# 

**JSP学习笔记**

**Java相关课程系列笔记之十**

**笔记内容说明**

JSP（程祖红老师主讲，占笔记内容100%）；

**目 录**

[一、 JSP基础](#_Toc8328)

[1.1什么是JSP](#_Toc8576)

[1.2为什么要使用JSP](#_Toc24369)

[1.3 JSP与Servlet的关系](#_Toc22208)

[1.4如何写一个JSP文件](#_Toc18139)

[1.5 JSP是如何运行的](#_Toc30752)

[1.6隐含对象](#_Toc3520)

[1.7指令](#_Toc1541)

[1.8 JSP注释](#_Toc4474)

[1.9案例：创建emplist.jsp页面，将表示逻辑交给JSP处理](#_Toc29505)

[二、 JSP标签和EL表达式](#_Toc24235)

[2.1什么是JSP标签](#_Toc21150)

[2.2 JSTL及其使用](#_Toc23262)

[2.3什么是EL表达式](#_Toc16832)

[2.4 EL表达式的使用](#_Toc19048)

[三、 JSTL中的几个核心标签](#_Toc1019)

[3.1 if](#_Toc31311)

[3.2 choose](#_Toc5404)

[3.3 forEach](#_Toc22153)

[3.4 url](#_Toc31540)

[3.5 set](#_Toc14095)

[3.6 remove](#_Toc6107)

[3.7 catch](#_Toc17041)

[3.8 import](#_Toc31890)

[3.9 redirect](#_Toc31701)

[3.10 out](#_Toc9001)

[3.11 JSP标签是如何运行的](#_Toc30470)

[3.12案例：将员工列表中的Java代码改为JSP标签，并添加分页](#_Toc12881)

[四、 自定义标签](#_Toc24188)

[4.1如何写一个自定义标签](#_Toc30659)

[4.2 JavaEE5.0中，如何使用EL表达式和JSTL](#_Toc27335)

[4.3案例：自定义标签](#_Toc19620)

[4.4案例：修改之前员工列表中的日期显示](#_Toc20477)

[五、 MVC](#_Toc32275)

[5.1什么是MVC](#_Toc29344)

[5.2使用MVC的目的](#_Toc25338)

[5.3如何使用MVC来开发一个Web应用程序（JavaEE）](#_Toc2203)

[5.4 MVC的优缺点](#_Toc20892)

[5.5案例：简易贷款（贷款数小于余额数\*10）](#_Toc12060)

[5.6修改5.5案例，使用户无法直接访问view.jsp页面](#_Toc31360)

# 一JSP基础

## 1.1 JSP概述

### 1.1.1 什么是JSP

JSP（Java Server Page）是sun公司制订的一种服务器端的动态页面生成技术规范，是运行在服务器端的动态页面技术。

JSP是一个以”.jsp”为后缀的文件，在该文件中，主要是HTML和少量的java代码，同时JSP会被容器转换成一个Servlet类。

### 1.1.2 为什么使用JSP

直接使用Servlet，虽然也可以生成动态页面。但是，编写繁琐：需要使用out.println来输出。并且维护困难，如果页面发生了改变，需要修改Java代码，所以sun指定了JSP规范。

### 1.1.3 JSP与Servlet的关系

JSP其实是一个以.jsp为后缀的文件，**容器**会自动将.jsp文件转换成一个.java文件。其实就是一个Servlet，然后调用该Servlet。所以，从本质上讲，JSP其实就是一个Servlet。

