

Web应用开发 之 表达式语言

赵小敏

浙江工业大学计算机科学与技术学院

表达式语言

- 5.1 理解表达式语言
- 5.2 EL运算符
- 5.3 使用EL访问数据
- 5.4 EL隐含变量

5.1 理解表达式语言

- 表达式语言（Expression Language, EL）是JSP 2.0新增的功能，它是一种可以在JSP页面中使用的[数据访问语言](#)。
- 表达式语言的主要目标是使动态网页的设计、开发和维护更加容易。作为一种数据访问语言，EL具有自己的语法，定义了运算符，还有一些保留字。
- 用EL可在JSP页面中访问应用程序数据，无需使用小脚本（<%和%>）或JSP请求时表达式（<%=和%>）。
- 作为JSP开发员，我们的工作创建EL表达式并将其添加到JSP的响应中。

5.1.1 表达式语言的调用

- 表达式语言的使用形式如下：

`${expression}`

- 表达式语言是以\$开头，后面是一对大括号，括号里面是合法的EL表达式。
- 可以出现在JSP页面的模板文本中，也可以出现在JSP标签的属性值中，只要属性允许常规的JSP表达式即可。

5.1.1 表达式语言的调用

- 在JSP模板文本中使用EL表达式:

客户名: \${customer.custName}

Email地址: \${customer.email}

- 在JSP标准动作的属性中使用EL表达式:

<jsp:include page = "\${expression1}" />

<c:out value = "\${expression2}" />

5.1.2 表达式语言的功能

- EL的目的是简化页面的表示逻辑，主要功能包括：
 - 提供了一组简单的运算符。
 - 访问作用域变量、JavaBeans对象、集合元素。
 - 提供一组隐含变量，访问请求参数、作用域数据等。
 - EL提供了在JSP中使用Java函数的功能。

5.1.3 表达式语言与JSP表达式的区别

- JSP表达式的使用格式为：

`<%=expression %>`

- 这里的expression为合法的Java表达式，它属于脚本语言的代码。在expression中可以使用由脚本声明的变量。
- EL表达式的格式为：
`${expression}`
- 这里的expression是符合EL规范的表达式，并且不需要包含在标签内。在EL表达式中不能使用脚本中声明的变量。

5.2 EL运算符

- EL作为一种简单的数据访问语言，提供了一套运算符。包括：
 - 算术运算符
 - 关系运算符和逻辑运算符
 - 条件运算符
 - empty运算符
 - 属性与集合访问运算符

5.2 EL运算符

- 5.2.1 算术运算符
- 5.2.2 关系与逻辑运算符
- 5.2.3 条件运算符
- 5.2.4 empty运算符
- 5.2.5 属性与集合元素访问运算符

5.2.1 算术运算符

算术运算符	说明	示 例	结果
+	加	$\{6.80 + -12\}$	-5.2
-	减	$\{15-5\}$	10
*	乘	$\{2 * 3.14159\}$	6.28318
/或div	除	$\{25 \text{ div } 4\}$ 与 $\{25/4\}$	6.25
%或mod	取余	$\{24 \text{ mod } 5\}$ 与 $\{24 \% 5\}$	4

5.2.2 关系与逻辑运算符

- EL的关系运算符与一般的Java代码的关系运算符类似

关系运算符	说明	示 例	结果
== 或 eq	相等	<code>\${3==5}</code> 或 <code>\${3 eq 5}</code>	false
!= 或 ne	不相等	<code>\${3!=5}</code> 或 <code>\${3 ne 5}</code>	true
< 或 lt	小于	<code>\${3<5}</code> 或 <code>\${3 lt 5}</code>	true
> 或 gt	大于	<code>\${3>5}</code> 或 <code>\${3 gt 5}</code>	false
<= 或 le	小于等于	<code>\${3<=5}</code> 或 <code>\${3 le 5}</code>	true
>= 或 ge	大于等于	<code>\${3>=5}</code> 或 <code>\${3 ge 5}</code>	false

5.2.2 关系与逻辑运算符

- 关系表达式产生的boolean型值可以与EL的逻辑运算符结合运算

逻辑运算符	说明	示 例	结果
&& 或 and	逻辑与	<code>\${(9.2 >=4) && (1e2 <= 63)}</code>	false
或or	逻辑或	<code>\${(9.2>= 4) (1e2 <= 63)}</code>	true
! 或not	逻辑非	<code>\${ not 4 >= 9.2) }</code>	true

5.2.3 条件运算符

- EL的条件运算符的语法是：

expression ? expression1 : expression2

- 表达式的值是基于expression的值，它是一个boolean表达式。
- 如果expression的值为true，则返回expression1结果；
- 如果expression的值为false，则返回expression2的结果。

