



简化对Java代码的访问 : JSP 2.0表达式语言

JSP, Servlet, & Struts Training Courses: <http://courses.coreservlets.com>
Available in US, China, Taiwan, HK, and Worldwide

JSP and Servlet Books from Sun Press: <http://www.coreservlets.com>
*Available in English, Chinese (simplified and traditional script),
and 12 other languages*

2

议程

- 应用表达式语言的驱动力
- 基本语法
- 表达式语言和MVC构架之间的关系
- 访问作用域变量
- 访问bean的属性，数组、List和Map中的元素
- 表达式语言运算符的应用
- 表达式的条件求值

3

Servlet和JSP：处理单个请求所适用的范围

- 仅Servlet适用于：
 - 输出为某种二进制类型，例如：图像
 - 没有输出，例比：搜索引擎示例中的转发和重定向
 - 页面的格式/布局高度可变，例如网站的入口
- 仅JSP适用于：
 - 输出大部分为字符数据，例如：HTML
 - 格式/布局绝大多数比较固定
- 组合使用（MVC构架）。仅在下述情况下需要：
 - 单个请求可能产生多个外观相差很大的结果
 - 开发团队规模很大，不同的小组成员分别负责Web开发和商业逻辑
 - 需要执行复杂的数据处理任务，但布局相对固定

4

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

用RequestDispatcher 实现MVC

1. 定义一系列的bean来表示数据
2. 使用一个servlet来处理请求
 - 由这个servlet读取请求参数，检查缺失和异常的数据，等等
3. 填充bean
 - 由servlet调用商业逻辑（与具体应用相关的代码）或数据访问代码，获得相应的结果。结果放置在第一步中定义的bean中。
4. 将bean存储在请求、会话或servlet上下文中
 - Servlet调用请求、会话或servlet上下文对象的setAttribute，存储表达请求结果的bean的引用。

5

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

用RequestDispatcher 实现MVC (续)

5. 将请求转发给JSP页面。

- 由servlet确定哪个JSP页面适用于当前的情形，并使用RequestDispatcher的forward方法将控制转移给该页面。

6. 从bean中提取数据。

- JSP页面使用jsp:useBean以及与第4步中的位置相匹配的作用域访问这些bean，然后使用jsp:getProperty输出bean的属性。
- JSP页面并不创建和修改bean，它只是提取并显示由servlet创建的数据。

6

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

MVC的缺点

- 主要的缺点是最后一步：在JSP页面中表达最后的结果。
 - jsp:useBean和jsp:getProperty
 - 笨拙而冗长
 - 不能访问bean的子属性
 - JSP脚本元素
 - 造成代码难以维护
 - 违背了引入MVC的初衷
- 我们想要的是：
 - 更简洁的访问
 - 能够访问子属性
 - 为Web开发人员提供简单的语法

7

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

表达式语言的优势

- **对存储对象的简洁访问。**
 - 如果要输出名为saleItem的“作用域变量”（用setAttribute存储在PageContext，HttpServletRequest，HttpSession，或ServletContext中的对象），只需使用`${saleItem}`。
- **提供对bean属性的简化记法。**
 - 要输出作用域变量company的companyName属性（即getCompanyName方法的结果），只需使用`${company.companyName}`。而要访问作用域变量company的president属性的firstName属性，只需使用`${company.president.firstName}`。
- **对集合元素的简易访问。**
 - 要访问数组、List或Map中的元素，只要使用`${variable[indexOrKey]}`。假如索引或键能够作为合法的Java变量名，则也可以bean的点号记法来替代集合的方括号记法。

8

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

表达式语言的优势（续）

- **能够简洁地访问请求参数、cookie和其他请求数据。**
 - 如果要访问标准类型的请求数据，可以使用几个预定义的隐式对象。
- **一套小但有用的简单运算符。**
 - 在EL语言中操作对象时，可以任意地使用所提供的算术、关系、逻辑或空值测试运算符。
- **条件化输出。**
 - 如果需要在不同的输出间做出选择，不必再采用Java脚本元素，取而代之，可以使用`${test ? option1 : option2}`。
- **自动类型转换。**
 - 表达式语言排除了大部分类型转换的需要，同时也消除了很多将字符串解析成数字的代码。
- **空值替代错误消息。**
 - 大多数情况下，缺失的值或NullPointerExceptions会导致空串，而非抛出异常。

