Simplifying Access to Java Code: The JSP 2.0 Expression Language

Marty Hall
hall@coreservlets.com
http://www.coreservlets.com/

2

For live Struts training, please see JSP/servlet/Struts/JSF training courses at http://courses.coreservlets.com/.

Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.

3

Agenda

- **Motivating use of the expression language**
- **Understanding the basic syntax**
- **Understanding the relationship of the expression language to the MVC architecture**
- **Referencing scoped variables**
- **Accessing bean properties, array elements, List elements, and Map entries**
- **Using expression language operators**
- **Evaluating expressions conditionally**
- **Using the expression language with Struts**

Drawback of MVC

- **Main drawback is the final step: presenting the results in the JSP page.**
  - jsp:useBean and jsp:getProperty
    - Clumsy and verbose
    - Cannot access bean subproperties
  - Struts bean:write tag
    - Cannot access bean subproperties
    - Still a little bit verbose
  - JSP scripting elements
    - Result in hard-to-maintain code
    - Defeat the whole purpose behind MVC.
- **Goal**
  - More concise access
  - Ability to access subproperties
  - Simple syntax accessible to Web developers

Advantages of the Expression Language

- **Concise access to stored objects.**
  - To output a "scoped variable" (object stored with setAttribute in the PageContext, HttpServletRequest, HttpSession, or ServletContext) named saleItem, you use ${saleItem}.
- **Shorthand notation for bean properties.**
  - To output the companyName property (i.e., result of the getCompanyName method) of a scoped variable named company, you use ${company.companyName}. To access the firstName property of the president property of a scoped variable named company, you use ${company.president.firstName}.
- **Simple access to collection elements.**
  - To access an element of an array, List, or Map, you use ${variable[indexOrKey]}. Provided that the index or key is in a form that is legal for Java variable names, the dot notation for beans is interchangeable with the bracket notation for collections.

Advantages of the Expression Language (Continued)

- **Succinct access to request parameters, cookies, and other request data.**
  - To access the standard types of request data, you can use one of several predefined implicit objects.
- **A small but useful set of simple operators.**
  - To manipulate objects within EL expressions, you can use any of several arithmetic, relational, logical, or empty-testing operators.
- **Conditional output.**
  - To choose among output options, you do not have to resort to Java scripting elements. Instead, you can use ${test ? option1 : option2}.
- **Automatic type conversion.**
  - The expression language removes the need for most typecasts and for much of the code that parses strings as numbers.
- **Empty values instead of error messages.**
  - In most cases, missing values or NullPointerExceptions result in empty strings, not thrown exceptions.

Activating the Expression Language

- **Available only in servers that support JSP 2.0 (servlets 2.4)**
  - E.g., Tomcat 5, not Tomcat 4
- **You must use the JSP 2.0 web.xml file**
  - Download a template from the source code archive at coreservlets.com, or modify the version in the Tomcat 5 jsp-examples Web app (*not* the ROOT Web app).

```
<?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>
```

Invoking the Expression Language

- **Basic form: ${expression}**
  - These EL elements can appear in ordinary text or in JSP tag attributes, provided that those attributes permit regular JSP expressions. For example:
    - <UL>
    - <LI>Name: ${expression1}
    - <LI>Address: ${expression2}
    - </UL>
    - <jsp:include page="${expression3}" />
- **The EL in tag attributes**
  - You can use multiple expressions (possibly intermixed with static text) and the results are coerced to strings and concatenated. For example:
    - <jsp:include page="${expr1}blah${expr2}" />

Escaping Special Characters

- **To get ${ in the page output**
  - Use \${ in the JSP page.
- **To get a single quote within an EL expression**
  - Use \'
- **To get a double quote within an EL expression**
  - Use \"

Preventing Expression Language Evaluation

- **What if JSP 1.2 page contains ${ ?**
- **Deactivating the expression language in an entire Web application.**
  - Use a web.xml file that refers to servlets 2.3 (JSP 1.2) or earlier.
- **Deactivating the expression language in multiple JSP pages.**
  - Use the jsp-property-group web.xml element
- **Deactivating the expression language in individual JSP pages.**
  - Use <%@ page isELEnabled="false" %>
- **Deactivating individual EL statements.**
  - In JSP 1.2 pages that need to be ported unmodified across multiple JSP versions (with no web.xml changes), you can replace $ with $, the HTML character entity for $.
  - In JSP 2.0 pages that contain both expression language statements and literal ${ strings, you can use \${ when you want ${ in the output.

