**servlet_part3.txt**

| | | | |
|---|---|---|---|
| **笔记本:** | 杰普实训三_笔记 | | |
| **创建时间:** | 2018/8/18 8:02 | **更新时间:** | 2018/8/18 17:54 |
| **作者:** | 家龙 | | |
| **URL:** | file://D:\资料\Desktop\servlet_part3.txt | | |

- 1.servlet中接收客户端传的参数

浏览器地址栏中直接输入url进行传参
http://127.0.0.1:8989/jd1617_servlet/ParamServlet?name=tom&age=20&like=0&like=1

如果只有一个参数键值对，则在URL后面用拼接?参数名=参数值
如果是有多个参数键值对，则第一个参数对使用？拼接，其余的参数对以&拼接
即：协议://ip:port/资源路径?参数名=参数值&参数名=参数值&参数名=参数值

同时如果是超链接也可以这样传参:

```
<a href="ParamServlet?name=tom&age=20&like=0&like=1">传参测试</a>
```

同时表单也可以这样传参:

```
<form action="ParamServlet" method="post">
    用户名:<input type="text" name="name"><br>
    年龄:<input type="text" name="age"><br>
    爱好:<input type="checkbox" name="like" value="0">篮球
        <input type="checkbox" name="like" value="1">足球
        <input type="checkbox" name="like" value="2">排球
        <br>
        <input type="submit" value="提交">
</form>
```

1.1 getParameter接收单一的参数,根据指定参数名获取参数值，如果多个参数值对应一个参数名，只取第一个。

```
String name = request.getParameter("name");
String age = request.getParameter("age");
```

1.2 getParameterValues一个参数对应多个值，如多选框，多个参数值对应一个参数名

```
String[] like = request.getParameterValues("like");
```

1.3 getParameterNames获得客户端本次传参中的所有参数名

```
Enumeration<String> names = request.getParameterNames();
while(names.hasMoreElements()){
```

```
            String name = names.nextElement();
            System.out.println(name);
        }
```

2.4 getParameterMap获取所有参数封装到Map中，其中key为参数名，value为参数值，因为一个参数名称可能有多个值，所以参数值是String[]，而不是String。

```
    Map<String, String[]> map = request.getParameterMap();
        for(String key:map.keySet()){
            System.out.println(key+" : "+Arrays.toString(map.get(key)));
        }
```

Get与Post传参：

GET请求：

请求参数会在浏览器的地址栏中显示，所以不安全；

请求参数长度有一定的长度限制(浏览器决定，而且是限制的是整个 URI 长度，而不仅仅是参数值数据长度，不同浏览器不同版本支持的长度不同)；

GET请求没有请求体，无法通过request.setCharacterEncoding()来设置参数的编码；

以下方式发送的请求都是GET请求

1.超链接

2.直接在浏览器地址栏输入回车

3.form表单上Method指明为GET

POST请求：

请求参数不会显示浏览器的地址栏，相对安全；

请求参数长度取决于服务器；

一般在常见的方式是在Form表单上指明method为POST。

所谓的请求长度限制是由浏览器和 web 服务器决定和设置的，各种浏览器和 web 服务器的设定均不一样，这依赖于各个浏览器厂家的规定或者可以根据 web 服务器的处理能力来设定。

- 2.中文参数乱码

2.1 get方式传参,servlet接收中文乱码

修改tomcat中的配置server.xml

在修改端口的标签中添加属性URIEncoding="XXX"

```
<Connector URIEncoding="UTF-8" connectionTimeout="20000" port="8888" protocol="HTTP/1.1" redirectPort="8443"/>
```

2.2 post方式传参,servlet接中文乱码

获取参数【之前】,先设置一下request中的编码:

```
request.setCharacterEncoding("UTF-8");
```

此设置只对Post方式有效。

2.3 servlet中使用io流给浏览器写回数据,浏览器显示中文乱码

在response获得out输出流之前,我们需要设置一下这个输出流是用什么编码来输入内容。

response.setCharacterEncoding("UTF-8");

默认情况下浏览器会采用中文简体(GBK)来解析响应正文

我们可以在servlet设置响应的头部,来通知浏览器本次响应正文中的内容编码是什么，以让浏览器采用某种解码方式。

response.setContentType("text/html;charset=utf-8");

注意:响应内容的整体格式(格式的控制由tomcat负责)

