

Web应用开发 之 Servlet技术模型

赵小敏
浙江工业大学计算机科学与技术学院

本节内容

- 处理请求
- 发送响应

2.3 处理请求

- **HTTP消息**是客户向服务器的请求或者服务器向客户的响应。
- HTTP消息的各部分

消息部分	说明
请求行或状态行	指定请求或响应消息的目的
请求头或响应头	指定元信息，如关于消息内容的大小、类型、编码方式
空行	
可选的消息体	请求或响应消息的主要内容

2.3.1 HTTP请求结构

请求行	→	POST /paipaistore/selectProduct HTTP/1.1
请求头	→	Accept = */*
		Accept-Language = zh-cn
		Accept-Encoding = gzip, deflate
	→	User-Agent = Mozilla/4.0 (compatible; MSIE 9.0; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)
		Host = localhost:8080
	→	Connection = Keep-Alive
空行		
数据	→	productname=HUAWEIMate30

2.3.1 HTTP请求结构

- 由客户向服务器发出的消息叫做HTTP请求。

1. 请求行

- HTTP的请求行由三部分组成：方法名、请求资源的URI和HTTP版本。这三部分由空格分隔。

2. 请求头

- 请求行之后的内容称为请求头（request header），它可以指定请求使用的浏览器信息、字符编码信息及客户能处理的页面类型等。

2.3.1 HTTP请求结构

- 接下来是一个空行。
- 空行的后面是请求的数据。

3. HTTP的请求方法

- 请求行中的方法名指定了客户请求服务器完成的动作。

方 法	说 明	方 法	说 明
GET	请求读取一个Web页面	DELETE	移除Web页面
POST	请求向服务器发送数据	TRACE	返回收到的请求
PUT	请求存储一个Web页面	OPTIONS	查询特定选项
HEAD	请求读取一个Web页面的头部	CONNECT	保留作将来使用

4. GET方法和POST方法

- 在所有的HTTP请求方法中，GET方法和POST方法是两种最常用的方法。
- GET方法用来检索资源。它的含义是“获得（get）由该URI标识的资源”。
- POST方法用来向服务器发送需要处理的数据，它的含义是“将数据发送（post）到由该URI标识的主动资源”。

GET和POST方法的比较

特征	GET方法	POST方法
资源类型	主动的或被动的	主动的
数据类型	文本	文本或二进制数据
数据量	一般不超过255个字符	没有限制
可见性	数据是URL的一部分，在浏览器的地址栏中对用户可见	数据不是URL的一部分而是作为请求的消息体发送，在浏览器的地址栏中对用户不可见
数据缓存	数据可在浏览器的URL历史中缓存	数据不能在浏览器的URL历史中缓存

2.3.2 发送HTTP请求

- 在客户端如果发生下面的事件，浏览器就向Web服务器发送一个HTTP请求。
 - 用户在浏览器的地址栏中输入URL并按回车键。
 - 用户点击了HTML页面中的超链接。
 - 用户在HTML页面中添写一个表单并提交。

2.3.3 处理HTTP请求

- 在HttpServlet类中，除定义了service()方法为客户提供服务外，还针对每个HTTP方法定义了相应的doXxx()方法，一般格式如下：

```
protected void doXxx (HttpServletRequest,  
                      HttpServletResponse)  
throws ServletException, IOException;
```

HTTP方法	HttpServlet方法	HTTP方法	HttpServlet方法
GET	doGet()	DELETE	doDelete()
POST	doPost()	OPTIONS	doOptions()
HEAD	doHead()	TRACE	doTrace()
PUT	doPut()		

2.3.4 分析请求

- 客户发送给服务器的请求信息被封装在 `HttpServletRequest` 对象中，其中包含了由浏览器发送给服务器的数据，这些数据包括请求参数、客户端有关信息等。

1. 检索请求参数

- **请求参数**是随请求一起发送到服务器的数据，它是以名/值对的形式发送的。可以使用ServletRequest接口中定义的方法检索由客户发送的参数

- `public String getParameter(String name)`

返回由name指定的请求参数值，如果指定的参数不存在，则返回null值。使用该方必须确信指定的参数只有一个值。

1. 检索请求参数

- `public String[] getParameterValues(String name)`: 返回指定参数`name`所包含的所有值，返回值是一个`String`数组。如果指定的参数不存在，则返回`null`值。
- `public Enumeration getParameterNames()`: 返回一个`Enumeration`对象，它包含请求中所有的请求参数名，元素是`String`类型的。如果没有请求参数，则返回一个空的`Enumeration`对象。
- `public Map getParameterMap()`: 返回一个包含所有请求参数的`Map`对象，该对象以参数名作为键、以参数值作为值。

请求参数传递的方法

(1) 通过表单指定请求参数，每个表单域可以传递一个请求参数，这种方法适用于GET请求和POST请求。

- 程序login.html
- 程序LoginServlet.java

login.html

The screenshot displays two side-by-side views related to a web application's login page.