## 1.2创建JSP文件

step1：创建一个以“.jsp”为后缀的文件

step2：在该文件里面，可以添加如下的内容

（1）HTML（CSS、JS）：控制页面在浏览器中的显示效果

1. HTML 标记

2. CSS样式

3. JavaScript

（2）Java代码：控制页面中可变的内容

1. Java代码片段：**<%** Java代码 **%>** 控制页面

2. JSP表达式：**<%=** Java表达式 **%> 替换out.println()**

3. JSP声明：<**%!**  **%> 在service方法外声明一个方法**

4. 隐含对象：在jsp中直接可以使用的对象out,request,reponsse

（3）指令： 1. 语法 <%@指令名 属性=值%>

2. 作用 通知容器，控制jsp文件转译成java时添加一些设定

## 1.3 JSP中的Java代码

### 1.3.1 JSP表达式

语法规则：**<%=**……**%>**

合法内容：变量、变量加运算符组合的表达式、有返回值的方法

转译规则：在service()方法中用out.print语句输出该变量、表达式、方法的值

### 1.3.2 JSP小脚本

语法规则：<%……%>

合法内容：能够写在方法里的Java代码片段都可以作为小脚本

转译规则：原封不动成为Servlet类的service()方法里面的一段代码

### 1.3.3 JSP声明

语法规则：<%!……%>

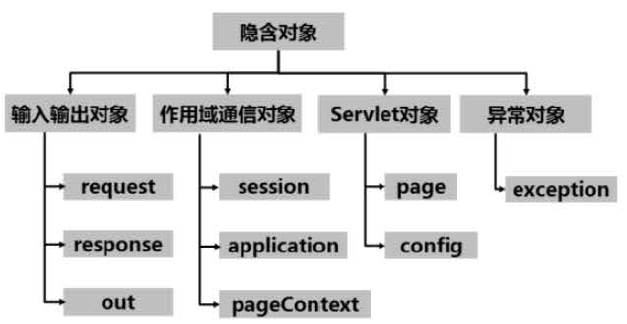
合法内容：成员属性或成员方法的声明

转译规则：成为JSP页面转译成的Servlet类中成员属性或成员方法

### 1.3.4 JSP隐含对象

隐含对象：容器自动创建，在JSP文件中可以直接使用的对象

对象作用：JSP预先创建的这些对象可以简化对HTTP请求、响应信息的访问



隐含对象的功能如下：



**exception对象**：当jsp页面运行时发生了异常，容器会将异常信息封装到该对象，可以使用该对象提供的方法来输出异常的信息。另外，必须在页面设置isErrorPage="true"指令才能使用该隐含对象。

**servletContext对象**：容器会为每一个servlet实例，创建唯一的一个符合servletContext接口的对象，称之为servlet上下文。该对象会一直存在，除非servlet实例被销毁。它作用：绑定数据：setAttribute、removeAttribute、getAttribute，servletContext可以被应用所有组件访问

**pageContext对象**：与servletContext对象对象，pageContext相对JSP页面来说的全局上下文，容器会为每一个JSP实例创建唯一一个符合pageContext接口的对象。从pageContext中可以获取任意一种其他的对象

config对象：即ServletConfig，可以使用该对象获得初始化参数

page对象：JSP实例本身，一般不用

## 1.4 JSP中的指令

语法规则：<%@指令名称 属性名=属性值 %>

指令作用：控制JSP在转译成Servlet类时生成的内容

常用指令：

1. page指令

2. include指令

3. taglib指令

### 1.4.1 page指令

Page指令主要作用：用于导包，设置页面属性

**import属性**：用于导包，多个包以逗号隔开，都在一个双引号里

例如：<%@page import="java.util.\*"%>

<%@page import="java.util.\*,java.text.\*"%>

**contenType属性**：设置response.setContentType的参数值，属性之间用空格隔开

例如：<%@page import="java.util.\*" contentType="text/html;charset=utf-8" %>

**pageEncoding属性**：设置容器读取.jsp的文件的解码格式

例如：<%@page contentType="text/html;charset=utf-8" pageEncoding="utf-8" %>

**session属性**：true/false，设置容器是否添加session语句。缺省值true

例如：<%@page session="false" %>

sessionID：<%=session.getId() %><!-- 报错：session cannot be resolved -->

**isELIgnored属性**：true/false，设置容器是否忽略el表达式。缺省值true，如果值为false，则告诉容器不要。J2EE5.0需要使用isELIgnored="false"，否则EL表达式无效。

**isErrorPage属性**：true/false，缺省值false，如果值为true，表示这是一个错误处理页面：即专门用来处理其他JSP产生的异常，只有值为true时，才能使用exception隐含对象去获取错误信息。

例如：<%=exception.getMessage() %>

errorPage属性：**设置**一个错误处理页面。

step1：a3.jsp页面，测试输入a3.jsp?num=100a

<%@page errorPage="a4.jsp" %>

<% String num=request.getParameter("num");

int sum=Integer.parseInt(num)+100;

out.println(sum); %>

step2：a4.jsp页面

<%@page isErrorPage="true" pageEncoding="UTF-8" contentType="text/html;charset=utf-8"%>

