# Jsp：

### 常规语法：

<%! 声明变量、方法、类 %>:作用域为整个页面

<%= 引用方法、变量、类.属性|方法 %>：

<% 普通代码 %>

<%@ 指令名 属性="属性值"%> ：指令名有page、include、和taglib三种

【page】：

**<%@ page**

**[ language="java" ]**

**[ extends="package.class" ]**

**[import="{package.class | package.\*},..." ]**

**[ session="true | false" ]**

**[ buffer="none | 8kb | sizekb" ]**

**[ autoFlush="true | false" ]**

**[ isThreadSafe="true | false" ]**

**[ info="text" ]**

**[ errorPage="relativeURL" ]**

**[contentType="mimeType;charset=characterSet]"|"text/html; charset=ISO-8859-1" ]**

**[ isErrorPage="true | false" ]**

**%>**

**【**include**】：**

**include指令：修改后要重新编译**

<%@ include file="relativeURL" %> 或

<%@ include file="相对位置" %>

**JSP动作组件：修改后立即生效**

< jsp:include page=“文件名” flush=“true”/>

**【**taglib**】：**

### JSP动作组件：

<jsp:include>：在页面被请求的时候引入一个文件。

< jsp:param>：在动作组件中引入参数信息。

< jsp:forward>：把请求转到一个新的页面。

< jsp:setProperty>：设置JavaBean的属性。

< jsp:getProperty>：输出某个JavaBean的属性。

< jsp:useBean>：寻找或者实例化一个JavaBean。

### JSP内置对象：



#### request对象

|  |  |
| --- | --- |
| **方 法 名 称** | **方 法 说 明** |
| getCookies() | 返回客户端的Cookie对象，结果是一个Cookie数组 |
| getHeader(String name) | 获得HTTP协议定义的请求头信息 |
| getAttribute(String name) | 返回name指定的属性值 |
| getattributeNames() | 返回request对象所有属性的名字 |
| getHeaderNames() | 返回所有request header的名字 |
| getHeaders(String name) | 返回所有指定名字的request header 的所有值 |
| getMethod() | 获得客户端向服务器端传送数据的方法 |
| getParameter(String name) | 获得指定的客户端传送给服务器的参数值 |
| getParameterNames() | 获得客户点传送给服务器端的所有的参数名 |
| getParameterValues(String name) | 获得指定参数的所有值 |
| getQueryString( ) | 获得查询字符串，由客户端以GET方式传送 |
| getRequestURI() | 获得发出请求字符串的客户端地址 |
| getServletPath() | 获得客户端所请求的脚本文件的文件路径 |
| setAttribute(String name, java.lang.Object o) | 设定名称为name的request属性值 |
| getServerName() | 获得服务器的名字 |
| getServerPort() | 获得服务器的端口号 |
| getRemoteAddr() | 获得客户端的IP地址 |
| getRemoteHost() | 获得客户端的名字 |
| getProtocol( ) | 获取协议 |

#### reponse对象：

* addCookie(Cookie cookie)：向response对象添加一个Cookie对象
  + <%Cookie mycookie = new Cookie(“name”,”hy1”)%>
* addHeader(String name, String Value):添加HTTP文件头
* containsHeader(String name)：判断指定名字的HTTP头是否存在
* sendError(int sec)：向客户端发送错误信息
  + <%Response.sendError(response.SC\_NO\_CONTENT)%>
* setHeader(String name, String value):设定指定名字的HTTP头
  + <%reponse.setHeader(“refresh”,”5”)%>
* setContenteType(String vlaue):设定返回reponse实体的类型
  + <%response.setContentType(“Application/pdf”)%>
* setRedirect(String url):将请求重新定位到一个新的页面

#### session对象：

* “session” 对象代表服务器与客户端所建立的会话，会话是同一个客户发起的多个请求的集合。当需要在不同的JSP页面中保留客户信息的情况下使用，比如在线购物、客户轨迹跟踪等。
* 通过调用HttpServletRequest的getSession()方法可以查看当前请求的会话对象
  + HttpSession session = request.getSession(true);
* Session 通过setAttibute()和getAttribute()方法来添加或者读取存储的属性。