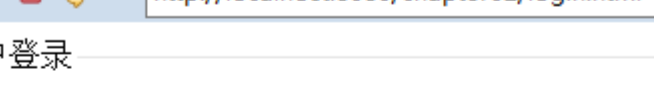
Left View (HTML Code):

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>登录页面</title>
</head>
<body>
    <form action="/login.do" method="post">
        <fieldset>
            <legend>用户登录</legend>
            <p>
                <label>用户名:<input type="text" name="username" /></label>
            </p>
            <p>
                <label>密    码:<input type="password" name="password" /></label>
            </p>
            <p>
                <label><input type="submit" value="登录" />
                    <input type="reset" value="取消" /> </label>
            </p>
        </fieldset>
    </form>
</body>
</html>
```

Right View (Visual Rendering):

A browser window titled "http://localhost:8080/chapter02/login.html" shows the rendered form:

- Title bar: http://localhost:8080/chapter02/login.html
- Form Title: 用户登录
- User Label: 用户名: [Text Input Field]
- Password Label: 密 码: [Password Input Field]
- Action Buttons: [Login Button] [Cancel Button]



用户登录

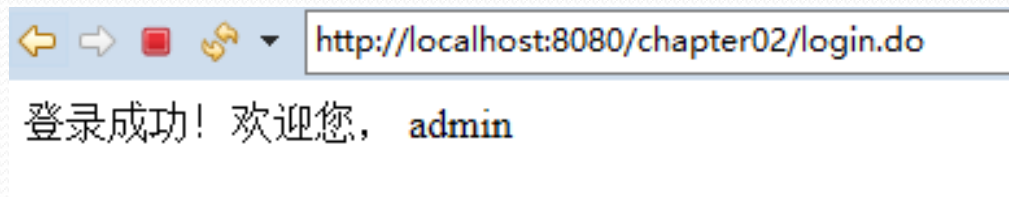
用户名:

密 码:

LoginServlet.java

```
@WebServlet("/login.do")
```

```
public class LoginServlet extends HttpServlet {  
    public void doPost(HttpServletRequest request,  
        HttpServletResponse response) throws ServletException, IOException {  
        String username = request.getParameter("username");  
        String password = request.getParameter("password");  
        response.setContentType("text/html;charset=UTF-8");  
        PrintWriter out = response.getWriter();  
        out.println("<!DOCTYPE html>");  
        out.println("<html><body>");  
        if ("admin".equals(username) && "admin".equals(password)) {  
            out.println("登录成功! 欢迎您, " + username);  
        } else {  
            out.println("对不起! 您的用户名或密码不正确. ");  
        }  
        out.println("</body></html>");  
    }  
}
```



请求参数传递的方法

(2) 通过查询串指定请求参数，将参数名和值附加在请求的URL后面，这种方法只适用于GET请求。

如

[http://localhost:8080/chapter02/login.do?username=admin
&password=admin](http://localhost:8080/chapter02/login.do?username=admin&password=admin)

- 问号后面内容为请求参数名和参数值对，若有多个参数，中间用“&”符号分隔，参数名和参数值之间用等号（=）分隔。
- 问号后面内容称为**查询串**（query string）。



2. 检索客户端有关信息

- 在`HttpServletRequest`接口中还定义了下面常用的方法用来检索客户端有关信息：
 - `public String getMethod()`
 - `public String getRemoteHost()`
 - `public String getRemoteAddr()`
 - `public int getRemotePort()`
 - `public String getProtocol()`
 - `public String getRequestURI()`
 - `public String getQueryString()`
 - `public String getContentType()`
 - `public String getCharacterEncoding()`

检索客户端有关信息示例ClientInfoServlet.java

```
@WebServlet("/client-information")
public class ClientInfoServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<html><head>");
        out.println("<title>客户端信息</title></head>");
        out.println("<body>");
        out.println("<p>客户端信息: </p>");
        out.println(request.getMethod() + " " + request.getRequestURI() + "
" + request.getProtocol() + "<br>");
        out.println("<p>客户主机名:" + request.getRemoteHost() + "</p>");
        out.println("<p>客户IP地址:" + request.getRemoteAddr() + "</p>");
        out.println("<p>端口号:" + request.getRemotePort() + "</p>");
        out.println("</body></html>");
    }
}
```

3. 检索HTTP请求头

- HTTP请求头是随请求一起发送到服务器信息，它是以“名/值”对的形式发送。

请求头	内 容
User-Agent	关于浏览器和它的平台的信息
Accept	客户能接受并处理的MIME类型
Accept-Charset	客户可以接受的字符集
Accept-Encoding	客户能处理的页面编码的方法
Accept-Language	客户能处理的语言
Host	服务器的DNS名字
Authorization	访问密码保护的Web页面时，客户用这个请求头来标识自己的身份
Cookie	将一个以前设置的Cookie送回服务器
Date	消息被发送的日期和时间
Connection	指示连接是否支持持续连接，值Keep-Alive表示支持持续连接

3. 检索HTTP请求头

- `public String getHeader(String name)`: 返回指定名称的请求头的值。
- `public Enumeration getHeaders(String name)`: 返回指定名称的请求头的Enumeration对象。
- `public Enumeration getHeaderNames()`: 返回一个Enumeration对象，它包含所有请求头名。
- `public int getIntHeader(String name)`: 返回指定名称的请求头的整数值。
- `public long getDateHeader(String name)`: 返回指定名称的请求头的日期值。

检索HTTP请求头示例ShowHeadersServlet.java

