# JSP注释

JSP文件中的注释分为显式注释和隐式注释。

1. **显式注释：**

**<!-- 注释内容 -->**

显式注释是从HTML语言继承而来的，显示注释的内容会发送给客户端，客户端可 以通过查看源代码的方式看到该注释。

**2、隐式注释**

**// 注释（单行）**

**/\* 注释（多行） \*/**

**<%-- 注释 --%>**

·隐式注释的内容不会发送给客户端，客户端无法看到；

·前两种注释是从Java继承而来的，它们只能用在Scriptlet中，最后一种注释是J

SP真正独有的。

# Scriptlet

Scriptlet（服务端小程序，也叫脚本小程序）是JSP的最重要组成部分，所有嵌入在HTML代码中的Java代码都必须使用Scriptlet标记出来。在JSP中一共有3种Scriptlet：

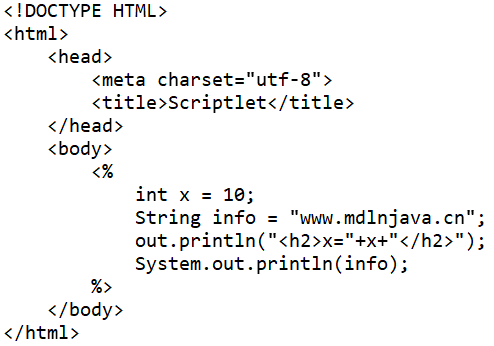
1）<% %>

2）<%! %>

3）<%= %>

## <% %>

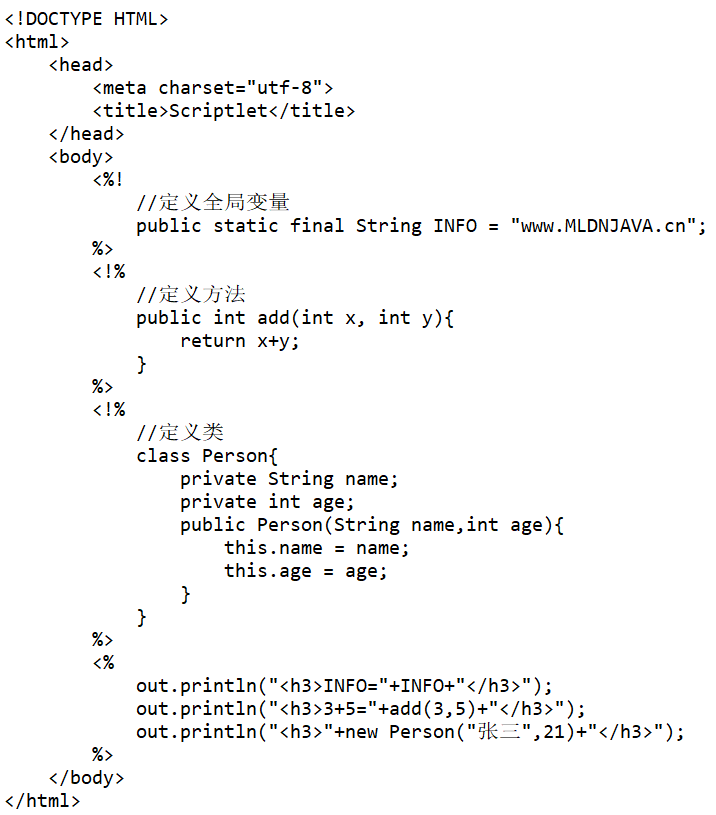
第一种Scriptlet使用<% %>表示，在<% %>中可以定义局部变量、编写语句等。例如jsp文件内容如下：



