# 回顾

# 1.访问控制

存有一个大问题:后台界面是可以直接访问,这个访问我们需要进行控制

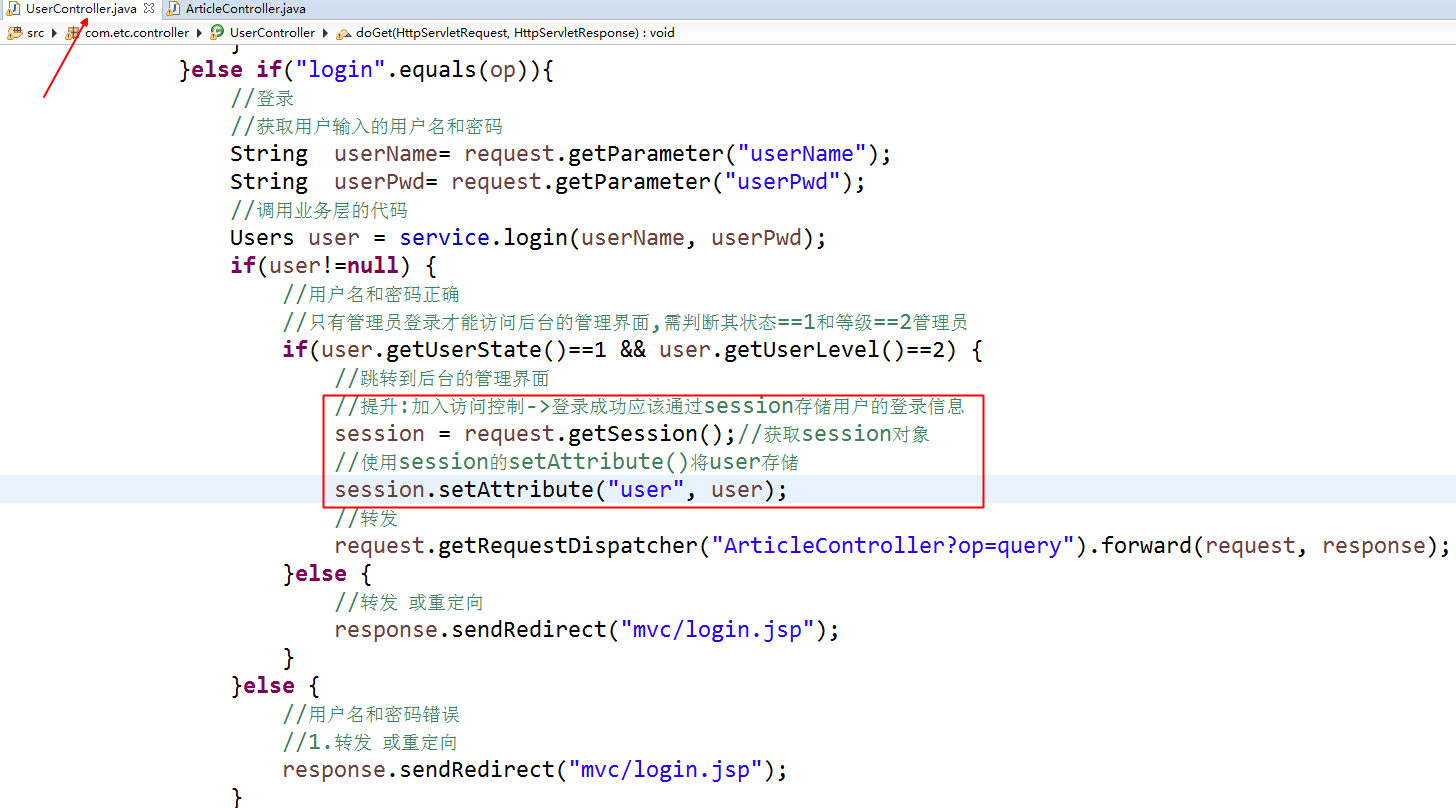
访问控制:添加商品,评论….都需要登录,称为访问控制 session

当用户登录后,会将用户信息存储到session.setAttribute(“name”,value)起来,只要浏览器没有关闭,存储信息一直都在,当浏览器关闭之后,存储的信息就消失了,再重新访问页面的话,就要重新登录了

只有登录的用户才能访问(session.getAttribute(“name”))网站的后台,如果没有登录,则跳转登录页面去.

