JSP概述

2019年5月22日 星期三 9:01

1. JSP概述

由于在servlet中书写前台页面代码非常复杂，于是Sun提出一门新的技术--jsp，将所有的页面内容重新书写到一个文件中，在这个文件中还可以书写java语句，这样的一个文件就称之为一个jsp文件。

在jsp页面内开发HTML代码十分便利，还可以在其中嵌入java语句，使开发变得非常简洁，直接在页面中就完成了页面构建和代码逻辑的全部内容。

2. JSP原理

在jsp页面被初次访问的时候，会被jsp翻译引擎翻译成一个servlet文件，这个文件是一个普通java文件，继承了一个具有servlet接口的类，所以当前文件才会成为一个servlet文件。

实验：寻找如下文件：

在[tomcat]/work/localhost/day14/index_jsp.java文件

JSP语法

2019年5月22日 星期三 9:33

1. JSP语法
   a. 模板元素

jsp页面中的HTML内容会在浏览器发生访问的时候，被jsp翻译引擎翻译为Servlet文件的模板元素。

```
<body>
  <div>aaaa</div>
</body>
```

```
out.write("   <body>\r\n");
out.write("     <div>aaaa</div>\r\n");
out.write("   </body>\r\n");
```

其实就是将HTML内容使用out.write()输出.

   b. 脚本表达式

语法：<%= 脚本表达式(具有返回值的表达式，或一个直接量) %>
在jsp页面被访问的时候，jsp翻译引擎将脚本表达式中的内容翻译为了原封不动的表达式结果值。

```
<h3>脚本表达式</h3>
<%=request.getContextPath() %>
</body>
```

```
out.write("     ");
out.print(request.getContextPath() );
out.write("\r\n");
```

翻译后的结果值直接放在servlet文件的对应位置，并使用out.print()输出表达式的值。

   c. 脚本片段

语法：<% 脚本片段(java语句) %>
在JSP页面被访问的时候，其中脚本片段中的内容会原封不动的放在翻译后的servlet文件制定位置。

```
<h3>脚本表达式</h3>
<%=request.getContextPath() %>
<h3>脚本片段</h3>
<%
    for(int i=0;i<5;i++){
    %>

    <font color='red'>I am li shuai;</font>

    <%
    }

 %>
```

```
    for(int i=0;i<5;i++){

  out.write("\r\n");
  out.write("   \t\t\r\n");
  out.write("   \t\t<font color='red'>I am li shuai;</font>\r\
n");
  out.write("   \t\t\r\n");
  out.write("   \t\t");

    }
```

在jsp页面中脚本片段可以分开书写，分开书写脚不片段必须要保证java语法完整，因为在jsp页面中的java语句会被翻译到servlet文件中，需要保证java语句在servlet文件中的完整性。

d. JSP声明

语法：<%! JSP声明 %>

在jsp页面被访问的时候，其中的jsp声明会被jsp翻译引擎翻译为当前servlet的成员变量，成员方法，静态代码块等内容。

```
<h3>JSP声明</h3>
<%! int i =0; %>
<%! public void m(){} %>
<%!static{} %>
```

```
public final class index_jsp
HttpJspBase
    implements org.apache.ja

 int i =0;
 public void m(){}
static{}
```

被翻译后的成员内容都会出现在类内方法外，作为成员内容使用。

e. JSP注释

<%-- JSP注释--%>

<%//abc%>	被翻译成java注释内容，由于是注释内容所以不会输出任何内容
<!-- def-->	HTML注释被翻译成模板元素，由于是HTML注释，所以不会输出任何内容，但是

