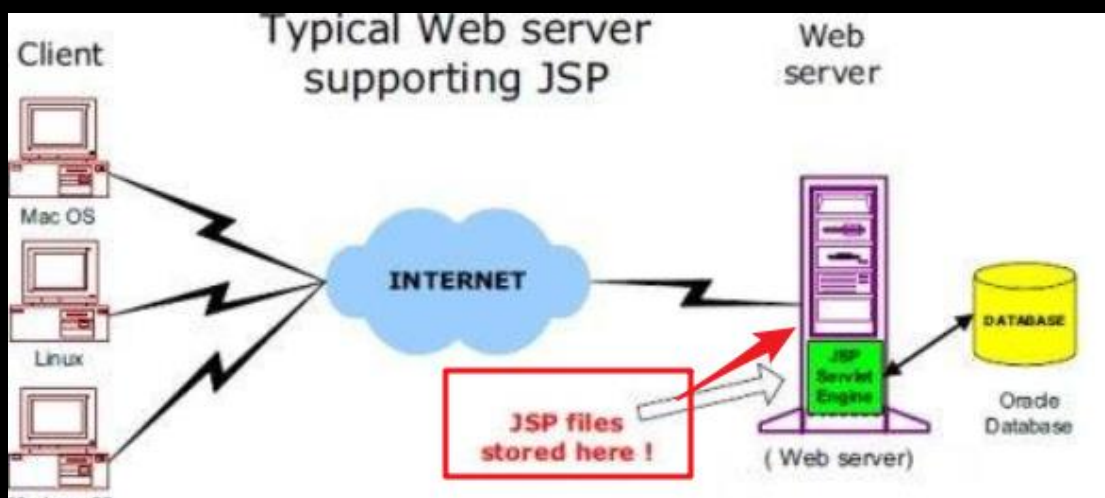


# JSP 快速入门

## 概述

- Java Server Pages, 是一种动态网页开发技术。标签通常以<%开头以%>结束。
- JSP 是一种 Java servlet, 主要用于实现 Java web 应用程序的用户界面部分
- JSP 通过网页表单获取用户输入数据、访问数据库及其他数据源, 然后动态地创建网页。
- 优点:
  - 性能更加优越
  - 服务器调用的是已经编译好的 JSP 文件, 而不像 CGI/Perl 那样必须先载入解释器和目标脚本
  - JSP 基于 Java Servlets API



- 运行过程:

- 浏览器发送一个 HTTP 请求给服务器
- Web 服务器识别 JSP 网页的请求。将该请求传递给 JSP 引擎。  
通过使用 URL 或者.jsp 文件来完成。
- JSP 引擎从磁盘中载入 JSP 文件，然后将它们转化为 servlet。  
这种转化只是简单地将所有模板文本改用 `println()` 语句，并且将所有的 JSP 元素转化成 Java 代码
- JSP 引擎将 servlet 编译成可执行类，并且将原始请求传递给 servlet 引擎
- Web 服务器的某组件将会调用 servlet 引擎，然后载入并执行 servlet 类。在执行过程中，servlet 产生 HTML 格式的输出并将其内嵌于 HTTP response 中上交给 Web 服务器
- Web 服务器以静态 HTML 网页的形式将 HTTP response 返回到浏览器并显示

## 生命周期

- 编译阶段：servlet 容器编译 servlet 源文件，生成 servlet 类
  - 解析 jsp 文件
  - 将 jsp 装化为 Servlet
  - 编译 Servlet
- 初始化：加载与 JSP 对应的 servlet 类，创建其实例，并调用它的初始化方法
  - 容器载入 JSP 文件后，它会在为请求提供任何服务前调用

jspInit()方法。如果需要执行自定义的 JSP 初始化任务，复写 jspInit()方法。

- 执行：调用与 JSP 对应的 servlet 实例的服务方法
  - JSP 引擎将会调用\_jspService()——一个 HttpServletRequest 对象和一个 HttpServletResponse 对象作为它的参数
- 销毁阶段：调用与 JSP 对应的 servlet 实例的销毁方法，然后销毁 servlet 实例
  - jspDestroy()

