Author：相忠良
Email: ugoood@163.com

起始于：April 18, 2018
最后更新日期：April 22, 2018

声明：本笔记依据传智播客方立勋老师 **Java Web** 的授课视频内容记录而成，中间加入了自己的理解。本笔记目的是强化自己学习所用。若有疏漏或不当之处，请在评论区指出。谢谢。
涉及的图片，文档写完后，一次性更新。

rw 读写模板的设置

在编译的时候只有写出rw之后使用alt+/就可以将模板代码全部展现出来。

rw读写模板代码：

```
InputStream in = null;
OutputStream out = null;
try {
  in = new FileInputStream(path);
  int len = 0;
  byte[] buffer = new byte[1024];
  out = new FileOutputStream("");
  while ((len = in.read(buffer)) > 0) {
    out.write(buffer, 0, len);
  }
} finally {
  if (in != null) {
    try {
      in.close();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  if (out != null) {
    try {
      out.close();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

MyEclipse设置模板方式：http://www.jb51.net/softjc/486530.html

day05 Request Response

Web服务器收到客户端的http请求，会针对每一次请求，分别创建一个用于代表请求的 request 对象，和代表响应的 response 对象：

- 要想获取客户机提交过来的数据，只需找 request 对象；
- 要向客户机输出数据，只需找 response 对象就行了。

1. HttpServletResponse 简介

`HttpServletResponse`响应，它封装了向客户机 发送数据、发送响应头和发送响应状态码 的方法。例如：

```
setStatus(int)
setHeader(String, String)
getWriter()
getOutputStream()
```

1.1 Response 的 OutputStream 输出中文的问题

程序已什么码表输出了，程序就一定要控制浏览器以什么码表打开。

1.用 响应头 的方式控制浏览器的码表，如下：

```
// servlet 中用 OutputStream 输出中文的问题
public class ResponseDemo1 extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        // 通过response对象为浏览器指定码表
        // 程序已什么码表输出了，程序就一定要控制浏览器以什么码表打开
        response.setHeader("Content-type", "text/html;charset=UTF-8");

        String data = "中国";
        OutputStream out = response.getOutputStream();
        out.write(data.getBytes("UTF-8")); // 以该码表输出
    }
}
```

2.用 html 的`<meta>`方式控制浏览器的码表，此方法 没有 向浏览器发送响应头。如下(我试验失败)：

```
public void doGet(HttpServletRequest request, HttpServletResponse respons
e)
            throws ServletException, IOException {

        String data = "中国";
        OutputStream out = response.getOutputStream();

        out.write("<meta http-equiv='content-type' content='text/html;cha
rset=UTF-8'>"
                .getBytes());
        out.write(data.getBytes("UTF-8"));
}
```

1.2 Response 的 Writer 输出中文(只能写字符串)

```
public class ResponseDemo2 extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        // 设置 response 码表
        response.setCharacterEncoding("UTF-8");
        // 设置浏览器应查的码表
        response.setHeader("content-type", "text/html;charset=UTF-8");

        String data = "中国";
        PrintWriter out = response.getWriter();
        out.write(data);
    }
}
```

上述代码设置浏览器码表，也可这样写：`response.setContentType("text/html;charset=UTF-8");`，且这句话也修改了 response 的码表。

2. response 实现文件下载

准备：项目`WebRoot`目录下建立`download`目录，并在该目录中预先放入`1.jpg`和`日本妞.jpg`文件。
分别下载这两个文件的代码：
注意：如果文件名为中文，则文件名需经过 **url** 编码，下面代码中有体现：

```java
// 以下载方式打开
public class ResponseDemo3 extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

//      String path = this.getServletContext().getRealPath("/download/1.j
pg");
        String path = this.getServletContext().getRealPath("/download/日本
妞.jpg");
        String filename = path.substring(path.lastIndexOf("\\") + 1);

        // 发送响应头，通知浏览器以下载方式打开，且文件名是 filename
//      response.setHeader("content-disposition", "attachment;filename="
//              + filename);

        // 如果文件名为中文，则文件名需经过 url 编码
        response.setHeader("content-disposition", "attachment;filename="
                + URLEncoder.encode(filename, "UTF-8"));