	仍然会出现在页面中。
<%--ghi--%>	在翻译过程中直接被抛弃，不会出现在翻译后的servlet文件中。

f. JSP指令

语法：<%@指令名称 %>

指令本身不会产生任何的输出内容，是用来控制jsp翻译引擎如何来翻译jsp页面的。

指令内容：

page	<%@page %> 用户指定当前页面依赖哪些配置翻译页面。
include	<%@include %> 在当前jsp页面引入其他jsp页面
taglib	<%@taglib %> 标签技术

page指令：

language="java"	当前JSP使用的开发语言
extends="package.class"	当前jsp翻译成servlet后要继承的类，注意此值必须是一个servlet的子类，一般情况下不要改
import="{package.class | package.*}, ..."	导入需要使用到的包 java.lang.*;javax.servlet.*;javax.servlet.jsp.*;javax.servlet.http.*;
session="true | false"	用来指定当前页面是否使用session，如果设置为true，则翻译过来的servlet中将会有对session对象的引用，于是可以直接在jsp中使用session隐式对象。但是这将导致一旦访问jsp就会调用request.getSession()方法，可能导致不必要的空间浪费。如果确定jsp中不需要session可以设为false
[buffer="none | 8kb | sizekb"]	out隐式对象所使用的缓冲区的大小
[autoFlush="true | false"]	out隐式对象是否自动刷新缓冲区，默认为true，不需要更改
[isThreadSafe="true | false"]	翻译过来的servlet是否实现SingleThreadModel
[errorPage="relative_url"]	如果页面出错，将要跳转到的页面，除了在jsp中使用此属性指定错误页面外也可以在web.xml中配置整个web应用的错误页面，如果两个都设置则jsp中的此属性起作用 <error-page> <error-code>404</error-code> <location>/error/404.jsp</location> </error-page> <error-page> <error-code>500</error-code> <location>/error/500.jsp</location> </error-page>
[isErrorPage="true | false"]	如果设置此属性为true,翻译过来的servlet中将含有Exception隐式对象,其中封装的就是上一个页面中抛出的异常对象
[contentType="mimeType [;charset=characterSet]" | "text/html ; charset=ISO-8859-1"]	和jsp乱码相关的指令,用来指定jsp输出时,设置的Content-Type响应头用来指定浏览器打开的编码
[pageEncoding="characterSet | ISO-8859-1"]	服务器翻译jsp时使用的编码集.如果向防止jsp乱码,应该保证文件的保存编码和jsp翻译成servlet用的编码以及输出到浏览器后浏览器打开的编码一致.此属性一旦设置好,翻译引擎会间接帮我们设置content-type属性.

g. include指令

在当前的jsp页面中引入其他的jsp页面，将组合之后的页面输出在浏览器。

<%@include file=""%>

h. taglib指令

标签技术

<%@taglib %>

2. 九大隐式对象

在jsp翻译成一个servlet后，其中都可以包含9大隐式对象，其中Exception、session需要手动开启。在每一个jsp页面中都会有自己独立的9大隐式对象，每一个隐式对象所具有的变量名可以直接使用，在自行定义新的变量时 应避免与已有隐式对象名称重复。

面试题经常考：

九大隐式对象：

```
page ---  代表当前Servlet的对象
request ---  代表当前请求的对象
response ---  代表当前响应的对象
session  ---  代表当前会话的对象
application ---  代表当前web应用的对象
config ---  代表当前servlet配置信息的对象
exception ---  代表当前页面产生的异常信息对象
out  ---  代表当前jsp页面向浏览器输出数据的对象
pageContext ---  代表当前jsp页面上下文的对象
```

3. PageContext详解

a. 功能一：可以作为其他八大隐式对象的入口。

```
getException方法返回exception隐式对象
getPage方法返回page隐式对象
getRequest方法返回request隐式对象
getResponse方法返回response隐式对象
getServletConfig方法返回config隐式对象
getServletContext方法返回application隐式对象
getSession方法返回session隐式对象
getOut方法返回out隐式对象
```

b. 功能二：域对象

相关的方法：

setAttribute()

getAttribute()

removeAttribute()

getAttributeNames()

生命周期：

当前jsp页面被访问的时候创建，当前jsp访问结束的时候销毁。

作用范围：

当前jsp页面。

主要功能：

当前jsp页面内共享数据。

c. 功能三:可以作为其他三大作用域的入口对象使用：

scope是int类型的一个值，代表作用域.