直接应用:

1. 登录成功应该通过session存储用户登录信息



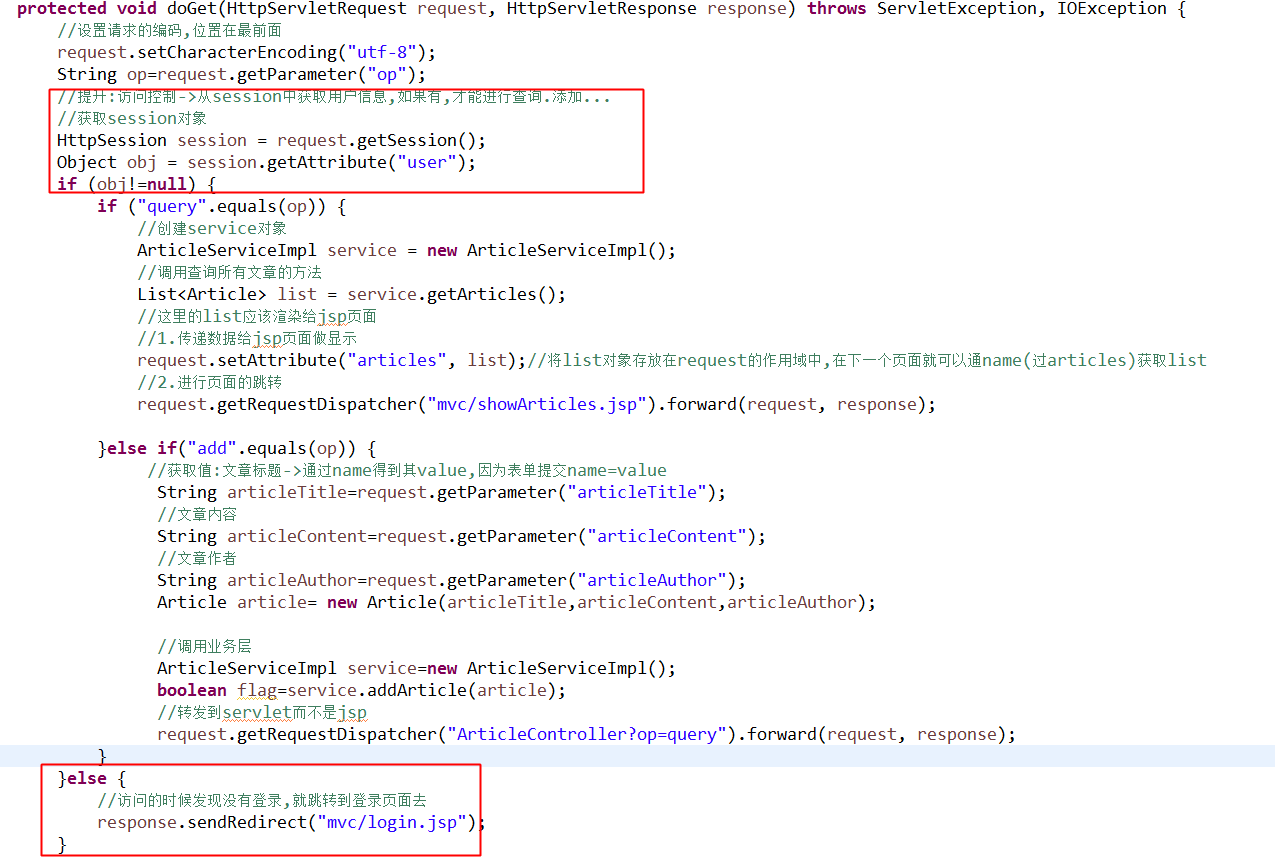
1. 后台ArticleController->判断是否有session信息

UsersController





ArticleController



1. 此时访问<http://localhost:9999/cms1.2/ArticleController?op=query>时会先跳转到登录页面,除了这个servlet要处理登录访问控制,UserController中的query/add操作也需要加入访问控制,

# 2. jsp如何传递参数给servlet

1. form表单的提交 addArticle.jsp

<form action=”servlet的url地址?参数=值&参数=值”>

<input type=”text” name=”username”/>

</form>

1. 使用url地址直接:

url地址?参数=值&参数=值

1. 使用超链接来提交

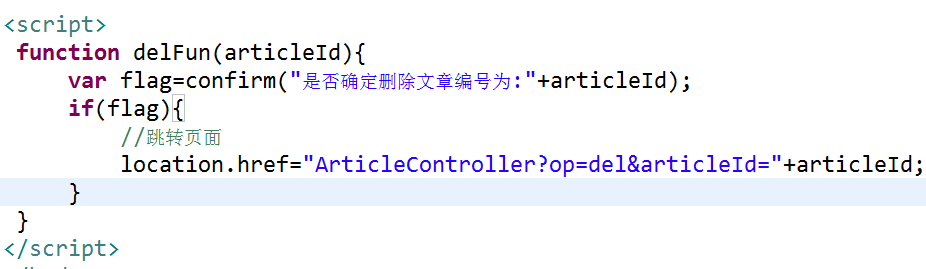
<a href=” url地址?参数=值&参数=值”>图片,文字 按钮</a>



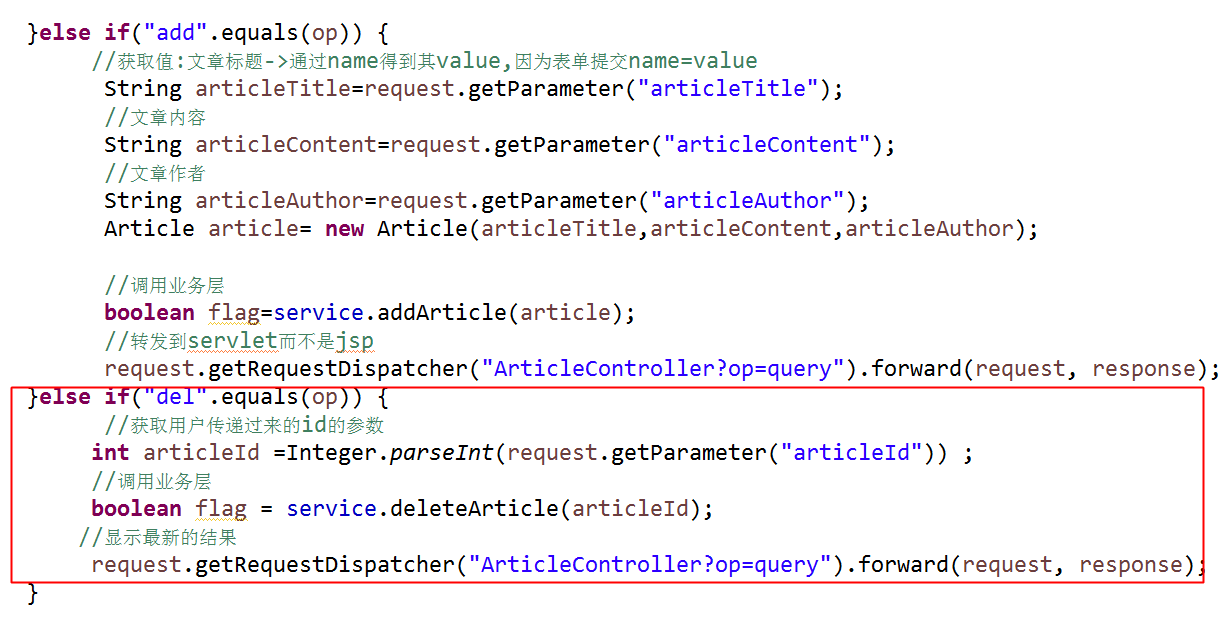
# 删除功能

showArticles.jsp

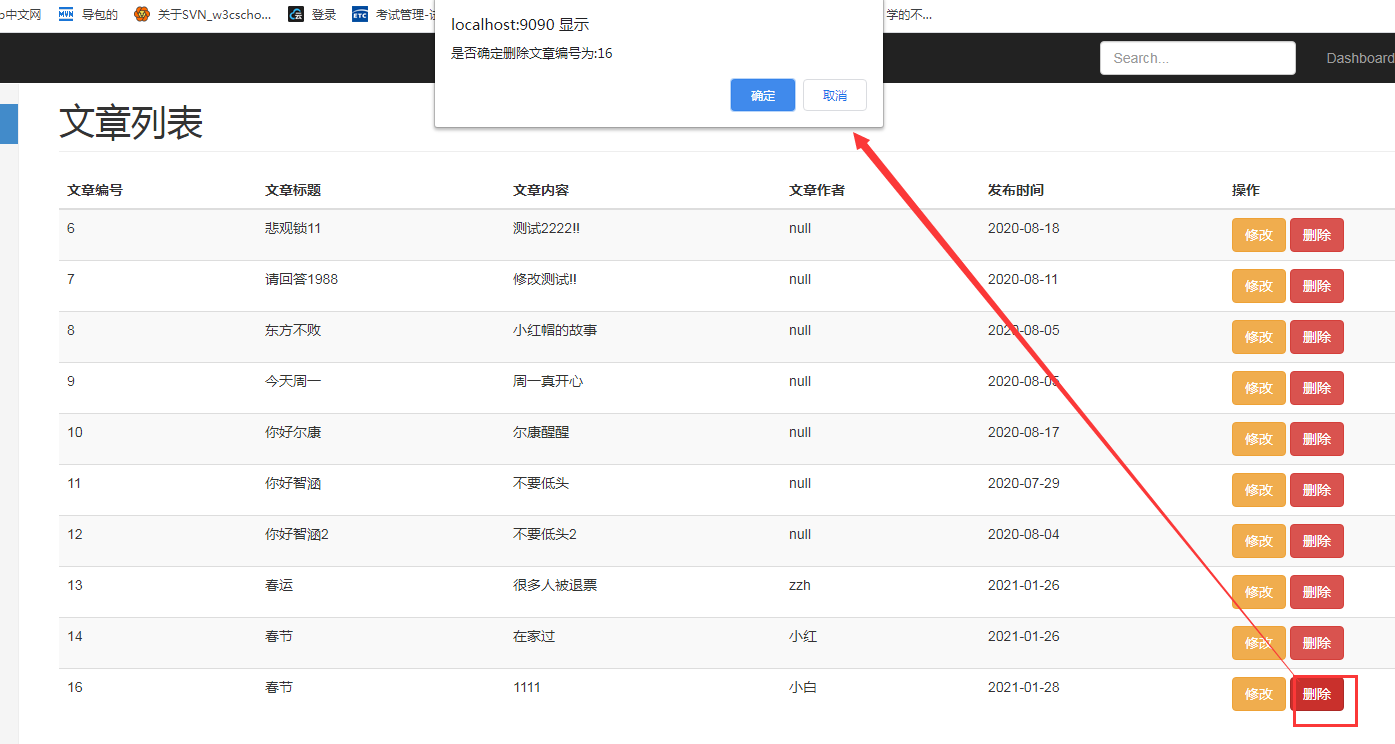




控制器:



效果:

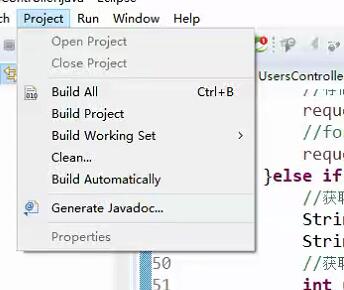
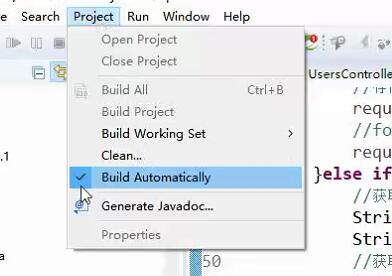