定义变量x，将其输出到浏览器页面；定义变量info，在Tomcat服务器控制台打印。

## <%! %>

第二种Scriptlet使用<%! %>表示，在<%! %>中可以定义全局变量常量、方法、类。例如jsp文件内容如下：



本程序分别在<%! %>中定义了全局常量，方法、类但是由于<%! %>中不能出现其他语句，所以编写了一个<% %>，在其中完成输出操作。

虽然在<%! %>中可以定义类或方法，但是从正确的开发思路上讲，很少有用户会这样操作。当JSP中需要类或者方法时，往往会通过JavaBean的形式调用。

## <%= %>

第三中Scriptlet的主要功能是输出一个变量或一个具体的常量，有时也将其称为表达式输出。例如jsp文件内容如下：



本程序使用表达式输出输出了变量及常量，并且这种方式输出可以更好地将HTML代码和Java代码相分离。前面学习的out.println( )虽然也可以进行输出，但是无法使HTML代码和Java代码相分离，所以最好的做法就是使用<%= %>只输出由Java（JSP）程序产生的变量，使HTM代码和Java代码相分离。

# Scriptlet标签

在程序中如果过多的出现<% %>会导致代码混乱，所以在新版本的JSP中提供了一种scriptlet标签，使用scriptlet标签可以完成<% %>同样的功能。语法如下：

**<jsp:scriptlet>**

**Java代码**

**</jsp:scriptlet>**

例如jsp文件内容如下：



在scriptlet标签中定义局部变量或语句，完成<% %>同样的功能，最后通过表达式输出向页面输出，但是是否使用此标签来代理<% %>并没有严格的规定，唯一的好处就在于scriptlet标签会更美观一些。

# Page指令

## 认识Page指令

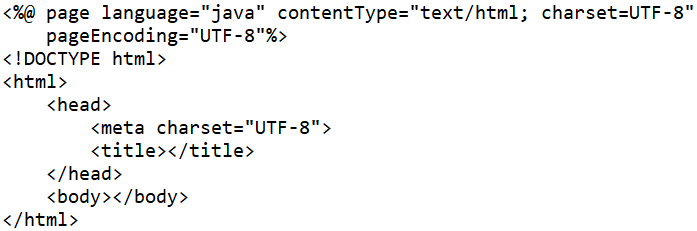
使用page指令可以定义一个JSP页面的相关属性。其常用的属性（对于其他属性，可以参考教材书或查阅相关资料）如下：

|  |  |
| --- | --- |
| **属性** | **描述** |
| language | 定义要使用的服务端语言，值写java |
| contentType | 定义页面响应的MIME类型和JSP字符编码。 |
| errorPage | 定义此页面出错时要跳转的显示页，并且显示页的isErrorPage属性必须设置为true，需要与isErrorPage搭配使用 |
| isErrorPage | 定义此页面是否是出错的处理页。值为true或false |
| pageEncoding | 定义JSP页面的字符编码，默认值为ISO-8859-1。 |
| import | 在此页面中导入需要操作的java包 |
| info | 此JSP页面的信息，用于描述该JSP页面 |

其指令语法为：

**<%@page 属性1=”值” 属性2=”值” %>**

使用示例：



注意：一个JSP文件中可以定义多个page指令，但只有import属性可以重复出现多次，对于其他属性只能出现一次。

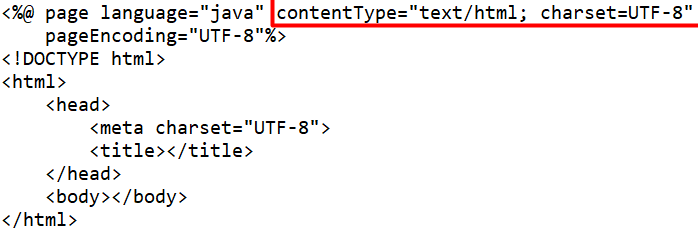
## 设置页面的MIME和编码

1、什么是MIME？

许多文件都是存在扩展名的，根据这些不同的扩展名可以打开不同的应用程序。而 MIME就是指定某个扩展名文件将使用何种应用程序打开的一个说明。当该JSP文件被访问时，浏览器会自动指定某应用程序来打开此JSP文件。