```
@WebServlet("/show-headers")
public class ShowHeadersServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<head><title>请求头信息</title></head>");
        out.println("服务器收到的请求头信息<p>");

        Enumeration<String> headers = request.getHeaderNames();
        while (headers.hasMoreElements()) {
            String header = (String) headers.nextElement();
            String value = request.getHeader(header);
            out.println(header + " = " + value + "<br>");
        }
        out.println("</body></html>");
    }
}
```

2.3.5 请求转发

- 在实际应用中可能需要将请求转发 (forward) 到其他资源。
- 使用 `ServletRequest` 接口中定义的方法，格式如下：

`RequestDispatcher getRequestDispatcher(String path)`

RequestDispatcher接口定义了两个方法

- `public void forward(ServletRequest request, ServletResponse response)`: 将请求转发到服务器上的另一个动态或静态资源（如Servlet、JSP页面或HTML页面）。
- `public void include(ServletRequest request, ServletResponse response)`: 将控制转发到指定的资源，并将其输出包含到当前输出中。

2.3.6 使用请求对象存储数据

`void setAttribute(String name, Object obj)`

`Object getAttribute(String name)`

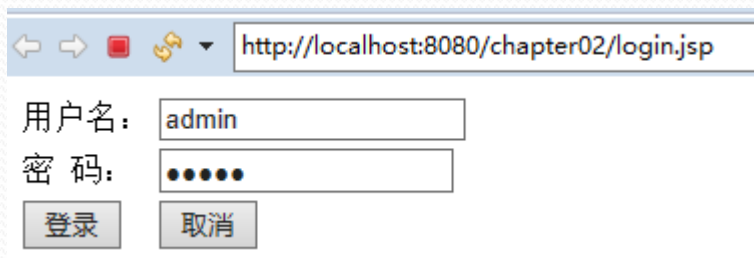
`void removeAttribute(String name)`

处理登陆的LoginServlet

