


© 2003-2004 Marty Hall



Simplifying Access to Java Code: The JSP 2.0 Expression Language

JSP and Servlet Training Courses: <http://courses.coreservlets.com>
 JSP and Servlet Books from Sun Press: <http://www.coreservlets.com>

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

JSP/Servlet training: <http://www.coreservlets.com>

Servlets and JSP: Possibilities for Handling a Single Request

- **Servlet only. Works well when:**
 - Output is a binary type. E.g.: an image
 - There is *no* output. E.g.: you are doing forwarding or redirection as in Search Engine example.
 - Format/layout of page is highly variable. E.g.: portal.
- **JSP only. Works well when:**
 - Output is mostly character data. E.g.: HTML
 - Format/layout mostly fixed.
- **Combination (MVC architecture). Needed when:**
 - A single request will result in multiple substantially different-looking results.
 - You have a large development team with different team members doing the Web development and the business logic.
 - You perform complicated data processing, but have a relatively fixed layout.

JSP/Servlet training: <http://www.coreservlets.com>

Implementing MVC with RequestDispatcher

- **Define beans to represent the data**
- **Use a servlet to handle requests**
 - Servlet reads request parameters, checks for missing and malformed data, etc.
- **Populate the beans**
 - The servlet invokes business logic (application-specific code) or data-access code to obtain the results. Results are placed in the beans that were defined in step 1.
- **Store the bean in the request, session, or servlet context**
 - The servlet calls setAttribute on the request, session, or servlet context objects to store a reference to the beans that represent the results of the request.

JSP/Servlet training: <http://www.coreservlets.com>

Implementing MVC with RequestDispatcher (Continued)

- **Forward the request to a JSP page.**
 - The servlet determines which JSP page is appropriate to the situation and uses the forward method of RequestDispatcher to transfer control to that page.
- **Extract the data from the beans.**
 - The JSP page accesses beans with jsp:useBean and a scope matching the location of step 4. The page then uses jsp:getProperty to output the bean properties.
 - The JSP page does not create or modify the bean; it merely extracts and displays data that the servlet created.

JSP/Servlet training: <http://www.coreservlets.com>

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
 - 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

JSP/Servlet training: <http://www.coreservlets.com>

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.

JSP/Servlet training: <http://www.coreservlets.com>

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.

JSP/Servlet training: <http://www.coreservlets.com>

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>
```

JSP/Servlet training: <http://www.coreservlets.com>

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:
 - ``
 - `Name: ${expression1}`
 - `Address: ${expression2}`
 - ``
 - `<jsp:include page="${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}" />`

JSP/Servlet training: <http://www.coreservlets.com>

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 `\"`

JSP/Servlet training: <http://www.coreservlets.com>

Preventing Expression Language Evaluation

- **What if JSP 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.

JSP/Servlet training: <http://www.coreservlets.com>

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>
```

13

ISP/Servlet training: <http://www.coreservlets.com>

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 %>`

14

ISP/Servlet training: <http://www.coreservlets.com>

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);
  }
}
```

15

ISP/Servlet training: <http://www.coreservlets.com>

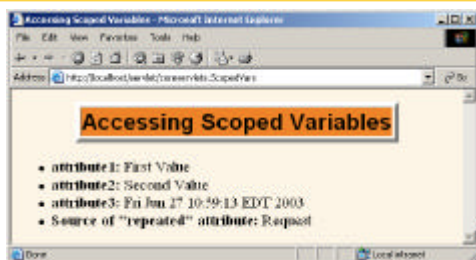
Example: Accessing Scoped Variables (Continued)

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

16

ISP/Servlet training: <http://www.coreservlets.com>

Example: Accessing Scoped Variables (Result)



17

ISP/Servlet training: <http://www.coreservlets.com>

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() %>`

18

ISP/Servlet training: <http://www.coreservlets.com>

Accessing Bean Properties (Continued)

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

19

ISP/Servlet training: <http://www.coreservlets.com>

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"]}`

20

ISP/Servlet training: <http://www.coreservlets.com>

Example: Accessing Bean Properties