## 语法

- 脚本程序可以包含任意量的 Java 语句、变量、方法或表达式，只要它们在脚本语言中是有效的

<%代码片段%>

也可以使用与其等价的 XML 语句：

<jsp: scriptlet>代码</jsp:scriptlet>

注意：任何文本、HTML 标签、JSP 元素必须写在脚本程序的外面。

- 声明：<%! 声明语句如：int I=0; %>
  - <%! int a = 100; String b = "str\_b"; %>
  - 实际上也是可以直接写在%%中的

- JSP 表达式

表达式中可以包含脚本语言，先被转化为 String，再插入 HTML

## 标签

- 格式: `<%=表达式%>`
- 注意: 不能使用分号结束表达式

- 注释

`<%-- --%>`

- JSP 隐含对象:

JSP支持九个自动定义的变量, 江湖人称隐含对象。这九个隐含对象的简介见下表:

对象	描述
request	<code>HttpServletRequest</code> 类的实例
response	<code>HttpServletResponse</code> 类的实例
out	<code>PrintWriter</code> 类的实例, 用于把结果输出至网页上
session	<code>HttpSession</code> 类的实例
application	<code>ServletContext</code> 类的实例, 与应用上下文有关
config	<code>ServletConfig</code> 类的实例
pageContext	<code>PageContext</code> 类的实例, 提供对JSP页面所有对象以及命名空间的访问
page	类似于Java类中的this关键字
Exception	<code>Exception</code> 类的对象, 代表发生错误的JSP页面中对应的异常对象

- 控制流语句:

- If-else:

好奇怪, 使用个`<%%>`[脚本拼接](#)的Java代码也能运行:

```
<% int day=3; if (day == 1 | day == 7) { %>
    <p> Today is weekend</p>
<% } else { %>
    <p> Today is not weekend</p>
<% } %>
```

- Switch-case

#### ■ For

#### ■ While

```
<%while ( fontSize <= 3){ %>
    <font color="green" size="<%= fontSize %>">
    JSP Tutorial
    </font><br />
    <%fontSize++;%>
```

#### ■ Do while

#### ● Jsp 常量:

- 布尔值(boolean): true 和 false;
- 整型(int): 与Java中的一样;
- 浮点型(float): 与Java中的一样;
- 字符串(string): 以单引号或双引号开始和结束;
- Null: null。

## Jsp 指令

指令用来设置与整个页面相关的属性

- <%@page attribute = "value"%>:定义页面的依赖属性，如脚本语言、error 缓存。一个 JSP 页面可以包含多个 page 指令

下表列出与Page指令相关的属性:

属性	描述
buffer	指定out对象使用缓冲区的大小
autoFlush	控制out对象的 缓存区
contentType	指定当前JSP页面的MIME类型和字符编码
errorPage	指定当JSP页面发生异常时需要转向的错误处理页面
isErrorPage	指定当前页面是否可以作为另一个JSP页面的错误处理页面
extends	指定servlet从哪一个类继承
import	导入要使用的Java类
info	定义JSP页面的描述信息
isThreadSafe	指定对JSP页面的访问是否为线程安全
language	定义JSP页面所用的脚本语言, 默认是Java
session	指定JSP页面是否使用session
isELIgnored	指定是否执行EL表达式
isScriptingEnabled	确定脚本元素能否被使用

- `<%@ include file = "relativeURL %">`: 包含其他文件
- `<%@ taglib uri="uri" prefix = "prefixOfTag" %>`: 引入标签库的定义, 可以是自定义标签

## JSP 动作元素

在请求处理阶段其作用。

- 利用 JSP 动作可以动态地插入文件、重用 JavaBean 组件、把用户重定向到另外的页面、为 Java 插件生成 HTML 代码。
- Jsp 行为行为标签使用 XML 语法结构来控制 Servlet 引擎
- 格式: `<jsp: action_name attribute = "value"/>`

语法	描述
jsp:include	用于在当前页面中包含静态或动态资源
jsp:useBean	寻找和初始化一个JavaBean组件
jsp:setProperty	设置 JavaBean组件的值
jsp:getProperty	将 JavaBean组件的值插入到 output中
jsp:forward	从一个JSP文件向另一个文件传递一个包含用户请求的 request对象
jsp:plugin	用于在生成的HTML页面中包含Applet和JavaBean对象
jsp:element	动态创建一个XML元素
jsp:attribute	定义动态创建的XML元素的属性
jsp:body	定义动态创建的XML元素的主体
jsp:text	用于封装模板数据

