# 修改Servlet模板



每次创建的Servlet中有很多的注释，每次删除好麻烦，可以修改生成这个Servlet的模板，方法如下：



注册在桌面右击图标打开的位置还不是MyEclipse的安装根目录，还需要往上一层，在这一层里进行搜索。打开模板文件后删除掉那些注释即可。

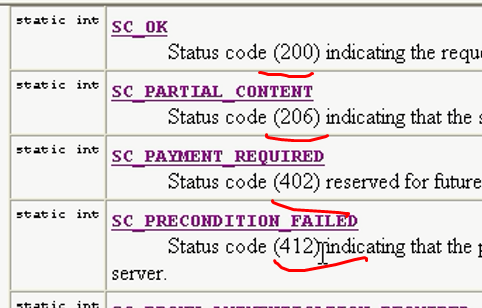
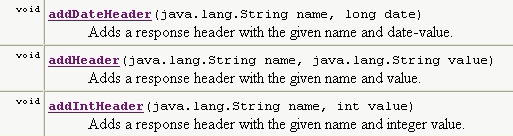
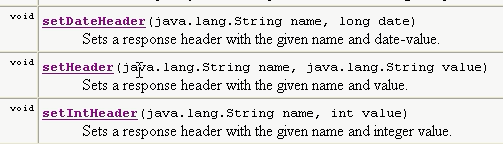
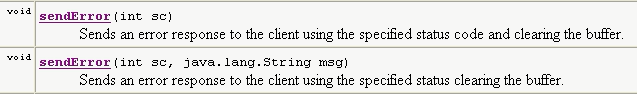
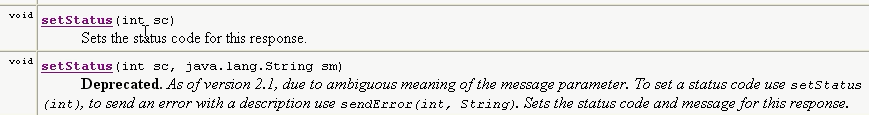
# 重命名Servlet名字的问题

当修改了XXXServlet.java的名字时，web.xml中的相关映射中的类名需要我们手动修改过来，系统并不会自动并我们改，即使我们改类名时使用的是重构也不行。

# 简介

* 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。
* request和response对象即然代表请求和响应，那我们要获取客户机提交过来的数据，只需要找request对象就行了。要向客户机输出数据，只需要找response对象就行了。

# HttpServletResponse

* HttpServletResponse对象代表服务器的**响应**。这个对象中封装了向客户端发送数据、发送响应头，发送响应状态码的方法。
* 可查看该类的API文档，该类有各种状态码的常量，如下：  
    
  这里只是列出了几个而已，还有很多很多的状态码常量  
    
  这3个方法中第2个是通用的，如有些头的参数值是一个数值，使用addIntHeader可以，使用addHeader传一个字符串形式的数字也可以  
    
  这3个方法在真实开发用的比较多一些，没有就添加头，有就修改值   
    
  传错误相关的响应码与消息  
    
    
  传没错的响应码与消息
* HttpServletResponse它的父类中的方法也需要看：  
    
  当调用此方法时，客户端就能立马看到结果，但是我们不需要调用这个方法，当给response写内容时，当缓存满了，它会自动调用此方法，或者doGet或doPost方法返回时Tomcat容器会帮我们调用。  
  默认缓存大小是8K  
    
  setContentLength，设置正文的长度，不设置也能知道  
  setContentType，，正文的类型，浏览器需要知道正文是什么类型，然后选择以什么样的方式处理，比如：如果正文是一个文件，则采用下载处理

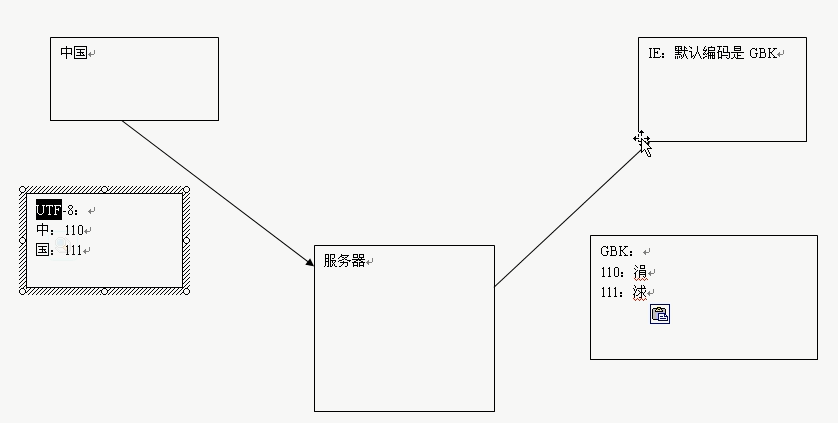
# HttpServletResponse常见应用

## 向客户端输出中文数据

### 用OutputStream(字节流)发送数据：

1、response.getOutputStream().write(“中国”.getBytes());//以默认编码发送数据(GBK)，而浏览器在读取时默认也使用本地编码读，所以没有乱码问题