5.2.3 条件运算符

- $\$(5 * 5) == 25 ? 1 : 0$ 的结果为 1;
- $\$(3 > 2) \&\& !(12 > 6) ? \text{"Right"} : \text{"Wrong"}$ 的结果为Wrong;
- $\$(("14" \text{ eq } 14.0) \&\& (14 \text{ le } 16) ? \text{"Yes"} : \text{"No"})$ 的结果为Yes;
- $\$(4.0 \text{ ne } 4) || (100 \leq 10) ? 1 : 0$ 的结果为 0。

5.2.4 empty运算符

- empty运算符的使用格式为:

`${empty expression}`

- 它判断expression的值是否为null、空字符串、空数组、空Map或空集合，若是则返回true，否则返回false。

5.2.5 属性与集合访问运算符

- 属性访问运算符用来访问对象的成员，集合访问运算符用来检索Map、List或数组对象的元素。这些运算符在处理隐含变量时特别有用。
- 在EL中，这类运算符有下面两个。
 - ① 点号（.）运算符。
 - ② 方括号（[]）运算符。

1. 点号 (.) 运算符

- 点号运算符用来访问Map对象一个键的值或bean对象的属性值，例如：param是EL的一个隐含对象，它是一个Map对象，下面代码返回param对象username请求参数的值：

`${param.username}`

- 假设customer是Customer类的一个实例，下面代码访问该实例的custName属性值：

`${customer.custName}`

2. 方括号（[]）运算符

- 方括号运算符除了可以访问Map对象键值和bean的属性值外，还可以访问List对象和数组对象的元素。例如：

`${param ["username"]}`

或 `${ param ['username']}`

`${customer["custName"]}`

2. 方括号（[]）运算符

- 程序eloperator.jsp
- 为了在JSP页面中输出文本 $\${2+5}$ ，需要在“\$”符号前使用转义字符“\”，否则将输出EL表达式的值。

5.3 使用EL访问数据

- 5.3.1 访问作用域变量
- 5.3.2 访问JavaBeans属性
- 5.3.3 访问集合元素
- 5.3.4 访问静态方法和静态属性

5.3.1 访问作用域变量

- 在JSP页面中，可以使用JSP表达式访问作用域变量。

做法如下：

- ① 在Servlet中使用setAttribute()将一个变量存储到某个作用域对象上，如HttpServletRequest、HttpSession及ServletContext等。
- ② 使用RequestDispatcher对象的forward()将请求转发到JSP页面，在JSP页面中调用隐含变量的getAttribute()返回作用域变量的值。

5.3.1 访问作用域变量

- 使用EL可方便地访问作用域变量。要输出作用域变量的值，只需在EL中使用变量名即可，例如：

`${variable_name}`

- 对该表达式，容器将依次在页面作用域、请求作用域、会话作用域和应用作用域中查找名为variable_name的属性。如果找到该属性，则调用它的toString()并返回属性值。如果没有找到，则返回空字符串（不是null）。

5.3.1 访问作用域变量

- 如何访问作用域变量的示例
- 程序 `VariableServlet.java`
- 程序 `variables.jsp`

VariableServlet.java

```
@WebServlet("/VariableServlet")
public class VariableServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException, IOException {
        request.setAttribute("attrib1", new Integer(100));
        HttpSession session = request.getSession();
        session.setAttribute("attrib2", "Java World!");
        ServletContext application = getServletContext();
        application.setAttribute("attrib3", new java.util.Date());
        request.setAttribute("attrib4", "请求作用域");
        session.setAttribute("attrib4", "会话作用域");
        application.setAttribute("attrib4", "应用作用域");
        // 请求转发到JSP页面
        RequestDispatcher rd =
            request.getRequestDispatcher("/variables.jsp");
        rd.forward(request, response);
    }
}
```


variables.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>
```

```
<html>
```

```
<head><title>访问作用域变量</title></head>
```

```
<body>
```

```
  <h3>访问作用域变量</h3>
```

```
  <ul>
```

```
    <li>属性1: ${attrib1}
```

```
    <li>属性2: ${attrib2}
```

```
    <li>属性3: ${attrib3}
```

```
    <li>属性4: ${attrib4}
```

```
  </ul>
```

```
</body></html>
```

5.3.2 访问JavaBeans属性

- 如果知道JavaBeans的完整名称和它的作用域，也可以使用JSP标准动作访问JavaBeans的属性：

```
<jsp:useBean id="employee" class="com.demo.Employee"  
scope="session" />
```