Preventing Use of Standard Scripting Elements

- **To enforce EL-only with no scripting, use scripting-invalid in web.xml**

```
<?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>
```

Accessing Scoped Variables

- **${varName}**
  - Means to search the PageContext, the HttpServletRequest, the HttpSession, and the ServletContext, *in that order*, and output the object with that attribute name.
  - PageContext does not apply with MVC.
- **Equivalent forms**
  - ${name}
  - <%= pageContext.findAttribute("name") %>
  - <jsp:useBean id="name"
                type="somePackage.SomeClass"
                scope="...">
  - <%= name %>

Example: Accessing Scoped Variables

```
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);
  }
}
```

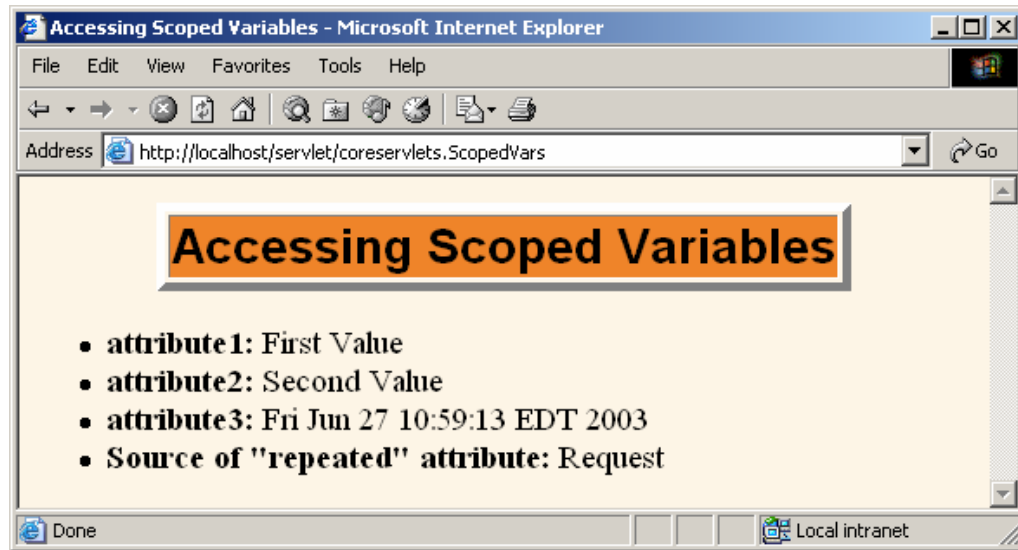# Example: Accessing Scoped Variables (Continued)

```
<!DOCTYPE …>
…
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
  Accessing Scoped Variables
</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>
```

Example: Accessing Scoped Variables (Result)

Accessing Scoped Variables - Microsoft Internet Explorer

File   Edit   View   Favorites   Tools   Help

Address http://localhost/servlet/coreservlets.ScopedVars

Accessing Scoped Variables

- **attribute1**: First Value
- **attribute2**: Second Value
- **attribute3**: Fri Jun 27 10:59:13 EDT 2003
- **Source of "repeated" attribute**: Request

Done   Local intranet

Accessing Bean Properties

- **${varName.propertyName}**
  - Means to find scoped variable of given name and output the specified bean property
- **Equivalent forms**
  - ${customer.firstName}

  - <%@ page import="coreservlets.NameBean" %>
    <%
    NameBean person =
        (NameBean)pageContext.findAttribute("customer");
    %>
    <%= person.getFirstName() %>

Accessing Bean Properties (Continued)

- **Equivalent forms**
  - ${customer.firstName}

  - <jsp:useBean id="customer"
                  type="coreservlets.NameBean"
                  scope="*request, session, or application*" />
    <jsp:getProperty name="customer"
                  property="firstName" />
- **This is better than script on previous slide.**
  - But, fails for subproperties.
  - No non-Java equivalent to
    - ${customer.address.zipCode}