setAttribute(String name,Obejct obj,int scope);

getAttribute(String name,int scope);

removeAttribute(String name,int scope)

getAttributeNamesInScope(int scope)

PageContext.APPLICATION_SCOPE	ServletContext
PageContext.SESSION_SCOPE	Session
PageContext. REQUEST_SCOPE	request
PageContext.PAGE_SCOPE	page

获取指定域中的数据：

```
<h4>获取指定域当中的数据</h4>
    page： <%=pageContext.getAttribute("country", PageContext.PAGE_SCOPE) %>
    request： <%=pageContext.getAttribute("country",PageContext.REQUEST_SCOPE) %>
    session： <%=pageContext.getAttribute("country",PageContext.SESSION_SCOPE) %>
    application:<%=pageContext.getAttribute("country",PageContext.APPLICATION_SCOPE) %>
```

直接获取pageContext域属性：

```
<h4>直接书写pageContext获取的是pageContext域</h4>
    <%=pageContext.getAttribute("country") %>
```

删除指定域中属性：

```
<h4>删除指定域中属性</h4>
    <%
            pageContext.removeAttribute("country", PageContext.PAGE_SCOPE);
     %>
```

d. 功能四：便捷的请求转发

```
pageContext. forward("path");
```

四大作用域

2019年5月22日 星期三 14:29

1. 域对象概念

   一个对象身上有一个可以被看见的范围,在这个范围内利用对象身上的map实现资源的共享,像这样一个对象就称之为域对象.

2. 四个域对象

   由大到小的顺序排列:

   ServletContext > Session > Request > PageContext

   web应用 会话 请求 jsp页面

   i. ServletContext

      代表web应用的对象

      生命周期:

      web应用加载的时候ServletContext对象创建，web应用被移出容器ServletContext对象销毁。

      作用范围:

      整个web应用的范围。

      主要功能:

      在整个web应用的范围内实现资源的共享。

   ii. Session

      代表session会话的对象

      生命周期:

      调用request.getSession()方法是session对象创建。

      超时死亡：如果一个session在30分钟内没有任何操作，则服务器认为这个session无人认领，可以销毁。

      自杀：session对象主动调用invalidate()，自行释放。

      意外身亡：服务器非正常关闭的时候session对象也会销毁。如果服务器正常关闭是仍有session对象没有释放，则session中的数据会序列化到本次磁盘上，作为一个文件保存，这个过程叫做钝化，在服务器再次启动的时候会将这个文件读取，这个过程称之为活化。

      作用范围:

      整个会话范围。

      主要功能:

      在整个会话范围内共享数据。

   iii. request

      代表request请求的对象

      生命周期:

      请求链开始的时候request对象创建，请求链结束的时候request对象销毁。

      作用范围:

整个请求链。

主要功能：

在整个请求链中实现数据的共享。

iv. pageContext

代表当前jsp页面上下文的对象

生命周期：

当前jsp页面加载的时候pageContext创建，访问结束时pageContext对象销毁

作用范围：

当前jsp页面。

主要功能：

在当前jsp页面内实现数据的共享。

JSP标签技术

2019年5月22日 星期三 14:59

1. JSP标签技术概述

   jsp当前可以书写HTML内容和Java语句，便于开发，但是面对大量逻辑需要处理和复杂的页面构成的时候，HTML和java的结合就无法做到合理维护和高效开发。这时可以将页面中的java语句都变成标签内容，让标签处理java语句对应的逻辑，也就是说利用标签代替java，这样就可以保证页面内只出现标签内容，这样的页面只有一种语言，相比之前便于维护和关闭，可以提升开发效率。