```
<jsp:setProperty name="employee" property="empName" value="Hacker" />
```

```
<jsp:getProperty name="employee" property="empName" />
```

5.3.2 访问JavaBeans属性

- 使用EL可以通过点号表示法很方便地访问JavaBeans的属性，如下所示：

`${employee.empName}`

- 使用表达式语言，如果没有找到指定的属性不会抛出异常，而是返回空字符串。

5.3.2 访问JavaBeans属性

- 使用表达式语言还允许访问嵌套属性。例如，如果Employee有一个address属性，它的类型为Address，而Address又有zipCode属性，可以使用简单形式访问zipCode属性。

`${employee.address.zipCode}`

- 不能使用<jsp:useBean>和<jsp:getProperty>实现。

5.3.2 访问JavaBeans属性

- 对JavaBeans属性访问的示例。
 - 程序Address.java
 - 程序Employee.java
 - 程序EmployeeServlet.java
 - 程序beanDemo.jsp
- 该例中有两个JavaBeans
 - ① Address有三个字符串类型的属性，city、street和zipCode;
 - ② Employee是在前面的类的基础上增加了一个Address类型的属性address表示地址。

```
public class Employee
{
    private String empName;
    private String email;
    private String phone;
    private Address address;
}
```

```
public class Address
{
    private String city;
    private String street;
    private String zipCode;
}
```

5.3.2 访问JavaBeans属性

- 在EmployeeServlet.java程序中创建了一个Employee对象并将其设置为请求作用域的一个属性，然后将请求转发到JSP页面
- 在JSP页面中使用EL访问客户地址的三个属性：

城市:\${employee.address.city}

街道:\${employee.address.street}

邮编:\${employee.address.zipCode}

5.3.3 访问集合元素

- 在EL中可以使用数组记法的运算符([])访问各种集合对象的元素，如数组、List对象或Map对象。
- 假设有一个上述类型的对象attributeName，可以使用下面形式访问其元素。

`$\${attributeName[entryName]}$`

5.3.3 访问集合元素

(1) 如果attributeName对象是数组，则entryName为下标。表达式返回指定下标的元素值。下面代码演示了访问数组元素。

```
<%
```

```
String[] fruit = {"apple","orange","banana"};  
request.setAttribute("myFruit", fruit);
```

```
%>
```

```
My favorite fruit is:${myFruit[2]}
```

- 上面一行还可以写成：

```
My favorite fruit is:${myFruit["2"]}
```


5.3.3 访问集合元素

(2) 如果attributeName对象是实现了List接口的对象，则entryName为索引。下面代码演示了访问List元素。

```
<%@ page import="java.util.ArrayList" %>
<%
    ArrayList<String> fruit = new ArrayList<String>();
    fruit.add("apple");
    fruit.add("orange");
    fruit.add("banana");
    request.setAttribute("myFruit", fruit);
%>
My favorite fruit is:${myFruit[2]}
```

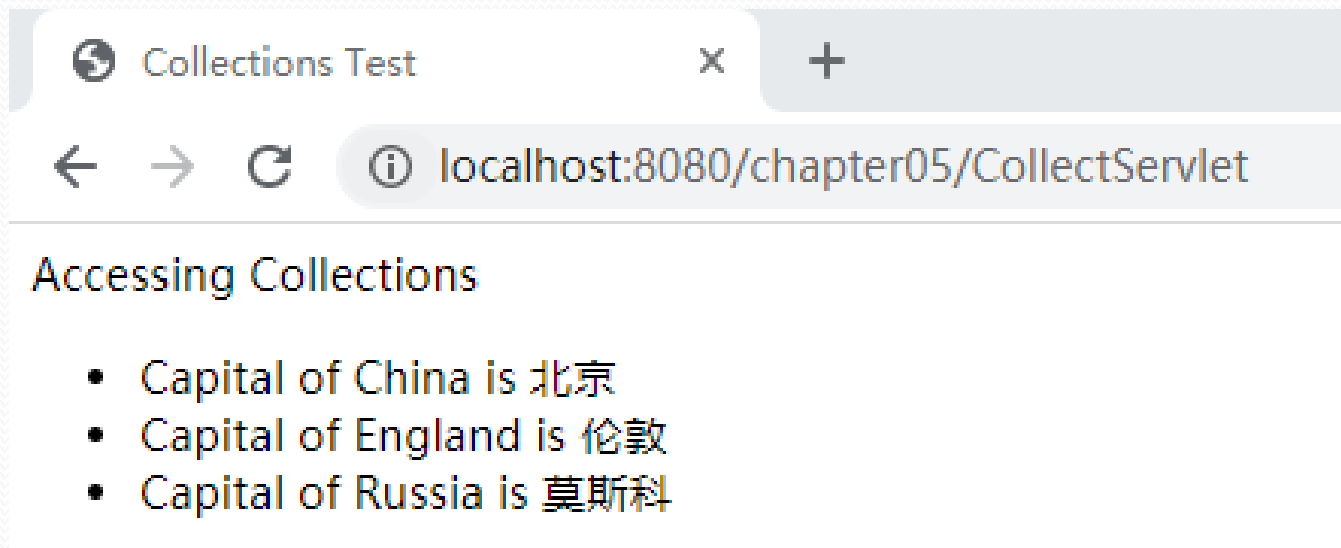
5.3.3 访问集合元素

(3) 如果attributeName对象是实现了Map接口的对象，则entryName为键，相应的值通过Map对象的get(key)获得的，例如：