1.响应状态行
2.消息报头/响应头部
3.\r\n
4.响应正文

给浏览器传输的内容都在响应正文中放着。

- 3.servlet请求跳转

之前我们接收到一个请求，要返回一个html页面，我们是使用的IO流的形式在响应中写出去。这些代码格式都是一样的，我们可以用servlet的请求转发来实现。

3.1服务器内部跳转（请求转发）：

所谓服务器内部跳转就是由服务器在执行完一个servlet之后，在继续访问下一个资源。

特征：

1.客户端只发出一次请求
2.服务器内部进行跳转的时候会把本次请求继续往下传递
3.客户端浏览器的地址栏中显示的是第一次访问的那个servlet的访问路径(相当于浏览器中地址栏的地址不会发生变化)
4.服务器内部跳转需要使用request来完成。

操作：

服务器内部跳转主要借助于RequestDispatccher对象。

RequestDispatcher dispatcher = request.getRequestDispatcher("资源路径");

获得一个指顶Web资源的包装器。资源可以使页面也可以是其他的servlet。

通过RequestDispatcher进行跳转的方式有两种:

1.forward(request,response)

会清空response里边的信息.

dispacher.forword(request,response);

2.include(request,response)

不会清空。而是将要跳转到的资源的信息包含到当前response里边，进行追加

dispacher.include(request,response);

注：WEB-INF目录中的内容，要使用服务器内部跳转访问。

3.2客户端重定向：

重定向是指页面重新定位到某个新地址，之前的请求失效，进入一个新的请求，且跳转后浏览器地址栏内容将变为新的指定地址

特征：

1.假设浏览器发请求访问我们的一个servlet,然后这个servlet里面又完成了客户端重定向,重定向到另一个资源中(可能是页面也能servlet),那么浏览器的地址栏中显示的是重定向到的那个资源的地址,浏览器的窗口中显示的重定向到的那个资源信息

2.每次进行客户端重定向,都会是一个全新的request和response。

3.客户端重定向需要使用response来完成。

操作：

response.sendRedirect("资源路径")

2.请求跳转

内部跳转

特点：地址栏地址不发生改变，请求只发送一次，请求始终是同一个

forward:response清空

include:response追加

客户端重定向：

特点：地址栏地址发生改变，请求发送多次，请求不是同一个

- 4.请求路径：

4.1.相对路径：

请求路径不以"/"开头的，是相对路径，相对路径的相对点是指当前请求所在的路径下。

如：通过http://localhost:8888/jd1704Servlet/index.html 访问到的页面，里面有一个超链接或者表单，请求路径为“ServletB”，则相当于相对当前路径，即http://localhost:8888/jd1704Servlet/ 发送请求：http://localhost:8888/jd1704Servlet/ServletB。

如果是在请求内部跳转结束后，浏览器页面上的静态资源，如果使用相对路径的话，则是相对当前浏览器上显示的地址。

4.2绝对路径

绝对路径是指以"/"开头的路径，“/”在不同位置表示的含义不同

1) 前台页面 “/”代表服务器根目录

/ 表示：http://ip:port/

```
<a href="url"></a>
<form action="url"></form>
<img src="url" alt="">
<link rel="stylesheet" href="url">
resp.sendRedirect("/");
```

2) 后台servlet中

如果是内部跳转： “/”代表项目根目录

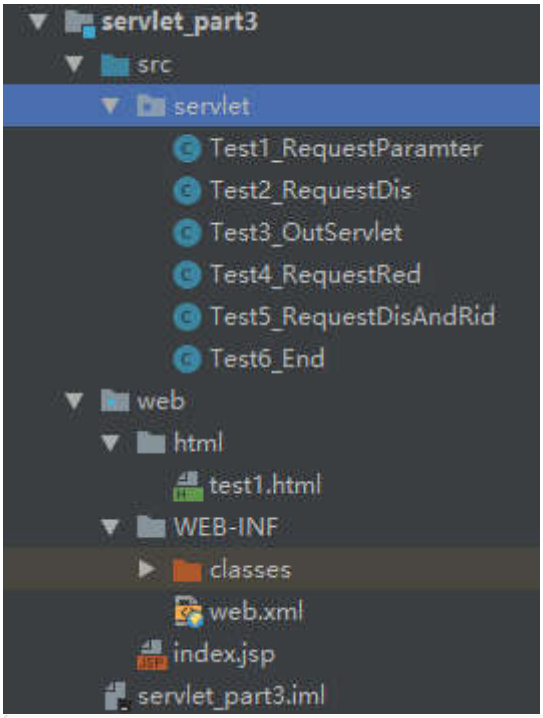
/ 表示： http://ip:port/jd1704Servlet

```
req.getRequestDispatcher("/.....")
```