**MIME：多功能Internet邮件扩充服务（Multipurpose Internet Mail Extensions）**

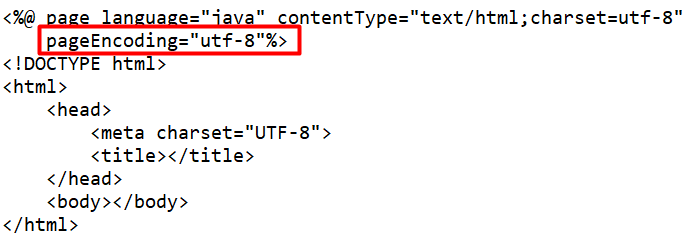
1. 通过contentType属性设置JSP页面的MIME和编码



1. 可以从Tomcat安装目录下的conf/web.xml文件中查询到所有已知的MIME类型及其对应的文件扩展名。
2. 如果设置的MIME类型错误，则也出现一个下载提示框。
3. 通过response对象设置头信息可以为下载的文件起一个指定的名称和后缀，而不再默认使用的名称。

## 设置文件编码

除了可以使用ContentType属性设置页面MIME类型外，还使用了charset指定了页面的编码，但在实际开发中往往还需要使用page指令中的pageEncoding属性设置页面的编码，否则页面会出现乱码。应用示例如下：



在JSP中，如果pageEncoding属性存在，那么JSP的编码将由pageEncoding决定；否则，将由contentType属性中的charset决定。如果两者都不存在，则默认使IOS-8859-1编码方式，因为pageEncoding的默认编码是ISO-8859-1编码。实际开发中建议两者都设置。

## 设置错误页

**1、认识**

当一个页面出现错误后，会自动跳转到另一个页面上进行错误信息的提示。要想完成这样的操作，可以通过Page指令的errorPage属性来指定错误页，完成页面出现错误后自动跳转到错误页的操作。需要满足以下两个条件：

**·通过Page指令的pageError属性指定错误出现时的要跳转的页面**

**·错误处理页必须使用isErrorPage属性标识自己为错误跳转页**

1）如果一个JSP页面运行时出现了错误，会跳转到errorPage属性指定的页面，被跳 转的页面中必须将isErrorPage属性的值设置为true。

2）出现的错误也包括Java代码中产生的而未处理的异常。

3）在错误页的操作中，如果出现了错误页无法显示的情况，则有可能是跳转的错误页 页也出现了错误，从而无法进行显示。此时可以在错误页文件中加入以下代码。

**<% response.setStatus(200); %>**

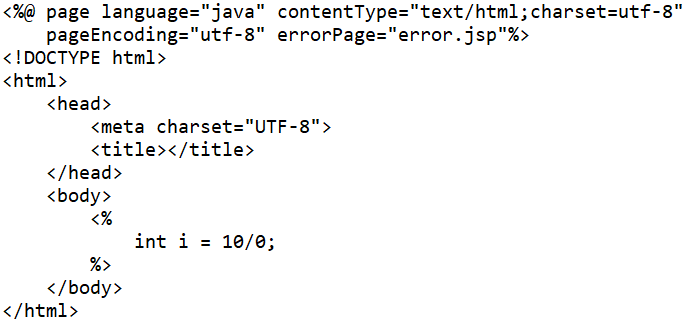
设置HTTP代码为200，强制表示本页没有错误。

4）这里的错误页的跳转属于服务器端跳转。

**2、错误页跳转的示例**

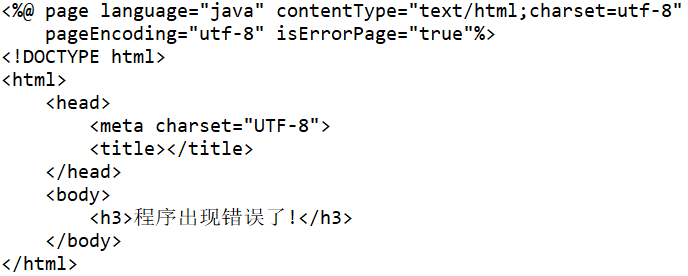
1）创建一个会出现错误的页面show.jsp，使用page指令指定errorPage属性，属性值