9

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

启用表达式语言

- 仅能够用于支持JSP 2.0 (servlets 2.4) 的服务器中
 - 例如：Tomcat 5，而非Tomcat 4
- 必须在web.xml文件中使用JSP 2.0
 - 从coreservlets.com提供的源代码文档中下载一个模板，或者修改Tomcat 5中jsp-examples Web应用所使用的版本（不是ROOT Web应用）。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation=
           "http://java.sun.com/xml/ns/j2ee web-app_2_4.xsd"
         version="2.4">
...
</web-app>
```

10

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

调用表达式语言

- 基本形式：**`${expression}`**
 - 这些EL元素可以出现在常规的正文，或者JSP的标签属性中，只要这些属性允许常规的JSP表达式。如下所示：
 - ``
 - `Name: ${expression1}`
 - `Address: ${expression2}`
 - ``
 - `<jsp:include page="${expression3}" />`
- 标签属性中的EL
 - 我们可以使用多个表达式（可能和静态文本混合在一起），结果会被强制转换成字符串，并拼接在一起。例如：
 - `<jsp:include page="${expr1}blah${expr2}" />`

11

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

转义特殊字符

- 要在页面输出中得到\${
 - 在JSP页面中使用\\${。
- 要在EL表达式中得到单引号
 - 使用\'
- 要在EL表达式中得到双引号
 - 使用\"

阻止表达式语言的求值

- 如果JSP页面中含有\${会怎样？
- 停用整个Web应用中的表达式语言。
 - 使用指向servlets 2.3 (JSP 1.2) 或更早版本的web.xml文件。
- 停用多个JSP页面中的表达式语言。
 - 使用web.xml中的jsp-property-group元素。
- 停用个别页面中的表达式语言。
 - 使用 <% @ page isELEnabled="false" %>
- 停用个别EL语句。
 - 对于需要跨多个JSP版本移植的JSP 1.2页面（不修改web.xml文件），可以将\$替换为$（对应\$的HTML字符实体）。
 - 如果JSP 2.0页面既含有表达式语言的语句，又含有字面 \${字符串时，可以使用\\${来输出\${。

阻止标准脚本元素的使用

- 要强制只使用EL而不使用脚本元素，可以在web.xml中使用scripting-invalid

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi=
    "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://java.sun.com/xml/ns/j2ee web-app_2_4.xsd"
  version="2.4">
  <jsp-property-group>
    <url-pattern>*.jsp</url-pattern>
    <scripting-invalid>true</scripting-invalid>
  </jsp-property-group>
</web-app>
```

14

JSP/servlet/Struts/JSE training: <http://www.coreservlets.com>

访问作用域变量

- **`${varName}`**
 - 表示在PageContext，HttpServletRequest，HttpSession，和ServletContext中，依照所列的顺序进行查找，输出这个属性名所对应的对象。
 - PageContext不适用于MVC。
- **等价的形式**
 - `${name}`
 - `<%= pageContext.findAttribute("name") %>`
 - `<jsp:useBean id="name" type="somePackage.SomeClass" scope="...">`
`<%= name %>`

15

JSP/servlet/Struts/JSE training: <http://www.coreservlets.com>

示例：访问作用域变量

```
public class ScopedVars extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        request.setAttribute("attribute1", "First Value");
        HttpSession session = request.getSession();
        session.setAttribute("attribute2", "Second Value");
        ServletContext application = getServletContext();
        application.setAttribute("attribute3",
                                new java.util.Date());
        request.setAttribute("repeated", "Request");
        session.setAttribute("repeated", "Session");
        application.setAttribute("repeated", "ServletContext");
        RequestDispatcher dispatcher =
            request.getRequestDispatcher("/el/scoped-vars.jsp");
        dispatcher.forward(request, response);
    }
}
```