### ● 属性：

- Id：可以通过 pageContext 调用
- Scope 属性：用于识别动作元素的生命周期，scope 属性决定了 id 的生命周期。4 个值：
  - ◆ Page
  - ◆ Request
  - ◆ Session
  - ◆ Application

### ● 动作元素 • <jsp:include>

- 作用：在 JSP 被转化为 Servlet 时导入文件，注意：应该是将文件导入到当前页面效果中。
- 格式：<jsp: include page="relative url" flush="true">

page	包含在页面中的相对URL地址。
flush	布尔属性，定义在包含资源前是否刷新缓存区。

- 动作元素 • <jsp: useBean>

- 作用：装载一个将 JSP 页面中使用的 javaBean，很有用

- 语法：

```
<jsp:useBean id="name" class="package.class" />
```

- 属性：

class	指定Bean的完整包名。
type	指定将引用该对象变量的类型。
beanName	通过 java.beans.Beans 的 instantiate() 方法指定Bean的名字。

- 动作元素 • <jsp: setProperty>

- 作用：设置已经实例化的 bean 对象属性。注意：必须使用先

<jsp: useBean>

- 用法：

- ◆ 用法 1:用在<jsp: useBean>的后面，无论如何都会执行

- ◆ 用法 2:用在<jsp: useBean>之中，只有在新建 bean 的实例中才会执行，现有实例是不会执行

- 格式：

```
<jsp:setProperty name="myName" property="someProperty" .../>
```

- 属性



name	name属性是必需的。它表示要设置属性的是哪个Bean。
property	property属性是必需的。它表示要设置哪个属性。有一个特殊用法：如果property的值是"*"，表示所有名字和Bean属性名字匹配的请求参数都将被传递给相应的属性set方法。
value	value 属性是可选的。该属性用来指定Bean属性的值。字符串数据会在目标类中通过标准的valueOf方法自动转换成数字、boolean、Boolean、byte、Byte、char、Character。例如，boolean和Boolean类型的属性值（比如"true"）通过 Boolean.valueOf转换，int和Integer类型的属性值（比如"42"）通过Integer.valueOf转换。value和param不能同时使用，但可以使用其中任意一个。
param	param 是可选的。它指定用哪个请求参数作为Bean属性的值。如果当前请求没有参数，则什么事情也不做，系统不会把null传递给Bean属性的set方法。因此，你可以让Bean自己提供默认属性值，只有当请求参数明确指定了新值时才修改默认属性值。

- 动作元素 · `<jsp: getProperty>`

- 获取 bean 属性，同样需要先使用 `<jsp: useBean>`
- 格式：

```
<jsp:getProperty name="myName" property="someProperty" .../>
```

- 动作元素 · `<jsp:forward>`

- 作用：把请求转到其他页面

- `<jsp:forward page="Relative URL" />`

- 唯一属性

page	page属性包含的是一个相对URL。page的值既可以直接给出，也可以在请求的时候动态计算，可以是一个JSP页面或者一个 Java Servlet。
------	--

- 动作元素 · `<jsp: plugin>`

- 插入插件

- 动作元素 · `<jsp:forward>` `<jsp:attribute>` `<jsp: body>`

- 动态定义 XML 元素

- 动作元素 · `<jsp: text>`

- 允许在 JSP 元素中使用写入文本的模版

■ 格式: `<jsp:text>Template data</jsp:text>`

■ `<>`转义: `<jsp:text><![CDATA[<br>]]></jsp:text>`

## 隐式对象

- Request: 每当客户端请求一个 JSP 页面时, JSP 引擎就会制造一个新的 request 对象来代表这个请求。request 对象提供了一系列方法来获取 HTTP 头信息, cookies, HTTP 方法等等
- Response: 当服务器创建 request 对象时会同时创建用于响应这个客户端的 response 对象。
- Out: javax.servlet.jsp.JspWriter 类的实例,用来在 response 对象中写入内容。

<code>out.print(dataType dt)</code>	输出Type类型的值
<code>out.println(dataType dt)</code>	输出Type类型的值然后换行
<code>out.flush()</code>	刷新输出流

