第 7 章 JSP 表达式

7.1 表达式语言（EL）作用（优点）

1.对存储对象的简洁访问

如果要输出名为“var”的“作用域变量”（用 setAttribute()方法存储在 pageContext、HttpServletRequest、HttpSession、ServletContext 中的对象）只需使用${var}。

2.提供对 JavaBean 属性的简化记法

要输出作用域变量 company 的 companyName 属性（即 getCompanyName 方法的结果），只需使用${company.companyName}

3.对集合元素的简易访问

要访问数组、List 或 Map 中的元素，只要使用${var[indexOfKey]}

4.能够简洁地访问请求参数、Cookie 和其他请求数据（隐含对象/内置对象）

如果要访问标准类型的请求参数，可以使用几个预定义的隐含对象。

7.2 表达式语言语法

1. 基本格式

${表达式}　　//${a}，${a+b}

2. 输出包含${}内容的方法

以“\”开头，例如：\${exp}。

3. EL 运算符

（1）算术运算符：+、-、*、/（或 div）、%（或 mod）

（2）逻辑运算符：==（或 eq）、!=（或 ne）、<（lt）、>（gt）、<=（或 le）、>=（或 ge）、&&（或 and）、||（或 or）、!（或 not）

（3）条件运算符：ask?ex1:ex2

（4）空运算符：empty(集合名)，例如，${empty(key_array) }

7.3 使用 EL 访问作用域变量

1.访问 request 作用域变量

（1）设置作用域变量

request.setAttribute("requestVar", "请求作用域变量");

（2）获取作用域变量

${requestVar }

（3）实例：符号使用方法

<% request.setAttribute("a", "100");

request.setAttribute("b", "200"); %>

显示：

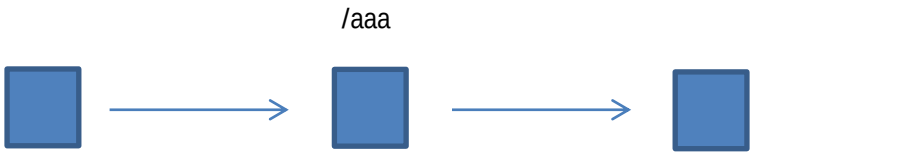
${a}+${b} = 100 + 200 //显示原值

${a + b} = 300

${a}+${b} =${a + b}//100 + 200=300

2. 示例：

a.jsp



a.jsp MyServlet 转发 b.jsp（显示数据）

（1）a.jsp

```
<a href=”aaa”>设置内置对象属性</a>
```

（2）MyServlet.java

```
request.setAttribute(“ln”,”辽宁”);
request.getRequestDispatcher(“b.jsp”).forward(request,response);
```

（3）b.jsp

```
${ln}
```

【拓展 + 学生回答】session 和 application 作用域

2.访问 session 作用域变量	3.访问 application 作用域变量
（1）设置作用域变量	（1）设置作用域变量
HttpSession session1 = request.getSession();	ServletContext application1 = getServletContext();

session1.setAttribute("sessionVar", "会话作用域变量"); （2）获取作用域变量 ${sessionVar }	application1.setAttribute("applicationVar", "环境作用域变量"); （2）获取作用域变量 ${applicationVar }

7.4 获取 JavaBean 属性

1.语法格式

① “.”运算符

```
${BeanName.PropertyName}
例如，年龄：${user.age }
```

② “[]”运算符

```
${BeanName[“PropertyName”]}
例如，用户名：${user["name"] }
```

2. 实例：获取 JavaBean 属性

（1）编写 JavaBean 类 User

```
public class User {
    private String name = “张三”;
    private int age;
    getters 和 setters
}
```

（2）创建 bean 对象

```
<%@ page contentType="text/html; charset=gb2312"    pageEncoding="gb2312"%>
<jsp:useBean id="user" class="com.yp.bean.UserBean" scope=" page/request/session/appliaction"/>
```

（3）使用存取运算符“.”（或“[]”）获取 JavaBean 属性

```
① “.”运算符：年龄：${user.age }
② “[]”运算符：用户名：${user["name"] }
```

7.5 EL 的内置对象

EL 表达式中定义的内置对象共有 11 个，分为 3 类：

1.pageContext 对象（JSP 页面上下文）

（1）作用：用于访问 JSP 内置对象，如请求、响应、会话、输出等。

（2）获取 HTTP 方法（get 或 post）

${pageContext.request.method}

（3）获取用户 IP 地址

${pageContext.request.remoteAddr}

==========

2.param 对象

（1）${param.var}

相当于 request.getParameter(“var”);

（2）作用：获取表单控件值

```
<input name=”controlName” />
${param.controlName}
```

（3）实例：

```
<form action="">
    <input type="text" name="mytxt" /> <input type="submit" value="ok" />
</form>
<c:if test="${param.mytxt > 100 }">
    <c:redirect url="welcome.jsp">
            <c:param name="username" value="Jhon" />
    </c:redirect>
</c:if>
```