表示此页面出现错误后要跳转去的页面。



2）创建错误处理页error.jsp，将page指令中的isErrorPage属性值设置为true，表示该

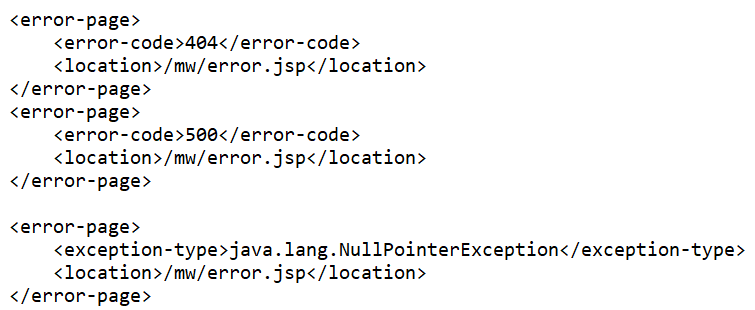
页面可以是错误显示页。



3）访问会出现错误的页面show.jsp，会跳转到错误显示页error.jsp。

## 全局错误处理

每一个JSP页面中可以使用Page指令的errorPage属性指定错误处理页，但是，也可以在项目中的web.xml文件中配置全局错误处理，全局错误处理可以处理两种类型的错误，一种是HTTP状态码所代表的错误，如404或500等，还有一种是异常的情况，如NullPointerException等。示例如下：



（修改完web.xml文件后，记得重启Tomcat服务器）此时，如果访问某个页面出现了404或500错误，则会跳转到error.jsp页面进行显示，如果访问某个页面出现了java.lang.NullPointerException异常，则会跳转到error.jsp页面进行显示（这里的错误页跳转属于服务器端跳转）。

如果既在项目的web.xml文件中配置了全局错误页，又在页面中使用了page指令的errorPage属性来指定错误页，则优先使用errorPage属性指定的错误页。

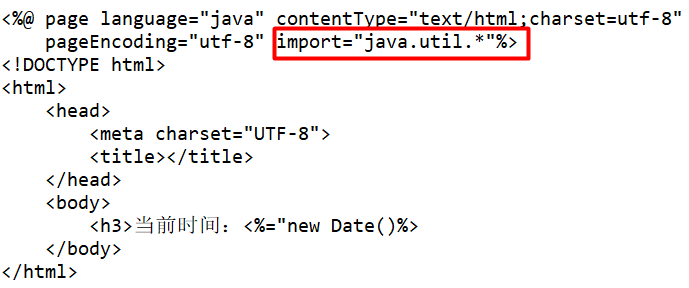
## 服务器端跳转与客户端跳转

当一个JSP文件出现错误并跳转到了错误页，页面的显示内容将变为错误页中设置的内 容，但是在地址栏上依然是show.jsp。也就是说，此时内容虽然改变了，显示的是另一 个文件jsp文件的内容，但地址栏并没有改变，这样的跳转，在程序中称为服务器端跳 转。在整个操作中，客户端对服务器只发送了一次请求，服务器对客户端也只会应了一 次。

而如果程序跳转后，页面的地址栏也发生改变了，则此种跳转属于客户端跳转。例如， 超链接的跳转。

## 导包操作

在page指令中可以使用import导入所需要的Java开发包。例如：



该示例通过page指令导入了java.util工具包，然后使用Date类创建一个日期类对象，并进行打印。

# 包含指令

## 包含指令的作用

在一般的页面开发中会有很多内容要重复地显示。例如，在一般的站点中，都会按照以下结构进行内容的显示，如下所示：



工具栏、页面头部、页面尾部基本上都是固定的，而中间的具体内容是不同的，那么这时就有以下两种做法：

·做法一：让每一个页面都分别包含工具栏、页面头部、页面尾部的代码；

·做法二：将工具栏、页面头部、页面尾部分别做成一个页面文件，然后在需

要地方导入（包含）；

很明显，第二种做法更加方便，因为采用包含的形式，可以减少代码的重复量；而做法有很多重复的代码。要想实现这样的包含功能，在JSP中可以通过静态包含和动态包含两种方式。

## 静态包含

静态包含是在JSP编译时插入一个包含文本或代码的文件，先包含在处理，属于代码包含，而包含的文件可以是JSP文件、HTML文件、文本文件，或是一段Java程序。但需要注意，在每一个完整的页面中，对于<html>、<head>、<title>、<body>这几个元素只能出现一次，如果重复出现，则可能会造成显示的错误。静态包含语法如下：