如果是服务器重定向： “/”代表服务器根目录

```
resp.sendRedirect("/....");
```

---




```
package servlet;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Arrays;
import java.util.Enumeration;
import java.util.Map;
import java.util.Set;

/**
```

```
* Created by TjI on 2018/8/18 8:41.
* 测试:请求参数的获取
*/
public class Test1_RequestParamter extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("接收到GET请求");
        //编码问题【路径，编码】
        //POST方式中文乱码
        req.setCharacterEncoding("utf-8");

        //1.获取单一key的参数名获取 对应的请求参数
        String sid = req.getParameter("sid");
        String username = req.getParameter("username");
        String password = req.getParameter("password");
        String time = req.getParameter("time");
        String hobby = req.getParameter("hobby");
        String address = req.getParameter("address");
        String self = req.getParameter("self");

        System.out.println("sid: "+sid);
        System.out.println("username: "+username);
        System.out.println("password: "+password);
        System.out.println("time: "+time);
        System.out.println("hobby: "+hobby);
        System.out.println("address: "+address);
        System.out.println("self: "+self);

        System.out.println("2-------------------------");
        //2.相同key 对应不同value，根据一个key值获取所有匹配的参数值
        String[] hobbies = req.getParameterValues("hobby");
        System.out.println(Arrays.toString(hobbies));

        System.out.println("3---------------------------");
        //3.获取所有的参数的key值
        Enumeration<String> parameterNames = req.getParameterNames();
        while (parameterNames.hasMoreElements()) {
            String key = parameterNames.nextElement();
            System.out.println(key+": "+Arrays.toString(req.getParameterValues(key)));
        }

        System.out.println("4---------------------------");
        //4.获取所有的参数键值对key:参数键值对的键名，value:与之对应的所有参数值组成的字符串数组
        Map<String, String[]> parameterMap = req.getParameterMap();
        Set<String> paramKeys = parameterMap.keySet();
        for (String paramKey : paramKeys) {
            System.out.println(paramKey+" : "+Arrays.toString(req.getParameterValues(paramKey)));
        }


        PrintWriter writer = resp.getWriter();
        writer.println("requestparamter.servlet");
        writer.flush();
        writer.close();


    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("接收到POST请求");
        doGet(req, resp);
    }
}
```

```
package servlet;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.*;

/**
* Created by TjI on 2018/8/18 9:54.
* 测试2：测试请求跳转
*/
public class Test2_RequestDis extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("接收到请求");
        System.out.println("可以获取参数，封装数据");
        System.out.println("如果这个请求需要访问数据库，思考在三层架构");
        System.out.println("假设数据处理代码有500行，现在处理结束了");
        System.out.println("返回响应--html页面");
        //怎么返回一个html页面
        resp.setContentType("text/html;charset=utf-8");
        //1.边看边写
        //2.边读边写
//        String realPath = getServletContext().getRealPath("/html/test1.html");
//        System.out.println(realPath);
//        FileInputStream fileInputStream = new FileInputStream(new File(realPath));
//        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(fileInputStream));
//        String line = "";
//        PrintWriter writer = resp.getWriter();
//        while ((line = bufferedReader.readLine()) != null) {
//            writer.print(line);
//        }
//        writer.flush();
//        writer.close();
        //3.跳转
        /*
        forward:response清空
        include:response追加
         */
//        req.getRequestDispatcher("html/test1.html").forward(req,resp);//清空
//        req.getRequestDispatcher("html/test1.html").include(req,resp);//追加
        //请求跳转到另一个请求中
        PrintWriter writer = resp.getWriter();
        writer.println("hello");
        req.getRequestDispatcher("OS.servlet").forward(req,resp);

    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        doPost(req, resp);
    }
}
```

```
package servlet;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

/**
* Created by TjI on 2018/8/18 10:17.
* 访问Test2OUtServeIt:请求从Test2_RequestDis中跳转过来
*/
public class Test3_OutServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("OutServlet");
        System.out.println("重定向之后: "+ req.getParameter("name"));
        PrintWriter writer = resp.getWriter();
        writer.println("Test3_OutServlet");
    }
}
```

```
package servlet;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
* Created by TjI on 2018/8/18 10:30.
* 重定向
*/
public class Test4_RequestRed extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("Test4_RequestRed");
        System.out.println("重定向之前: "+ req.getParameter("name"));
        //重定向
        resp.sendRedirect("OS.servlet");
    }
}
```

```
package servlet;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
* Created by TjI on 2018/8/18 10:44.
*/
public class Test5_RequestDisAndRid extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("接收到get请求");
        //服务器内部跳转: /:项目根目录http://localhost:8080/servlet_part3
//        req.getRequestDispatcher("day3/E.servlet").forward(req,resp);
//        req.getRequestDispatcher("/day3/E.servlet").forward(req,resp);

        //客户端重定向: /:服务器根目录http://localhost:8080/
```

```
//        resp.sendRedirect("day3/E.servlet");//404
        resp.sendRedirect("/day3/E.servlet");
    }


    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("接收到post请求");
        doPost(req, resp);
    }
}
```

```
package servlet;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
* Created by TjI on 2018/8/18 10:49.
*/
public class Test6_End extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        //服务器内部跳转:项目名/day3/...
        req.getRequestDispatcher("../html/test1.html").forward(req,resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        doGet(req, resp);
    }
}
```