16

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

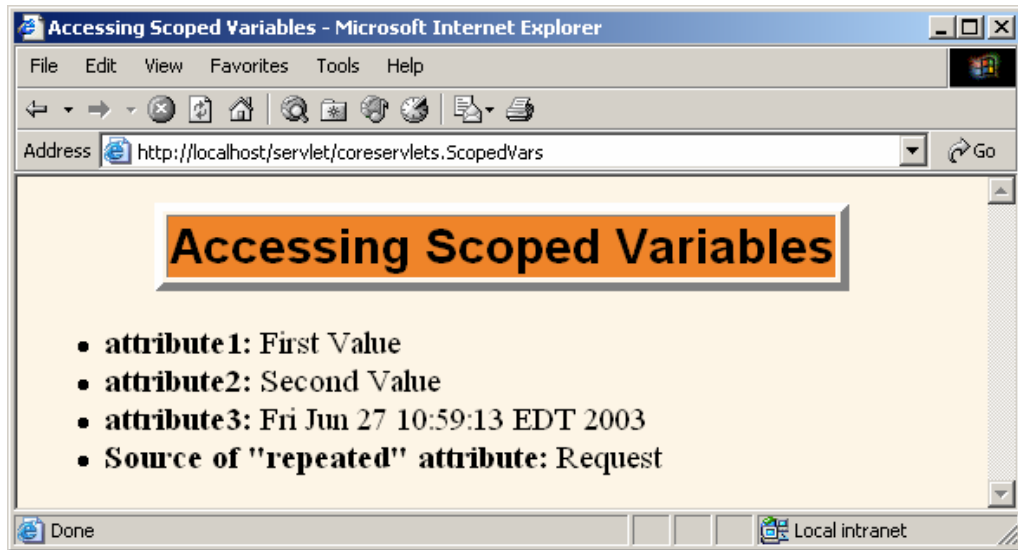
示例：访问作用域变量（续）

```
<!DOCTYPE ...>
...
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
    Accessing Scoped Variables
  </TH>
</TABLE>
<P>
<UL>
  <LI><B>attribute1:</B> ${attribute1}
  <LI><B>attribute2:</B> ${attribute2}
  <LI><B>attribute3:</B> ${attribute3}
  <LI><B>Source of "repeated" attribute:</B>
    ${repeated}
</UL>
</BODY></HTML>
```

17

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

示例：访问作用域变量（结果）



18

JSP/servlet/Struts/JSE training: <http://www.coreservlets.com>

访问bean的属性

- **`${varName.propertyName}`**
 - 表示查找给定名称的作用域变量，并输出指定的bean属性
- **等价的形式**
 - `${customer.firstName}`
 - ```
<% @ page import="coreservlets.NameBean" %>
<%
NameBean person =
 (NameBean)pageContext.findAttribute("customer");
%>
<%= person.getFirstName() %>
```

19

JSP/servlet/Struts/JSE training: <http://www.coreservlets.com>

## 访问bean的属性 ( 续 )

- 等价的形式
  - `${customer.firstName}`
  - ```
<jsp:useBean id="customer"
              type="coreservlets.NameBean"
              scope="request, session, or application" />
<jsp:getProperty name="customer"
                 property="firstName" />
```
- 这要好于之前的幻灯片上的脚本。
 - 但要求知道作用域
 - 且不能访问子属性。
 - 下面的语句没有对应的非Java语句
`${customer.address.zipCode}`

20

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

点号和数组记法的等效性

- 等价的形式
 - `${name.property}`
 - `${name["property"]}`
- 使用数组记法的理由
 - 访问数组、列表或其他集合
 - 参见随后的幻灯片
 - 在请求期间计算属性的名称。
 - `{name1[name2]}` (name2周围没有引号)
 - 使用不合乎Java变量名规定的名称
 - `{foo["bar-baz"]}`
 - `{foo["bar.baz"]}`