        InputStream in = null;
        OutputStream out = null;
        try {
            in = new FileInputStream(path);
            int len = 0;
            byte[] buffer = new byte[1024];
            out = response.getOutputStream();
            while ((len = in.read(buffer)) > 0) {
                out.write(buffer, 0, len);
            }
        } finally {
            if (in != null) {
                try {
                    in.close();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
            if (out != null) {
                try {
                    out.close();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

3. response 输出随机认证码图片 - 涉及了控制浏览器不留缓存 - 常用汉字码表

刷新按钮作用：不管服务器有没有缓存，都向服务器发请求。就是把上一次的申请再干一次。
下面代码涉及了 控制浏览器不留缓存。

```java
// 验证码， 谷歌浏览器不支持，IE支持本程序
public class ResponseDemo4 extends HttpServlet {

    // ctrl + shift + x 小写变大写
    public static final int WIDTH = 120;
    public static final int HEIGHT = 35;

    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        BufferedImage image = new BufferedImage(WIDTH, HEIGHT,
                BufferedImage.TYPE_INT_RGB);

        Graphics g = image.getGraphics();

        // 1. 设置背景色
        setBackGround(g);

        // 2. 设置边框
        setBorder(g);

        // 3. 画干扰线
        drawRandomLine(g);

        // 4. 写随机数
        drawRandomNum((Graphics2D) g);

        // 5. 图形写给浏览器
        response.setContentType("image/jpeg");
        // 发头控制浏览器不要缓存
        response.setDateHeader("expries", -1);
        response.setHeader("Cache-Control", "no-cache");
        response.setHeader("Pragma", "no-cache");

        ImageIO.write(image, "jpg", response.getOutputStream());
    }

    private void drawRandomNum(Graphics2D g) {
        g.setColor(Color.RED);
        g.setFont(new Font("宋体", Font.BOLD, 20));

        // unicode 汉字码表范围 [\u4e00-\u9fa5]
        // 常用汉字
        String base = "去网上搜unicode常用汉字码表，粘贴过来即可";

        int x = 5; // 从这开始写
        for (int i = 0; i < 4; i++) {

            // -30 -- 30度旋转设定
```

```
            int degree = new Random().nextInt() % 30;

            String ch = base.charAt(new Random().nextInt(base.length
())) + "";
            g.rotate(degree * Math.PI / 180, x, 20); // 设置旋转弧度
            g.drawString(ch, x, 20);
            g.rotate(-degree * Math.PI / 180, x, 20);
            x += 30;
        }
    }

    private void drawRandomLine(Graphics g) {
        g.setColor(Color.GREEN);
        // 4-5条干扰线
        for (int i = 0; i < 5; i++) {

            int x1 = new Random().nextInt(WIDTH);
            int y1 = new Random().nextInt(HEIGHT);

            int x2 = new Random().nextInt(WIDTH);
            int y2 = new Random().nextInt(HEIGHT);

            g.drawLine(x1, y1, x2, y2);
        }
    }

    private void setBorder(Graphics g) {
        g.setColor(Color.BLUE);
        g.drawRect(1, 1, WIDTH - 2, HEIGHT - 2);
    }

    private void setBackGround(Graphics g) {
        g.setColor(Color.WHITE);
        g.fillRect(0, 0, WIDTH, HEIGHT);
    }
}
```

4. JavaScript 编码 点验证码换一张图片

涉及2文件：`src/ResponseDemo4`(上节有了)和`WebRoot/register.html`。

register.html代码如下：
我实践的结果是，小手 `cursor:hand` 在 **IE** 和 **Chrome** 下都不显示。

```
<!DOCTYPE html>
<html>
<head>
<title>register.html</title>

<script type="text/javascript">
    function changeImage(img) {
        /* 点图片，能换一张，而不受缓存影响 */
        img.src = img.src + "?" + new Date().getTime();
    }
</script>
</head>

<body>
    <form action="">
        用户名：<input type="text" name="username"><br /> 密码：<input
            type="password" name="password"><br /> 认证码：<input type="te
xt"
            name="checkcode">

        <!-- 小手点验证码 -->
        <img src="/day06/servlet/ResponseDemo4" onclick="changeImage(thi
s)"
            alt="换一张" style="cursor:hand"><br /> <input type="submit"
            value="注册">
    </form>
</body>
</html>
```