2. 常用的JSP标签技术
   a. jsp标签技术

      Sun公司开发

   b. el表达式

      便捷易用 - 使用广泛 - 只能获取值 不能设置值。

   c. jstl标签技术

      标签库数据庞大，可以覆盖多数的开发需求。

   d. 自定义标签

   e. 其他第三方标签

EL表达式

2019年5月22日 星期三 14:58

1. EL表达式概述

   EL 全名为Expression Language，用来替代<%= %>脚本表达式

   EL具有获取数据、执行运算、获取常用开发对象、调用java方法这四方面的功能

   javaEE目前内置了EL表达式，可以在jsp页面中直接使用

2. EL表达式使用方式

   ${el表达式}

3. EL表达式的特点

   只能获取数据

   不能设置数据

   不能遍历数据

4. 功能一：获取数据

   a. 获取常量

```
<h4>获取常量</h4>
    脚本片段书写：<br/>
    <%=123 %>
    <%="abc" %>
    <%=true %>
    <%=1.1 %>
    <br/>el表达式书写：<br/>
    ${123}
    ${"abc" }
    ${true }
    ${1.1 }
```

   b. 自动搜寻域

   所谓自动搜寻域就是从最小范围的域中开始检索一个指定名称的域属性，如果在较小范围内找到了这个域属性，则直接将找到的域属性返回，当前范围的域中如果没有指定名称的域属性，则向更大的域中去搜寻，直到找到指定名称的域属性。

   总结：由小到大的搜索域中的属性。

```
<h4>自动搜寻域</h4>
<%
    application.setAttribute("name", "兰刚");
    session.setAttribute("name","朴乾");
    request.setAttribute("name","李帅");
    //pageContext.setAttribute("name","张三");

 %>
 <%=pageContext.findAttribute("name") %>

 ${name }
```

   el表达式自动搜寻域，只需要指定一个域属性的名称即可，它会自动在四个作用域中寻找指定名称的域属性。

   c. 获取指定域中的数据

```
<h4>获取指定域中的属性</h4>
    <%
        application.setAttribute("job", "IT");
        session.setAttribute("job","teacher");
        request.setAttribute("job", "doctor");
        pageContext.setAttribute("job", "work");
     %>
     <%=pageContext.getAttribute("job", PageContext.APPLICATION_SCOPE) %>
    el表达式获取指定域的属性：
    ${applicationScope.job}
    ${sessionScope.job }
    ${requestScope.job }
    ${pageScope.job }
```

el表达式中域名称书写形式：

脚本片段	el表达式
APPLICATION_SCOPE	applicationScope
SESSION_SCOPE	sessionScope
REQUEST_SCOPE	requestScope
PAGE_SCOPE	pageScope

d. 获取数组中的元素

el表达式可以获取数组中指定下标的元素，但是数据必须要在某一个域中，才可以获取其中的数据。

```
<h4>el表达式数组中获取数据</h4>
<%
    String[] names = {"殷天正","黛琦丝","谢逊","韦一笑"};
    pageContext.setAttribute("names", names);
 %>
${names[1]}
```

获取一个元素的时候，需要指定元素的下标。

e. 获取集合中的元素

```
<h4>获取集合中的元素</h4>
    <%
        List<String> list = new ArrayList<String>();
        list.add("大数据");
        list.add("小数据");
        list.add("中数据");
        pageContext.setAttribute("list", list);
     %>
     ${list[0]}
```

集合中获取元素和数组中获取元素的方式类似，都可以通过指定一个下标来获取元素。

f. 获取Map中的元素

```
<h4>获取Map中的元素</h4>
    <%
        Map<String,String> map = new HashMap<String,String>();
        map.put("name", "李帅");
        map.put("武功","黯然销魂掌");
        map.put("wife","姑姑");
```

```
        map.put("wife.one","小龙女");
        pageContext.setAttribute("map", map);
        /* 书写一个字符串数据添加到域中，即可获取变量名中包含的数据 */
        pageContext.setAttribute("wife", "wife");

    %>
    ${map["name"] }
    ${map.武功 }
    ${map[wife] }
    ${map["wife.one"] }
```

g. 获取javabean中属性

```
<h3>获取javabean中的元素</h3>
    <%
        Person p = new Person();
        p.setName("lishuai");
        p.setAge(18);
        p.setAddr("bj");
        pageContext.setAttribute("p", p);

    %>
    ${p.name }
    ${p.age }
    ${p.addr }
```