2、response.getOutputStream().write("中国".getBytes("UTF-8"));//以UTF-8编码发送数据，浏览器(默认用GB2312)会出现乱码

画图描述出现该问题的原因。  
 

解决办法：

2.1通过更改浏览器的编码方式：IE/”查看”/”编码”/”UTF-8”(不可取)

2.2通过设置响应头告知客户端编码方式：response.setHeader(“Content-type”, “text/html;charset=UTF-8”);//告知浏览器数据类型及编码  
 还有另一个更方便的方法：  
 内部原理也是添加头

2.3通过meta标签模拟请求头:out.write("<meta http-equiv='Content-Type' content='text/html; charset=utf-8' />".getBytes()); // 这是把内容写到了Html中，相当于在Html中设置头的效果，虽然不是响应头，功能类似响应头。  


2.4通过以下方法：response.setContentType("text/html;charset=UTF-8");

总结：程序以什么编码输出，就需要告知客户端以什么编码显示。

小细节：输出字符“1”用response.getOutputStream().write(1);出现的问题？ 写的时候是把1的字节码写出去了，但是IE显示时是把读取到的字节当字符来显示，所以会去查找GBK码表，把对应的字节的字符显示出来，而我们的字节是表示一个int值，并不是一个字符。解决办法把1变成String即可：”1”。使用getWriter()的write也一样，如果直接写数值，在浏览器将会看到乱码，使用getWriter的print方法则可以直接输出int数值。还有一个println方法，这个方法一般不用，因为它不能实现换行，它只是会输出\r\n，但这并不是html中的换行标志</br>，但是如果我们查看源代码是换行的，因为记事本决定文本的换行就是通过\n\r来决定。

### 用PrintWriter(字符流)发送数据：

示例：response.getWriter().write(“中国“); Servlet是老外发明的，写字符的编码默认使用的是ISO-8859-1编码，所以会发生乱码

解决办法：

setCharacterEncoding(“UTF-8”);// 告诉response使用Writter写字符的时候以”UTF-8”编码去写，这个方法一定要在获取response的流之前先设置，否则不生效。

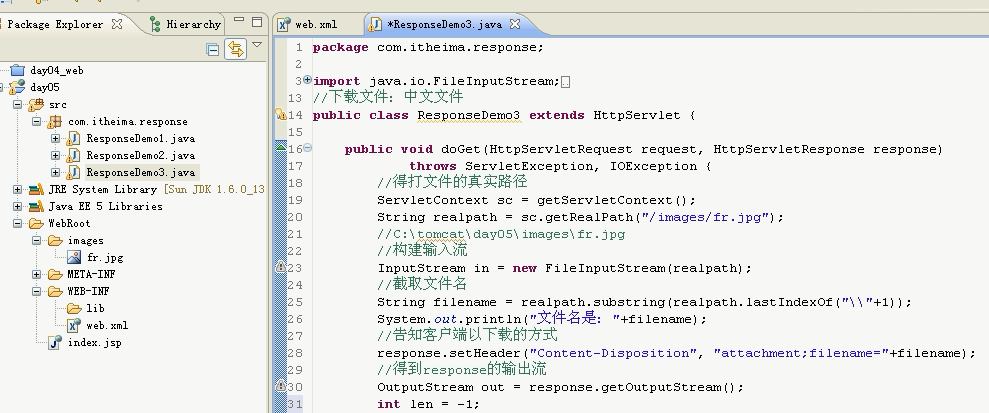
response.setHead(“Context-type”,”text/html;charset=UTF-8”);//告诉客户端以UTF-8编码读取

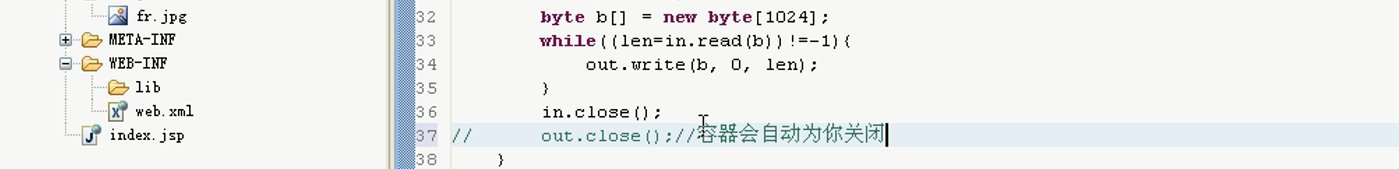
由于经常改动编码，response提供了一种更简单的方式：

response. setContentType(“text/html;charset=UTF-8”);其作用相当于以上两条代码。也就是这个方法内部会调用setCharacterEncoding来设置编码，同时还添加请求头。