浏览器输入`http://localhost:8080/day06/register.html`查看结果。

5. 实用技术 - 用 response 的 refresh 控制浏览器定时刷新

1.控制浏览器3秒刷新代码：

```
public class ResponseDemo5 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletRespons
e response)
            throws ServletException, IOException {

        response.setHeader("refresh","3"); // 每3秒刷新

        String data = new Random().nextInt(10000) + "";
        response.getWriter().write(data);
    }
}
```

2.登录成功后，跳转代码：

```
protected void doGet(HttpServletRequest request, HttpServletResponse resp
onse)
    throws ServletException, IOException {

  // 假设这是处理登录的sevlet

  // 假设程序运行到此，用户登录成功了

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

  response.setHeader("refresh", "3;url='/day06/index.jsp'");
  response.getWriter().write("恭喜你登录成功，浏览器将在3秒后，跳到首页，若没
跳，请点<a href=''>超链接</a>");
}
```

3.登录成功后，实用的自动跳转技术： 用到，想法还是 **servlet** 不适合输出数据，数据在 **servlet** 中产生后，应传给一个**jsp**文件，由**jsp**修饰格式后，再输出，代码如下：
涉及2文件，`ResponseDemo5` 和 `WebRoot/message.jsp`，分别如下：

```
// 控制浏览器刷新，跳转实用代码

@Override
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {

  // 假设这是处理登录的sevlet

  // 假设程序运行到此，用户登录成功了，要跳转，实用代码
  // 用 <meta> 模拟 响应头 让jsp传给浏览器
  // 因为 sevlet 不适合输出数据，数据需jsp修饰后输出才是王道！故而才用<meta>模拟响应头
  String message = "<meta http-equiv='refresh' content='3;url=/day06/index.jsp'>恭喜你登录成功，浏览器将在3秒后，跳到首页，若没跳，请点<a href=''>超链接</a>";

  this.getServletContext().setAttribute("message", message);
  // 让jsp负责输出数据，需再建立一个message.jsp文件
  this.getServletContext().getRequestDispatcher("/message.jsp")
      .forward(request, response);
}
```

`WebRoot/message.jsp`，下面代码注意把encoding改为UTF-8:

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%
    String path = request.getContextPath();
    String basePath = request.getScheme() + "://"
            + request.getServerName() + ":" + request.getServerPort()
            + path + "/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<base href="<%=basePath%>">
<title>My JSP 'message.jsp' starting page</title>
</head>

<body>
    <!-- 代码从这开始 -->
    <%
        String message = (String) application.getAttribute("message");
        out.write(message);
    %>
</body>
</html>
```

6. 实用技术 - 用 Expires 头控制浏览器缓存

本节故事是：控制浏览器缓存某个东西1个小时。背景是有些内容几乎无变化，这就得控制客户浏览器缓存这个内容，否则服务器会累死。
办法就是 通过服务器向浏览器发送控制缓存的响应头。

实例：客户访问`index.jsp`，点击“查看图书”超链接，然后跳转到`ResponseDemo6`，显示图书内容`aaaaaaaaaa`。因为图书内容不怎么变化，故需用`expires`头控制缓存。

`WebRoot/index.jsp`代码如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%
String path = request.getContextPath();
String basePath = request.getScheme()+"://"+request.getServerName()+":"+r
equest.getServerPort()+path+"/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <base href="<%=basePath%>">

    <title>My JSP 'index.jsp' starting page</title>

  </head>

  <body>
    <a href="/day06/servlet/ResponseDemo6">查看图书</a>
  </body>
</html>
```

`ResponseDemo6`代码如下，注意用`System.currentTimeMillis()`获取当前时间，若直接用`1000*3600`是不行的：

```
// 控制缓存
public class ResponseDemo6 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOExce
ption {

        // 注意 System.currentTimeMillis() 获取当前时间
        response.setDateHeader("expires",
                System.currentTimeMillis() + 1000 * 3600); // 缓存1小时

        String data = "aaaaaaaaaaaaa";
        response.getWriter().write(data);
    }
}
```

7. response实现请求重定向和response的一些细节

请求重定向：你管我借钱，我没有，我让你找别人。
特点：重定向后，浏览器地址栏内容会发生变化，且浏览器相当于发送服务器2次请求。
请求重定向能不用尽量不要用，它加重了服务器负担**(2**次请求**)**。