5. 执行运算

算术运算：

```
<h3>el表达式功能二：运算</h3>
    <h4>算术运算</h4>
    ${2+3 }
    ${2-3 }
    ${3/2 }
    ${3%2 }
    <hr>
    ${2+"3" }
    <hr>
    ${"2"+"3" }
    在加法中如果有字符串类型的数据，尝试将这个数据转换成一个数值类型，如果能够转换则转换过后执行运算，如果不能转换则会报错。
    <%-- ${2+"a" } --%>
```

关系运算：

```
<h4>关系运算</h4>
    ${3>2 }
    ${3<2 }
    ${3>=2 }
    ${3<=2 }
    ${3!=2 }
```

逻辑运算：

```
<h4>逻辑运算</h4>
    ${true and false }
    ${true or false }
    ${not(true) }

    ${true && false }
    ${true || false }
```

```
${!true}
```

Empty运算：

如果对象为null，字符串保存的值为""，数组和集合以及map中没有数据，使用empty运算的结构都为true，其他任何形式都为false.

```
<h4>empty运算</h4>
    <%
        String a = null;
        String str = "aaaa";
        String[] arr = {"1"};
        ArrayList list = new ArrayList();
        list.add("abc");
        Map map = new HashMap();

        pageContext.setAttribute("a",a);
        pageContext.setAttribute("str",str);
        pageContext.setAttribute("arr",arr);
        pageContext.setAttribute("list",list);
        pageContext.setAttribute("map", map);

     %>
    ${empty a}
    ${empty str}
    ${empty arr}
    ${empty list}
    ${empty map}
```

三元表达式：

```
<h4>三元表达式</h4>
    <%
        Map<String,String> mapx = new HashMap<String,String>();
        mapx.put("a","aa");
        mapx.put("b","bb");
        mapx.put("c","cc");
        pageContext.setAttribute("mapx", mapx);

     %>
     ${empty mapx.b?"yes":"no" }
```

6. 获取常用开发对象

EL内置的11个内置对象。

a. 代表域的内置对象

pageScope	代表pageContext域
requestScope	代表request域
sessionScope	代表session域
applicationScope	代表ServletContext域

b. 代表请求参数的内置对象

param	代表当前请求中的请求参数信息，获取的结果组成Map<String, String>
paramValues	代表当前你请求中的请求参数信息，获取的结果组成Map<String,String[]>

c. 代表请求头的内置对象

header	代表请求中的请求头信息，获取的结果组成Map<String,String>
headerValues	代表请求中的请求头信息，获取的结果组成Map<String,String[]>

d. 代表web应用初始化配置信息的内置对象

initParam	代表当前web应用的初始化配置信息的对象

e. 代表当前请求头中的cookie组成的内置对象

cookie	代表当前请求头中的cookie，获取的结果组成Map<String, Cookie>

f. 代表pageContext（页面中的上下文）的内置对象
pageContext

7. 调用java方法

a. 略

jstl标签技术

2019年5月24日 星期五 9:23

1. JSTL标签库概述

JSTL全称为JavaServerPages Standard Tag Library。

由JCP（Java Community Process）指定标准。

是提供给 Java Web 开发人员一个标准通用的标签函数库。

可以和 EL 配合来取代传统直接在页面上嵌入 Java 程序（Scripting）的做法，以提高程序可读性、维护性和方便性。

2. 在页面中引入jstl标签库

java EE 5包及其以上版本都已经支持jstl标签库，所以在页面中直接书写指令引入使用即可。

<%@taglib uri="" prefix=""%>

3. JSTL标签库的子库

核心标签库 core - c

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

国际化标签 fmt

数据库标签 sql

XML标签 xml

JSTL函数(EL函数)

4. 命名空间或名称空间概念

在uri中包含两部分内容，一个是url，是常见的location地址保存形式。还有一个URN保存的内容是一个名称，这个名称看起来像是一个地址，但是本质存储的是一个Naming，所以把这中命名方式称之为名称空间或者命名空间。