<%=exception.getMessage() %><!--获取到错误信息：For input string: "100a" -->

### 1.4.2 include指令

include指令主要作用：

1. 在JSP页面转换成Servlet时，能够将其他文件包含进来

2. 可以包含JSP文件也可以是静态的HTML文件，

3. 方便的在每个JSP页面中包含导航栏、版权声明、logo等

4. 插入的页面并未运行，而是机械的将内容插入

指令语法：[%@include file="head.jsp" %](mailto:%25@include%20file=%22head.jsp%22%20%25)

### 1.4.3 taglib指令

taglib指令：导入JSP标签<%@taglib uri="命名空间" prefix="前缀" %>

uri：在standard.jar/META-INF/c.tld中查找

prefix：前缀，用于代表命名空间

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

<c:if test="${1>0}">今天运气太好了</c:if>

## 1.5 JSP运行机制

### 1.5.1 代码转换

step1：容器依据.jsp文件生成.java文件，也就是先转换成一个Servlet

HTML（CSS、JS）放到service方法里，使用out.write输出

<% %>也放到service方法里，照搬，不改动。

<%= %>也会放到service方法里，使用out.print输出。

<%! %>给Servlet添加新的属性或者新的方法

这样就把一个JSP变成了一个Servlet容器。

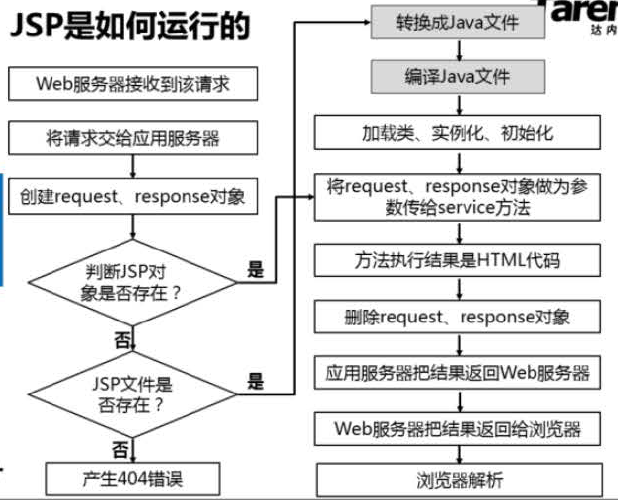
注意事项：out.writer方法只能输出简单的字符串，对象是没法输出的。优点是把null自动转换成空字符串输出。

### 1.5.2 servlet请求

step2：容器接下来就会调用Servlet来处理请求了，会将之前生成的.java文件进行编译、然后实例化、初始化、调用相应的方法处理请求

### 1.5.3 创建隐含对象

隐含对象之所以能直接使用这些对象，是因为容器会自动添加创建这些对象的代码。JSP仅仅是个草稿，最终会变为一个Servlet。



## 1.6 JSP注释

### <!-- 注释内容 -->

允许注释的内容是Java代码，如果是Java代码，会被容器执行，但是执行的结果会被浏览器忽略（不会显示出来）

### <%-- 注释内容 --%>

注释的内容不能是Java代码，如果是Java代码，会被容器忽略。

# 二、JSTL标签和EL表达式

## 2.1 JSP标签

sun公司制定的一种技术规范，利用类似于HTML标签的JSP标签来代替JSP中的Java代码。这样作的好处是，JSP页面会更加简洁，并且更好维护。比如，将这样的页面交给美工，可以很方便地去修改。

## 2.2 JSTL标签

### 2.2.1 JSTL概述

JSTL （JSP Standard Taglib）Java标准标签库，apache开发的一套标签，捐献给了sun后sun将其命名为JSTL。

如何使用：

1. 将JSTL标签对应的jar（标签类）文件拷贝到WEB-INF\lib下，standard.jar、jstl.jar

2. 使用taglib指令引入JSP标签

### 2.2.2 JSTL原理

根据taglib指令中uri+prefix确定唯一的一个空间，确定空间的规则是到jstl.jar包中的tld文件匹配uri+shortname节点。

匹配上之后，根据写的标签名，在tld文件中寻找<tag>标记下与名称相同的<name>节点，找到后查看下面<tag-class>节点中的类名，即该标签背后对应的类。

## 2.3 EL表达式

EL表达式（Expression Language）是一套简单的计算规则，用于给JSP标签的属性赋值，也可以直接输出。

注意事项：新建工程，J2EE选1.4可直接使用EL表达式，JavaEE5.0需要使用isELIgnored属性，否则无法使用详见1.7。

## 2.4 EL表达式的使用

取值+运算

### 2.4.1 EL表达式取值：访问bean属性

通常情况下JavaBean指的就是一个公共类，含有一个空的构造方法、一些属性以及访问这些属性的get/set方法，这些属性名和方法名都需要满足一定的规范。

**第一种方式**： ${绑定名} 直接获取绑定对象 ${绑定名.属性名} ${user.name}

容器会依次从4个隐含对象中pageContext、request、session、application中查找绑定名为"user"的对象。接下来，会调用该对象的"getName"方法（自动将n变大写然后加get），最后输出执行结果。