什么时候必须用请求重定向？

1. 登录，登录后跳到首页。地址栏发生变化，让用户知道他/她到首页上去了；
2. 购物，网上购物成功后，应该重定向到购物车显示页面(不要用转发技术)。

实例代码：

```
// 实现请求重定向
public class ResponseDemo7 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletRespons
e response)
            throws ServletException, IOException {

        /*
        response.setStatus(302);
        response.setHeader("location", "/day06/index.jsp");
        */

        response.sendRedirect("/day06/index.jsp"); // 这句=上面2句
    }
}
```

浏览器输入`http://localhost:8080/day06/servlet/ResponseDemo7`查看地址栏变化及结果。

8. response 的 getOutputStream 和 getWriter 不能同时调用，他们竟然互相排斥

如题，代码如下：

```
public class ResponseDemo8 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOExce
ption {
        response.getOutputStream();
        response.getWriter();
    }
}
```

上面代码运行时要抛异常：

```
java.lang.IllegalStateException: getOutputStream() has already been calle
d for this response
    org.apache.catalina.connector.Response.getWriter(Response.java:609)
    org.apache.catalina.connector.ResponseFacade.getWriter(ResponseFacad
e.java:211)
    cn.wk.response.ResponseDemo8.doGet(ResponseDemo8.java:16)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:622)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
    org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:5
2)
```

方立勋老师挺有意思，他课堂上说，学生们以后开发时肯定会犯上面所说的错，并报异常，这种异常见不到，不算是合格的开发人员。看到这，我就呵呵了，相忠良肯定不会犯这错误^^，玩笑话啦。

方老师说，有如下情况：

```
public class ResponseDemo8 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOExce
ption {

        response.getOutputStream();
        this.getServletContext.getRequestDispatcher("/servlet/ResponseDem
o9").forward(request, response);
    // 上面是转发，不是重定向哦，转发时客户浏览器地址栏是不变的！
    }
}
```

```
public class ResponseDemo9 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOExce
ption {

        response.getWriter();
    }
}
```

转着转着就懵逼了，报了那个`java.lang.IllegalStateException`异常。

重定向就不会抛这种异常，因为那是**2**次请求，有**2**个 **response**！ 转发是**1**次请求，只有**1**个**response**对象！

通过 **response** 对象弄出的流，我们不需关，**web**容器帮我们关。但若不是通过 **response** 弄出的流，咱自己得关，服务器不管！

9. request 简介

`HttpServletRequest` 对象代表客户端的请求，当客户端通过 **HTTP** 协议访问服务器时，**HTTP** 请求头中的所有信息都封装在这个对象中，开发人员通过这个对象的方法，可或得客户的这些信息。 request 里肯定有代表客户请求的请求行，请求头和请求数据的相关方法。通过 servlet API 查看之。

URI: /news/1.html 它能标识任意资源，URI是爸爸，URL是崽。开发中 URI 用的比 URL 多。
URL: http://www.sina.com/news/1.html 它只能标识互联网上的资源

例子：

```
// request 简单示例
public class RequestDemo1 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOExce
ption {

        System.out.println(request.getRequestURI());
        System.out.println(request.getRequestURL());

        System.out.println(request.getQueryString());

        System.out.println("---------------------");

        System.out.println(request.getRemoteAddr()); // client ip
        System.out.println(request.getRemoteHost()); // client host name

        System.out.println(request.getLocalAddr()); // WEB 服务器的 ip
        System.out.println(request.getLocalName()); // WEB 服务器的主机名

        System.out.println(request.getRemotePort());

        System.out.println(request.getMethod()); // 返回 GET POST 等
    }
}
```

浏览器中输入 `http://localhost:8080/day06/servlet/RequestDemo1?name=aaa&pass=666` 查看结果为：

```
/day06/servlet/RequestDemo1
http://localhost:8080/day06/servlet/RequestDemo1
name=aaa&pass=666
---------------------
0:0:0:0:0:0:0:1
0:0:0:0:0:0:0:1
0:0:0:0:0:0:0:1
0:0:0:0:0:0:0:1
51607
GET
```

`0:0:0:0:0:0:0:1`这个事，访问时localhost换成127.0.0.1就行了，具体情况网上查。据说通常发生在客户机和服务器在同一台机器上。感觉这个结论不靠谱。
并且，客户机浏览器的 port 也不像方老师所说的随机，我机器上的表现却是固定的，但也不完全固定，无论 chrome 还是 ie。