21

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

示例：访问bean的属性

```
public class BeanProperties extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        NameBean name = new NameBean("Marty", "Hall");
        CompanyBean company =
            new CompanyBean("coreservlets.com",
                            "J2EE Training and Consulting");
        EmployeeBean employee =
            new EmployeeBean(name, company);
        request.setAttribute("employee", employee);
        RequestDispatcher dispatcher =
            request.getRequestDispatcher
                ("/el/bean-properties.jsp");
        dispatcher.forward(request, response);
    }
}
```

22

JSP/servlet/Struts/JSF training: <http://www.coreservlets.com>

示例：访问bean的属性（续）

```
public class EmployeeBean {
    private NameBean name;
    private CompanyBean company;

    public EmployeeBean(NameBean name, CompanyBean company) {
        setName(name);
        setCompany(company);
    }

    public NameBean getName() { return(name); }

    public void setName(NameBean newName) {
        name = newName;
    }

    public CompanyBean getCompany() { return(company); }

    public void setCompany(CompanyBean newCompany) {
        company = newCompany;
    }
}
```

23

JSP/servlet/Struts/JSF training: <http://www.coreservlets.com>

示例：访问bean的属性（续）

```
public class NameBean {
    private String firstName = "Missing first name";
    private String lastName = "Missing last name";

    public NameBean() {}

    public NameBean(String firstName, String lastName) {
        setFirstName(firstName);
        setLastName(lastName);
    }

    public String getFirstName() {
        return(firstName);
    }

    public void setFirstName(String newFirstName) {
        firstName = newFirstName;
    }
    ...
}
```

24

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

示例：访问bean的属性（续）

```
public class CompanyBean {
    private String companyName;
    private String business;

    public CompanyBean(String companyName, String business) {
        setCompanyName(companyName);
        setBusiness(business);
    }

    public String getCompanyName() { return(companyName); }

    public void setCompanyName(String newCompanyName) {
        companyName = newCompanyName;
    }

    public String getBusiness() { return(business); }

    public void setBusiness(String newBusiness) {
        business = newBusiness;
    }
}
```

25

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

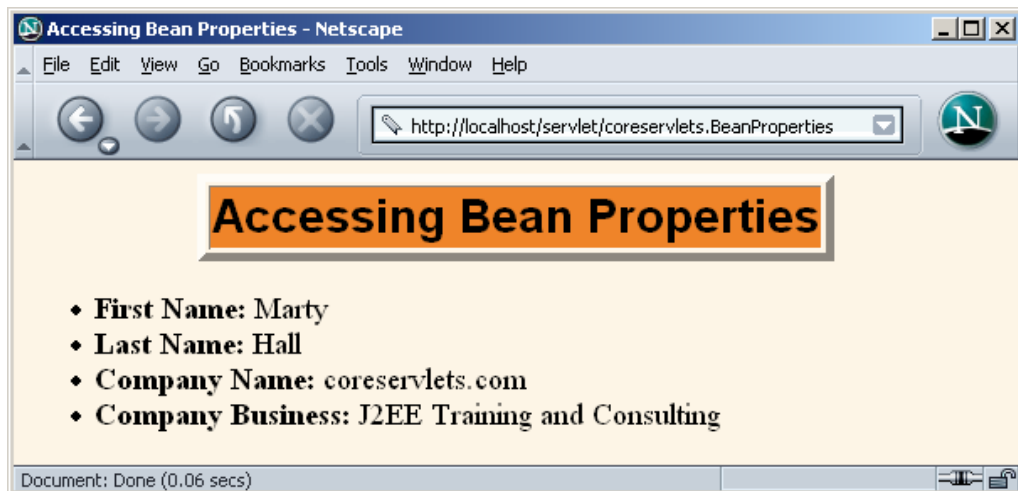
示例：访问bean的属性（续）

```
<!DOCTYPE ...>
...
<UL>
  <LI><B>First Name:</B>
    ${employee.name.firstName}
  <LI><B>Last Name:</B>
    ${employee.name.lastName}
  <LI><B>Company Name:</B>
    ${employee.company.companyName}
  <LI><B>Company Business:</B>
    ${employee.company.business}
</UL>
</BODY></HTML>
```

