.[? One? ,? One? ]

答案D

通过 String bStr = aStr;这句代码使 bStr 和 aStr 指向同一个地址空间，所以最后 aStr 和 bStr 的结 果应该是一样，String 类是定长字符串，调用一个字符串的方法以后会形成一个新的字符串。

在调用toUpperCase()和trim()方法时都会返回新的字符串对象，

而题目中在调用完这两个方法后并没有使用变量接收方法所返回的新字符串对象，而aStr和bStr所引用的仍然是原来的字符串对象。

所以选项D正确。

**4.下面代码的打印结果是**

String s1="abc";

String s2="abc";

System.out.println(s1==s2);

System.out.println(s1.equals(s2));

A.true,true

B.true,false

C.false, false

D.false,ture

答案A

此题目考查的是对于java常量池的理解, java中的常量池技术，是为了方便快捷地创建某些对象而出现的，当需要一个对象时，就可以从池中取一个出来（如果池中没有则创建一个），则在需要重复创建相等变量时节省了很多时间。常量池其实也就是一个内存空间，常量池存在于方法区中。根据代码顺序，先在常量池中创建”abc“，并指向s1,而后在创建s2时，由于常量池中已经存在”abc“，只需指向s2就可以，而不需要再创建。”==”在这里比较的是对象引用，故结果为”true”,String 中的equals方法经过重写后操作为“比较此字符串与指定的对象。当且仅当该参数不为 null，并且是表示与此对象相同的字符序列的 String 对象时，结果才为 true”。很明显，s1和s2的字符序列相同，故而结果为true

因此A是正确的选项。

**5. 以下关于hashCode说法正确的是（ ）**

A.两个对象如果hashCode值不同，则两个对象可能相同。

B.两个对象如果hashCode值相同，则两个对象必定相同。

C.两个对象如果hashCode值相同，则两个对象可能相同。

D.两个对象如果hashCode值不同，则两个对象必定不同。

答案CD

先通过 hashcode 来判断某个对象是否存放某个桶里，但这个桶里可能有很多对象，那么我们就需要再通过 equals 来在这个桶里找到我们要的对象。

**6.以下集合对象中哪几个是线程安全的（）**

A.LinkedList

B.ArrayList

C.Vector

D.Hashtable

答案CD

Collection 是对象集合， Collection 有两个子接口 List 和 Set,

List 可以通过下标 (1,2..) 来取得值，值可以重复,而 Set 只能通过游标来取值，并且值是不能重复的

ArrayList ， Vector ， LinkedList 是 List 的实现类

ArrayList 是线程不安全的， Vector 是线程安全的，这两个类底层都是由数组实现的

LinkedList 是线程不安全的，底层是由链表实现的

Map 是键值对集合

HashTable 和 HashMap 是 Map 的实现类

HashTable 是线程安全的，不能存储 null 值

HashMap 不是线程安全的，可以存储 null 值

**7.关于 Java 语言，下列描述正确的是（ ）**

A.switch不能够作用在String类型上

B.List, Set, Map都继承自Collection接口

C.Java语言支持goto语句

D.GC是垃圾收集器，程序员不用担心内存管理

答案AD

B. Map没有继承Collection接口；

C. java不支持goto语句。

**8.链表具有的特点是（ ）**

A.不必事先估计存储空间

B.可随机访问任一元素

C.插入删除不需要移动元素

D.所需空间与线性表长度成正比

答案ACD

A. 采用动态存储分配，不会造成内存浪费和溢出。

B. 不能随机访问，查找时要从头指针开始遍历

C. 插入、删除时，只要找到对应前驱结点，修改指针即可，无需移动元素

D. 需要用额外空间存储线性表的关系，存储密度小

**20:40实训室2-晚自习**

时间差不多了，机构的题目是真的做起来要命，到处都是陷阱，而且考察点好偏，难道面试题都是这种吗？这可就有点病态了。考验基础功是不假，只是这样的题做起来可真是莫名的恼火……哎~要气就气自己太辣鸡了吧。

话说回来这周四貌似是又要月考了来着，要都是像这样的题估计得跪……

我的同桌已经自行决定要留级了，由于3月的班已经人满了，所以他要留级的话也只能去4月班了，话句话来说就是这2个月全部重新来过，白白浪费了2个月的时间和生活费。

事到如今就算我想帮也帮不了什么了，看样子下个月又要淘汰一批人留级了。

今日感想总结：其他学员也好，舍友同桌也好，当初乐观的心态已经一去不复返了。虽然可以说自己作死占了很大一部分比重吧，但我还是觉得有些朋友因为各方面因素的影响，可能真的不适合学编程。

今日评分：3/10分

今天就到这里，明天继续投稿。

感想支持！

================================================================

**Acer程序员群：790482850**

**up主的java学习日记word文件及全部代码（随日记同步更新）：**

Adress：github.com/wt62635