10. request 获取请求头和请求数据

```java
// request 获取头相关的方法
public class RequestDemo2 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOExce
ption {

        String headValue = request.getHeader("Accept-Encoding");
        System.out.println(headValue);

        System.out.println("----------------------");

        Enumeration e = request.getHeaders("Accept-Encoding");
        while (e.hasMoreElements()) {
            String value = (String) e.nextElement();
            System.out.println(value);
        }

        System.out.println("----------------------");

        e = request.getHeaderNames();
        while (e.hasMoreElements()) {
            String name = (String) e.nextElement();
            String value = request.getHeader(name);
            System.out.println(name + " = " + value);
        }
    }
}
```

浏览器输入`http://localhost:8080/day06/servlet/RequestDemo2`，我机器控制台的输出是：

```
gzip, deflate, br
---------------------
gzip, deflate, br
---------------------
host = localhost:8080
connection = keep-alive
cache-control = max-age=0
upgrade-insecure-requests = 1
user-agent = Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHT
ML, like Gecko) Chrome/65.0.3325.181 Safari/537.36
accept = text/html,application/xhtml+xml,application/xml;q=0.9,image/web
p,image/apng,*/*;q=0.8
accept-encoding = gzip, deflate, br
accept-language = zh-CN,zh;q=0.9
cookie = _ga=GA1.1.1376610490.1484797437; _gid=GA1.1.1268285374.152430278
5
```

注意：不同浏览器(IE 和 Chrome)发出的请求头根本不一样，差别还挺大的。

10.1 用户带数据给服务器的方式，两种：超链接中或表单，重要例子

本节案例很重要！

该案例涉及3个文件和2个工具包：`RequestDemo2`，`WebRoot/test.html`，`cn.wk.User`普通java类，`commons-beanutils-1.7.0.jar`和`commons-logging-1.1.1.jar`工具包。

`test.html`代码：

```html
<!DOCTYPE html>
<html>
  <head>
    <title>带数据给RequestDemo2.html</title>

    <meta name="keywords" content="keyword1,keyword2,keyword3">
    <meta name="description" content="this is my page">
    <meta name="content-type" content="text/html; charset=UTF-8">

  </head>

  <body>
    <!-- 带数据方式1 -->
    <a href="/day06/servlet/RequestDemo2?username=xxx">点点</a>

    <!-- 带数据方式2 -->

    <form action="/day06/servlet/RequestDemo2" method="post">
        用户名1: <input type="text" name="username"><br/>
        用户名2: <input type="text" name="username"><br/>
        密码: <input type="text" name="password"><br/>
        <input type="submit" value="提交"><br/>
    </form>

  </body>
</html>
```

`cn.wk.User` 普通java类代码：

```java
package cn.wk;

public class User {
  //因为变态的弄了2个username,所以String[],正常不会这样干
    private String[] username;
    private String password;

    public String[] getUsername() {
        return username;
    }
    public void setUsername(String[] username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}
```

RequestDemo2 代码，里面有很重要的东西，涉及了 **BeanUtils** 工具包的使用：

```java
// request 获取请求头 和 请求数据
public class RequestDemo2 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOExce
ption {

        System.out.println("-------获取数据方式1---------");
        String value = request.getParameter("username");
        if (value != null && !value.trim().equals("")) {
            System.out.println(value);
        }

        System.out.println("-------获取数据方式2---------");

        String[] values = request.getParameterValues("username");
        for (int i = 0; value != null && i < values.length; i++)
            System.out.println(values[i]);

        System.out.println("-----最重要的获取数据方式3  用BeanUtils.populat
e()----");
        // 使用对象 封装浏览器送来的数据
        Map<String, String[]> map = request.getParameterMap();
        User user = new User();
        try {
            BeanUtils.populate(user, map); // map集合数据填充bean
            // BeanUtils.copyProperties(dest, orig); bean的拷贝
        } catch (Exception e) {
            e.printStackTrace();
        }

        // 服务器必须以debug方式启动，  才能观察断点
        System.out.println(user); // 这打断点，观察是否传入进user对象
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(req, resp);
    }
}
```

浏览器输入 `http://localhost:8080/day06/test.html`，页面中输入数据后提交，再查看控制台结果。我的 `test.html` 页面出乱码了。