- Session: 跟踪在各个客户端请求间的会话
- Application: 直接包装了 servlet 的 ServletContext 类的对象
- Config: config 对象是 javax.servlet.ServletConfig 类的实例, 直接包装了 servlet 的 ServletConfig 类的对象,允许开发者访问 Servlet 或者 JSP 引擎的初始化参数, 比如文件路径等
- pageContext: pageContext 对象是 javax.servlet.jsp.PageContext 类的实例, 用来代表整个 JSP 页面。
- page: 页面实例的引用, 即 this

➤ exception: 包装了异常信息

# 客户端请求

请求信息时发送信息头，HTTP 信息头：

信息	描述
Accept	指定浏览器或其他客户端可以处理的MIME类型。它的值通常为 <b>image/png</b> 或 <b>image/jpeg</b>
Accept-Charset	指定浏览器要使用的字符集。比如 ISO-8859-1
Accept-Encoding	指定编码类型。它的值通常为 <b>gzip</b> 或 <b>compress</b>
Accept-Language	指定客户端首选语言，servlet会优先返回以当前语言构成的结果集，如果servlet支持这种语言的话。比如 en, en-us, ru等等
Authorization	在访问受密码保护的网页时识别不同的用户
Connection	表明客户端是否可以处理HTTP持久连接。持久连接允许客户端或浏览器在一个请求中获取多个文件。 <b>Keep-Alive</b> 表示启用持久连接
Content-Length	仅适用于POST请求，表示 POST 数据的字节数
Cookie	返回先前发送给浏览器的cookies至服务器
Host	指出原始URL中的主机名和端口号
If-Modified-Since	表明只有当网页在指定的日期被修改后客户端才需要这个网页。 服务器发送 304码给客户端，表示没有更新的资源
If-Unmodified-Since	与If-Modified-Since相反， 只有文档在指定日期后仍未被修改过，操作才会成功
Referer	标志着所引用页面的URL。比如，如果你在页面1，然后点了个链接至页面2，那么页面1的URL就会包含在浏览器请求页面2的信息头中
User-Agent	用来区分不同浏览器或客户端发送的请求，并对不同类型的浏览器返回不同的内容

## HttpServletRequest 类

request 对象是 javax.servlet.http.HttpServletRequest 类的实例, 每当客户端请求一个页面时，JSP 引擎就会产生一个新的对象来代表这个请求。