```
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",
                "JZEE 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);
    }
}
```

21

ISP/Servlet training: <http://www.coreservlets.com>

Example: Accessing Bean Properties (Continued)

```
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;
    }
}
```

22

ISP/Servlet training: <http://www.coreservlets.com>

Example: Accessing Bean Properties (Continued)

```
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;
    }
    --
}
```

23

ISP/Servlet training: <http://www.coreservlets.com>

Example: Accessing Bean Properties (Continued)

```
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;
    }
}
```

24

ISP/Servlet training: <http://www.coreservlets.com>

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>

```

ISP/Servlet training: <http://www.coreservlets.com>

Example: Accessing Bean Properties (Result)



ISP/Servlet training: <http://www.coreservlets.com>

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}`

ISP/Servlet training: <http://www.coreservlets.com>

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);
  }
}

```

ISP/Servlet training: <http://www.coreservlets.com>

Example: Accessing Collections (Continued)

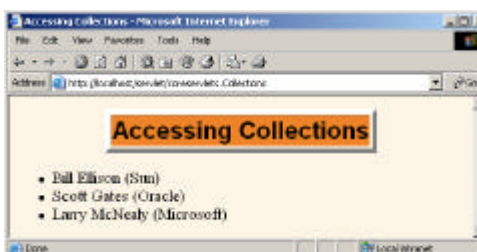
```

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

```

ISP/Servlet training: <http://www.coreservlets.com>

Example: Accessing Collections (Result)



ISP/Servlet training: <http://www.coreservlets.com>

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

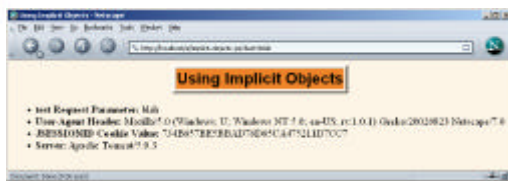
ISP/Servlet training: <http://www.coreservlets.com>

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>
```

ISP/Servlet training: <http://www.coreservlets.com>

Example: Implicit Objects (Result)



ISP/Servlet training: <http://www.coreservlets.com>

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

ISP/Servlet training: <http://www.coreservlets.com>

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>`${1<2}`<TD>`${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}`
...

```

ISP/Servlet training: <http://www.coreservlets.com>

Example: Operators (Result)

The screenshot shows a web browser window with the title "EL Operators". The page content includes a table with three sections: Arithmetic Operators, Relational Operators, and Logical Operators. The table shows expressions and their results.

Arithmetic Operators		Relational Operators	
Expression	Result	Expression	Result
<code>`\${3+2-1}`</code>	4	<code>`\${1<2}`</code>	true
<code>`\${"1"+2}`</code>	3	<code>`\${"a"<"b"}`</code>	true
<code>`\${1 + 2*3 + 3/4}`</code>	7.75	<code>`\${2/3 == 3/2}`</code>	false
<code>`\${3%2}`</code>	1	<code>`\${3/4 == 0.75}`</code>	true
<code>`\${(5 div 2) mod 3}`</code>	1	<code>`\${!true}`</code>	false
Logical Operators		Empty Operator	
Expression	Result	Expression	Result
<code>`\${(1-2)&&(4-3)}`</code>	false	<code>`\${empty ""}`</code>	true
<code>`\${(1-2) (4-3)}`</code>	true	<code>`\${empty null}`</code>	true
<code>`\${!(1-2)}`</code>	true	<code>`\${empty paramMap}`</code>	true

ISP/Servlet training: <http://www.coreservlets.com>

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

37 JSP/Servlet training: <http://www.coreservlets.com>

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);
    }
}
```

38 JSP/Servlet training: <http://www.coreservlets.com>

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); }
}
```

39 JSP/Servlet training: <http://www.coreservlets.com>

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>...
```

40 JSP/Servlet training: <http://www.coreservlets.com>

Example: Conditional Expressions (Result)

	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	-219.32

41 JSP/Servlet training: <http://www.coreservlets.com>

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

42 JSP/Servlet training: <http://www.coreservlets.com>



Questions?

JSP and Servlet Training Courses: <http://courses.coreservlets.com>
JSP and Servlet Books from Sun Press: <http://www.coreservlets.com>