11. 通过表单收集客户机数据 - 多种 form 数据

项类型展示

> 下面的2行没解决我的乱码问题
> eclipse导入项目后HTML文件都是乱码的（原UTF-8，现GBK）
> windows->perferences->General->Content Types->Text->HTML，然后将Default encoding设置为utf-8即可。

乱码问题困扰着我。

`WebRoot/form.html` 代码如下 **(多种 form 数据项类型展示)：**

```
<!DOCTYPE html>
<html>
  <head>
    <title>form.html</title>

    <meta name="keywords" content="keyword1,keyword2,keyword3">
    <meta name="description" content="this is my page">
    <meta name="content-type" content="text/html; charset=UTF-8">

  </head>

  <!-- 常用的 表单输入项 类型 -->
  <body>
    <form action="/day06/servlet/RequestDemo3" method="post">
        用户名：<input type="text" name="username"><br/>
        密码：<input type="password" name="password"><br/>
        性别：
            <input type="radio" name="gender" value="male">男
            <input type="radio" name="gender" value="female">女<br/>

        所在地：
            <select name="city">
                <option value="beijing">北京</option>
                <option value="shanghai">上海</option>
                <option value="chaihe">柴河</option>
            </select>
            <br/>

        爱好：
            <input type="checkbox" name="likes" value="sing">唱歌
            <input type="checkbox" name="likes" value="dance">跳舞
            <input type="checkbox" name="likes" value="basketball">篮球
            <input type="checkbox" name="likes" value="football">足球
            <br/>

        备注：<textarea rows="6" cols="60" name="description"></textarea><
br/>

        大头照：<input type="file" name="image"><br/>

        <!-- 隐藏输入项 用户不可见 但表单提交时 一同提交 -->
        <input type="hidden" name="id" value="12356">

        <input type="submit" value="提交"><br/>
    </form>
  </body>
</html>
```

`RequestDemo3` 代码：

用户提交的数据一定先检查后使用！

```
public class RequestDemo3 extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        System.out.println(request.getParameter("username"));
        System.out.println(request.getParameter("password"));
        System.out.println(request.getParameter("gender"));
        System.out.println(request.getParameter("city"));

        // 数据一定 先检查 后使用
        String[] likes = request.getParameterValues("likes"); // 遍历时小
心likes为空
        for (int i = 0; likes != null && i < likes.length; i++) {
            System.out.println(likes[i]);
        }

        System.out.println(request.getParameter("description"));

        // 大头照涉及文件上传和下载，在这先不处理

        System.out.println(request.getParameter("id"));
    }

    public void doPost(HttpServletRequest request, HttpServletResponse re
sponse)
            throws ServletException, IOException {
        doGet(request, response);
    }
}
```

浏览器输入`http://localhost:8080/day06/form.html`可能有乱码，填入数据提交，控制台查看结果。

12. request乱码(试验失败，待解决)

涉及2文件，`WebRoot/form3.html`和`RequestDemo4`，我机器上本实验做的不成功。

`WebRoot/form3.html`如下，2个表单，上面的用post方法，下面的用get方法：

另一个纳闷的事是`<meta name="content-type" content="text/html; charset=UTF-8">`对浏览器码表的设置对我机器不起作用，不知为啥！

```html
<!DOCTYPE html>
<html>
  <head>
    <title>form3.html</title>

    <meta name="keywords" content="keyword1,keyword2,keyword3">
    <meta name="description" content="this is my page">
    <meta name="content-type" content="text/html; charset=UTF-8">

  </head>

  <body>
    <form action="/day06/servlet/RequestDemo4" method="post">
        用户名：<input type="text" name="username"><br/>
        <input type="submit" value="提交"><br/>
    </form>

    <form action="/day06/servlet/RequestDemo4" method="get">
        用户名：<input type="text" name="username"><br/>
        <input type="submit" value="提交"><br/>
    </form>
  </body>
</html>
```

```java
// post 提交方式 乱码解决
request.setCharacterEncoding("UTF-8"); //只对post有效
String username = request.getParameter("username");
System.out.println(username);
```

我机器下面代码无法去除乱码：

```java
// get 提交方式 乱码解决
String username = request.getParameter("username");
username = new String(username.getBytes("iso-8859-1"), "UTF-8");
System.out.println(username);
```