**<%@ include file=”要包含的文件路径”%>**

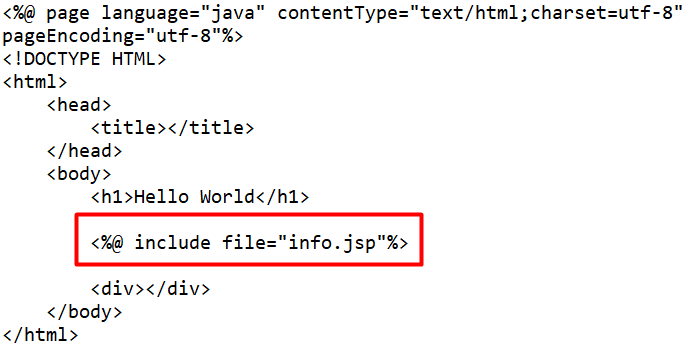
静态包含将会先把代码包含进来，然后再将全部的代码进行集中的编译处理。

示例：

1. 定义要包含的文件info.html，内容如下：



1. 定义include\_demo.jsp文件，将info.html包含进来



## 动态包含

动态包含可以自动区分被包含的页面是静态还是动态。如果是静态页面，则与静态包含一样，将代码包含进来，然后在进行集中的编译处理；如果是动态页面，则先进行动态的处理，然后再将处理后的结果包含进来，同时还可以向被包含的页面传递参数。语法如下：

不传递参数：

**<jsp:include page=”{要包含的文件路径 | <%=表达式%>}” flush=”true|false”/>**

传递参数：

**<jsp:include page=”{要包含的文件路径 | <%=表达式%>}” flush=”true|false”>**

**<jsp:param name=”参数名” value=”参数值”/>**

**……**

**</jsp:include>**

上述语法中，flush属性值为false表示这个网页完全被读进行以后才输出。在每一个JSP的内部都会有一个buffer，所以如果是true，当buffer满了就输出，一般将此属性的值设置为true，当然也可以不用设置，默认值为true。

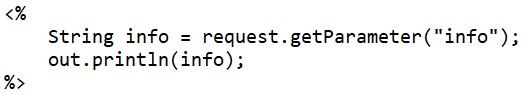
示例：

* 1. 创建info\_a.html和info\_b.jsp文件

**info\_a.html：**

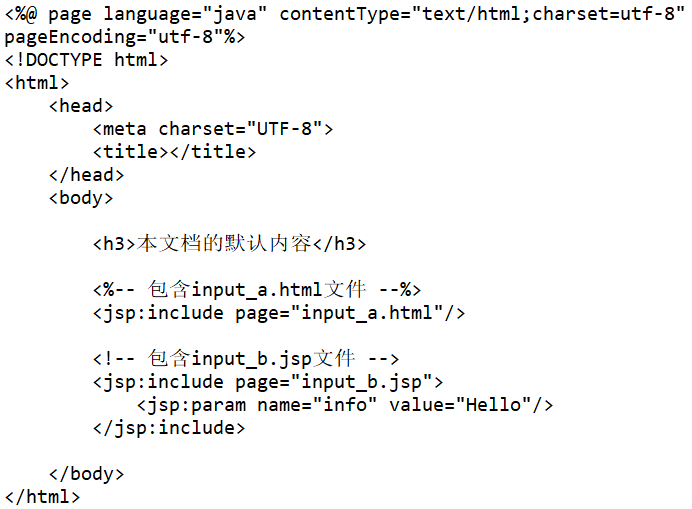


**info\_b.html：**



在此文档中接收了向该文件中传递的参数，并向页面输出。

* 1. 创建include.jsp文件



## 静态包含与动态包含的区别

使用<%@include%>指令可以实现静态包含，使用<jsp:include>标签可以实现动态包含，但使用哪种更好呢？但是使用动态包含更好，理由如下：

1. 动态包含属于先处理后包含，而静态包含属于先包含后处理。

例如：假设包含页a.jsp中定义了变量a，在b.jsp中也定义了变量a，此时如果使用静态包含，就会出现错误，因为静态包含是先包含后处理，即先将代码包含进行，然后再对其进行编译，此时就会出现变量重名的错误；而如果使用动态包含，则先将被包含的页面进行编译，然后将内容包含进行，不涉及到代码的包含，不会出现变量重名的错误。