```
Map<String,String> capital = new HashMap<String,String>();  
capital.put("England","伦敦");  
capital.put ("China","北京");  
capital.put ("Russia","莫斯科");  
request.setAttribute("capital", capital);  
The capital of China is:${capital["China"]}  
The capital of Russia is:${capital.Russia}
```

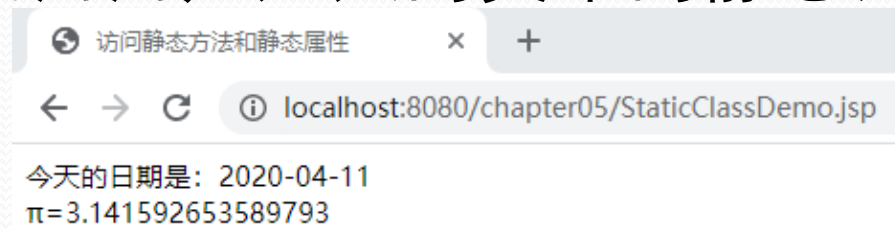
5.3.3 访问集合元素

- 程序CollectServlet.java
- 程序 collections.jsp



5.3.4 访问静态方法和静态属性

- 使用EL可以访问任何Java类中定义的静态方法和静态属性。但首先必须用page指令的import属性导入有关的类或包。
 - `<%@page import="java.time.LocalDate" %>`
- 下面代码使用EL调用LocalDate类的now()方法。
 - 今天的日期是: `${LocalDate.now()}` `
`
 - `π=${Math.PI}`
- 使用EL也可以调用用户定义的类中的静态方法和静态变量。



5.4访问EL隐含变量

- 在JSP页面的脚本中可以访问JSP隐含变量，如 `request`、`session`、`application`等。
- EL表达式中定义了一套自己的隐含变量。使用EL可以直接访问这些隐含变量。

5. 4访问EL隐含变量

变量名	说 明
pageContext	包含JSP常规隐含对象的PageContext类型对象
param	包含请求参数字符串的Map对象
paramValues	包含请求参数字符串数组的Map对象
header	包含请求头字符串的Map对象
headerValues	包含请求头字符串数组的Map对象
initParam	包含Servlet上下文参数的参数名和参数值的Map对象
cookie	匹配Cookie域和单个对象的Map对象
pageScope	包含page作用域属性的Map对象
requestScope	包含request作用域属性的Map对象
sessionScope	包含session作用域属性的Map对象
applicationScope	包含application作用域属性的Map对象

1. pageContext变量

- pageContext是PageContext类型的变量。
PageContext类依次拥有request、response、session、out和servletContext属性，使用pageContext变量可以访问这些属性的属性。

`${pageContext.request.method}`

➤ `${pageContext.request.remoteAddr}`

2. param和paramValues变量

- param和paramValues变量用来从ServletRequest中检索请求参数值。
- param变量是调用给定参数名的getParameter(String name)的结果，使用EL表示如下。

`${param.name}`

- paramValues是使用getParameterValues(String name)返回给定名称的参数值的数组。要访问参数值数组的第一个元素，可使用下面代码。

`${paramValues.name[0]}`

3. header和headerValues变量

- header和headerValues变量是从HTTP请求头中检索值，它们的运行机制与param和paramValues类似。如使用EL显示了请求头host的值：

`${header.host}`或`${header["host"]}`

- headerValues.host是一个数组，它的第一个元素可使用下列表达式之一显示。

`${headerValues.host[0]}`

`${headerValues.host["0"]}`

`${headerValues.host['0']}`

4. cookie变量

- 使用EL的cookie隐含变量得到客户向服务器发回的Cookie数组，即调用request对象的getCookies()的返回结果。
- 如果要访问cookie的值，则需要使用Cookie类的属性value（即getValue方法）。
- **`${cookie.userName.value}`**输出名为userName的Cookie的值。如果没有找到这个cookie对象，则输出空字符串。
- 使用cookie变量还可以访问会话Cookie的ID值，例如：**`${cookie.JSESSIONID.value}`**

5. initParam变量

- initParam变量存储了Servlet上下文的参数名和参数值。
- 假设在web.xml中定义了如下初始化参数。