实际开发中，使用Writer的方式写出内容比较常用，因为浏览器显示的内容一般就是字符。

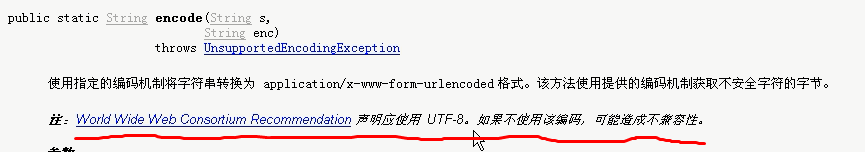
## 文件下载



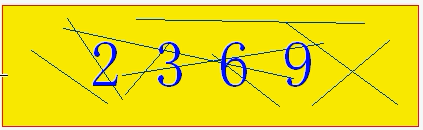
  
上面截取文件名的地方，“+1”需要写到括号外，表示最后一个反斜杠的位置的索引 + 1

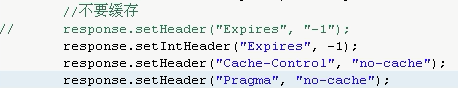
中文文件名的文件下载

* + 问题：文件名乱码或无法下载
  + 解决：将文件名进行编码  
    

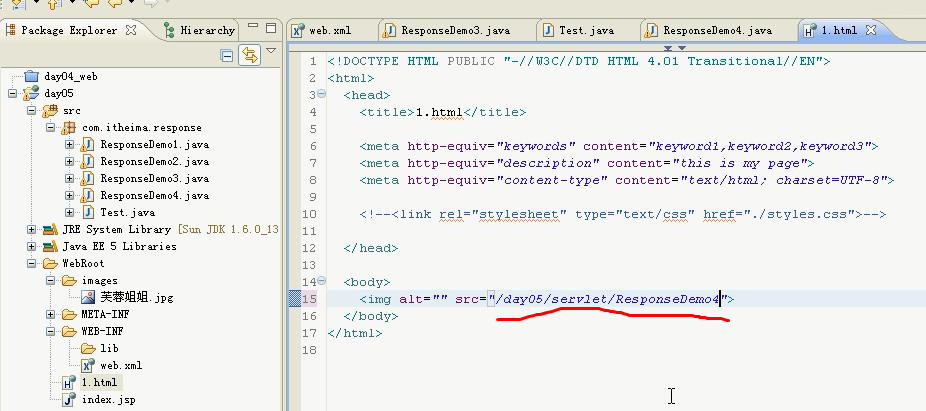
  
  
从API文档可知，W3C告诉我们声明编码时应该使用UTF-8。

## 生成验证码



* 随机图片(专业术语叫CAPTCHA图像):Completely Automated Public Turing Test to Tell Computers and Humans Apart (全自动区分计算机和人类的测试)
* 相关主要类(JDK 查看API)
  + BufferedImage:代表一幅内存图像
  + Graphics：代表画笔
  + ImageIO：用于输出图像
  + 放在html页面上<img src/>
* 注意：浏览器默认会缓存图片，所以当用户点击IE上的刷新按钮时，验证码图片并不会重新请求，而是使用缓存的图片，所以需要告诉IE不要缓存：
  + response.addHeader("Pragma", "no-cache");
  + response.setHeader("Cache-Control", "no-cache");
  + response.setHeader("Expires", "-1");  
    

在html中使用到图片：



在html中有一个img标签，src指向了一个Servlet，不论是一个静态资源图片还是一个Servlet，都是向服务器发一个请求，当请求的是Servlet时，则可以通过Servlet动态的返回图片内容，可以返回一个静态的图片，也可以动态的创建一张图片并返回。  
Servlet代码如下：  


完善Html网页：

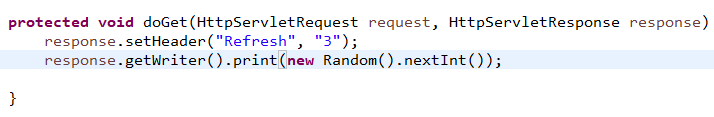
  
点击看不清时需要刷新验证码，就是重新请求图片的地址，但是如果图片地址不变，浏览器不会重新请求，所以需要动态的修改请求的url，可以在url后面通过?连接一些参数，这些参数就随便写了，如使用时间，这样每次发送的请求图片url都是不一样的，因为参数值不一样。  