**特点**：①会自动将null转换成""输出。

②如果绑定名称对应的值不存在，会不报null指针异常，会输出""。

注意事项：依次是指先从pageContext中查找，如果找不到，再查找request，如果找到了，则不再向下查找。如果要**指定查找范围**，可以使用pageScope、requestScope、sessionScope、applicationScope来指定查找的范围。${requestScope.绑定名.属性名}

第二种方式：${绑定名["属性名"]} ${绑定名[“属性名”][“属性名”]}

与第一种方式是等价的，容器会依次从4个隐含对象中pageContext、request、session、application中查找绑定名为"user"的对象。接下来，会调用该对象的"getName"方法（自动将n变大写然后加get），最后输出执行结果。

**特点：**①中括号[]里面可以出现变量。

②中括号[]里面可以出现下标从0开始的数组。

注意事项：

* 中括号[]里的字符串用双引号、单引号都可以
* EL表达式中没引号的为变量，有引号的为字符串

### 2.4.2 EL表达式计算

EL可以进行一些简单的计算，计算的结果可以用来给JSP标签的属性赋值，也可以直接输出。

1. 算术运算：“+”、“-”、“\*”、“/”、“%”，但"+"号操作不能连接字符串

2. 关系运算：“>”、“>=”、“<”、“<=”、“!=”、"=="，同时“eq”也可判断是否相等

3. 逻辑运算："&&"、"||"、"!" ，与Java中的一样

4. empty运算：判断是否是一个空字符串，或者是一个空的集合

### 2.4.3EL表达式获取请求参数值

${param.username}等价于request.getParameter("username");

${paramValues.city}等价于request.getParameterValues("city");

# 

# 三、JSTL核心标签

## 3.1 JSTL常用标签

### 3.1.1 if

语法：<c:if test="" var="" scope=""></c:if>

test属性：当test属性值为true，执行标签体的内容，test属性可以使用EL表达式。

var属性：用来指定绑定名称。

scope属性：指定绑定范围，可以是page（pageContext）、request、session、application

注意事项：可以在if标签里写Java代码。

### 3.1.2 choose

语法：<c:choose><!-- 用于分支，当满足某个条件，执行某一个分支 -->

<c:when test=""><!-- 分支，可多次出现 -->

</c:when>

...

<c:otherwise><!-- 当其他分支都不满足条件，则执行该标签的内容 -->

</c:otherwise>

</c:choose>

注意事项：

when和otherwise必须要放到choose标签里面才能使用。

when可以出现1次或者多次，otherwise可以出现0次或者1次，表例外

### 3.1.3 forEach

用法一：<c:forEach var="" items="" carStatus="">

</c:forEach>

items属性：用来指定要遍历的集合，可以使用EL表达式。

var属性：指定绑定名，绑定范围是pageContext，绑定值是从集合中取出的某个元素。

carStatus属性：指定绑定名，绑定范围是pageContext，绑定值是一个由容器创建的一个对象，该对象封装了当前迭代的状态。比如，该对象提供了getIndex、getCount方法，其中，getIndex会返回当前迭代的元素的下标（从0开始），getCount会返回当前迭代的次数（从1开始）。