```
<context-param>
```

```
    <param-name>email</param-name>
```

```
    <param-value>jack@163.com</param-value>
```

```
</context-param>
```

- 可以使用下面的EL表达式得到参数email的值。

```
${initParam.email}
```

5. pageScope、requestScope、sessionScope和applicationScope变量


- 这几个隐含变量是用来访问不同作用域的属性。
- 在会话作用域中添加一个表示商品价格的totalPrice属性，然后使用EL访问该属性值。

```
<%session.setAttribute("totalPrice",1000);%>
```

```
${sessionScope.totalPrice}
```

- 访问应用作用域的属性应使用applicationScope变量而不是使用pageContext变量。

EL隐含变量的使用示例implicitEL.jsp

 <http://localhost:8080/chapter05/implicitEL.jsp?userName=admin>

EL隐含变量的使用

EL表达式	值
<code>\${pageContext.request.method}</code>	GET
<code>\${param.userName}</code>	admin
<code>\${header.host}</code>	localhost:8080
<code>\${cookie.userName.value}</code>	张三
<code>\${initParam.email}</code>	jack@163.com
<code>\${sessionScope.employee.address.street}</code>	留和路288号

小结

- 表达式语言（EL）是一种数据访问语言，通过它可以很方便地在JSP页面中访问应用程序数据，无需使用小脚本和请求时表达式。
- 表达式语言最重要的目的是创建无脚本的JSP页面。为了实现这个目的，EL定义了自己的运算符、语法等，能够替代传统的JSP中的声明、表达式和小脚本。
- 使用EL可访问作用域变量、JavaBeans属性、集合元素，使用EL隐含变量可访问请求参数、请求头、Cookie和作用域变量

练习

1、有下面JSP页面，叙述正确的是 (A a没定义，当做空值，不做计算)

```
<html><body>
```

```
${(5 + 3 + a > 0) ? 1 : 2}
```

```
</body></html>
```

A. 语句合法，输出1

B. 语句合法，输出2

C. 因为a没有定义，因此抛出异常

D. 因为表达式语法非法，因此抛出异常

练习

- 2、如果使用EL显示请求的URI，下面哪个是正确的？
(B)

A. `${pageScope.request.requestURI}`

B. `${pageContext.request.requestURI}`

C. `${ request.requestURI}`

D. `${requestScope.request.requestURI}`

练习

- 3、给定一个HTML表单，其中使用了有一个名为hobbies的复选框，如下所示：

兴趣： ☐文学
 ☐体育
 ☐电脑

下面哪些表达式能够计算并得到hobbies参数的第一个值？（ AC ）

- A. `${param.hobbies}`
- B. `${ paramValues. hobbies }` 得到数组首地址 字符串
- C. `${paramValues.hobbies[o]}`
- D. `${ paramValues. hobbies[1]}`
- E. `${ paramValues.[hobbies][o]}` 抛出异常

练习

- 4、一个Web站点将管理员的Email地址存储在一个名为master-email的ServletContext参数中，如何使用EL得到这个值？（ AC ）

- A. `email me`
- B. `email me`
- C. `email me`
- D. `email me`

练习

- 5、下面页面的输出结果是什么？

`<%@ page isELIgnored="true"%>` 忽略EL表达式

`<html><head>`

`${(5 + 3 > 0) ? true : false}` 原样输出

`</body></html>`

练习

6、某银行的客户管理系统采用MVC模式设计实现，其中有一个录入客户信息的功能，请按如下要求编写程序。

(1) 编写一个类为`bank.com.model.Customer`的JavaBeans，包括3个属性：`custName`表示客户姓名，`email`表示客户邮箱，`phone`表示客户手机号；

(2) 编写一个输入客户信息页面`bank/inputCustomer.jsp`，通过表单输入客户信息，将请求转发到映射地址为`CustomerServlet.do`的`bank.com.control.CustomerServlet`类进行处理；

(3) 编写一个`bank.com.control.CustomerServlet`类，从输入客户信息页面得到客户信息，并将客户信息通过作用域共享后转发至客户信息显示页面`bank/displayCustomer.jsp`；

(4) `bank/displayCustomer.jsp`显示客户的信息，要求用EL访问数据。

Thank You !