3.paramValues 对象

（1）作用：获取多个复选表单控件值

（2）${paramValues.var}相当于 request.getParameterValues(“var”);

在 JSP 页面放置一个复选框，代码如下：

```
<input type = "checkbox" name = "hobby" value = "1" id = "hobby">1
<input type = "checkbox" name = "hobby" value = "2" id = "hobby">2
<input type = "checkbox" name = "hobby" value = "3" id = "hobby">3
```

使用表达式获取 hobby 值，代码如下：

```
${paramValues.hobby[0]}
${paramValues.hobby[1]}
${paramValues.hobby[2]}
```

4.header 对象

（1）作用：获取 HTTP 请求的一个具体请求头的 header 值。

（2）请求头信息

```
Accept: */*          浏览器可接受的 MIME 类型。
Host: localhost:8080          初始 URL 中的主机和端口。
Referer : null          包含一个 URL，用户从该 URL 代表的页面出发访问当前请求的页面。
Accept-Charset：浏览器可接受的字符集。
Accept-Language : zh-CN          浏览器所希望的语言种类。
Accept-Encoding : gzip, deflate          浏览器能够进行解码的数据编码方式，比如 gzip。
User-Agent : Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/7.0)          浏览器类型。
Connection : Keep-Alive          表示是否需要持久连接。
Cookie : JSESSIONID=E43BC896921041C6BB3733C085FB7938          这是最重要的请求头信息之一。
```

（3）${header.var}相当于 request.getHeader(“var”);

例如：要获取 HTTP 请求的 header 的 Host 属性，可以用：

${header.host}或者${header[host]}

获取 user-agent 属性，必须使用下面格式：

${header["user-agent"]}

【外引】百度搜索

5.requestScope 对象（简单讲解【查看示例】）

（1）作用：获取生命周期为 request 的键值

作用域	EL 内置对象	属性	属性值
请求	requestScope	a	10
会话	sessionScope	a	20
共享	applicationScope	a	30

（2）实例 1、获取 request 作用域属性值

```
<%request.setAttribute("requestKey", 200);%>
${requestScope.requestKey }
```

（3）实例 2、获取 request 作用域的 JavaBean 属性值

<jsp:useBean id="request_user" class="com.yp.bean.UserBean" scope="request"/>

<jsp:setProperty property="name" name="request_user" value="requestZhangsan"/>

${requestScope.request_user.name}

【拓展 + 学生回答】

9.pageScope 对象
（1）作用：获取 page 作用域变量（属性）值
（2）实例 1、获取 page 作用域变量值
//设置键值对（变量）
<%pageContext.setAttribute("pageKey", 100);%>
//获取键值
${pageScope.pageKey }
（3）实例 2、获取 page 作用域的 JavaBean 属性值
<jsp:useBean id="page_user" class="com.yp.bean.UserBean" scope="page"/>
//设置 bean 属性值
<jsp:setProperty property="name" name="page_user" value="pageZhangsan"/>
//获取 page 生命周期的 bean 属性
${pageScope.page_user.name}
	10.sessionScope 对象
	（1）作用：获取生命周期为 session 的键值
	（2）实例 1
	<%session.setAttribute("sessionKey", 300);%>
	${sessionScope.sessionKey }
	（3）实例 2
	<jsp:useBean id="session_user" class="com.yp.bean.UserBean" scope="session"/>
	<jsp:setProperty property="name" name="session_user" value="sessionZhangsan"/>
	${sessionScope.session_user.name}
		11.applicationScope 对象
		（1）作用：获取生命周期为 application 的键值
		（2）实例 1
		<%application.setAttribute("applicationKey", 400);%>
		${applicationScope.applicationKey }
		（3）实例 2
		<jsp:useBean id="application_user" class="com.yp.bean.UserBean" scope="application"/>
		<jsp:setProperty property="name" name="application_user" value="applicationZhangsan"/>
		${applicationScope.application_user.name}

==========

7.6 访问集合元素

1.访问数组元素

（1）设置数组

String array[] = { "A", "B", "C" };

（2）存储数组到范围变量中

request.setAttribute("key_array", array);

（3）获取数组值

${key_array[0] },${key_array[1] },${key_array[2] }

2.访问列表集合元素

（1）创建列表并添加列表元素

```
ArrayList<Object> list = new ArrayList<Object>();
list.add(100);
list.add("E");
list.add(new Date());
```

（2）存储列表到范围变量中

request.setAttribute("key_list", list);

（3）获取列表元素值

${key_list[0] },${key_list[1] },${key_list[2] }

~~3.访问 HashMap 集合元素（选讲）~~

（1）创建 HashMap 对象并添加元素

```
HashMap<String,Object> map = new HashMap<String,Object>();
map.put("key_a", 300);
map.put("key_b", "刘德华");
map.put("key_c", new Date());
```

（2）存储列表到范围变量中

request.setAttribute("key_map ", map);

（3）获取列表元素值

${key_map["key_a"] },${key_map["key_b"] },${key_map["key_c"] }