Equivalence of Dot and Array Notations

- **Equivalent forms**
  - ${name.property}
  - ${name["property"]}
- **Reasons for using array notation**
  - To access arrays, lists, and other collections
    - See upcoming slides
  - To calculate the property name at request time.
    - {name1[name2]}   (no quotes around name2)
  - To use names that are illegal as Java variable names
    - {foo["bar-baz"]}
    - {foo["bar.baz"]}

Example: Accessing Bean Properties

```java
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);
  }
}
```

Example: Accessing Bean Properties (Continued)

```java
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;
  }
}
```

Example: Accessing Bean Properties (Continued)

```java
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;
  }
  …
}
```

Example: Accessing Bean Properties (Continued)

```java
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;
  }
}
```

Example: Accessing Bean Properties (Continued)

```
<!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>
```

Example: Accessing Bean Properties (Result)
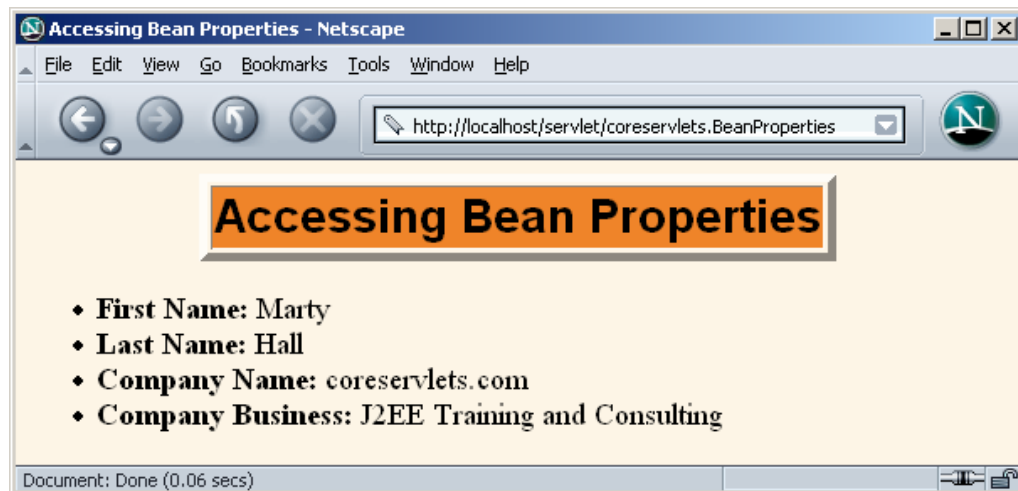
Accessing Collections

- **${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}

Example: Accessing Collections

```
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);
  }
}
```

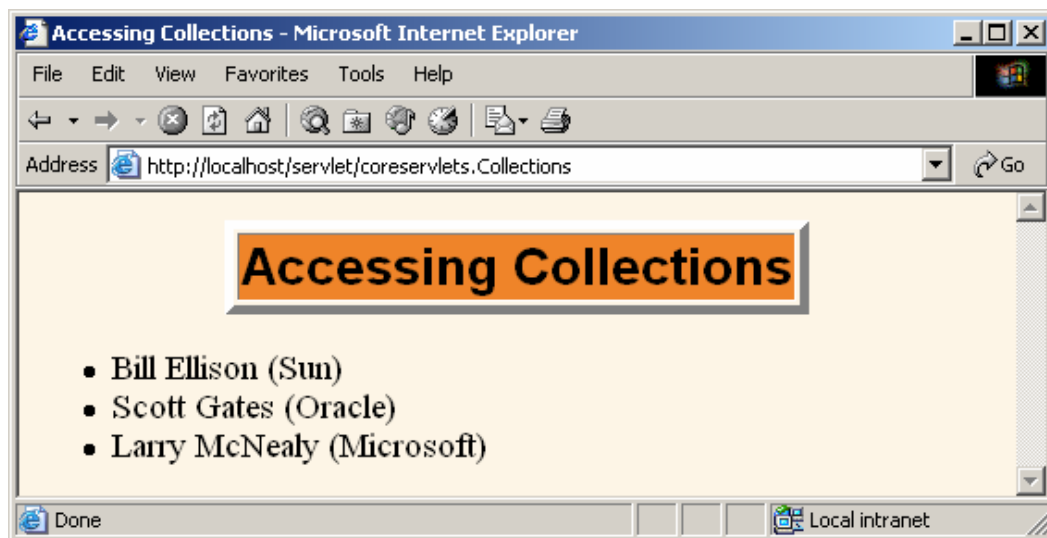# Example: Accessing Collections (Continued)

```
<!DOCTYPE …>
…
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
  Accessing Collections
</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>
```

