# Servlet内容补充

1. **两种响应方式（输出流输出和跳转）不能混用，要么用输出流输出，要么直接跳转，不能一部分输出一部分跳转**。

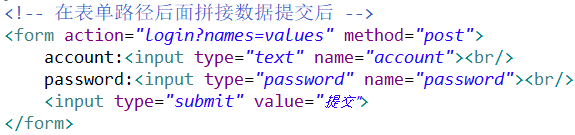
2.WebContent目录下的文件是可以直接访问的。

3. 表单提交上的数据是用getParameter方法，getAttribute拿的数据是setAttribute提交上的(即作用域中的数据)。 parameter的数据是客户端发送过来的，而attribute则是服务端的。

4.一个Sevlet类是可以配置多个访问路径的。在注解中配置多个访问路径如下所示：



5.**注意在form表单提交数据时，若在action路径后面拼接数据则不能采用get方式提交数据，否则拼接的数据会被表单数据覆盖掉。而采用post方法提交则可以。但是不建议在form表单的路径后面拼接数据**。





注意如果表单中提交数据的方式是post方法，自己对数据的操作是在doGet()方法中进行的，若doPost()方法中没有调用doGet()方法则对数据的操作是无效的。**自己有的时候会忘记在doPost()方法中调用doGet()方法**。

6.当某些数据不想在网页上显示又需要被提交时，可以使用**hidden**的方式：。

注意使用jquery隐藏某个div时采用的使**hide()**方法。

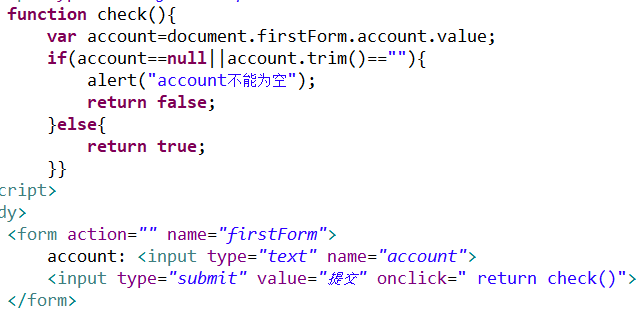
js方法传递字符串，记得添加''

## 一、表单数据通过校验才提交

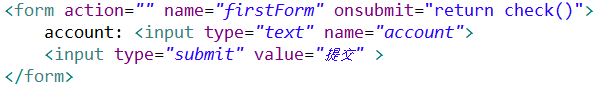
**法一**：将submit按钮改为button类型。点击按钮校验正确自己提交数据。



**法二**：使用submit按钮，给鼠标点击事件添加 **onclick=”return** **xxx()”**函数。



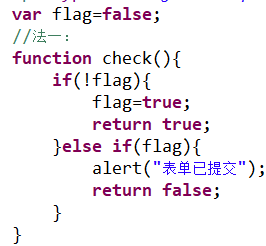
**法三**：使用submit按钮，给form表单添加onsubmit事件(点击submit按钮就会触发该事件)，**onsubmit=”return xxx()”**函数。

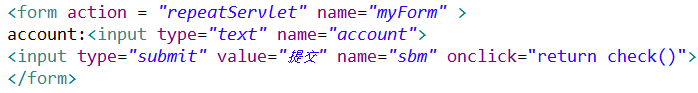


**注意form表单中的action为空时，form表单中的数据默认提交给自己。**

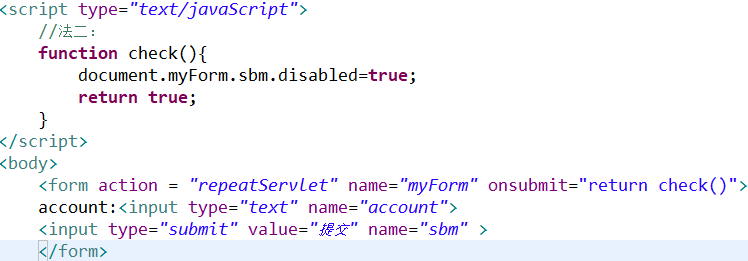
## 二、解决表单数据重复提交

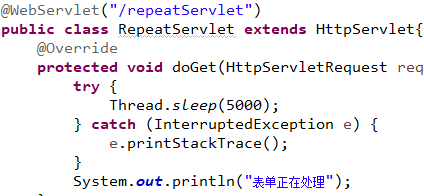
**法一**：





**法二**：将提交按钮点击一次，然后让按钮不可用。**若要想提交按钮失效之后同时数据也会提交要使用onsubmit事件**。





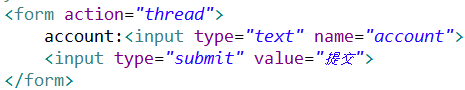
注意给提交按钮设置为disabled不能直接采用下面这种形式，下面这种只会点击一下按钮之后，按钮失效，同时也会终止提交数据。