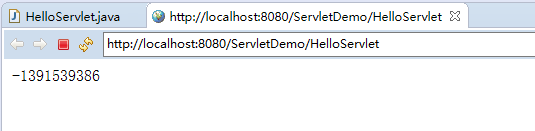

## 通过response实现请求重定向

第四天笔记中有

## 发送http头，控制浏览器定时刷新网页

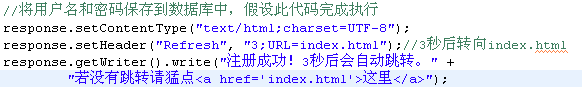
1、输出一个随机数，控制浏览器每2S刷新一次

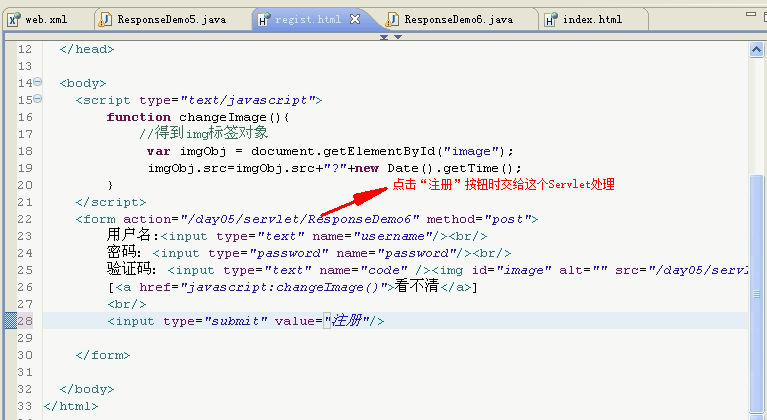




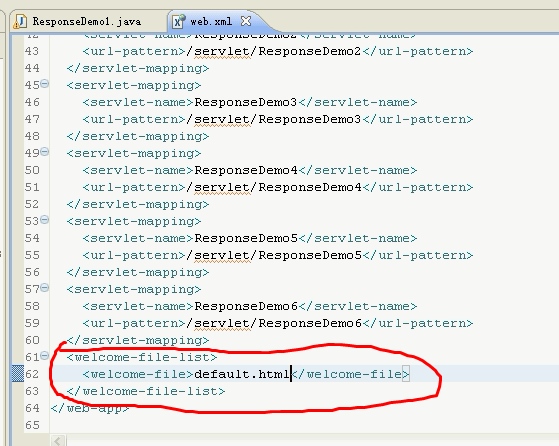
这里每3秒就会自动刷新，原理应该是每3秒再往服务器发送一次请求。

1. 利用Refresh刷新到其他组件。模拟注册页面，注册处理交由一个Servlet，在Servlet处理完成后，显示注册成功并转向主页。(meta也可实现)