Example: Accessing Collections (Result)

Referencing Implicit Objects (Predefined Variable Names)

- **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

Example: Implicit Objects

```
<!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>
```

Example: Implicit Objects (Result)
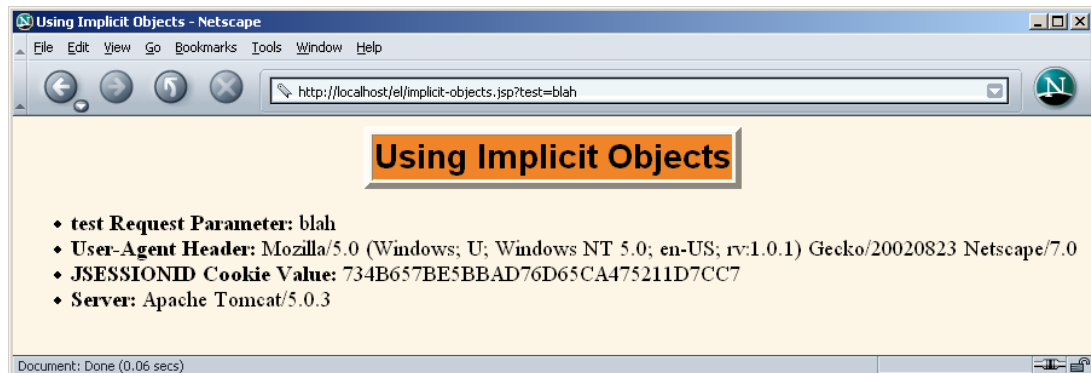
Expression Language Operators

- **Arithmetic**
  - + - * / div % mod
- **Relational**
  - == eq != ne < lt > gt <= le >= ge
- **Logical**
  - && and || or ! Not
- **Empty**
  - Empty
  - True for null, empty string, empty array, empty list, empty map. False otherwise.
- **CAUTION**
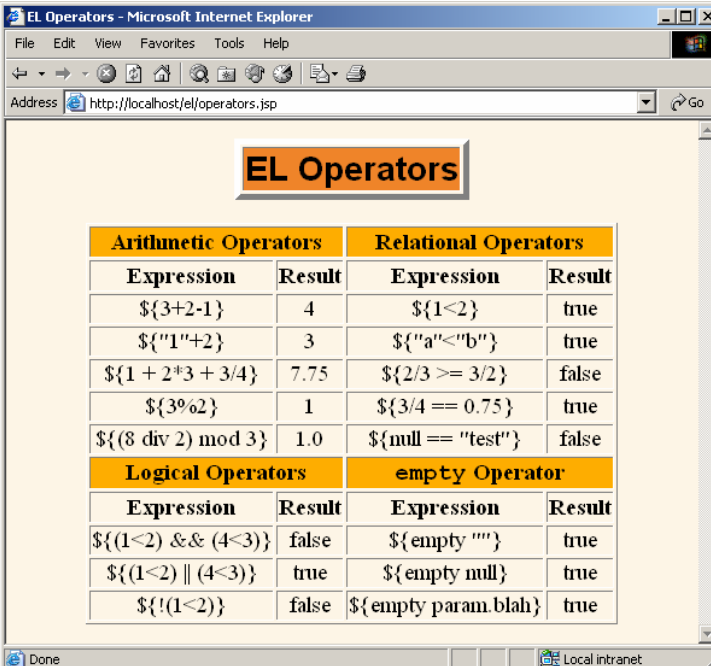 - Use extremely sparingly to preserve MVC model