26

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

示例：访问bean的属性（结果）



27

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

访问集合

- **`${attributeName[entryName]}`**
- **Works for**
 - Array. Equivalent to
 - `theArray[index]`
 - List. Equivalent to
 - `theList.get(index)`
 - Map. Equivalent to
 - `theMap.get(keyName)`
- **Equivalent forms (for HashMap)**
 - `${stateCapitals["maryland"]}`
 - `${stateCapitals.maryland}`
 - But the following is illegal since 2 is not a legal var name
 - `${listVar.2}`

28

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

示例：访问集合

```
public class Collections extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        String[] firstNames = { "Bill", "Scott", "Larry" };
        ArrayList lastNames = new ArrayList();
        lastNames.add("Ellison");
        lastNames.add("Gates");
        lastNames.add("McNealy");
        HashMap companyNames = new HashMap();
        companyNames.put("Ellison", "Sun");
        companyNames.put("Gates", "Oracle");
        companyNames.put("McNealy", "Microsoft");
        request.setAttribute("first", firstNames);
        request.setAttribute("last", lastNames);
        request.setAttribute("company", companyNames);
        RequestDispatcher dispatcher =
            request.getRequestDispatcher("/el/collections.jsp");
        dispatcher.forward(request, response);
    }
}
```

29

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

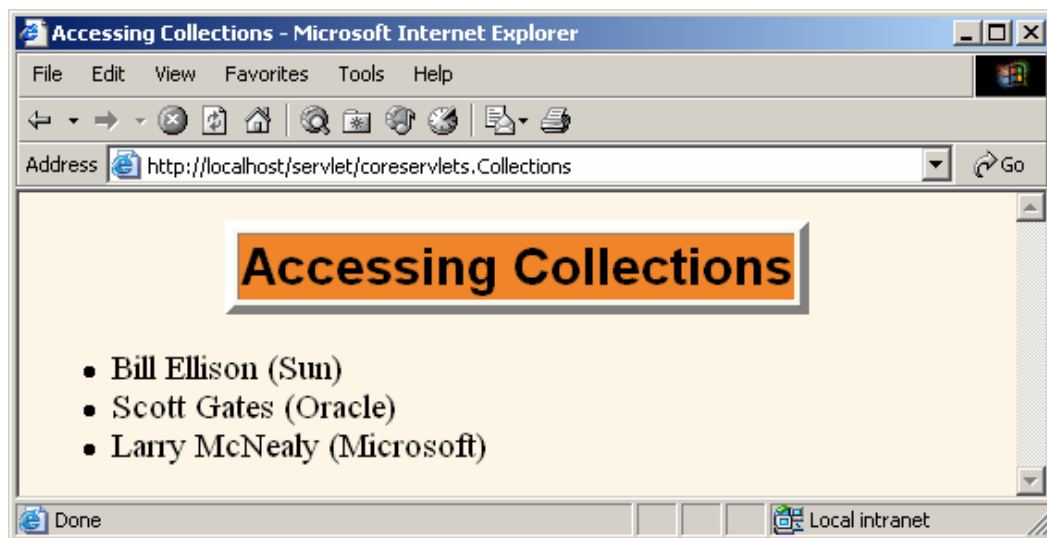
示例：访问集合（续）

```
<!DOCTYPE ...>
...
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
    Accessing Collections
  </TR>
</TABLE>
<P>
<UL>
  <LI>${first[0]} ${last[0]} (${company["Ellison"]})
  <LI>${first[1]} ${last[1]} (${company["Gates"]})
  <LI>${first[2]} ${last[2]} (${company["McNealy"]})
</UL>
</BODY></HTML>
```

30

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

示例：访问集合（结果）



31

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

引用隐式对象 (预定义变量名)