超链接提交的中文也只能手工处理(因为也是get提交，按上面get提交乱码解决方式处理)：

```html
<a href="/day06/servlet/RequestDemo4?username=中国">点点</a>
```

13. request 实现请求转发和 mvc 设计模式 -

涉及到 forward 转发时的幺蛾子 - 不能转发2次

request 域

MVC思想 (model->javabean view -> jsp controller -> servlet)

RequestDemo6 生成数据，用 request 转发给 /WebRoot/message.jsp 文件显示：下面例子简单，但非常重要。另外，不能转发两次！详细情节看下面代码：

forward请求转发的特点：

1. 客户端只发1次请求，而服务器端有多个资源调用；
2. 客户端浏览器地址栏无变化。

```java
// 请求转发，以及使用request域对象把数据带给转发资源
public class RequestDemo6 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {

        // MVC思想 (model->javabean view -> jsp controller -> servlet)

        String data = "aaaaaaa";

        req.setAttribute("data", data);

        // request也可实现转发
        if (true) {
            req.getRequestDispatcher("/message.jsp").forward(req, resp);
            return; // return保证了以后不会再次转发了
        }

        // 但不能再次转发
        // 将抛异常 java.lang.IllegalStateException: Cannot forward afte
r response
        // has been committed
        req.getRequestDispatcher("/index.jsp").forward(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {

        doGet(req, resp);
    }
}
```

/WebRoot/message.jsp 的代码，里面涉及了 EL 表达式，jsp中用 EL 表达式取出数据输出：

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%
String path = request.getContextPath();
String basePath = request.getScheme()+"://"+request.getServerName()+":"+r
equest.getServerPort()+path+"/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <base href="<%=basePath%>">

    <title>My JSP 'message.jsp' starting page</title>

  </head>

  <body>

  ${data}   <!-- EL 表达式 -->

  <!-- 下面写法也可，推荐用 EL -->
<%
  String data = (String)request.getAttribute("data");
  out.write(data);
%>

  </body>
</html>
```

forward细节：forward 时，会清空response中的数据，如下：

```
// forward细节： forward时，会清空response中的数据
public class RequestDemo7 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {

        String data = "aaaaaaa";
        resp.getWriter().write(data);

        req.getRequestDispatcher("/index.jsp").forward(req, resp); //会覆
盖上面的resp向浏览器送出的信息
    }

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

14. request 实现页面包含 include 代替 forward (此技术不用，通常由 jsp 处理包含，而不是由 servlet 处理)

如题，代码如下，本例涉及1个servlet，2个jsp:

```
public class RequestDemo8 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {

        req.getRequestDispatcher("/public/head.jsp").include(req, resp);

        resp.getWriter().write("hahahahaha<br/>");

        req.getRequestDispatcher("/public/foot.jsp").include(req, resp);
    }
}
```

`/public/head.jsp`：

```
<body>
  head <br>
</body>
```

`/public/foot.jsp`：

```
<body>
  foot <br>
</body>
```

浏览器输入`http://localhost:8080/day06/servlet/RequestDemo8`，输出结果为：

```
head
hahahahaha
foot
```

15. web 工程中各类地址的写法

写地址的原则：

1. 以`/`开头；
2. 若地址是写给服务器用的，`/`就代表当前 web 应用；
3. 若地址是写给浏览器用的，`/`就代表网站；

我认为，第1条有用，其他2条无所谓。

```java
// 用地址的地方
public class ServletDemo1 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {

        // 1.
        req.getRequestDispatcher("/").forward(req, resp);

        // 2.
        resp.sendRedirect("");

        // 3.
        this.getServletContext().getRealPath("");

        // 4.
        this.getServletContext().getResourceAsStream("");

        // 5.
        /*
         * <a href="">点点</a>
         *
         * <form action="/day06/form1.html">
         *
         * </form>
         * */
    }
}
```

16. 利用 referer 防盗链

```java
// 防盗链
public class RequestDemo9 extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        // 判断 来访者从哪个页面来的
        String referer = request.getHeader("referer");

        // 符合盗链者条件
        if (referer == null || !referer.startsWith("http://localhost")) {
            response.sendRedirect("/day06/index.jsp");
            return;
        }

        String data = "凤姐日记";
        response.getWriter().write(data);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse re
sponse)
            throws ServletException, IOException {

        doGet(request, response);
    }
}
```