```
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    public void doPost(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        // 用户名和口令均为admin, 认为登录成功
        if(username.equals("admin")&&password.equals("admin")){
            request.setAttribute("username", username);
            RequestDispatcher rd = request.getRequestDispatcher("/welcome.jsp");
            rd.forward(request, response);
        }else{
            RequestDispatcher rd = request.getRequestDispatcher("/login.html");
            rd.forward(request, response);
        }
    }
}
```

登陆成功页面welcome.jsp显示存储的数据

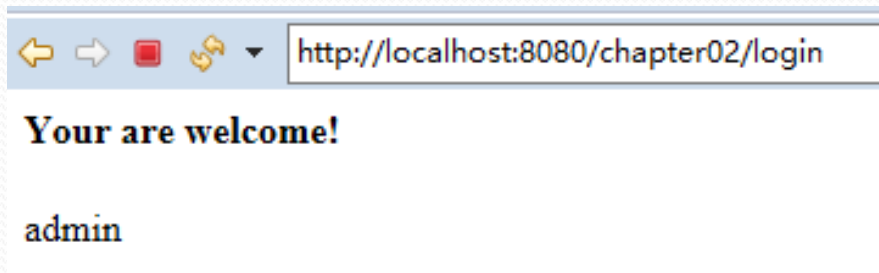
```
<html>  
<body>  
<h4>Your are welcome!</h4>  
${username}  
</body>  
</html>
```



http://localhost:8080/chapter02/login.jsp

用户名:

密 码:



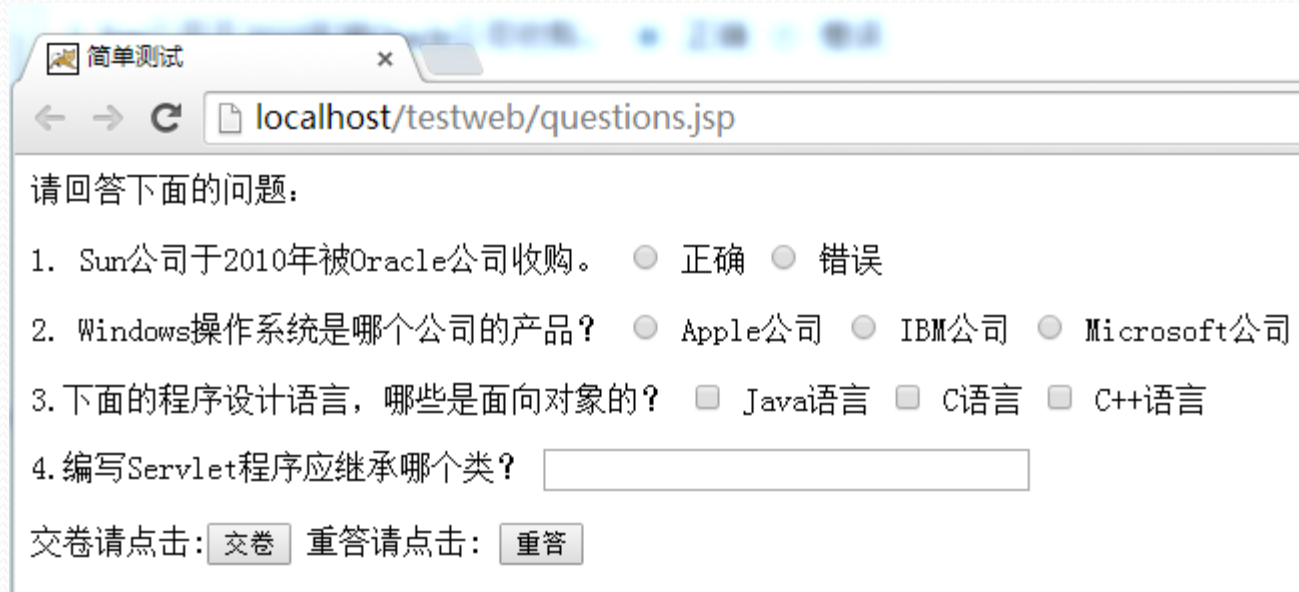
http://localhost:8080/chapter02/login

Your are welcome!

admin

2.3.7 实例：一个简单的考试系统

- 开发一个简单的考试系统，在JSP页面中建立一个表单，通过POST方法传递参数。
- 程序questions.jsp
- 程序SimpleTestServlet.java



简单测试

localhost/testweb/questions.jsp

请回答下面的问题：

1. Sun公司于2010年被Oracle公司收购。 ☐ 正确 ☐ 错误
2. Windows操作系统是哪个公司的产品？ ☐ Apple公司 ☐ IBM公司 ☐ Microsoft公司
3. 下面的程序设计语言，哪些是面向对象的？ ☐ Java语言 ☐ C语言 ☐ C++语言
4. 编写Servlet程序应继承哪个类？

交卷请点击： 重答请点击：

2.3.8 文件上传

- 文件上传是将客户端的一个或多个文件传输到服务器上保存。
- 实现文件上传首先需要在客户端的HTML页面中通过一个表单打开一个文件，然后提交给服务器。
- 上传文件表单的<form>标签中应该指定enctype属性，它的值应该为“multipart/form-data”，<form>标签的method属性应该指定为“post”，同时表单应该提供一个<input type="file">的输入域用于指定上传的文件。

客户端上传文件示例fileUpload.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>上传文件</title>
</head>
<body>
<form action="fileUpload.do" enctype="multipart/form-data" method="post">
  <table>
    <tr> <td colspan="2" align="center">文件上传</td>
    </tr>
    <tr><td>会员号: </td>
      <td><input type="text" name="mnumber" size="30" /></td>
    </tr>
    <tr> <td>文件名: </td>
      <td><input type="file" name="fileName" size="30" /></td>
    </tr>
    <tr>
      <td align="right"><input type="submit" value="提交" /></td>
      <td align="left"><input type="reset" value="重置"/></td>
    </tr>
  </table>
</form>
</body>
</html>
```

2.3.8 文件上传

- 在服务器端，可以使用请求对象的`getInputStream()`返回`ServletInputStream`输入流对象，文件内容就包含在该对象中，另外还包含表单域的名称和值、上传的文件名、内容类型等信息。

2.3.8 文件上传

- 当表单提交时，浏览器将表单各部分的数据发送到服务器端，每个部分之间使用分隔符分隔开。
- 通过请求对象的下面两个方法来处理上传的文件。
 - ① `public Part getPart(String name)`: 返回用name指定名称的Part对象。
 - ② `public Collection<Part> getParts()`: 返回所有Part对象的一个集合。

2.3.8 文件上传

- Part是Servlet 3.0 API新增的一个接口，定义在javax.servlet.http包中。它提供了下面的常用方法：
 - public InputStream **getInputStream()** throws IOException: 返回Part对象的输入流对象。
 - public String **getContentType()**: 返回Part对象的内容类型。
 - public String **getName()**: 返回Part对象的名称。
 - public long **getSize()**: 返回Part对象的大小。
 - public String **getHeader(String name)**: 返回Part对象指定的MIME头的值。
 - public Collection<String> **getHeaders(String name)**: 返回name指定的头值的集合。
 - public Collection<String> **getHeaderNames()**: 返回Part对象头名称的集合。
 - public void **delete()** throws IOException: 删除临时文件。
 - public void **write(String fileName)** throws IOException: 将Part对象写到指定的文件中。