1. 动态包含可以向被包含的页面传递参数，而静态包含不可以。

**包含指令：**

通过包含指令可以在一个JSP文件中引入一个或多个JSP文件或者HTML文件。

在JSP中要想实现包含功能，可以通过静态包含和动态包含两种方式完成。

**静态包含：**

静态包含指令是在JSP文件未编译前通过静态包含指令插入一个或多个文件，然后将这 些文件与此JSP文件合在一起进行编译，然后再将一份整体的内容展现给用户。静态包 含指令的语法为：

**<%@ include file=”要包含文件的路径” %>**

**动态包含：**

动态包含指令可以自动区分被包含的页面是静态的还是动态的。如果是静态页面，则与 静态包含一样，将内容包含进来处理；而如果被包含的页面是动态页面，则分别对其进 行处理，然后将处理后的结果包含进来。

动态包含指令分为两种，一种是传递参数的包含，一种是不传递参数的包含。

·不传递参数：

**<jsp : include page=”要包含的文件路径”/>**

**·**传递参数：

**<jsp : include page=”要包含文件的路径”>**

**<jsp:param name=”参数名” value=”参数值”/>**

**......**

**</jsp:include>**

如果传递了参数到被包含的页面，则在被包含的页面中可以使用 requeset.getParameter( “参数名” )取得对应的参数值。

**静态包含与动态包含的区别：**

1）静态包含的操作属于先包含后处理，相当于把被包含的文件代码拿处理来对包含 语句进行替换。而动态包含属于先处理后包含，即分别处理被包含的文件，将被包 含的文件的显示结果拿出来对包含语句进行替换。

2）如果使用静态包含则容易出现变量重复导致程序错误。所以使用动态包含会更加方 便，并且还可以向被包含的文件进行参数的传递。

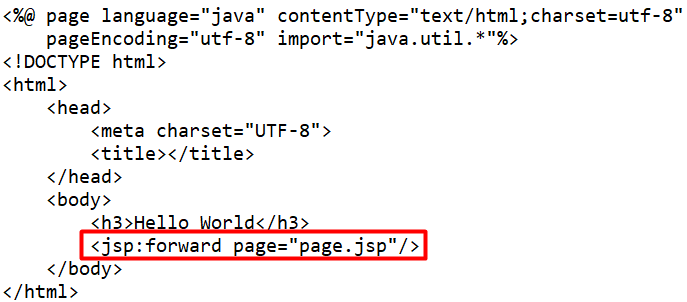
# 跳转指令

在Web中可以使用<jsp : forward>指令，将一个用户的请求（request）从一个页面传递到另外一个页面，完成跳转（服务器端跳转）的操作。需要注意，通过跳转指令完成的跳转属于服务器端跳转。其语法为：

**1、不传递参数跳转**

**<jsp : forward page=”要跳转文件的路径 | <%=表达式%>” />**

例如：



当解析到<jsp:forward>标签处，就会发生跳转。

**2、传递参数跳转**

**<jsp forward page=”要跳转文件的路径” />**

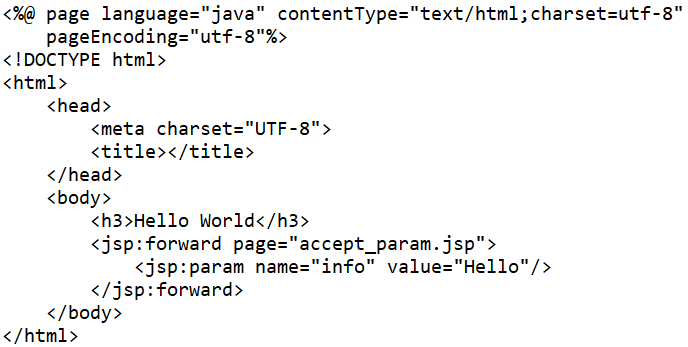
**<jsp : param name=”参数名称” value=”参数内容” />**

**......**

**</jsp : forward>**

如果向跳转的页面传递了参数，则在跳转的页面中可以使用request.getParameter( “参数名” )取得对应的参数值。例如：

**input\_param.jsp**



**accept\_param.jsp**