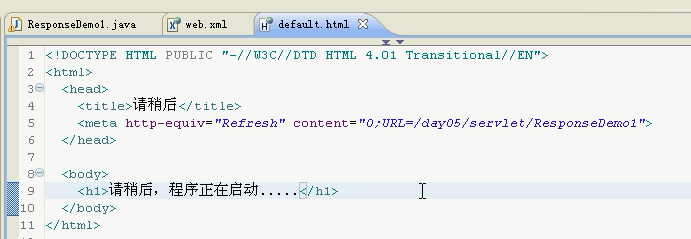
  
这里的index.html是放在应用的根目录的。

这样3秒一到就跳转到另一个界面了，且是直接覆盖当前页面，也就是说跳转成功后每3秒刷一次的页面就不存在了。  
  
  
注册界面如下：  
  
点击这里的“注册”按钮，交给ResponseDemo6这个Servlet处理，代码如下：  
  
这个Servlet将输出如下界面，并且在2秒后会自动跳转到index.html，如下：  
  


1. 思考：web.xml中能使用Servlet作为默认主页吗？

不行，但是可以通过html页面自动刷新到Servlet上，如下：  
设置主页为一个html  


default.html如下：



所以这也是一样使用场景，通过Header设置刷新头，可以设置Servlet为主页

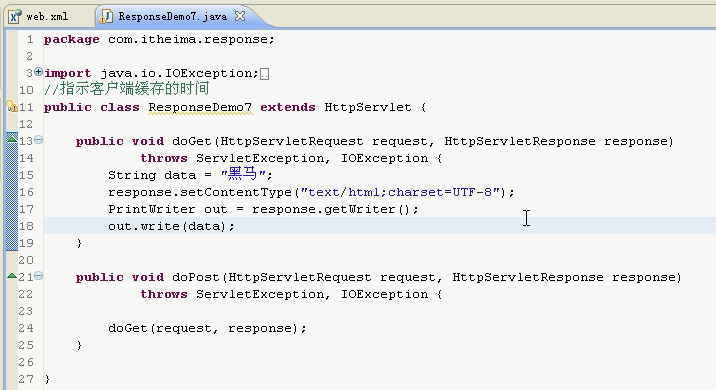
## response细节

* getOutputStream和getWriter方法分别用于得到输出二进制数据、输出文本数据的ServletOuputStream、Printwriter对象。
* getOutputStream和getWriter这两个方法互相排斥，调用了其中的任何一个方法后，就不能再调用另一方法。 会抛异常。
* Servlet程序向ServletOutputStream或PrintWriter对象中写入的数据将被Servlet引擎从response里面获取，Servlet引擎将这些数据当作响应消息的正文，然后再与响应状态行和各响应头组合后输出到客户端。
* Serlvet的service方法结束后，Servlet引擎将检查getWriter或getOutputStream方法返回的输出流对象是否已经调用过close方法，如果没有，Servlet引擎将调用close方法关闭该输出流对象。

## 控制资源的缓存时间

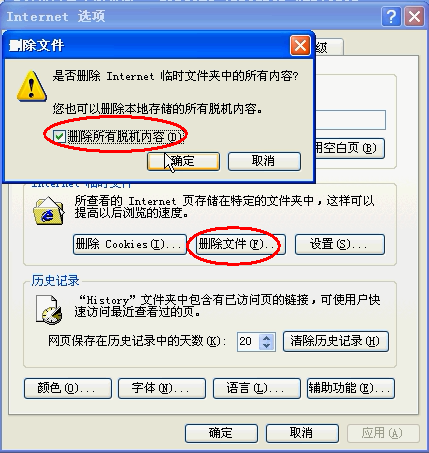
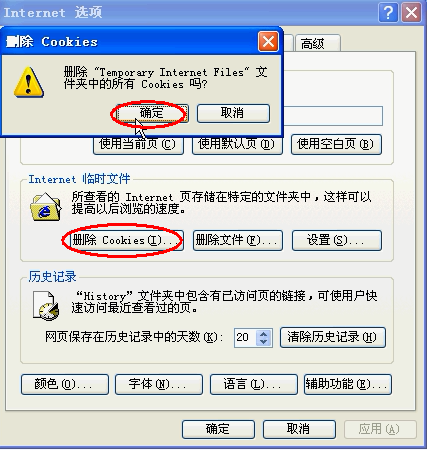
* 发送http头，控制浏览器缓存当前文档内容
* response.setDateHeader(“Expires”, System.*currentTimeMillis*()+1000\*60\*60);//缓存1小时,注意此处是相对于1970年1月1日00:00:00的时间
* 作用：有些不怎么变化的数据，利用缓存能减轻服务器的负担。
* 查看一下IE的缓存文件