用法二：<c:forEach var="" begin="" end="">

</c:forEach>

begin属性：

如果指定了items，那么迭代就从items[begin]开始进行迭代；

如果没有指定items，那么就从begin开始迭代。它的类型为整数。

end属性：

如果指定了items，那么就在items[end]结束迭代；

如果没有指定items，那么就在end结束迭代。它的类型也为整数。

step属性：forEach迭代的步长。

### 3.1.4 url

语法：<c:url value="">

当用户禁止cookie以后，会自动在地址后面添加sessionId

当使用绝对路径时，会自动在地址前添加应用名

value属性：指定地址，在表单提交、链接当中，可以使用该标签。

### 3.1.5 set和remove

语法：<c:set var="" scope="" value="">，绑定一个对象到指定的范围。

value属性：绑定值。

语法：<c:remove var="" scope="">，解除绑定

### 3.1.7 catch

语法：<c:catch var="">，处理异常，会将异常信息封装成一个对象，绑定到pageContext对象

### 3.1.8 import

语法：<c:import url="">

url属性：指定一个jsp文件的地址，jsp会在运行时调用这个jsp。

### 3.1.9 redirect

语法：<c:redirect url="">，重定向到另外一个地址。

url属性：指定重定向的地址。

### 3.1.10 out

语法：<c:out value="" default="" escapeXml="">，用于输出el表达式的值。

value属性：指定输出的值。

default属性：指定缺省值。

escapeXml属性：设置成true，会将value中的特殊字符替换成相应的实体。缺省值就是true。

## 3.2 JSP标签运行机制

1. 容器依据命名空间找到标签的描述文件（.tld文件）

2. 接下来，依据标签的名称找到标签类的类名

3. 然后将该标签实例化

4. 最后，调用标签实例的相应的方法。

注意事项：容器会从WEB-INF下查找，如果找不到，还会查找WEB-INF\lib下的jar文件。

## 3.3自定义标签

### 3.3.1自定义标签步骤

1. 写一个Java类（标签类），且必须继承SimpleTagSupport类

2. 在doTag方法里面（覆盖doTag方法），编写相应的处理逻辑

2.1 标签有哪些属性，则标签类也有哪些属性，并且类型要匹配

2.2 此外，这些属性必须提供相应的set方法

3. 在.tld文件当中，描述该标签

3.1 .tld文件可以放在WEB-INF下或它的子文件夹下

3.2 也可以放到META-INF下

3.3可以参考c.tld文件来写

注意事项：

* 简单标签技术是新技术，新的规范。
* <body-content></body-content>的作用是告诉容器标签是否有标签体，即开始、结束标签之间的内容，如果有标签体可以有三个值：

1. empty：没有标签体。

2. scriptless：可以有标签体，但标签体里面不能够出现Java代码，三种形式都不行

3. JSP：有标签体，并且标签体的内容可以是Java代码。

只有复杂标签技术支持JSP。简单标签技术只支持empty和scriptless。

## 3.4 JavaEE5.0中EL表达式和JSTL

|  |  |  |
| --- | --- | --- |
| Tomcat5.5 | 对应Servlet2.4规范 | 对应J2EE1.4 |
| Tomcat6.0 | 对应Servlet2.5规范 | 对应JavaEE5.0 |
| Tomcat7.0 | 对应Servlet3.0规范 | 对应JavaEE6.0 |

在JavaEE5.0以上版本中，已经将JSTL标签库对应的jar文件包含进来了，不用再去将那两个jar文件拷贝到WEB-INF\lib下了。

## 

# 四、MVC

## 5.1什么是MVC

Model View Controller，是一种软件架构的思想，将一个软件的模块划分成三种不同类型的模块，分别是模型：封装业务逻辑、视图：实现表示逻辑、控制器协调模型和视图模型：

1. 返回的结果要先交给控制器

2. 由控制器来选择合适的视图来展示处理结果

3. 另外，视图发送请求给控制器，由控制器来选择对应的模型来处理

## 5.2使用MVC的目的

使用MVC思想来设计软件，最根本的目的是为了实现模型的复用：

1. 模型只返回处理结果，并不关心这些结果如何展现

2. 展现由不同的视图来处理

3. 另外可以使用不同的视图来调用同一个模型

## 5.3 MVC开发Web应用程序（JavaEE）

1. 使用Java类来实现模型，将业务逻辑写在Java类里面，写完之后可以立即测试

2. 使用Servlet或者Filter来实现控制器

3. 使用JSP来实现视图

注意事项：

* 一般模型产生应用异常：即不是系统的原因产生的异常，而是用户使用不当造成的，需要提示用户采取正确的操作时，抛自定义异常给控制器。自定义异常可自己写个最简单的类，但是，当有许多应用异常时可采用异常编号方式，控制器再返回结果给视图。
* 对于客户端，WEB-INF下的文件都是受保护的。不能直接访问，只有服务器之间的组件可以访问。某个结果页面不想被用户访问，可以设置主页，然后使用ActionServlet采取转发方式。

## 5.4 MVC的优缺点

优点：

1. 可以实现模型的复用。

2. 模型或者视图发生改变，不会相互影响。

3. 方便测试

缺点：

1. 会增加设计的难度

2. 代码量也会增加

3. 相应地也会增加软件的开发成本