服务器端处理上传文件的servlet示例

```
@WebServlet(name="FileUploadServlet",urlPatterns={"/fileUpload.do"})
@MultipartConfig(location="C:\\tools\\temp\\",fileSizeThreshold=1024)
public class FileUploadServlet extends HttpServlet{
    public void doPost(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException,IOException{
        // 返回Web应用程序文档根目录
        String path1 = this.getServletContext().getRealPath("/");
        System.out.println(path1);
        String path = "C:\\tools\\temp\\";
        String mnumber = request.getParameter("mnumber");
        Part p = request.getPart("fileName");

        String message="";
        if(p.getSize() >1024*1024){    // 上传的文件不能超过1MB大小
            p.delete();
            message = "文件太大，不能上传！ ";
        }else{
            path = path + "\\student\\" +mnumber;
            //System.out.println(path);
            File f = new File(path);
            if( !f.exists()){ // 若目录不存在，则创建目录
                f.mkdirs();
            }
            String h = p.getHeader("content-disposition");
            // 得到文件名
            String fname = h.substring(h.lastIndexOf("\\")+1, h.length()-1);
            p.write(path + "\\ "+ fname);
            message = "文件上传成功！ ";
        }
        request.setAttribute("message", message);
        RequestDispatcher rd = request.getRequestDispatcher("/fileUpload.jsp");
        rd.forward(request, response);
    }
}
```

2.3.8 文件上传

- 对实现文件上传的Servlet类必须使用 **@MultipartConfig** 注解，使用该注解告诉容器该Servlet能够处理multipart/form-data的请求。
- 使用 **@MultipartConfig** 注解，HttpServletRequest对象才可以得到表单数据的各部分。
- 使用 **@MultipartConfig** 注解可以配置容器存储临时文件的位置，文件和请求数据的大小限制以及阈值大小。

@MultipartConfig注解的常用元素

元素名	类 型	说 明
location	String	指定容器临时存储文件的目录位置
maxFileSize	long	指定允许上传文件的最大字节数
maxRequestSize	long	指定允许整个请求的multipart/form-data数据的最大字节数
fileSizeShreshold	int	指定文件写到磁盘后阈值的大小

2.4 发送响应

- 2.4.1 HTTP响应结构
- 2.4.2 理解ServletResponse
- 2.4.3 理解HttpServletResponse
- 2.4.4 发送状态码和错误消息

2.4.1 HTTP响应结构

- 由服务器向客户发送的HTTP消息称为HTTP响应（HTTP response）。
- 一个典型的HTTP响应消息

状态行	HTTP/1.1 200 OK
响应头	Date: Tue, 01 Sep 2004 23:59:59 GMT Content-Type: text/html Content-Length: 52
空行	
响应数据	<html> <body> <h1>Hello, John!</h1> </body></html>

1. 状态行与状态码

- 状态行由三部分组成，各部分由空格分隔：
 - HTTP版本
 - 说明请求结果的响应状态码
 - 描述状态码的短语
- HTTP/1.1 404 Not Found // 表示没有找到与给定的URI匹配的资源
- HTTP/1.1 500 Internal Error // 表示服务器检测到一个内部错误

2. 响应头

- 响应头是服务器向客户端发送的消息。

Date响应头表示消息发送的日期。

Content-Type响应头指定响应的内容类。

Content-Length指示响应内容的长度。

3. 响应数据

- 空行的后面是响应的数据。

```
<html><body>
```

```
    <h1>Hello, World!</h1>
```

```
</body></html>
```

2.4.2 输出流与内容类型

- Servlet使用输出流向客户发送响应。
- 在发送响应数据之前还需通过响应对象的 `setContentType()` 方法设置响应的内容类型。
- `public PrintWriter getWriter()`
- `public ServletOutputStream getOutputStream() throws IOException`
- `public void setContentType(String type)`

1. 使用PrintWriter

- PrintWriter对象被Servlet用来动态产生页面。调用响应对象的 `getWriter()` 方法返回 `PrintWriter`类的对象，它可以向客户发送文本数据。

```
PrintWriter out = response.getWriter();
```

2. 使用ServletOutputStream

- 如果要向客户发送二进制数据（如JAR文件），应该使用OutputStream对象。

```
ServletOutputStream sos = response.getOutputStream();
```

3. 设置内容类型

- 在向客户发送数据之前，一般应该设置发送数据的 MIME (Multipurpose Internet Mail Extensions) 内容类型。
- MIME 是描述消息内容类型的因特网标准
- `response.setContentType("text/html;charset=UTF-8");`

常见的MIME内容类型

类型名	含义
application/msword	Microsoft Word文档
application/pdf	Acrobat 的pdf文件
application/vnd.ms-excel	Excel 电子表格
application/vnd.ms-powerpoint	PowerPoint演示文稿
application/jar	JAR文件
application/zip	ZIP压缩文件
audio/midi	MIDI音频文件
image/gif	GIF图像
image/jpeg	JPEG图像
text/html	HTML文档
text/plain	纯文本
video/mpeg	MPEG视频片段