Request 对象提供了一系列方法来获取 HTTP 信息头。

使用隐藏对象 request 即可调用

**Cookie[] getCookies()**

返回客户端所有的Cookie的数组

**Enumeration getAttributeNames()**

返回request对象的所有属性名称的集合

**Enumeration getHeaderNames()**

返回所有HTTP头的名称集合

**Enumeration getParameterNames()**

返回请求中所有参数的集合

**HttpSession getSession()**

返回request对应的session对象，如果没有，则创建一个

**HttpSession getSession(boolean create)**

返回request对应的session对象，如果没有并且参数create为true，则返回一个新的session对象

**Locale getLocale()**

返回当前页的Locale对象，可以在response中设置

**Object getAttribute(String name)**

返回名称为name的属性值，如果不存在则返回null。

**ServletInputStream getInputStream()**

返回请求的输入流

**String getAuthType()**

返回认证方案的名称，用来保护servlet，比如 "BASIC" 或者 "SSL" 或 null 如果 JSP没设置保护措施

**String getCharacterEncoding()**

返回request的字符编码集名称

**String getContentType()**

返回request主体的MIME类型，若未知则返回null

**String getContextPath()**

返回request URI中指明的上下文路径

**String getHeader(String name)**

返回name指定的信息头

**String getMethod()**

返回此request中的HTTP方法，比如 GET, POST, 或 PUT

**String getParameter(String name)**

返回此request中name指定的参数，若不存在则返回 null

**String getPathInfo()**

返回任何额外的与此request URL相关的路径

**String getProtocol()**

返回此request所使用的协议名和版本

**String getQueryString()**

返回此 request URL包含的查询字符串

**String getRemoteAddr()**

返回客户端的IP地址

**String getRemoteHost()**

返回客户端的完整名称

**String getRemoteUser()**

返回客户端通过登录认证的用户，若用户未认证则返回 null

**String getRequestURI()**

返回request的URI

**String getRequestedSessionId()**

返回request指定的session ID

**String getServletPath()**

返回所请求的servlet路径

**String[] getParameterValues(String name)**

返回指定名称的参数的所有值，若不存在则返回null

**boolean isSecure()**

返回request是否使用了加密通道，比如HTTPS

**int getContentLength()**

返回request主体所包含的字节数，若未知的返回-1

```
int getIntHeader(String name)
```

返回指定名称的request信息头的值

```
int getServerPort()
```

返回服务器端口号

# 服务器响应

响应头	描述
Allow	指定服务器支持的request方法（GET，POST等等）
Cache-Control	指定响应文档能够被安全缓存的情况。通常取值为 <b>public</b> ， <b>private</b> 或 <b>no-cache</b> 等等。Public意味着文档可缓存，Private意味着文档只为单用户服务并且只能使用私有缓存。No-cache 意味着文档不被缓存。
Connection	命令浏览器是否要使用持久的HTTP连接。 <b>close</b> 值 命令浏览器不使用持久HTTP连接，而keep-alive 意味着使用持久化连接。
Content-Disposition	让浏览器要求用户将响应以给定的名称存储在磁盘中
Content-Encoding	指定传输时页面的编码规则
Content-Language	表述文档所使用的语言，比如en，en-us,, ru等等
Content-Length	表明响应的字节数。只有在浏览器使用持久化 (keep-alive) HTTP 连接时才有用
Content-Type	表明文档使用的MIME类型
Expires	指明啥时候过期并从缓存中移除
Last-Modified	指明文档最后修改时间。客户端可以 缓存文档并且在后续的请求中提供一个 <b>If-Modified-Since</b> 请求头
Location	在300秒内，包含所有的有一个状态码的响应地址，浏览器会自动重连然后检索新文档
Refresh	指明浏览器每隔多久请求更新一次页面。
Retry-After	与503 (Service Unavailable)一起使用来告诉用户多久后请求将会得到响应
Set-Cookie	指明当前页面对应的cookie