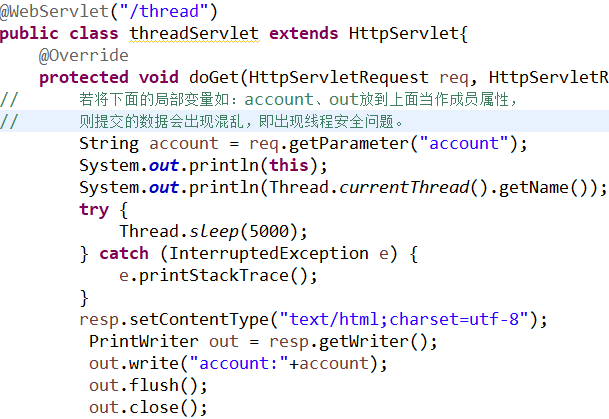


## 三、Servlet的线程安全问题(重点)

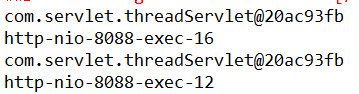
### 1.Servlet是多线程单实例

**注意Servlet类是以多线程的方式处理请求，每一个请求都是一个线程，但是使用的都是同一个对象**。**Servlet是多线程单实例(多个线程共同使用同一个Servlet对象)**，这样会造成线程安全问题。





在同一个浏览器打开两个窗口或者不同浏览器中发送请求，执行上面的代码可以得到类似如下的结果：即可以发现使用的是同一个Servlet对象，但是线程确是不同的。



将Servlet中方法的局部变量变为成员属性会造成线程安全问题。

### 2.解决Servlet线程安全问题

**法一：**给方法(如：doGet())加上同步**synchronized**可以解决线程安全问题，但是加上同步之后，会出现服务器只有处理完一个请求之后才会处理下一个请求的现象，这会造成用户等待响应的时间过长，导致用户体验极差。

**法二：**将Servlet类实现**SingleThreadModel**接口，但是会导致每一个请求(每一个线程)都创建一个Servlet对象，会消耗服务器的资源。



**法三：**推荐针对用户有关的数据，不要使用全局变量。

## 四、Cookie

**如果浏览器的响应是304则说明是从缓存中拿的数据，而不是从服务器中拿到的最新数据**。

**Cookie是服务器端保存在客户端的一些信息**。浏览器访问对应的网站的时候，会自动将cookie提交上去。

**Cookie具有不可跨域名性：**如浏览器访问Google只会携带Google的Cookie，而不会携带Baidu的Cookie。**浏览器判断一个网站是否能操作另一个网站Cookie的依据是域名**。Google与Baidu的域名不一样，因此Google不能操作Baidu的Cookie。

浏览器提交Cookie时，只会提交Cookie中的name和value属性的值。

#### 1.常用方法和构造器

查看源码看Cookie的构造器有哪些。

Cookie的构造器中的参数都是String类型**new Cookie(String name，String value)**，这说明**Cookie只能存储String类型的数据**。

Cookie**.setMaxAge(int num)**：设置Cookie的有效时间，以**秒**为单位。**如果时间为正数，则该Cookie在maxAge秒之后失效，即使关闭了浏览器，只要还在有效时间内该Cookie信息就依然存在。如果为负数，该Cookie为临时Cookie，关闭浏览器即失效，浏览器也不会以任何形式保存该Cookie。如果为0，表示删除该Cookie。默认为–1。**

HttpServletResponse**.addCookie(Cookie cookie)**：将Cookie对象添加到respose中。

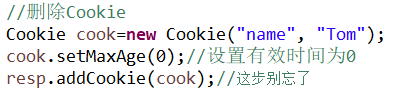
HttpServletRequest**.getCookies()**：返回一个Cookie数组。

Cookie**.getName()**：以String类型返回cookie的name

Cookie**.getValue()**：以String类型返回cookie的value值。

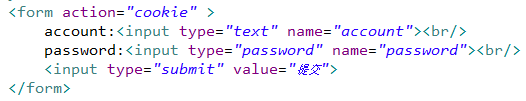
Cookie并不提供修改、删除操作。如果要修改某个Cookie，只需要新建一个同名的Cookie，添加到response中覆盖原来的Cookie。若要删除某个Cookie，只需要将Cookie的maxAge设置为0即可。

#### 2.创建Cookie对象





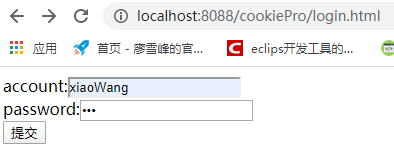
#### 3.自动登录的例子：

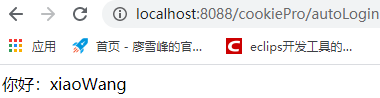






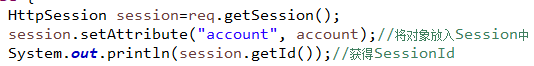
点击下面的提交按钮之后，服务器中将account、password的内容保存到Cookie中。关闭浏览器，重新打开，在地址栏中输入地址：localhost:8088/cookiePro/autoLogin，则可以实现自动登录，并在页面上显示“你好：xxx”，xxx是Cookie记录的上次保存的内容。





