**随堂测试②**

一．单选题（每空3分），共30分。在以下每小题给出的A 、B、C、D四个选项中，只有一个选项正确，请选出正确答案，并将答案填在题中括号内）。

1. 关于URI与URL的区别，以下描述不正确的是（ C ）。
2. URI 类与URL 类相比，它更能适应相关的规范
3. URI 类只关心资源的标识和对URI的解析
4. 如果要获取内容，则使用URI;如果只是标识资源，则使用URL
5. URL 类在存放之前，就已经对所有的URI进行了“绝对化”的处理
6. 以下哪种编码方式经常会导致中文乱码（C）？
7. GBK
8. ASCII
9. UTF-8
10. GB2312
11. http服务器的响应码以“4”开头表示（ A ）。
12. 客户端错误
13. 服务器错误
14. 重定位或重定向
15. 响应成功
16. 下面关于URLConnection的内容，不正确的是（ B ）。
17. URLConnection可向服务器发送数据
18. URLConnection是URL的子类
19. URLConnection提供了对HTTP头信息访问
20. URLConnection可以配置发给服务器的参数
21. 下面关于DNS缓存，描述不正确的是（ B ）。
22. 修改域名指向可能需要几个小时才能生效
23. 按IP地址使用getByName查找，需要检查DNS
24. DNS查询可能需要几秒
25. 本机、本地DNS、其他DNS会缓存DNS查询
26. URL中的符号#的作用是（ D ）。
27. 传递参数
28. 标记一个新的网页地址
29. 清除缓存
30. 标记网页中的一个位置
31. 使用请求方法（ A ）需要设置 setDoOutput(true) 。
32. POST
33. HEAD
34. OPTIONS
35. GET
36. 下面关于GET 与 POST 的区别, 错误的是（ D ）。
37. get 是从服务器上获取数据，post 是向服务器传送数据
38. 在做数据查询时，建议用Get方式；而在做数据添加、修改或删除时，建议用Post方式
39. get 传送的数据量较小，不能大于2KB。post 传送的数据量较大，一般被默认为不受限制
40. get 安全性较高，post 安全性相对低。
41. 关于Cookie 与 Session的说法，错误的是（ C ）。
42. Cookie保存在客户端
43. Session在服务器端创建
44. Cookie在客户端创建
45. Session保存在服务器端
46. 通过getCanonicalHostName方法能得到（ D ）。
47. 域名
48. IP地址
49. IP地址和域名
50. IP地址或域名

二．判断题 （每小题2分，共10分，请给正确的打“√”，错误的打“×”，并填写题中括号内）。

1. InetAddress类的getLocalHost( )方法不会进行DNS查询，将直接返回主机名和回送地址。（ F ）
2. 不同服务器响应Http首部的属性数量是一样的 （ F ）
3. getHeaderFieldKey(int n), n=1时表示请求方法的字段名（ F ）
4. POST方法主要面向不能重复的不安全操作（ T ）
5. URI中必须使用ASCII字符（ F ）

三．综合题（每小题20分，共60分）

1. 试简析InetAddress类、URI类、URL类和URLConnection类的作用，分别在什么编程需求下可以选用？

InetAddress类：

作用：用于表示IP地址和主机名。

适用场景：获取或操作IP地址和主机名。

URI类（Uniform Resource Identifier）：

作用：用于表示统一资源标识符，即标识互联网上的资源。

适用场景：解析、操作或构建统一资源标识符（URI）。

URL类（Uniform Resource Locator）：

作用：用于表示统一资源定位符，即指向互联网上资源的地址。

适用场景：打开网络连接、读取远程资源。

URLConnection类：

作用：用于建立与URL所指向资源的连接，并提供了读取和写入远程资源的方法。

适用场景：对远程资源进行更底层的操作，例如发送请求、下载文件等。

1. 客户端编程：创建一个简单的HTTP客户端程序：EasyHTTPClient 类，它访问www.szu.edu.cn/xxgk/xxjj.htm, 把得到的HTTP响应结果保存到本地文件系统的一个文件中。
2. **import** java.net.\*;
3. **import** java.io.\*;
5. **public** **class** EasyHTTPClient{
6. String host="www.szu.edu.cn/xxgk/xxjj.htm";
7. **int** port = 80;
8. Socket socket;
9. **public** **void** createSocket() **throws** Exception    {
10. socket = **new** Socket(host, 80);
11. }
13. **public** **void** communicate() **throws** Exception {
14. StringBuffer sb = **new** StringBuffer("GET " + "/index.jsp" + " HTTP/1.1\r\n");
15. sb.append("Host: "+host+"\r\n");
16. sb.append("Accept: \*/\*\r\n");
17. sb.append("Accept-Language: zh-cn\r\n");
18. sb.append("Accept-Encoding: gzip, deflate\r\n");
19. sb.append("User-Agent: MyHTTPClient\r\n");
20. sb.append("Connection: Keep-Alive\r\n\r\n");
22. // 发出HTTP请求
23. OutputStream socketOut = socket.getOutputStream();
24. socketOut.write(sb.toString().getBytes());
25. socketOut.flush();
27. // 接收响应结果
28. InputStream socketIn = socket.getInputStream();
29. FileOutputStream fileOut = **new** FileOutputStream("response.data");
30. **byte**[] buff = **new** **byte**[1024];
31. **int** len = -1;
32. **while** ((len = socketIn.read(buff)) != -1)   {
33. fileOut.write(buff, 0, len);
34. }
35. fileOut.close();
36. socket.close();
37. System.out.println("响应数据已经保存到response.data文件中");
38. }
40. **public** **static** **void** main(String args[]) **throws** Exception  {
41. EasyHTTPClient client = **new** EasyHTTPClient();
42. client.createSocket();
43. client.communicate();
44. }
45. }

3.使用httpURLConnection实现web页面的下载。

1. **import** java.io.FileOutputStream;
2. **import** java.io.InputStream;
3. **import** java.net.HttpURLConnection;
4. **import** java.net.URL;
6. **public** **class** WebPageDownloader {
7. **public** **static** **void** main(String[] args) {
8. **try** {
9. URL url = **new** URL("http://example.com/page.html");
10. HttpURLConnection connection = (HttpURLConnection) url.openConnection();
11. connection.setRequestMethod("GET");
13. connection.connect();
15. InputStream inputStream = connection.getInputStream();
16. FileOutputStream fileOutputStream = **new** FileOutputStream("page.html");
18. **byte**[] buffer = **new** **byte**[1024];
19. **int** bytesRead;
20. **while** ((bytesRead = inputStream.read(buffer)) != -1) {
21. fileOutputStream.write(buffer, 0, bytesRead);
22. }
24. inputStream.close();
25. fileOutputStream.close();
26. connection.disconnect();
28. System.out.println("Web page downloaded successfully.");
29. } **catch** (Exception e) {
30. e.printStackTrace();
31. }
32. }
33. }