5. 核心函数库

a. c:set标签

设置或修改域中的值，

设置或修改map中的值，

设置修改javaBean中的数据

```
<h4>c:set标签，设置或修改值</h4>
    <c:set var="a" value="bj" scope="request"></c:set>
    ${a }
    <c:set var="a" value="sh" scope="request"></c:set>
    ${a}
<h3>c:set设置或修改map中的值</h3>
    <%
        Map<String,String> map = new HashMap<String,String>();
        pageContext.setAttribute("map", map);
     %>
     <c:set target="${map }" property="addr" value="重庆"></c:set>
    ${map }
```

```
    ${map.addr}
    <c:set target="${map }" property="addr" value="宝安"></c:set>
    ${map.addr }
<h4>c:set设置或修改javabean中的属性</h4>
    <%
        Person p = new Person("李帅",18,"北京");
        pageContext.setAttribute("p",p);
     %>
     <c:set target="${p }" property="name" value="朴乾"></c:set>
    ${p.name }
    ${p.age }
    <!-- javabean中不存在的属性不能直接设置值 -->
    <%-- <c:set target="${p }" property="gender" value="男"></c:set>--%>
    <%-- ${p.gender }  --%>
```

b. c:remove 删除指定域中的域属性

```
<h3>c:remove</h3>
  <h4>c:remove删除指定域中的域属性</h4>
  <c:set var="name" value="张三丰" scope="request"></c:set>
  ${name}
  <br>
    <c:remove var="name" scope="request"></c:remove>
    调用remove： ${name}

<h4>删除全部域中的域属性</h4>
    <%
        application.setAttribute("namex","大一");
        session.setAttribute("namex","大二");
        request.setAttribute("namex","大三");
        pageContext.setAttribute("namex","大四");

     %>
    <%-- <c:remove var="namex"></c:remove> --%>
     <c:remove var="namex" scope="request"></c:remove>

     ${applicationScope.namex}
     ${sessionScope.namex}
     ${requestScope.namex}
     ${pageScope.namex}
```

c. c:if判断

```
<h3>c:if</h3>
     <h4>c:if判断</h4>
    <c:set var="i" value="10" scope="request"></c:set>
    <c:if test="${i<=10 }" var="flag">
        no~~~
    </c:if>

    <c:if test="${i>10 }" var="f">
        yes
    </c:if>
    ${flag}
    ${f}
```

d. c:catch 捕获异常

```
<h3>c:catch</h3>
    <h4>c:catch捕获异常信息</h4>
    <c:catch var="e">
        <%
            String str = null;
            str.toUpperCase();
         %>
    </c:catch>
    ${e }
```

e. c:choose 多重判断

可以实现一个类似于if else的判断结构。

```
<h3>c:choose</h3>
    <h4>c:choose实现多重判断</h4>
    <c:set var="b" value="18" scope="request"></c:set>
    <c:choose >
        <c:when test="${b<10 }" >小于10</c:when>
        <c:when test="${b<100 }">大于10小于100</c:when>
        <c:when test="${b<1000 }">大于100小于1000</c:when>
        <c:otherwise>大于1000</c:otherwise>
    </c:choose>
```

多重判断的结果不需要使用变量展示，满足c:when或者从c:otherwise结果会打印在页面当中。

f. c:forEach 遍历循环

可以循环指定数据内容，也可以遍历一个数据结构。

forEach中有varStatus属性，属性可以引用的内容如下：

属性	类型	意义
index	number	现在指到成员的索引
count	number	总共指到成员的总数
first	boolean	现在指到的成员是否为第一个成员
last	boolean	现在指到的成员是否为最后一个成员

i. 如果是利用自身数据遍历:

```
<c:forEach begin="10" end="100" step="1" var="o" varStatus="stat">
    ${stat.index }
</c:forEach>
```

循环中会认为10即为第十个下标。打印结果为 10 11 12

ii. 如果是遍历其他数据结构：

```
<%
    List<String> list = new LinkedList<String>();
      list.add("danny");
      list.add("jenny");
      list.add("liming");
      list.add("laowang");
      pageContext.setAttribute("list", list);
 %>
<c:forEach items="${list }" var="h" varStatus="stat">
    ${h }
    ${stat.index }
</c:forEach>
```