设置内容类型为Excel表格

- 通过将响应内容类型设置为“**application/vnd.ms-excel**”可将输出以Excel电子表格的形式发送给客户浏览器，这样客户可将结果保存到电子表格中。
- 输出内容可以用制表符分隔的数据或HTML表格数据等，并且还可以使用Excel内建的公式。下面的Servlet使用制表符分隔数据生成Excel电子表格。

输出Excel表格示例

```
@WebServlet(name = "ExcelServlet", urlPatterns = { "/excel.do" })  
  
public class ExcelServlet extends HttpServlet {  
    public void doGet(HttpServletRequest request, HttpServletResponse  
        response) throws ServletException, IOException {  
        // 设置响应的内容类型  
        response.setContentType("application/vnd.ms-excel;charset=gb2312");  
        PrintWriter out = response.getWriter();  
  
        out.println("学号\t姓名\t性别\t年龄\t所在系");  
        out.println("95001\t李勇\t男\t20\t信息");  
        out.println("95002\t刘晨\t女\t19\t数学");  
    }  
}
```


导出学生成绩示例

- 从文件中获取学生成绩，并导出


所有学生的成绩

学号	姓名	语文	数学	英语
2017268100101	张三	90	70	85
2017268100102	李四	85	75	80
2017268100103	王五	97	80	75
2017268100104	赵六	92	78	80


[导出所有学生成绩](#)

文件下载

你要打开还是保存此文件？

 名称: 学生成绩.xls
类型: Microsoft Excel 97-2003 工作表
来源: localhost

打开(O) 保存(S) 取消

 来自 Internet 的文件可能对你有所帮助，但某些文件可能危害你的计算机。如果你不信任其来源，请不要打开或保存该文件。[有何风险?](#)

- 程序ScoreSearch.java、ExportScore.java

```

@WebServlet("/scoreSearch.do")
public class ScoreSearch extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<html><head>");
        out.println("<title>所有学生的成绩</title>");
        out.println("</head><body>");
        out.println("<table width=400 border=1>");
        out.println("<h4>所有学生的成绩</h4>");
        out.println("<tr><td>学号</td><td>姓名</td><td>语文</td><td>数学</td><td>英语</td>");
        String str="";
        try{
            BufferedReader br = new BufferedReader(new
FileReader("C:\\java\\temp\\stuScore.txt"));
            while((str=br.readLine())!=null){
                String s[]=str.split("\t");
                out.println("<tr align=center>");
                for(int i=0;i<s.length;i++){
                    out.println("<td width=20%>"+s[i]+"</td>");
                }
                out.println("</tr>");
            }
            br.close();
        }catch(IOException e){ }
        out.println("</table>");
        out.println("<a href=\"exportScore.do\">导出所有学生成绩</a>");
        out.println("</body></html>");
    }
}

```

```
@WebServlet("/exportScore.do")
public class ExportScore extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setHeader("Content-Encoding", "gb2312");
        response.setHeader("Content-Disposition",
            "attachment; filename=" + java.net.URLEncoder.encode("学生成绩.xls", "UTF-8"));
        response.setContentType("application/vnd.ms-excel;charset=gb2312");

        PrintWriter out = response.getWriter();
        out.println("学号\t姓名\t语文\t数学\t英语");
        String str = "";
        try {
            BufferedReader br = new BufferedReader(new
                FileReader("C:\\java\\temp\\stuScore.txt"));
            while ((str = br.readLine()) != null) {
                System.out.println(str);
                String s[] = str.split(" ");
                for (int i = 0; i < s.length; i++) {
                    out.print(s[i] + "\t");
                }
                out.println("");
            }
            br.close();
        } catch (IOException e) { }
    }
}
```

2.4.3 设置响应头

- 响应头是随响应数据一起发送到浏览器的附加信息。
- `public void setHeader(String name, String value)`
- `public void setIntHeader(String name, int value)`
- `public void setDateHeader(String name, long date)`
- `public void addIntHeader(String name, int value)`
- `public void addDateHeader(String name, long date)`

典型的响应头名及其用途

响应头名称	说明
Date	指定服务器的当前时间
Expires	指定内容被认为过时的时间
Last-Modified	指定文档被最后修改的时间
Refresh	告诉浏览器重新装载页面
Content-Type	指定响应的内容类型
Content-Length	指定响应的内容的长度
Content-Disposition	为客户指定将响应的内容保存到磁盘上的名称
Content-Encoding	指定页面在传输过程中使用的编码方式

2.4.3 设置响应头

- ShowTimeServlet通过设置Refresh响应头实现每5秒钟刷新一次页面。

程序ShowTimeServlet.java

- 要告诉浏览器在5秒钟后跳转到http://host/path页面，也可以使用下面语句。

```
response.setHeader("Refresh","5;URL=http://host/path/");
```

2.4.3 设置响应头

- 在HTML页面中通过在<head>标签内添加下面代码也可以实现页面跳转功能。
- `<meta http-equiv="Refresh" content="5;URL= Login.html">`