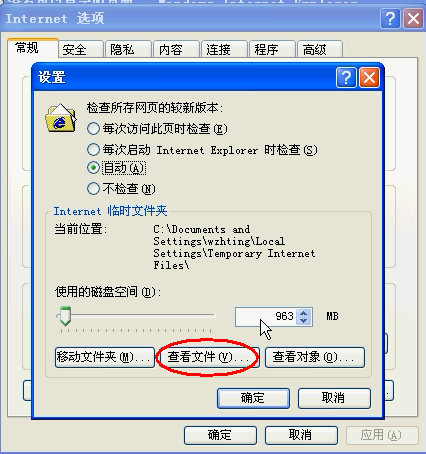
示例代码：



默认情况下浏览器会缓存页面的。

先清除浏览器的所有缓存，操作如下：

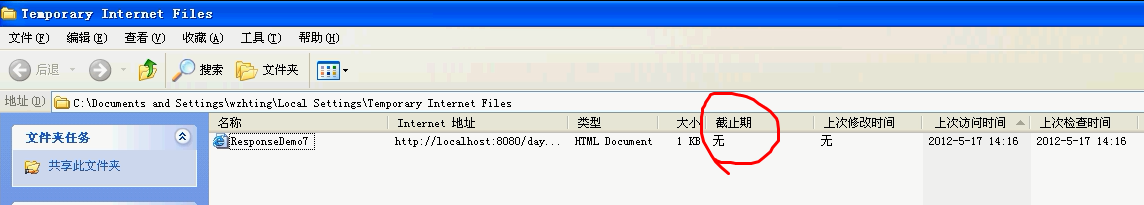
 

这时候查看缓存目录中是什么都没有的，如下：

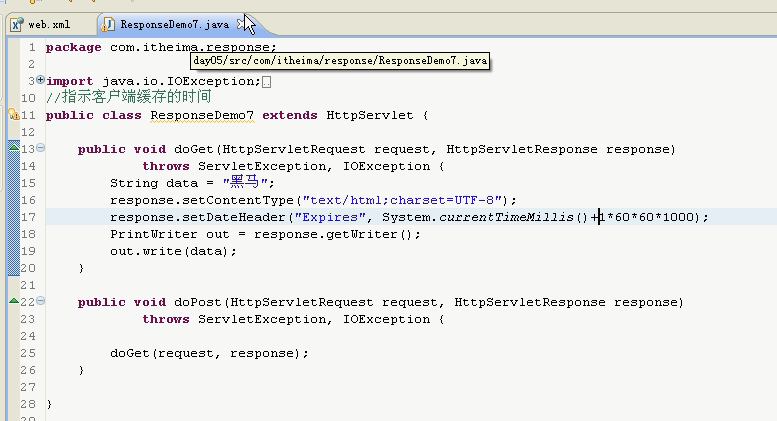


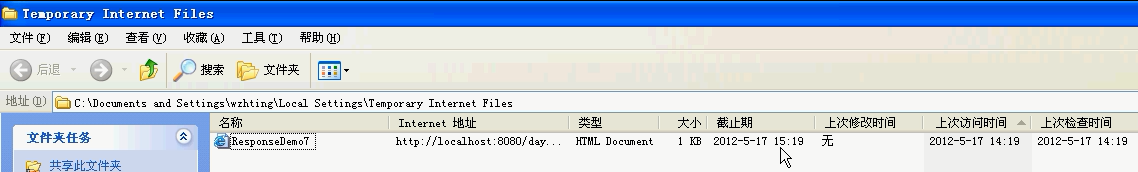
这时在浏览器上访问一下ResponseDemo7这个Servlet，这时就可以在缓存目录中看到有缓存文件，如下：



这里并没有显示出具体的缓存时间。

这时我们在代码中加入缓存时间有效期的header，如下：





# HttpServletRequest

* 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，可以获得客户这些信息。
* 查看request的API文档

## request常用方法

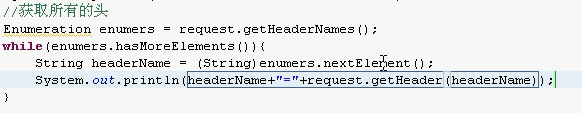
* **获得客户机信息**
  + getRequestURL方法返回客户端发出请求时的完整URL。
  + getRequestURI方法返回请求行中的资源名部分。
  + getQueryString 方法返回请求行中的参数部分。
  + getRemoteAddr方法返回发出请求的客户机的IP地址
  + getRemoteHost方法返回发出请求的客户机的完整主机名
  + getRemotePort方法返回客户机所使用的网络端口号，注：不是8080，8080是服务器的端口，客户机使用的端口是电脑随便分配的一个空闲端口。
  + getLocalAddr方法返回WEB服务器的IP地址。
  + getLocalName方法返回WEB服务器的主机名
  + getMethod得到客户机请求方式

注：getHeaders(String name)这是获取指定名字的多个头，也就是说同样名字的Header可以有多个，它们的值可能不相同。

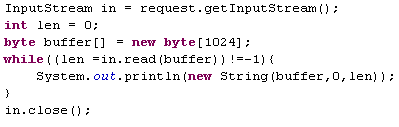
常用方法返回值示例如下：



## 获得客户机请求头

* + **getHead(name)方法**
  + getHeaders(String name)方法
  + getHeaderNames方法   
    

## 获得客户机请求参数(客户端提交的数据)

* + getParameter(name)方法
  + getParameterValues（String name）方法。这说明请求参数也可以相同，也就是说可以传多个同名的参数。所以才有这个方法获取指定名字参数的多个值。
  + getParameterNames方法
  + getParameterMap方法 //做框架用，非常实用。返回的Map是这样的 ：Map<String, String[]>，key就是参数名，因为同样名字的参数可以有多个，所以使用数组装value。
  + getInputStream 以流的方式获取请求正文的内容，如post提交的参数是放在请求正文里的，骑过读流也能获取到所有的请求参数。如果是get请求的参数就获取不 到了，因为参数不是放在请求正文里的  
    

可以获取get请求中的参数，也可以获取表单中post请求中的参数。 在取参数值时，如果有传这个参数，如post请求，即使你什么也不输入，则取到的参数值为空字符串，如果这个参数在提交时并没有，也就是说抓包时都看不 到有这个参数，如果这时取这个参数，值 为null。

获取属性要保存时，有的人会先写一个JavaBean，然后调用getParameter(name)方法得到属性值设置给JavaBean，最后保存JavaBean到数据库，这是比较low的做法。高级做法是使用getParameterMap方法得到所有的参数，然后通过反射来给对象填充值。补充一下内省知识点，如下：

  
这是一个JavaBean，自省呢是看方法名的，setUserName就是一个写属性，getUserName就是一个读属性，而username1字段怎么重命名无所谓，对内省没有影响，只要你方法名不变就行，通过内省填充JavaBean属性，示例如下：