# 4.清除缓存方法

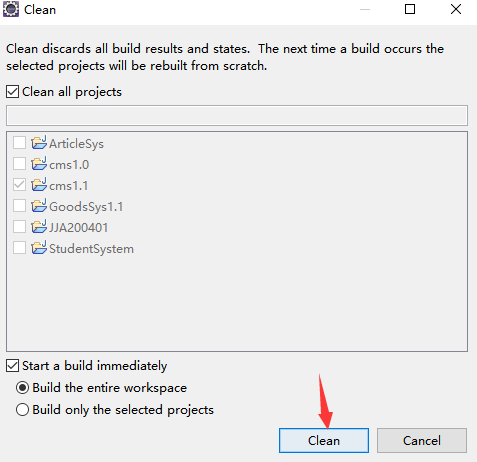
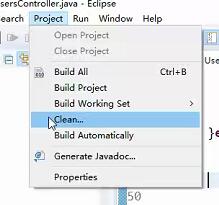
1. 把eclipse中server下的项目先remove掉
2. 找到tomcat的webapps目录,把你自己部署的项目删掉



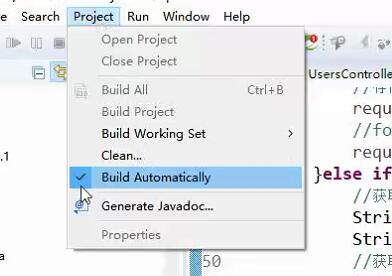
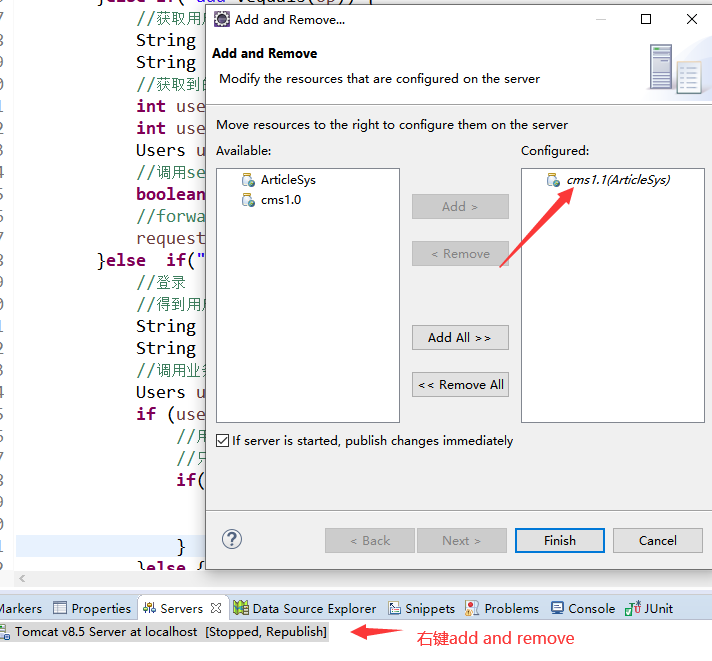
1. 找到tomcat的word目录,把localhost目录直接删掉
2. 把eclipse中project的build Automatically的勾去掉