- **pageContext.** The PageContext object.
 - E.g. `${pageContext.session.id}`
- **param and paramValues.** Request params.
 - E.g. `${param.custID}`
- **header and headerValues.** Request headers.
 - E.g. `${header.Accept}` or `${header["Accept"]}`
 - `${header["Accept-Encoding"]}`
- **cookie.** Cookie object (not cookie value).
 - E.g. `${cookie.userCookie.value}` or `${cookie["userCookie"].value}`
- **initParam.** Context initialization param.
- **pageScope, requestScope, sessionScope, applicationScope.**
 - Instead of searching scopes.
- **Problem**
 - Using implicit objects usually works poorly with MVC model

32

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

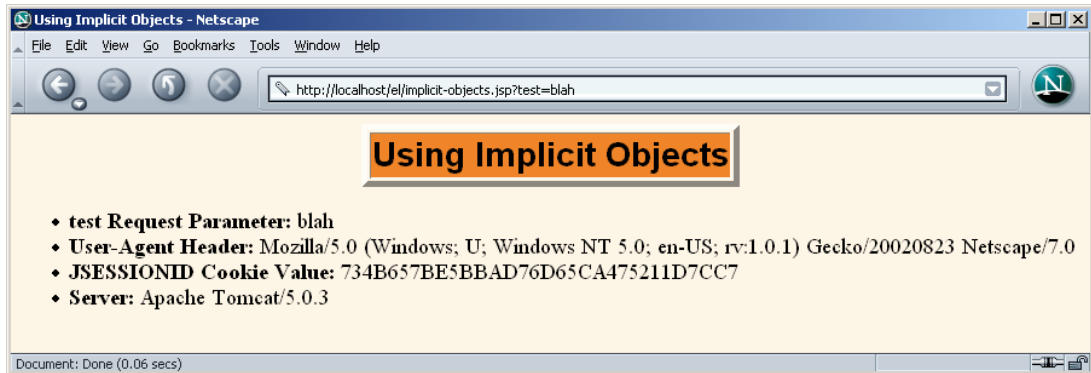
示例：隐式对象

```
<!DOCTYPE ...>
...
<P>
<UL>
  <LI><B>test Request Parameter:</B>
    ${param.test}
  <LI><B>User-Agent Header:</B>
    ${header["User-Agent"]}
  <LI><B>JSESSIONID Cookie Value:</B>
    ${cookie.JSESSIONID.value}
  <LI><B>Server:</B>
    ${pageContext.servletContext.serverInfo}
</UL>
</BODY></HTML>
```

33

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

示例：隐式对象（结果）



34

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

表达式语言的运算符

- **算术运算符**
 - + - * / div % mod
- **关系运算符**
 - == eq != ne < lt > gt <= le >= ge
- **逻辑运算符**
 - && and || or ! Not
- **空运算符**
 - Empty
 - 参数为null，空串，空数组，空列表，空map时为true。否则为false。
- **注意**
 - 为保护MVC模型，在使用时要极为克制。