User u = **new** User();

PropertyDescriptor pd = **new** PropertyDescriptor("username", User.**class**);

Method writeMethod = pd.getWriteMethod(); // 得到User的setUsername方法

writeMethod.invoke(u, "Even"); //调用 u.setUsername("Even")

这样搞也很麻烦的，所以出来了一个构架，叫BeanUtils，这是Aphache出品的，这需要用到两个jar包：commons-beanutils-1.8.3.jar，这个jar包它又依赖于：commons-logging-1.1.1.jar，使用示例如下：

User u = **new** User();

BeanUtils.*setProperty*(u, "username", "Even");

这只是设置了对象的一个属性，如果属性多也需要写很多代码，可以使用下面的方法一句话完成：

BeanUtils.*populate*(bean, map); 第一个参数为需要填充属性的JavaBean，第二个参数则包含了所有的属性名和对应的值，工具会自动帮我们填充。这里的map我们就可以使用request. getParameterMap（）中返回的map对象。使用这种方式就要求请求参数的名字要和JavaBean中的名字保持一致。对于Map中的value是一个String数组，Beanutils会取索引为0的元素填进去。如果JavaBean的属性是一个数组，则Map中的value正好就是一个数组，会把整个数组填充进去。且Beanutils还能根据属性的类型进行类型转换再赋值，如属性是int类型，则会把value转换为int再填充。注：类型转换只限于基本数据类型：int、float、long、boolean等。因为其它类型不知道要怎么转换，所以我们可以给Beanutils设置一个转换器，告诉它如何转换，比如JavaBean的属性类型为Date，你就要告诉它如何把String转换为Date，如下：  
ConvertUtils.*register*(**new** DateLocaleConverter(), Date.**class**);

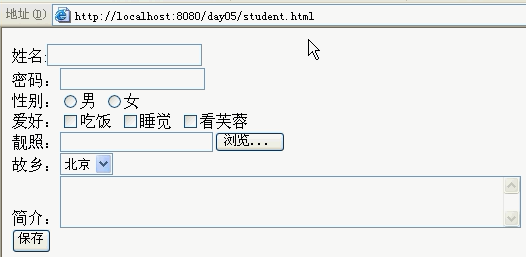
这里，代码中的第二个参数是说如果JavaBean的属性是Date，则按DateLocaleConverter这个转换器来进行转换，BeanUtils中实现了很多的常用对象的转换器，如果是一些自己定义的对象，则需要自己实现一个转换器。 BeanUtils的使用可以查看它的文档。

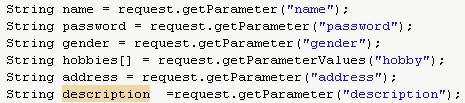
## 各种表单输入项数据的获取

* 各种表单输入项数据的获取
  + text、password、radio、checkbox、
  + file、select、textarea、 hidden、
  + image、button给js编程用



如上代码预览效果如下：





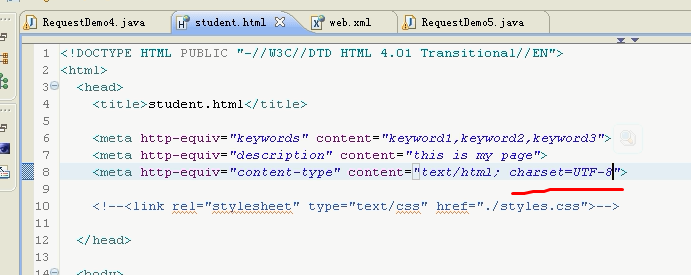
如果用户不输入，结果为空字符串，如果是单选或复选类型的，如果不选结果为null，为了预防这种情况，可以设置一个默认值，在选项里面加入：checked=”checked”



复选只有一个的情况下，未选为null，选择了值为on，所以这里给input标签加入value没有意义，要看有没有被选择，就看值是否为on即可。

* **请求参数的中文乱码问题**
  + 浏览器是什么编码就以什么编码传送数据
  + 解决：request.setCharacterEncoding(“UTF-8”);//POST有效
  + **new** String(username.getBytes(“ISO-8859-1”),“UTF-8”);//GET方式
  + 超链接：<a href=*“/JavaWeb/RequestDemo2?name=中国”*>cn</a>//GET方式，这个方式需要进行URLEncoder，如果把这个get请求直接打入到浏览器，浏览器可能会自动帮我们做URLEncoder的事。
  + 更改Tomcat的配置解决URL编码问题：<Connector URIEncoding=“UTF-8”/>