Example: Operators

```
…
<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>${3+2-1}
    <TD>\${1&lt;2}<TD>${1<2}
  <TR ALIGN="CENTER">
    <TD>\${"1"+2}<TD>${"1"+2}
    <TD>\${"a"&lt;"b"}<TD>${"a"<"b"}
  <TR ALIGN="CENTER">
    <TD>\${1 + 2*3 + 3/4}<TD>${1 + 2*3 + 3/4}
    <TD>\${2/3 &gt;= 3/2}<TD>${2/3 >= 3/2}
  <TR ALIGN="CENTER">
    <TD>\${3%2}<TD>${3%2}
    <TD>\${3/4 == 0.75}<TD>${3/4 == 0.75}
…
```

Example: Operators (Result)

Evaluating Expressions Conditionally

- **${ test ? expression1 : expression2 }**
  - Evaluates test and outputs either expression1 or expression2
- **Problems**
  - Relatively weak
    - c:if and c:choose from JSTL are much better
  - Tempts you to put business/processing logic in JSP page.
  - Should only be used for presentation logic.
    - Even then, consider alternatives

Example: Conditional Expressions

```
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);
  }
}
```

Example: Conditional Expressions (Continued)

```
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); }
}
```

Example: Conditional Expressions (Continued)

```
…
<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>…
```
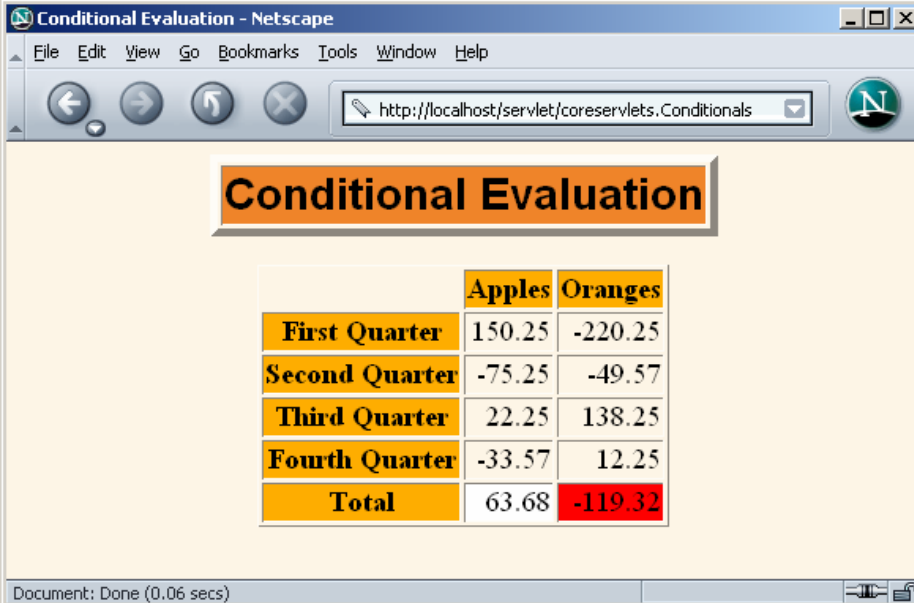
Example: Conditional Expressions (Result)

Apache Struts: Typical Processing Flow

1. **HTML form uses html:form and html:text to create a form that is associated with a bean**
2. **Form submits data to a URL of the form** *blah*.**do**
3. **That address is mapped by struts-config.xml to an Action object, whose execute method handles the request.**
4. **The execute method is automatically given a "form bean" corresponding to request parameters, but can create other results beans and store them in request, session, or application scope.**

Apache Struts: Typical Processing Flow (Continued)

5. **The execute method uses mapping.findForward to return conditions.**
6. **The struts-config.xml file maps those conditions to JSP pages to be displayed.**
7. **The JSP pages use bean:write to output the properties of the bean.**
   - bean:write is more concise than jsp:useBean and jsp:getProperty, but more verbose than JSP 2.0 expression language
   - bean:write cannot access bean subproperties
   - So, replace step 7 with the JSP 2.0 EL.
     - Note that bean:write automatically filters HTML characters, but EL does not