循环中下标会从0开始计数,打印结果为 danny 0 jenny1 . . . . .

```
<h3>c:forEach实现循环</h3>
    <h4>c:forEach循环数字</h4>
    <c:forEach begin="1" end="100" step="1" var="q">
        ${q }
    </c:forEach>

    <h4>c:forEach遍历链表</h4>
    <%
        List<String> list = new LinkedList<String>();
          list.add("danny");
          list.add("jenny");
          list.add("liming");
          list.add("laowang");
          pageContext.setAttribute("list", list);
     %>
     <c:forEach items="${list }" var="h">
          ${h }
     </c:forEach>
     <c:forEach begin="10" end="100" step="1" var="o" varStatus="stat">
          ${stat.index }
     </c:forEach>
     <br>
    问题：遍历一个10到100的全部偶数，如果当前总数的下标为3的倍数，则将这个数据变为红色。
    <br>
     <c:forEach begin="10" end="100" step="2" var="r" varStatus="s">
          <c:if test="${s.count%3 == 0}">
              <font color="red">${r }</font>
          </c:if>
          <c:if test="${s.count%3 != 0 }">
              <font color="blue">${r }</font>
          </c:if>
     </c:forEach>
```

修改EasyMall页面

2019年5月24日 星期五 11:13

1. 将页面中的java代码替换为el表达式和jstl标签

替换各个页面中的: <%=request.getContextPath()%>	${pageContext.request.contextPath}
修改regist.jsp <%=request.getAttribute("msg")==null?"": request.getAttribute("msg")%>	${requestScope.msg} 或者: ${msg}
修改login.jsp <% =request.getParameter("username")==null?"":request .getParameter("username")%>	${param.username}
修改regist.jsp <% =request.getParameter("nickname")==null?"":request .getParameter("nickname")%>	${param.nickname}
修改regist.jsp <% =request.getParameter("email")==null?"":request.get Parameter("email")%>	${param.email}

修改login.jsp
<%
Cookie remnameC = null;
Cookie[] cs = request.getCookies();
if(cs != null){
for(Cookie c:cs){
if("remname".equals(c.getName())){
remnameC = c;
}
}
}
String username = "";
if(remnameC !=null){
username =
URLDecoder.decode(remnameC.getValue(),
"utf-8");
}
 	
%>
	1.添加jquery和js代码
	<script type="text/javascript" src="js/jquery-1.4.2.js">
	</script>
	<script type="text/javascript" >
	//从cookie中读取数据remname
	window.onload = function(){
	$("#username").val(decodeURI('${cooki
	e.remname.value}'));
	
	}
	</script>
	2.将input[name='username']中的value属性删除。

修改login.jsp <%= username==""?"":"checked='checked'"%>	1. 先导入jstl标签库的核心标签库 2. <input type="checkbox" name="remname" value="true" <c:if test="${not empty cookie.remname }"> checked="checked" </c:if> />记住用户名
修改head.jsp <% if(request.getSession(false)!=null && request.getSession().getAttribute("usern	1. 先导入jstl标签库的核心标签库 2. <c:if test="${sessionScope.username == null }"> <a href="${pageContext.request.contextPath }/lo

```
ame") != null){
%>
<a href="#">欢迎 <%
=request.getSession().getAttribute("user
name") %> 回来</a>  
|  
<a
href="${pageContext.request.contextPath
}/LogOutServlet">注销</a>
<%
    }else{
    %>
<a
href="${pageContext.request.contextPath
}/login.jsp">登录</a>  
|  
<a
href="${pageContext.request.contextPath
}/regist.jsp">注册</a>
    <%
    }
 %>
```

```
gin.jsp">登录</a>  | 
 
      <a
      href="${pageContext.request.contextPat
      h }/regist.jsp">注册</a>
 </c:if>
 <c:if test="${sessionScope.username !=
null }">
        <a href="#">欢迎 
${sessionScope.username } 回来
</a>  |  
      <a
      href="${pageContext.request.contextPat
      h }/LogOutServlet">注销</a>
 </c:if>
```