1. 点击project中的clean,直到底部eclipse图标不是绿色之后



1. 重新部署cms1.1,再把project的build Automatically的勾上



7.启动tomcat

# 总结

1. 案例的用户查询与增加:mvc架构(每一个模块都有一个servlet),上下文路径使用

<%String contextPath=request.getContextPath(); %>

<%=contextPath%>/.... 跳转页面的顺序

1. 登录:登录权限状态正常并且等级管理员才可以登录,登录完后把用户存储session(会话)
2. 清除缓存:按讲义步骤
3. Jsp传递参数给servlet:form/url/a
4. 删除功能:直接使用a标签的传值

# 6.JSP内置对象

内置：jsp本身就包含，且不用去实例化可以直接使用(web容器) 。

### 2.1.10九大内置对象总体说明:

我们发现通过jsp得到的那个.java文件(本质上就是一个servlet文件);这个文件包含了jsp的内置对象

|  |  |  |
| --- | --- | --- |
| Jsp内置对象名(直接使用) | 类型(原始类型) Servlet中 | 基本使用 |
| request | HttpServletRequest | 请求处理(设置请求编码，获取请求参数，设置(获取)属性,设置页面的转发) |
| response | HttpServletReponse | 响应的处理(页面的跳转，设置响应的编码,设置操作cookie的信息) |
| session | HttpSession(request.getSession()) | 会话(设置属性信息) |
| out | jspWriter(servlet中可以使用response.getWriter来获取) | 页面打印输出 |
| application | ServletContext | 全局对象，可以设置其属性信息，在这个**应用程序范围**内都可以访问,容器重新启动才消失 |
| page | Object | 页面，代指当前的servlet对象 (this) ,基本用不到 |
| pageContext | PageContext | 页面上下文,可以设置属性setAttribute() getAttribute() |
| exception | Throwable | 异常处理有关(页面出现exception) |
| Config | ServletConfig | 处理和配置有关的操作 |

内置对象依次介绍:

### 2.1.2Request对象:

request对象是httpServletRequest类型的对象。代表客户端的请求信息，主要作用接受通过http协议传送到服务器的数据。

1. Request的常见方法：
2. 可以直接平移到Servlet中来:

3.作用域：可以在请求的下一个页面(servlet)中获取值,如果**一直**转发,那么数据就一直都可以访问(传递)。

### 2.1.3Response:原型

response 代表的是对客户端的响应，主要是将JSP容器处理过的对象传回到客户端。

原型:HttpServletResponse

页面重定向的使用:

同样可以复制到servlet中去使用.

### 2.1.4 Session会话

1 Session对象是一个JSP内置对象，它在第一个JSP页面被装载时自动创建，完成会话期管理。从一个客户打开浏览器并连接到服务器开始，到客户关闭浏览器离开这个服务器结束，被称为一个会话。当一个客户访问一个服务器时，可能会在这个服务器的几个页面之间切换，服务器应当通过某种办法知道这是一个客户，就需要Session对象。

常见的其他方法：

servlet:

原型:HttpSession

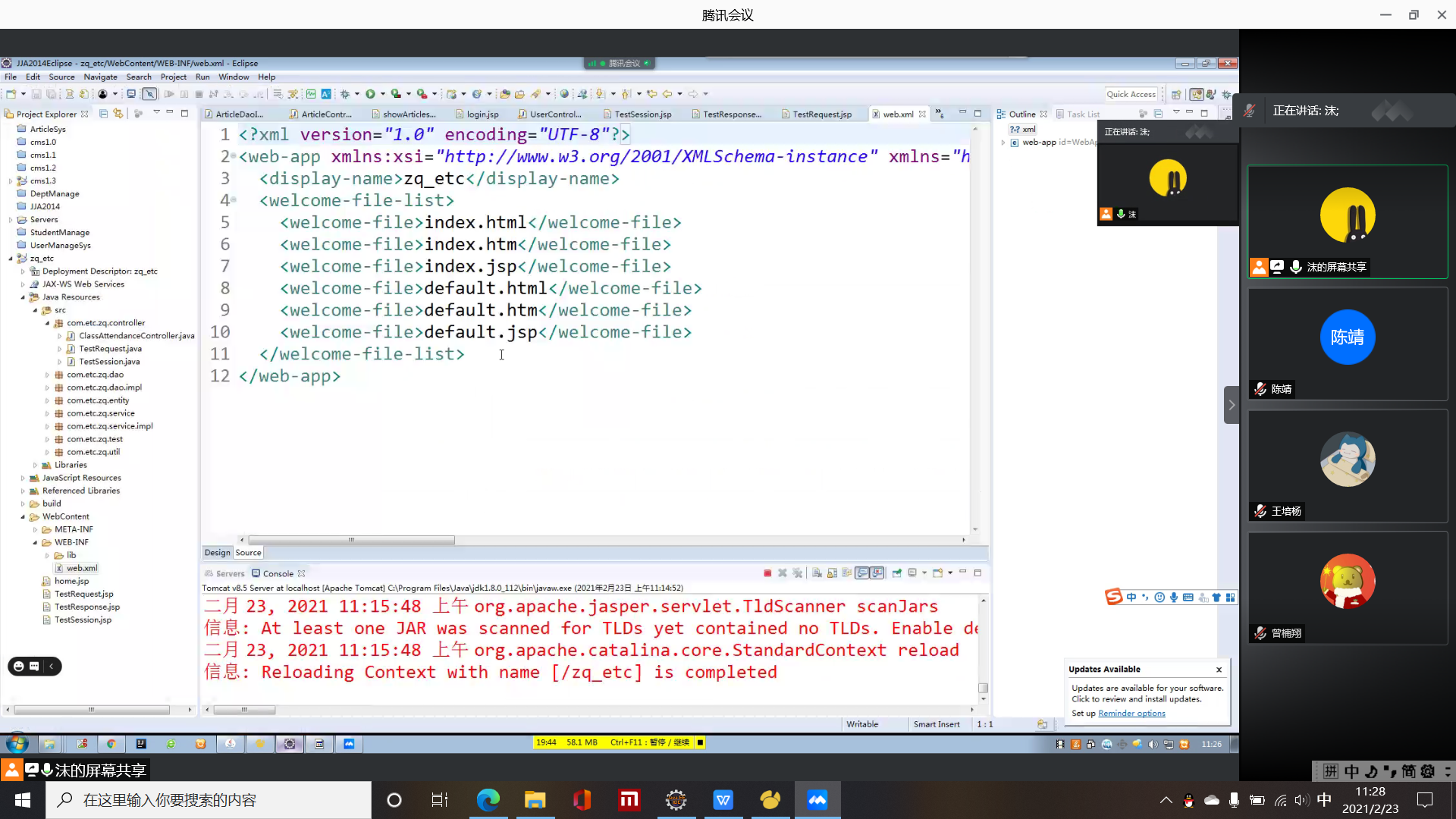
会话:当我们打开浏览器来访问一个网站,一个会话 就开始了,只要浏览器没有关闭,会话就一直存在,当浏览器关闭,会话结束;

其他页面:

2.作用域:会话期间

只要你的浏览器没有关闭,则在网站任何页面(jsp,servlet)都可以获取session中的值;当用户第一次访问url的时候,就会自动生成一个jsessionId存在本地的cookie;

同一个浏览器访问,jsessionid应该是一致的



### 2.1.5 session超时

Session超时理解为：浏览器和服务器之间创建了一个Session，由于客户端长时间（休眠时间）没有与服务器交互，服务器将此Session销毁，客户端再一次与服务器交互时之前的Session就不存在了。

设置Session超时时间方式：

方式一：

在web.xml中设置session-config如下：

<session-config>

<session-timeout>2</session-timeout>

</session-config>

即客户端连续两次与服务器交互间隔时间最长为2分钟，2分钟后session.getAttribute()获取的值为空

session.getCreationTime() 获取session的创建时间

session.getLastAccessedTime() 获取上次与服务器交互时间

session.getMaxInactiveInterval() 获取session最大的不活动的间隔时间，以秒为单位120秒。

方式二：

在Tomcat的/conf/web.xml中session-config,默认值为：30分钟

<session-config>

<session-timeout>30</session-timeout>

</session-config>

方式三：

在Servlet中设置

HttpSession session = request.getSession();

session.setMaxInactiveInterval(60);//单位为秒

说明：

1.优先级：Servlet中API设置 > 程序/web.xml设置 > Tomcat/conf/web.xml设置

### 2.1.6 session应用场景(访问控制)

1)下载文件，判断用户是否登录，如果有登录过，则可以下载否则提示登录。



2)只有登录的用户才能进入网站的后台

### 2.1.7 out对象

out对象：页面打印输出

原型:JspWriter

Servlet中使用out:

### 2.1.8 Application对象

类似于系统的“全局变量”，用于在同一个服务器内的所有用于之间的数据共享，对于整个web服务器，application对象有且只有一个实例。

原型:ServletContext:servlet上下文

常见方法：

|  |  |
| --- | --- |
| void setAttribute(String key,Object value) | 以key/value的形式保存对象值 |
| Object getAttribute(String key) | 通过key获取对象值 |

可以在其他jsp中也写这句也可以获取得到。

就算把浏览器关了，依然有该属性值。

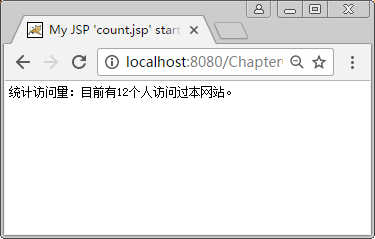
属性值什么时候消失?容器(Tomcat)关闭后,重启

使用setAttribute设置了属性之后,在当前的整个应用程序中都可以使用。

### 2.1.8 Application对象应用场景

如何实现网站的用户统计功能？

简单的记录访问网站的在线人数:



统计：

<%

Integer count = (Integer) application.getAttribute("count");

**if**(count != **null**){

count += 1;

}**else**{

count = 1;

}

application.setAttribute("count", count);

%>

显示：

<%

Integer i = (Integer) application.getAttribute("count");

out.print("统计访问量：目前有"+ i +"个人访问过本网站。");

%>

### 2.1.9 PageContext

主要作用是获取其他对象:

### 2.1.10 Page对象

表示当前页，类似java中的this。在JSP页面中使用较少。

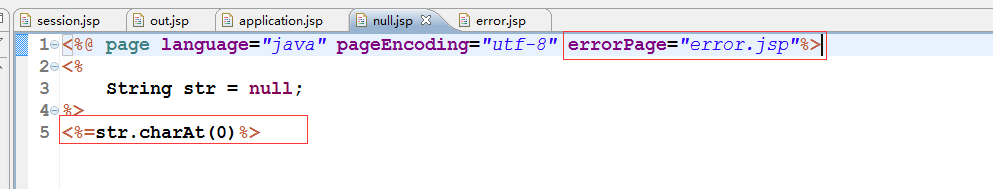
Servlet当前类的对象

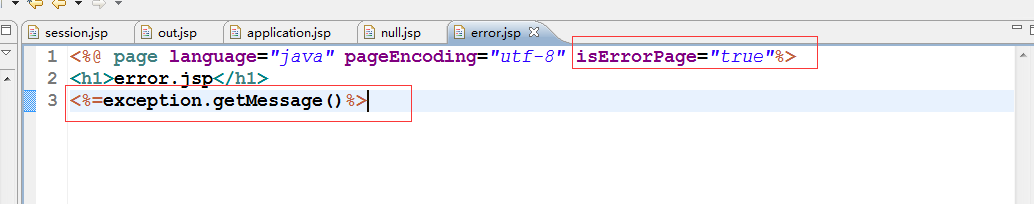
单例

原型:Object

### 2.1.11 Exception对象

|  |
| --- |
| 表示JSP页面运行时产生的异常和错误信息，该对象只有在错误页面中才能使用。错误页面：page指令中isErrorPage=”true”的页面。 |





### 2.1.12 Config对象

1.原型:ServletConfig

用于存放JSP编译后的初始数据。在JSP页面中使用较少。

2. 模拟创建一个servlet来设置config参数值

|  |
| --- |
|  |
| 在本页面得到初始化参数的值 |
|  |
| 新建一个servlet:看看能不能访问刚才那个servlet的值:  说明了这个config对象设置的初始化参数,只能在servlet有效,设置临时文件存储目录等  能不能有一个办法,能对所有的servlet都有效  Application->ServletContext |

## (面试题)转发和重定向

1. 转发(request.getRequestDispatcher(“index.jsp”).forward()) ;操作的时候浏览器地址栏还停留在原来的url地址，而页面内容已经显示的是index.jsp的内容;而重定向(response.sendRedirect(“index.jsp”)) ;此时浏览器地址栏和显示的内容页是一致，都是index.jsp.

转发:

此时,地址栏没变,内容已经是跳转后的那个页面中的内容

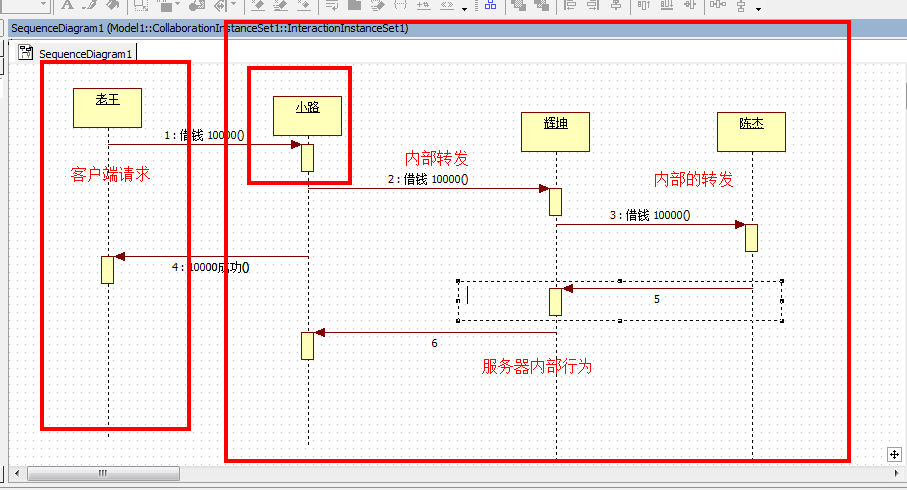
重定向:

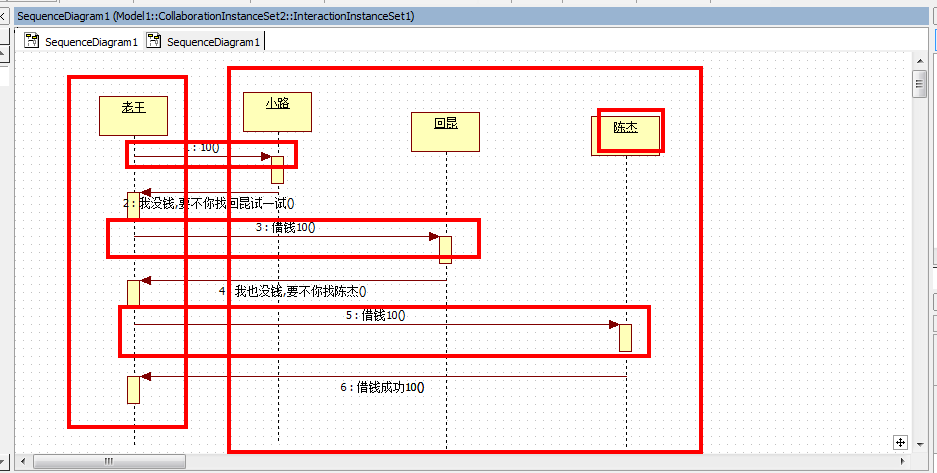
地址栏发生改变,是重定向到那个页面的地址

1. 转发的时候可以携带数据（request.setAttribute(“key”,value)）;之后可以在新的页面获取(request.getAttribute(“key”)) ;而重定向不能使用setAttribute()/getAttribute()的操作来设置或者获取数据. 传递数据只能通过url直接传递;

Index.jsp:

3、二者的原理不同: 转发是服务器内部的行为,对于请求者来说，根本不关注内部的行为只关注自己发送请求的响应; 重定向其实是浏览器重新发送的请求,多次请求者和服务器进行交互的操作.





4、相对来讲，因为是服务器内部行为,内部转发，所以效率会高一些.

转发是服务器**内部**的行为，所以转发页面也是同一个网站的不同部分;但是不能跳转到外部网站;而重定向可以是内部或者外部的网站url.

转发:



重定向:

5、使用场合:如果需要携带数据(很多对象)到下一个请求地址，则可以使用转发，否则可以使用重定向.

## 关于中文乱码

### 1.4.1 post请求

后台接受表单数据,需要进行编码格式

### 1.4.2 get请求

模拟一个简单的get请求:

也设置了请求的编码，但是其实对get是无效的

直接url访问servlet，并传递了一个中文的字符串:

在控制台的输出结果就是中文乱码：

出现这个问题的原因是，浏览器传递数据的时候，如果使用get,对传递的数据进行编码(iso-8859-1) ;

我们解决的方案是将这个编码重新设置为支持中文的编码格式(utf-8 gbk gb2312等)

解决方案1，如下，直接对接收到的字符串进行转码：

解决方案2：修改web服务器的配置: server.xml

useBodyEncodingForURI=”true” 或者 URIEncoding=”UTF-8”

二者的差别： useBodyEncodingForURI 页面的编码是什么，浏览器发送get请求的时候的编码就是什么 ;

URIEncoding :直接这是uri的编码是固定的值，和页面本身是什么编码没有关系. UltraEdit



常见乱码处理：

中文乱码处理:

* + - 1. html页面? <meta charset=”utf-8”>
      2. JSP页面? <%page pageEncoding=”utf-8”%>
      3. 传递数据post提交? Request. setCharacterEncoding("utf-8");
      4. 传递数据get提交? 修改tomcat server.xml 加入URIEncoding=”UTF-8”
      5. 从servlet响应的数据jsp乱码处理:response. setCharacterEncoding("utf-8");

# 2.2 属性作用域

有一些对象 *setAttribute(“key”,valu*e) ; 设置了属性之后,在哪里可以使用getAttribute(“key”);得到他的值呢?

|  |  |  |
| --- | --- | --- |
| 对象名 | 属性范围说明 | 场景 |
| pageContext | 在当前页中使用 |  |
| request | 在请求的下一个页面(servlet)中获取值,如果一直转发,那么数据就一直可以访问 | 例如:AticleController中执行query后,转发到showArticle.jsp,此时要传递数据,数据可以放在resquest请求范围内 |
| session | 在会话的范围一直有效,只要会话生存时间存在,或者浏览器没有关闭,数据就可以一直访问,不管页面之间是转发还是重定向 | 例如:用户登录,将用户登录信息保存在session会话中,购物车商品信息,临时存储可以存在session |
| application | 在整个应用程序中都有效,直到容器停止了,例如tomcat停止 | 例如:统计网站整体信息,例如访问统计操作等 |

request作用域：

# 作业

文章和用户的模糊查询

预习:configjstl,jsp的标签库+el表达式