客户端在提交时，以什么样的编码提交的参数呢？取决于html的这个属性：



我们读的时候就要先获取到这个属性，再以对应的编码来读参数：

 // 只对post请求的参数有用。

通过上面的方法设置了编码之后，再调用getParameter方法获取的参数就会是以UTF-8编码来获取的了。

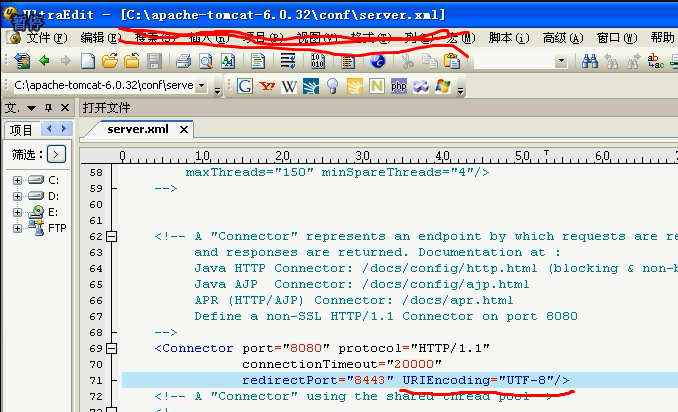
如果是get请求的参数中有中文，上面的方法是解决不了问题的，因为get请求中的值在传递时是以ISO-8859-1的编码进行传输的，需要这样来获取get请求中的中文参数值：

如果html指定了utf-8编码，但是提交的表单方式为get，获取中文value如下:

String username = request.getParameter("username");

username = **new** String(username.getBytes("ISO-8859-1"), "UTF-8");

也可以通过server.xml配置来告诉Sevelt使用UTF-8来能参数：

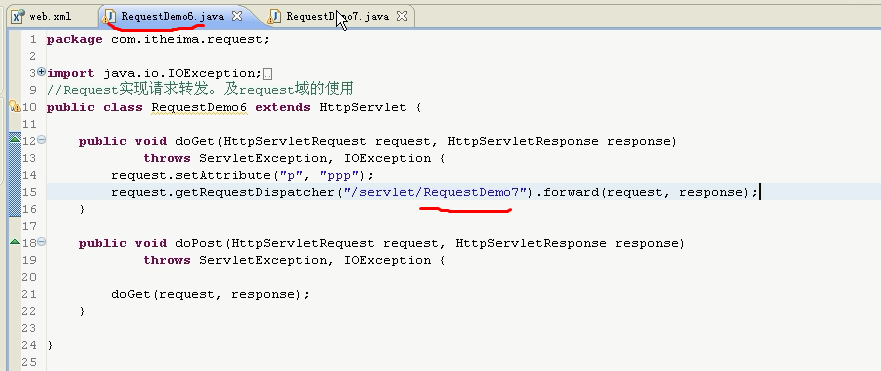
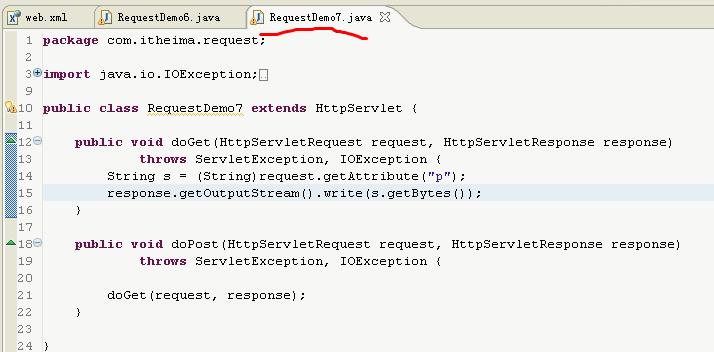


注：修改了这个需要重启Tomcat使其生效，然后Servlet中直接取get请求中的参数就是中文的了，这说明Servelet知道我们的网页是UTF-8编码的，因为get请求又是以ISO-8859-1传过来的，Servelt在取参数的时候就会自动帮我们进行编码转换。 这个方式看起来不错，但是很少使用，因为这种方式太依赖于服务器了，假设你部署到JBoss服务器，里面没有这个属性设置呢？

## request常见应用2

* request对象实现请求转发：请求转发指一个web资源收到客户端请求后，通知服务器去调用另外一个web资源进行处理。  
  注：这个是服务器行为，服务器把客户端对一个Servelt的请求处理交给另一个Servlet来处理，对客户端来说，它只发了一个请求。
* request对象提供了一个getRequestDispatcher方法，该方法返回一个RequestDispatcher对象，调用这个对象的forward方法可以实现请求转发。
* request对象同时也是一个域对象，开发人员通过request对象在实现转发时，把数据通过request对象带给其它web资源处理。
  + setAttribute方法
  + getAttribute方法
  + removeAttribute方法
  + getAttributeNames方法

示例代码如下：



这里实际上是由RequestDemo7进行的处理。

如果直接访问RequestDemo7的话，会出现空指针异常，因为没有属性“p”

5分56秒