# test1.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Test1</title>
</head>
<body>
    <!--http://127.0.0.1:8080/servlet/html/test1.html-->
    <!--http://127.0.0.1:8080/servlet/html/reqparam.servlet-->
    <!--相对路径:相对当前请求/最近的一次请求-->
    <!--a请求方式:GET-->
    <a href="../reqparam.servlet">超链接GET方式发送请求</a>
    <!--表单控件中的值 name的值作为key,value的值作为value ?key=value&key=value-->
    <form action="../reqparam.servlet" method="get">
        <input type="hidden" name="sid" value="10086"><br>
        <input type="text" name="username" value="张三"><br>
        <input type="password" name="password" value="123456"><br>

        <input type="radio" name="time" value="AM">
        <input type="radio" name="time" value="PM" checked="checked"><br>

        <input type="checkbox" name="hobby" value="篮球">
        <input type="checkbox" name="hobby" value="网球">
        <input type="checkbox" name="hobby" value="台球"><br>
```

```
    <select name="address">
        <option value="10">小店</option>
        <option value="11" selected>尖草坪</option>
    </select><br>

    <textarea rows="3" cols="10" name="self">自我介绍</textarea><br>

    <input type="submit" value="提交GET方式">

</form>

<hr>
<form action="../reqparam.servlet" method="post">
    <input type="hidden" name="sid" value="10086"><br>
    <input type="text" name="username" value="张三"><br>
    <input type="password" name="password" value="123456"><br>

    <input type="radio" name="time" value="AM">
    <input type="radio" name="time" value="PM" checked="checked"><br>

    <input type="checkbox" name="hobby" value="篮球">
    <input type="checkbox" name="hobby" value="网球">
    <input type="checkbox" name="hobby" value="台球"><br>

    <select name="address">
        <option value="10">小店</option>
        <option value="11" selected>尖草坪</option>
    </select><br>

    <textarea rows="3" cols="10" name="self">自我介绍</textarea><br>

    <input type="submit" value="提交POST方式">

</form>


</body>
</html>
```

## web.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaeehttp://xmlns.jcp.org/xml/ns/javaee/web-
app_4_0.xsd"
version="4.0">
<servlet>
<servlet-name>Test1_RequestParamter</servlet-name>
<servlet-class>servlet.Test1_RequestParamter</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>Test1_RequestParamter</servlet-name>
<url-pattern>/reqparam.servlet</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>Test2_RequestDis</servlet-name>
<servlet-class>servlet.Test2_RequestDis</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>Test2_RequestDis</servlet-name>
<url-pattern>/RD.servlet</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>Test3_OutServlet</servlet-name>
<servlet-class>servlet.Test3_OutServlet</servlet-class>
</servlet>
```

```
<servlet-mapping>
<servlet-name>Test3_OutServlet</servlet-name>
<url-pattern>/OS.servlet</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>Test4_RequsetRed</servlet-name>
<servlet-class>servlet.Test4_RequestRed</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>Test4_RequsetRed</servlet-name>
<url-pattern>/RR.servlet</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>Test5_RequestDisAndRid</servlet-name>
<servlet-class>servlet.Test5_RequestDisAndRid</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>Test5_RequestDisAndRid</servlet-name>
<url-pattern>/RDAR.servlet</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>Test6_End</servlet-name>
<servlet-class>servlet.Test6_End</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>Test6_End</servlet-name>
<url-pattern>/day3/E.servlet</url-pattern>
</servlet-mapping>
</web-app>
```