Struts Example: Form Bean

```
package coreservlets;
import org.apache.struts.action.*;

public class ContactFormBean extends ActionForm {
  private String firstName = "First name";
  private String lastName = "Last name";
  private String email = "user@host";
  private String faxNumber = "xxx-yyy-zzzz";
  private String warning = "";

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

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

Struts Example: Value Bean

```java
package coreservlets;

public class MessageBean {
  private String message = "";

  public String getMessage() {
    return(message);
  }

  public void setMessage(String message) {
    this.message = message;
  }
}
```

Struts Example:
struts-config.xml

```xml
…
<struts-config>
  <form-beans>
    <form-bean name="contactFormBean"
               type="coreservlets.ContactFormBean"/>
  </form-beans>
  <action-mappings>
    <action path="/actions/signup2"
            type="coreservlets.SignupAction2"
            name="contactFormBean"
            scope="session"
            input="/forms/signup2.jsp">
      <forward name="missing-value"
               path="/forms/signup2.jsp"
               redirect="true"/>
      <forward name="success"
               path="/WEB-INF/results/confirmation.jsp"/>
    </action>
  </action-mappings>
</struts-config>
```

# Struts Example: Confirmation Page (Classic Style)

```
…
Congratulations. You are now signed up for the
Single Provider of Alert Memos network!
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<UL>
  <LI>First name:
  <bean:write name="contactFormBean" property="firstName"/>
  <LI>Last name:
  <bean:write name="contactFormBean" property="lastName"/>
  <LI>Email address:
  <bean:write name="contactFormBean" property="email"/>
  <LI>Fax number:
  <bean:write name="contactFormBean" property="faxNumber"/>
</UL>
…
```

Struts Example: Confirmation Page (JSP 2.0 Style)

```
…
Congratulations. You are now signed up for the
Single Provider of Alert Memos network!
<UL>
  <LI>First name: ${contactFormBean.firstName}
  <LI>Last name: ${contactFormBean.lastName}
  <LI>Email address: ${contactFormBean.email}
  <LI>Fax number: ${contactFormBean.faxNumber}
</UL>
…
```

Struts Example: Results
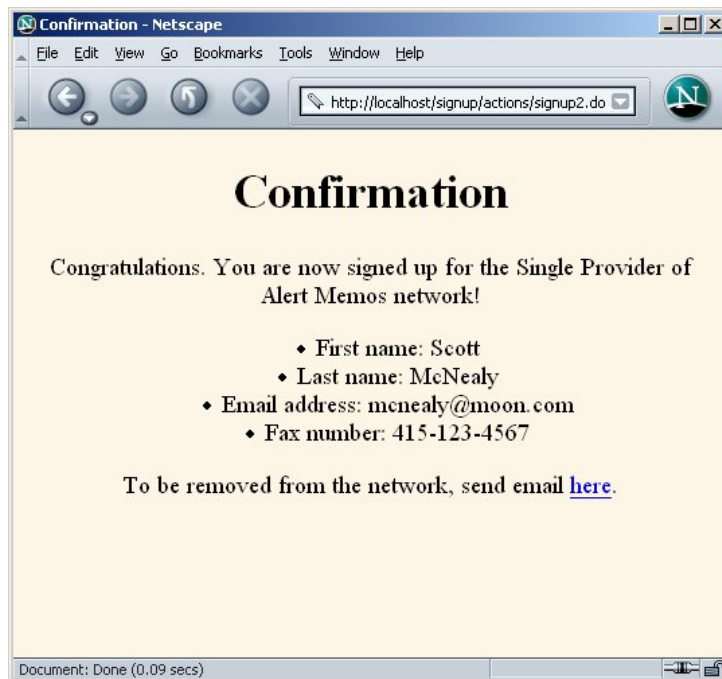
Summary

- **The JSP 2.0 EL provides concise, easy-to-read access to**
  – Bean properties
  – Collection elements
  – Standard HTTP elements such as request parameters, request headers, and cookies
- **The JSP 2.0 EL works best with MVC**
  – Use only to output values created by separate Java code
  – Resist use of EL for business logic
- **The JSP 2.0 EL fits well with Apache Struts**
  – More powerful and concise replacement for bean:write

Questions?