## 五、HttpSession内容补充

每获取一个Session对象服务器会自动创建一个SessionId，当访问时会自动把sessionId提交，该sessionId可以帮服务器识别会话属于哪个session。



**HTTP协议是无状态的协议。一旦数据交换完毕，客户端与服务器端的连接就会关闭，再次交换数据需要建立新的连接。这就意味着服务器无法从连接上跟踪会话。**

1.**session是保存在服务器端的**。客户端(浏览器)访问服务器的时候，服务器把客户端信息以某种形式记录在服务器上，这就是session。

2. 针对一个web项目，一个浏览器是共享一个session，就算有两个web项目部署在同一个服务器上，针对两个项目的session是不同的。

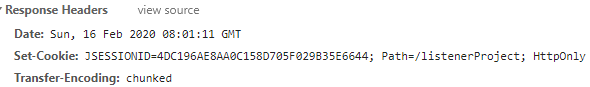
3. session是基于Cookie技术实现，重启浏览器后再次访问原有的连接依然会创建一个新的session，因为默认情况下Cookie在关闭浏览器后就会消失，但是原来服务器的session还在，只有等到了销毁的时间才会自动销毁。

4. 服务器一般把session放在内存里。每个用户都会有一个独立的session。

5. session生成后，只要用户继续访问，服务器就会更新session的最后访问时间，并维护该session。用户每访问服务器一次，无论是否读写session，服务器都认为该用户的session“活跃（active）”了一次。

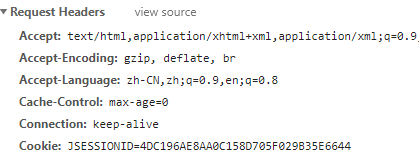
## 六、HttpSession和Cookie之间的联系

1.当浏览器第一次发送请求时，若访问的Servlet中调用了req.getSession()方法或访问的是jsp页面，则服务器会创建session对象，此时服务器将会以响应的方式给客户端传入一个JSESSIONID=ID的值，该ID就是session对象对应的sessionId，可以在Response Headers中查看到。同时服务器自动在客户端中创建一个Cookie对象，该Cookie保存的key就是JSESSIONID，value就是ID，该Cookie的MaxAge属性一般为-1，即表示关闭浏览器Cookie就会失效。



2.当浏览器第二次发送请求时，浏览器会携带Cookie中JSESSIONID=ID的值的信息，在Request Headers中可以看到。然后服务器就可以根据请求中的JSESSIONID即sessionId找到与之相关联的session对象。只要该浏览器不关闭，之后的每次请求(不管是请求静态页面还是动态)都会携带这个Cookie中的JSESSIONID的信息。

sessionId是服务器和客户端链接时候随机分配的，一般来说是不会有重复，但若有大量的并发请求，也是有可能重复的。



## 七、HttpSession和Cookie的区别

1. Cookie和session都是会话技术，Cookie是运行在客户端，session是运行在服务器端。Cookie通过在客户端记录信息确定用户身份，session通过在服务器端记录信息确定用户身份。

2. Cookie有大小限制以及浏览器中存储的Cookie的个数也有限制。session是没有大小限制，和服务器的内存大小有关。

3. Cookie有安全隐患，通过拦截或在本地文件得到Cookie后进行攻击。若主要考虑安全则应使用session。

4.session是保存在服务端的，保存一定时间才会被销毁，若session过多则会对服务器造成压力。若主要考虑服务器性能则使用Cookie。

5.Cookie只能存储String类型的数据，而session则可以存储任意类型的数据。

## 八、报异常

**Cannot forward after response has been committed**异常，有如下解决办法：

**原因一**：往输出流中写了东西，同时又执行了跳转。

比如在文件流中输出了内容，同时又采用服务器或者客户端跳转。

**原因二**：PrintWriter out中调用了print()方法，同时也进行了跳转。

我们写的servlet类，就是用来响应用户的访问请求的，而out.print()是要把内容输出到(响应)页面，也是对用户访问的响应。若在跳转之前有out.println()相当于已经response响应了用户请求，所以再req.forward(req,resp)就会提示已经重复提交。

**原因三**：有可能调用了父类的doGet()和doPost()方法，super.doPost(request,response)。

serive()方法执行过程中判断用户访问方式（post、get等）后，调用相应的doGet()、doPost()方法去响应用户。当显示调用父类的doGet()、doPost()方法，会先执行父类的doGet()、doPost()方法，这就等于已经响应了一次，然后又去跳转又去响应，就会报错重复提交。所以去掉super.doPost(request,response)。

**原因四**：其它

解决办法：在执行客户端或服务器跳转的代码后面不要再写其它代码，同时加上return语句。