35

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

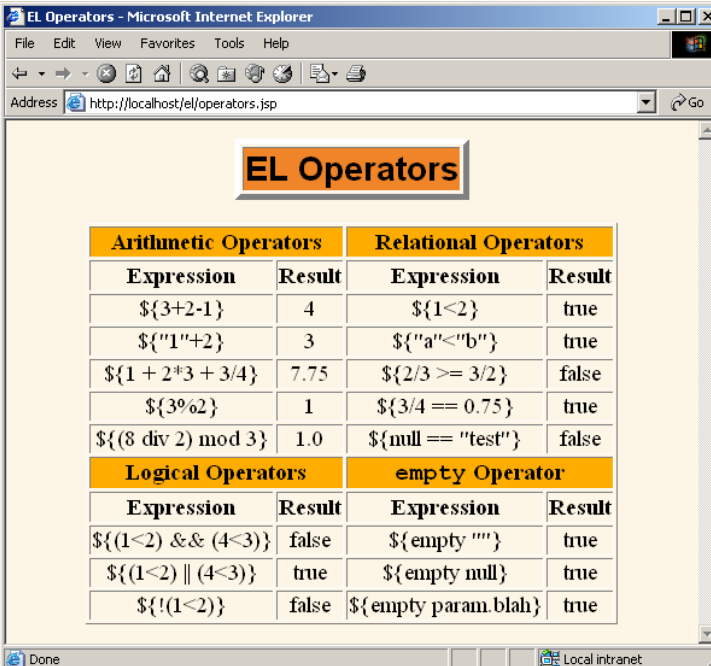
示例：运算符

```
...
<TABLE BORDER=1 ALIGN="CENTER">
  <TR><TH CLASS="COLORED" COLSPAN=2>Arithmetic Operators
    <TH CLASS="COLORED" COLSPAN=2>Relational Operators
  <TR><TH>Expression<TH>Result<TH>Expression<TH>Result
  <TR ALIGN="CENTER">
    <TD>\${3+2-1}<TD>${3+2-1}
    <TD>\${1<2}<TD>${1<2}
  <TR ALIGN="CENTER">
    <TD>\${"1"+2}<TD>${"1"+2}
    <TD>\${"a"<"b"}<TD>${"a"<"b"}
  <TR ALIGN="CENTER">
    <TD>\${1 + 2*3 + 3/4}<TD>${1 + 2*3 + 3/4}
    <TD>\${2/3 >= 3/2}<TD>${2/3 >= 3/2}
  <TR ALIGN="CENTER">
    <TD>\${3%2}<TD>${3%2}
    <TD>\${3/4 == 0.75}<TD>${3/4 == 0.75}
  ...
```

36

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

示例：运算符（结果）



Arithmetic Operators		Relational Operators	
Expression	Result	Expression	Result
$\{3+2-1\}$	4	$\{1<2\}$	true
$\{1+2\}$	3	$\{a<b\}$	true
$\{1 + 2*3 + 3/4\}$	7.75	$\{2/3 \geq 3/2\}$	false
$\{3\%2\}$	1	$\{3/4 == 0.75\}$	true
$\{(8 \text{ div } 2) \text{ mod } 3\}$	1.0	$\{\text{null} == \text{"test"}\}$	false
Logical Operators		empty Operator	
Expression	Result	Expression	Result
$\{(1<2) \&\& (4<3)\}$	false	$\{\text{empty} \text{" "}\}$	true
$\{(1<2) \ \ (4<3)\}$	true	$\{\text{empty null}\}$	true
$\{!(1<2)\}$	false	$\{\text{empty param.blah}\}$	true

37

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

表达式的条件求值

- **`{ test ? expression1 : expression2 }`**
 - 对test求值，输出表达式1或表达式2
- **存在的问题**
 - 相对较弱
 - JSTL提供的c:if和c:choose比它要好得多。
 - 诱使开发人员将商业（业务）/处理逻辑放在JSP页面中。
 - 只应用在表示逻辑中。
 - 即使如此，也要尽量使用其他一些替代方式。

示例：条件求值

```
public class Conditionals extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        SalesBean apples =
            new SalesBean(150.25, -75.25, 22.25, -33.57);
        SalesBean oranges =
            new SalesBean(-220.25, -49.57, 138.25, 12.25);
        request.setAttribute("apples", apples);
        request.setAttribute("oranges", oranges);
        RequestDispatcher dispatcher =
            request.getRequestDispatcher
                ("/el/conditionals.jsp");
        dispatcher.forward(request, response);
    }
}
```

示例：条件求值（续）

```
public class SalesBean {
    private double q1, q2, q3, q4;

    public SalesBean(double q1Sales,
                     double q2Sales,
                     double q3Sales,
                     double q4Sales) {
        q1 = q1Sales; q2 = q2Sales;
        q3 = q3Sales; q4 = q4Sales;
    }

    public double getQ1() { return(q1); }
    public double getQ2() { return(q2); }
    public double getQ3() { return(q3); }
    public double getQ4() { return(q4); }
    public double getTotal() {
        return(q1 + q2 + q3 + q4); }
}
```