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta http-equiv="Refresh" content="5;URL= Login.html">
<title>页面跳转</title>
</head>
<body>
页面在5秒后跳转
</body>
</html>
```

2.4.4 响应重定向

- Servlet可能决定不直接向浏览器发送响应，而是将响应重定向到其他资源。

public void sendRedirect(String location)

- location为指定的新的资源的URL，该URL可以是绝对URL（如http://www.microsoft.com），也可以是相对URL。若路径以“/”开头，则相对于服务器根目录（如，/helloweb/login.html），若不以“/”开头，则相对于Web应用程序的文档根目录（如，login.jsp）。
- 程序RedirectServlet.java

2.4.4 响应重定向

- 关于 `sendRedirect()` 方法，应该注意如果响应被提交，即响应头已经发送到浏览器，就不能调用该方法，否则将抛出 `java.lang.IllegalStateException` 异常。

```
PrintWriter out = response.getWriter();
```

```
out.println("<html><body>Hello
```

```
World!</body></html>");
```

```
out.flush();           // 响应在这一点被提交了
```

```
response.sendRedirect("http://www.baidu.com");
```

2.4.5 发送状态码和错误消息

- 服务器向客户发送的响应的第一行是状态行，它由三部分组成：HTTP版本、状态码和状态码的描述信息，如下是一个典型的状态行：

HTTP/1.1 200 OK

- 由于HTTP的版本是由服务器决定的，而状态的消息与状态码有关，因此，在Servlet中一般只需要设置状态码。

2.4.5 发送状态码和错误消息

- 状态码200是系统自动设置的，Servlet不需要指定该状态码。对其他状态码，可以由系统自动设置，也可用响应对象的setStatus()方法设置，该方法的格式为：

```
public void setStatus (int sc)
```

- 可以设置任意的状态码。参数sc表示要设置的状态码
- 对于404状态码，其消息为Not Found，
HttpServletResponse接口中为该状态码定义的常量名为SC_NOT_FOUND。

2.4.5 发送状态码和错误消息

- 在HTTP协议1.1版中定义了若干状态码，这些状态码由3位整数表示，一般分为5类

状态码范围	含 义	示 例
100~199	表示信息	100表示服务器同意处理客户的请求
200~299	表示请求成功	200表示请求成功，204表示内容不存在
300~399	表示重定向	301表示页面移走了，304表示缓存的页面仍然有效
400~499	表示客户的错误	403表示禁止的页面，404表示页面没有找到
500~599	表示服务器的错误	500表示服务器内部错误，503表示以后再试

2.4.5 发送状态码和错误消息

- HTTP为常见的错误状态定义了状态码，这些错误状态包括：资源没有找到、资源被永久移动以及非授权访问等。所有这些代码都在接口

`HttpServletResponse`中作为常量定义。

- `HttpServletResponse`也提供了 `sendError()` 方法用来向客户发送状态码，该方法有两个重载的形式，如下所示。

```
public void sendError (int sc)
```

```
public void sendError (int sc, String msg)
```

2.4.5 发送状态码和错误消息

- 第一个方法使用一个状态码，第二个方法同时指定显示消息。服务器在默认情况下创建一个HTML格式的响应页面，其中包含指定的错误消息。
- 例如，如果Servlet发现客户不应访问其结果，它将调用 `sendError (HttpServletResponse.SC_UNAUTHORIZED)`
- 程序StatusServlet.java

作业

- 1、用html和servlet编程实现输入三角形的三个边长并计算和输出三角形的面积。 要求如下：
 - (1) 编写一个input.html页面，页面中包括输入三个边长和提交按钮的表单。
 - (2) 编写一个文件名为TriangleServlet.java的Servlet，其URL为/computeTriangleArea.do，响应来自input.html的请求。如果能够构成三角形，则将计算得到的面积构成字符串“三角形面积=XXX”并在页面输出，要求面积保留2位小数，否则输出“三条边长无法构成三角形”，如果输入的边长为非数字型或不是正数，则在页面输出“输入的边长有误！”。

作业

- 2、JSON(JavaScript Object Notation, JS 对象表示法) 是一种轻量级的数据交换格式, 由于其易于人阅读和编写, 同时也易于机器解析和生成, 已广泛应用于各业务系统接口的数据交换。JSON是一个标记符的序列, 本质是一个字符串。JSON值可以是对象、数组、数字、字符串或者三个字面值(false、null、true)中的一个, 对象由花括号括起来的逗号分割的成员构成, 成员是字符串键和上文所述的值由逗号分割的键值对组成, 如:
{"name": "John Doe", "age": 18, "address": {"country": "china", "zip-code": "10000"}}。数组是由方括号括起来的一组值构成, 如[3, 1, 4, 1, 5, 9, 2, 6]。目前, Java程序一般采用阿里巴巴的FastJSON、谷歌的GSON、SpringMVC内置的解析器jackson等第三方jar包实现JSON串的生成(序列化)和解析(反序列化)。但我们也可以通过处理字符串的方式来处理JSON串。假设某成绩系统导出Java程序设计的成绩为JSON格式, 保存为c:/temp/ javascore.json, 请你编写文件解析的web应用程序, 将该文件通过uploadfile.html上传, 由映射地址为parsejson.do的ParseJsonServlet解析并以表格方式在页面进行显示, 表格下方有超链接“导出excel”可将解析的数据导出为excel文件。

(1) javascore.json 文件格式如下:

```
[  
  {  
    "stuid": "201826630601",  
    "name": "王明",  
    "courseName": "java程序设计",  
    "score": 90  
  },  
  {  
    "stuid": "201826630602",  
    "name": "张三",  
    "courseName": "java程序设计",  
    "score": 80  
  },  
  .....  
  {  
    "stuid": "201826630604",  
    "name": "王五",  
    "courseName": "java程序设计",  
    "score": 85  
  }  
]
```

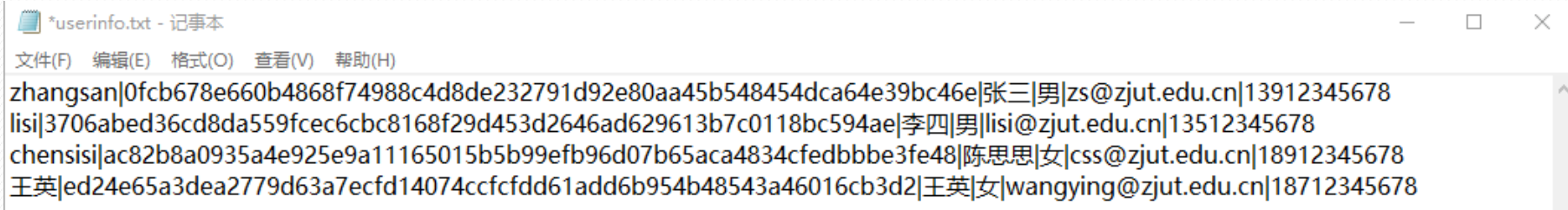
(2) 导出的 excel 文件格式如下:

A 序号	B 学号	C 姓名	D 课程名称	E 成绩
1	201826630601	王明	java程序设计	90
2	201826630602	张三	java程序设计	80
3	201826630603	李四	java程序设计	95
4	201826630604	王五	java程序设计	85

选做: 列出的成绩清单前面有复选框, 可选中其中某几个学生的成绩, 可对选中的成绩进行下载导出 excel 文件。

作业

- 3、实现一个简单的企业会员注册和登录功能。具体要求如下：
- (1) 注册页面register.html包括用户名、密码、姓名、性别、邮箱、手机号等信息，要求对所有字段进行非空判断，密码为数字、字母和特殊字符的组合（不少于8位），对邮箱和手机号进行有效性判断，否则弹出对话框进行提示。注册信息处理由映射地址为register.do的RegisterServlet.java进行处理，将所有用户注册的登录名、密码、姓名、性别、邮箱、手机号等信息按行保存到userinfo.txt中，每个字段用“|”分隔，要求将密码以SHA256加密（可用JDK自带的java.security.MessageDigest实现）后写入文件，userinfo.txt文件格式如下：
- 如果当前注册的用户名不与已注册的用户名冲突，则提示“注册成功”，5秒后自动跳转或有超链接至登录页面，否则提示“你的用户名已被注册，请返回重新注册”。
- (2) 登录页面login.html。用户登录页面需判断用户名或密码是否为空，登录成功与否由映射地址为login.do的LoginServlet.java处理，该Servlet从userinfo.txt查询判断用户名与密码是否一致，如果一致则跳转至欢迎页面welcome.html提示“登录成功！”，否则跳转至失败页面failed.html提示“用户名或密码错误，请重新登录”，其中“重新登录”为超链接，可以跳转至登录页面login.html。



```
*userinfo.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
zhangsan|0fcb678e660b4868f74988c4d8de232791d92e80aa45b548454dca64e39bc46e|张三|男|zs@zjut.edu.cn|13912345678
lisi|3706abed36cd8da559fcec6cbc8168f29d453d2646ad629613b7c0118bc594ae|李四|男|lisi@zjut.edu.cn|13512345678
chensisi|ac82b8a0935a4e925e9a11165015b5b99efb96d07b65aca4834cfedbbbe3fe48|陈思思|女|css@zjut.edu.cn|18912345678
王英|ed24e65a3dea2779d63a7ecfd14074ccfcfd61add6b954b48543a46016cb3d2|王英|女|wangying@zjut.edu.cn|18712345678
```

注册登录页面效果

用户注册

用 户 名

密 码

姓 名

性 别 ☒男 ☐女

邮 箱

手 机 号

[立即注册](#)

<http://localhost:8080/web2020/hw2/register.do>

注册成功! 去[登录](#)

企业会员登录

用 户 名

密 码

[立即登录](#)

没有帐号? [立即注册](#)

<http://localhost:8080/web2020/hw2/login.do>

登录成功!

<http://localhost:8080/web2020/hw2/login.do>

用户名或密码错误, 请[重新登录](#)

Thank You !