## HTTPServletResponse

**String encodeRedirectURL(String url)**

对sendRedirect()方法使用的URL进行编码

**String encodeURL(String url)**

将URL编码，回传包含Session ID的URL

**boolean containsHeader(String name)**

返回指定的响应头是否存在

**boolean isCommitted()**

返回响应是否已经提交到客户端

**void addCookie(Cookie cookie)**

添加指定的cookie至响应中

**void addDateHeader(String name, long date)**

添加指定名称的响应头和日期值

**void addHeader(String name, String value)**

添加指定名称的响应头 and 值

**void addIntHeader(String name, int value)**

添加指定名称的响应头和int值

**void flushBuffer()**

将任何缓存中的内容写入客户端

**void reset()**

清除任何缓存中的任何数据，包括状态码和各种响应头

**void resetBuffer()**

清除基本的缓存数据，不包括响应头和状态码

**void sendError(int sc)**

使用指定的状态码向客户端发送一个出错响应，然后清除缓存

**void sendError(int sc, String msg)**

使用指定的状态码和消息向客户端发送一个出错响应

**void sendRedirect(String location)**

使用指定的URL向客户端发送一个临时的间接响应

**void setBufferSize(int size)**

设置响应体的缓存区大小

**void setCharacterEncoding(String charset)**

指定响应的编码集（MIME字符集），例如UTF-8

**void setContentLength(int len)**

指定HTTP servlets中响应的内容的长度，此方法用来设置 HTTP Content-Length 信息头

**void setContentType(String type)**

设置响应的内容的类型，如果响应还未被提交的话

**void setDateHeader(String name, long date)**

使用指定名称和值设置响应头的名称和内容

**void setHeader(String name, String value)**

使用指定名称和值设置响应头的名称和内容

**void setIntHeader(String name, int value)**

使用指定名称和值设置响应头的名称和内容

**void setLocale(Locale loc)**

设置响应的语言环境，如果响应尚未被提交的话

**void setStatus(int sc)**

设置响应的状态码

## JSP HTTP 状态码



状态码	消息	描述
100	Continue	只有一部分请求被服务器接收，但只要没被服务器拒绝，客户端就会延续这个请求
101	Switching Protocols	服务器交换机协议
200	OK	请求被确认
201	Created	请求已完成，新的资源被创建
202	Accepted	请求被接受，但未处理完
203	Non-authoritative Information	
204	No Content	
205	Reset Content	
206	Partial Content	
300	Multiple Choices	一个超链接表，用户可以选择一个超链接并访问，最大支持5个超链接
301	Moved Permanently	被请求的页面已经移动到了新的URL下
302	Found	被请求的页面暂时性地移动到了新的URL下
303	See Other	被请求的页面可以在一个不同的URL下找到
304	Not Modified	
305	Use Proxy	
306	<i>Unused</i>	已经不再使用此状态码，但状态码被保留
307	Temporary Redirect	被请求的页面暂时性地移动到了新的URL下
400	Bad Request	服务器无法识别请求
401	Unauthorized	被请求的页面需要用户名和密码
402	Payment Required	<i>目前还不能使用此状态码</i>
403	Forbidden	禁止访问所请求的页面
404	Not Found	服务器无法找到所请求的页面
405	Method Not Allowed	请求中所指定的方法不被允许
406	Not Acceptable	服务器只能创建一个客户端无法接受的响应
407	Proxy Authentication Required	在请求被服务前必须认证一个代理服务器
408	Request Timeout	请求时间超过了服务器所能等待的时间，连接被断开
409	Conflict	请求有矛盾的地方
410	Gone	被请求的页面不再可用
411	Length Required	"Content-Length"没有被定义，服务器拒绝接受请求
412	Precondition Failed	请求的前提条件被服务器评估为false
413	Request Entity Too Large	因为请求的实体太大，服务器拒绝接受请求
414	Request-url Too Long	服务器拒绝接受请求，因为URL太长。多出现在把"POST"请求转换为"GET"请求时所附带的大量查询信息

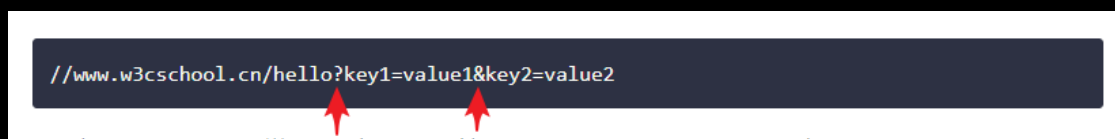
415	Unsupported Media Type	服务器拒绝接受请求，因为媒体类型不被支持
417	Expectation Failed	
500	Internal Server Error	请求不完整，服务器遇见了出乎意料的情况
501	Not Implemented	请求不完整，服务器不提供所需要的功能
502	Bad Gateway	请求不完整，服务器从上游服务器接受了一个无效的响应
503	Service Unavailable	请求不完整，服务器暂时重启或关闭
504	Gateway Timeout	网关超时
505	HTTP Version Not Supported	服务器不支持所指定的HTTP版本

设置HTTP状态码的方法	
下表列出了HttpServletResponse 类中用来设置状态码的方法：	
S.N.	方法 & 描述
1	<b>public void setStatus ( int statusCode )</b> 此方法可以设置任意的状态码。如果您的响应包含一个特殊的状态码和一个文档，请确保在用PrintWriter返回任何内容前调用setStatus方法
2	<b>public void sendRedirect(String url)</b> 此方法产生302响应，同时产生一个 Location 头告诉URL 一个新的文档
3	<b>public void sendError(int code, String message)</b> 此方法将一个状态码(通常为 404)和一个短消息，自动插入HTML文档中并返回给客户端

## JSP 表单处理

浏览器中使用 GET、POST 方法向服务器提交数据

- GET: 将请求的编码信息添加在网址后面，网址与编码信息通过？分隔，多个值使用&分隔，最大为 1024B



- POST: 敏感信息如密码等使用 POST 传递，POST 提交数据是隐式的、不可见的。

```
<form action="form1" method="POST">
  name:<input type = "text" name = "name"><br>
```

**POST GET**

使用表单时需要在 input 里添加 Name 属性，以便在 getParameter 时使用

## JSP 读取表单数据

- **getParameter():** 使用 request.getParameter() 方法来获取表单参数的值。
- **getParameterValues():** 获得如checkbox类（名字相同，但值有多个）的数据。接收数组变量，如checkbox类型
- **getParameterNames():**该方法可以取得所有变量的名称，该方法返回一个Enumeration。
- **getInputStream():**调用此方法来读取来自客户端的二进制数据流。