40

JSP/servlet/Struts/JSF training: <http://www.coreservlets.com>

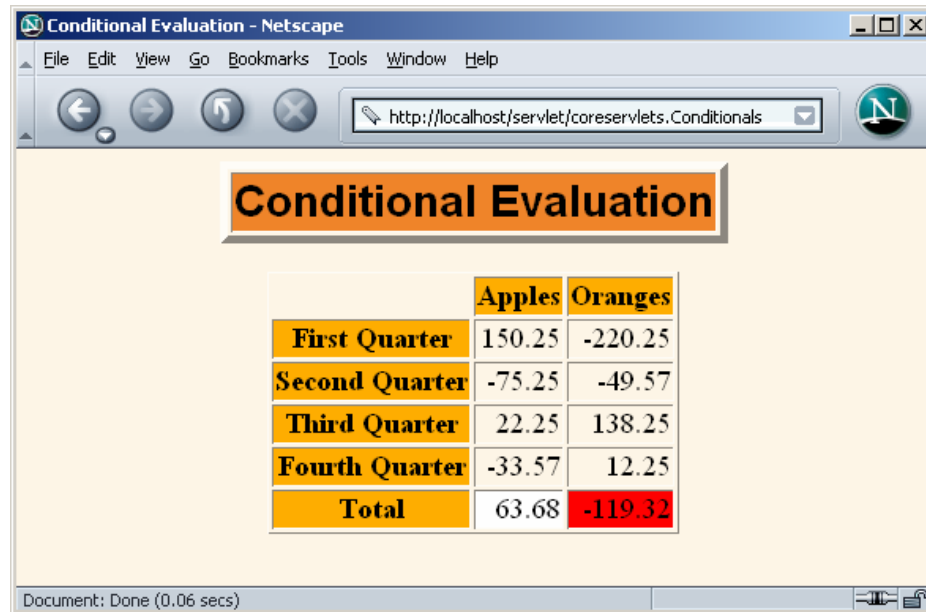
示例：条件求值（续）

```
...
<TABLE BORDER=1 ALIGN="CENTER">
  <TR><TH>
    <TH CLASS="COLORED">Apples
    <TH CLASS="COLORED">Oranges
  <TR><TH CLASS="COLORED">First Quarter
    <TD ALIGN="RIGHT">${apples.q1}
    <TD ALIGN="RIGHT">${oranges.q1}
  <TR><TH CLASS="COLORED">Second Quarter
    <TD ALIGN="RIGHT">${apples.q2}
    <TD ALIGN="RIGHT">${oranges.q2}
  ...
  <TR><TH CLASS="COLORED">Total
    <TD ALIGN="RIGHT"
      BGCOLOR="${(apples.total < 0) ? "RED" : "WHITE" }">
      ${apples.total}
    <TD ALIGN="RIGHT"
      BGCOLOR="${(oranges.total < 0) ? "RED" : "WHITE" }">
      ${oranges.total}
  </TABLE>...
```

41

JSP/servlet/Struts/JSF training: <http://www.coreservlets.com>

示例：条件求值（结果）



Conditional Evaluation

	Apples	Oranges
First Quarter	150.25	-220.25
Second Quarter	-75.25	-49.57
Third Quarter	22.25	138.25
Fourth Quarter	-33.57	12.25
Total	63.68	-119.32

42

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>

小结

- 使用JSP 2.0 EL可以简洁、清晰易读地访问：
 - Bean的属性
 - 集合元素
 - 标准的HTTP元素，如请求参数、请求报头和cookie。
- JSP 2.0 EL和MVC配合的最好
 - 只用来输出由单独的Java代码所创建的值。
- 切不可将EL用在商业（业务）逻辑中

43

JSP/servlet/Struts/JSP training: <http://www.coreservlets.com>



问题?

JSP, Servlet, & Struts Training Courses: <http://courses.coreservlets.com>
Available in US, China, Taiwan, HK, and Worldwide

JSP and Servlet Books from Sun Press: <http://www.coreservlets.com>
*Available in English, Chinese (simplified and traditional script),
and 12 other languages*