#### application对象：

|  |  |
| --- | --- |
| **方法名称** | **方法说明** |
| getAttribute(String name) | 返回由name指定的属性值 |
| setAttribute(String name, Object o) | 用o设置属性值 |
| getattributeNames() | 返回所有对象属性的名字 |
| getInitParamter(String name) | 返回初始化参数 |
| getServerInfo() | 获得当前版本servlet编译器信息 |
| getMimeType() | 获取组件MIME类型 |
| getReadPath() | 获取组件在服务器上的真实路径 |

#### Out对象：

* “out” 对象代表了向客户端发送数据的对象，与“response” 对象不同，通过“out” 对象发送的内容将是浏览器需要显示的内容，是文本一级的，可以通过“out” 对象直接向客户端写一个由程序动态生成HTML文件。
* 常用的方法除了pirnt和println之外，还包括clear、clearBuffer、flush、getBufferSize和getRemaining，这是因为“out” 对象内部包含了一个缓冲区，所以需要一些对缓冲区进行操作的方法。

#### config对象：

* “config” 对象提供一些配置信息，常用的方法有getInitParameter和getInitParameterNames，以获得Servlet初始化时的参数。

### 表达式语言：

${expr[能获取到一个确切值的语句]}

#### EL有效表达式支持的内容类型

|  |  |
| --- | --- |
| 内容类型 | 合法取值 |
| **Boolean** | **true 和 false** |
| **Integer** | **与 Java 类似。可以包含任何正数或负数，例如 24、-45、567** |
| **Floating Point** | **与 Java 类似。可以包含任何正的或负的浮点数，例如 -1.8E-45、4.567** |
| **String** | **任何由单引号或双引号限定的字符串。对于单引号、双引号和反斜杠，使用反斜杠字符作为转义序列。必须注意，如果在字符串两端使用双引号，则单引号不需要转义。** |
| **Null** | **null** |

#### EL支持的隐式对象 ：直接拿来用的

|  |  |  |
| --- | --- | --- |
| 类别 | 标识符 | 描述 |
| **JSP** | **pageContext** | **JSP 页的上下文。** |
| **作用域** | **pageScope** | **与page作用域的属性的名称和值相关联的 Map 类** |
| **requestScope** | **与request作用域的属性的名称和值相关联的 Map 类** |
| **sessionScope** | **与session作用域的属性的名称和值相关联的 Map 类** |
| **applicationScope** | **与application作用域的属性的名称和值相关联的 Map 类** |
| **请求参数** | **param** | **存储请求参数名称－值对的 Map 类。** |
| **paramValues** | **将请求参数的所有值作为 String 数组存储的 Map 类** |
| **请求头** | **header** | **按名称存储请求头部主要值的 Map 类** |
| **headerValues** | **将请求头部的所有值作为 String 数组存储的 Map 类** |
| **Cookie** | **cookie** | **按名称存储请求附带的 cookie 的 Map 类** |
| **初始化参数** | **initParam** | **按名称存储 Web 应用程序上下文初始化参数的 Map 类** |

#### EL案例：

程序4－41：el2.jsp

<%@page contentType="text/html"%>

<%@page pageEncoding="UTF-8"%>

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

<%

HashMap values = new HashMap();

values.put("money", "沪指今日大涨200点！");

values.put("热点", "嫦娥一号发回第一幅月球照片！");

values.put("奥运", "奥运会准备一切就绪！");

String newscolor="blue";

request.setAttribute("news",values);

request.setAttribute("color",newscolor);

%>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>EL表达式的存取符和隐含对象</title>

</head>

<body>

<h1> Hello ${param.name}!</h1>

<font color="${requestScope.color}" >

<h3>

财经：${requestScope.news.money}<br>

热点：${requestScope.news.热点}<br>

奥运：${requestScope.news["奥运"]}<br>

其它：${requestScope.news.other}

</h3></font>

</body>

</html>