## 复选框

使用复选框时这里 Name 属性也必须是不同的，同样需要使用 getParameter("name")来访问，得到的是： on / null

```
<!-- 获取所有参数: -->
<%Enumeration paramNames = request.getParameterNames()
    while(paramNames.hasMoreElements()){
        String a = (String)paramNames.nextElement();
        out.println(a);
        out.println(request.getHeader(a)+"    ");
    }
%>
```

## JS 过滤器

作用：

- 在请求访问后端资源是拦截它。
- 管理从服务器返回给客服端的响应

类型：

- 认证过滤器
- 数据压缩过滤器
- 加密过滤器
- 触发资源访问事件的过滤器
- 图像转换过滤器

- 登录和验证过滤器
- MIME 类型链过滤器
- 令牌过滤器
- 转换 XML 内容的 XSL/T 过滤器

未完待续.....

## Cookie 处理

未完待续.....

## Session

未完待续.....

## 文件上传

未完待续.....

## 日期处理

未完待续.....

## 页面重定向

未完待续.....

## 点击量统计

未完待续.....

# 自动刷新 发送邮件

## JSTL

Jsp 标准标签库 jstl 封装了 JSP 通用核心标签

JSP 标准标签库，javaSeerver Pages Standard Tag Libriary

JSTL 支持通用的、结构化的任务，比如迭代，条件判断，XML 文档操作，国际化标签，SQL 标签。它还提供了一个框架来使用集成 JSTL 的自定义标签。

需要导入包：standard.jar jstl.jar

分类：

### 1. 核心标签

引入：

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

标签	描述
<a href="#"><u>&lt;c:out&gt;</u></a>	用于在JSP中显示数据，就像<%= ... >
<a href="#"><u>&lt;c:set&gt;</u></a>	用于保存数据
<a href="#"><u>&lt;c:remove&gt;</u></a>	用于删除数据
<a href="#"><u>&lt;c:catch&gt;</u></a>	用来处理产生错误的异常状况，并且将错误信息储存起来
<a href="#"><u>&lt;c:if&gt;</u></a>	与我们在一般程序中用的if一样
<a href="#"><u>&lt;c:choose&gt;</u></a>	本身只当做<c:when>和<c:otherwise>的父标签
<a href="#"><u>&lt;c:when&gt;</u></a>	<c:choose>的子标签，用来判断条件是否成立
<a href="#"><u>&lt;c:otherwise&gt;</u></a>	<c:choose>的子标签，接在<c:when>标签后，当<c:when>标签判断为false时被执行
<a href="#"><u>&lt;c:import&gt;</u></a>	检索一个绝对或相对 URL，然后将其内容暴露给页面
<a href="#"><u>&lt;c:forEach&gt;</u></a>	基础迭代标签，接受多种集合类型
<a href="#"><u>&lt;c:forTokens&gt;</u></a>	根据指定的分隔符来分隔内容并迭代输出
<a href="#"><u>&lt;c:param&gt;</u></a>	用来给包含或重定向的页面传递参数
<a href="#"><u>&lt;c:redirect&gt;</u></a>	重定向至一个新的URL。
<a href="#"><u>&lt;c:url&gt;</u></a>	使用可选的查询参数来创建一个URL

## 2. 格式化标签

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

标签	描述
<a href="#"><u>&lt;fmt:formatNumber&gt;</u></a>	使用指定的格式或精度格式化数字
<a href="#"><u>&lt;fmt:parseNumber&gt;</u></a>	解析一个代表着数字，货币或百分比的字符串
<a href="#"><u>&lt;fmt:formatDate&gt;</u></a>	使用指定的风格或模式格式化日期和时间
<a href="#"><u>&lt;fmt:parseDate&gt;</u></a>	解析一个代表着日期或时间的字符串
<a href="#"><u>&lt;fmt:bundle&gt;</u></a>	绑定资源
<a href="#"><u>&lt;fmt:setLocale&gt;</u></a>	指定地区
<a href="#"><u>&lt;fmt:setBundle&gt;</u></a>	绑定资源
<a href="#"><u>&lt;fmt:timeZone&gt;</u></a>	指定时区
<a href="#"><u>&lt;fmt:setTimeZone&gt;</u></a>	指定时区
<a href="#"><u>&lt;fmt:message&gt;</u></a>	显示资源配置文件信息
<a href="#"><u>&lt;fmt:requestEncoding&gt;</u></a>	设置request的字符编码

### 3. SQL 标签

提供关系型数据库交互的标签

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

标签	描述
<a href="#"><u>&lt;sql:setDataSource&gt;</u></a>	指定数据源
<a href="#"><u>&lt;sql:query&gt;</u></a>	运行SQL查询语句
<a href="#"><u>&lt;sql:update&gt;</u></a>	运行SQL更新语句
<a href="#"><u>&lt;sql:param&gt;</u></a>	将SQL语句中的参数设为指定值
<a href="#"><u>&lt;sql:dateParam&gt;</u></a>	将SQL语句中的日期参数设为指定的java.util.Date 对象值
<a href="#"><u>&lt;sql:transaction&gt;</u></a>	在共享数据库连接中提供嵌套的数据库行为元素，将所有语句以一个事务的形式来运行

### 4. XML 标签

```
<%@ taglib prefix="x"
    uri="http://java.sun.com/jsp/jstl/xml" %>
```

## 5. Jstl 函数

```
<%@ taglib prefix="fn"
      uri="http://java.sun.com/jsp/jstl/functions" %>
```

<a href="#">fn.contains()</a>	测试输入的字符串是否包含指定的子串
<a href="#">fn.containsIgnoreCase()</a>	测试输入的字符串是否包含指定的子串，大小写不敏感
<a href="#">fn.endsWith()</a>	测试输入的字符串是否以指定的后缀结尾
<a href="#">fn.escapeXml()</a>	跳过可以作为XML标记的字符
<a href="#">fn.indexOf()</a>	返回指定字符串在输入字符串中出现的位置
<a href="#">fn.join()</a>	将数组中的元素合成一个字符串然后输出
<a href="#">fn.length()</a>	返回字符串长度
<a href="#">fn.replace()</a>	将输入字符串中指定的位置替换为指定的字符串然后返回
<a href="#">fn.split()</a>	将字符串用指定的分隔符分隔然后组成一个子字符串数组并返回
<a href="#">fn.startsWith()</a>	测试输入字符串是否以指定的前缀开始
<a href="#">fn.substring()</a>	返回字符串的子集
<a href="#">fn.substringAfter()</a>	返回字符串在指定子串之后的子集
<a href="#">fn.substringBefore()</a>	返回字符串在指定子串之前的子集
<a href="#">fn.toLowerCase()</a>	将字符串中的字符转为小写
<a href="#">fn.toUpperCase()</a>	将字符串中的字符转为大写
<a href="#">fn.trim()</a>	移除首位的空白符



# EL

## JSP 表达式语言——EL

作用：

- 创建算术表达式/逻辑表达式
- 访问 javaBean 中的数据

注意：EL 可以使用整数、浮点、字符串、true、false、null

运算符也是有效的

设置属性值：

```
<jsp:setProperty name="box" property="perimeter" value="100"/>
```

**box添加属性perimeter, 值为100**

EL 格式：

使用\${para}来获取表达式的值

如果一个属性有子属性：使用点运算符：.

如\${student.id}

- 取消对\${}中表达式的计算：

```
<%@ page isELIgnored ="true|false" %>
```


- EL 中的函数

定义函数：

`${namespaceName : funcName(prar1,prar2)}`

如果需要使用自定义函数，需要使用 taglib 引入该函数库

- 隐含对象

隐含对象	描述
pageScope	page 作用域
requestScope 	request 作用域
sessionScope	session 作用域
applicationScope	application 作用域
param 	Request 对象的参数，字符串
paramValues	Request对象的参数，字符串集合
header	HTTP 信息头，字符串
headerValues	HTTP 信息头，字符串集合
initParam	上下文初始化参数
cookie	Cookie值
pageContext	当前页面的pageContext

您可以在表达式中使用这些对象，就像使用变量一样。接下来会给出几个例子来更

## ■ PageContext

pageContext 对象是 JSP 中 pageContext 对象的引用。通过 pageContext 对象，您可以访问 request 对象。比如，访问 request 对象传入的查询字符串

```
${pageContext.request.queryString}
```

