

© 2003 Marty Hall



Simplifying Access to Servlet Code: 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 Expression Language (EL) Use
- Invoking the Expression Language
- Disabling the Expression Language
- Preventing Use of Classic Scripting Elmts
- Understanding the EL and MVC Architecture
- Referencing Scoped Variables
- Accessing Bean Properties, Array Elements, List Elements and Map Entries
- Using Expression Language Operators
- Conditionally Evaluating EL Expressions

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

Uses of JSP Constructs

Simple Application

- Scripting elements calling servlet code directly
- Scripting elements calling servlet code indirectly (by means of utility classes)
- Beans
- Servlet/JSP combo (MVC)
- MVC with JSP expression language
- Custom tags

Complex Application

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

Servlet-JSP Releases

- **Servlets 2.2 – JSP 1.1**
 - Tomcat 3.x
 - 1st Edition of Core Servlets and JavaServer Pages
- **Servlets 2.3 – JSP 1.2**
 - Tomcat 4.x
 - `Projects1stEd.zip` by J. Blessing
- **Servlets 2.4 – JSP 2.0**
 - Tomcat 5.x
 - JSP 2.0 = JSP 1.2 + JSTL 1.0
 - 2nd Edition of Core Servlets and JavaServer Pages
 - `Projects2ndEd.zip` by J. Blessing

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

EL Motivation

- **Lots of success with Velocity**
 - <http://jakarta.apache.org/velocity>
- **Simplifies access to Servlet objects**
 - JavaBeans, properties, collections, parameters, cookies
 - Simple operators (+, -, *, /, etc.)
 - Conditional output (?:)
 - Automatic type conversion
 - Removes most need of type casts and code to parse strings as numbers
 - Empty strings returned instead of errors
 - "" returned instead of null or exception being thrown
 - Much shorter symbols than classic JSP tags

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

Invoking the EL

`#{expression}`

- **For Example:**

```
<UL>
  <LI>Name: ${cust.name}
  <LI>Address: ${cust.addr}
</UL>
<jsp:include page="${expr}"/>
```
- **Instead of:**

```
<jsp:usebean id="cust" class="package.Customer"/>
<UL>
  <LI>Name: <jsp:getProperty name="cust" property="name"/>
  <LI>Address: <jsp:getProperty name="cust" property="addr"/>
</UL>
<jsp:include page="<%= expr %>"/>
```

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

Disabling EL Evaluation

- **If JSP2.0 is supported, disable EL:**
 - For entire webapp by using the web.xml changes
 - Across multiple JSP pages by specifying the proper `jsp-property-group` tag in web.xml
 - In individual JSP pages by using the page directive's `isELEnabled` attribute
 - In individual expressions by replacing:
 - \$ with `$`; char entity in JSP 1.2 and earlier
 - \${ with `\${` in JSP 2.0 or later

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

Enabling EL For Entire WebApp

- **Replace:**

```
<!DOCTYPE web-app
PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
...
</web-app>
```
- **in web.xml with:**

```
<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>

Disabling EL across multiple JSPs

- **in web.xml write:**

```
<?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>/legacy/*.jsp</url-pattern>
    <el-ignored>true</el-ignored>
  </jsp-property-group>

</web-app>
```

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

Disabling EL in some JSPs

- **Use the following page attribute:**

```
<% @ page isELEnabled="false" %>
```
- **This is a new page attribute for JSP 2.0**

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

Disabling Single EL expressions

- **In JSP 1.2 and prior servers**
 - Replace \$ with `$`; HTML character entity
- **In JSP 2.0 and later servers**
 - Replace \${ with `\${`

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

Disable Classic JSP Scripting

- **in web.xml write:**

```
<?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>
```

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

Accessing Scoped Variables

- `<jsp:useBean .../>` associates a scope with each object
- `${expression}` will search for an object named *expression* in the following order:
 - PageContext
 - HttpServletRequest
 - HttpSession
 - ServletContext
- If you never reuse a name in a different scope (which is recommended) then this will work for all your old webapps!

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

Example 16.5

- **ScopedVars.java**
 - The servlet that stores data in each scope
- **scoped-vars.jsp**
 - The JSP 2.0 page that extracts the data values
- **Run example 16.5.ScopeExample**

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

Accessing Bean Properties

- “Dot” notation is very powerful in the EL
 - `${customer.firstName}` instead of:

```
<% page import="coreservlets.NameBean" %>
<% NameBean person = (NameBean)
    pageContext.findAttribute("customer"); %>
<%= person.getFirstName() %>
```
- Properties can even have properties
 - `${customer.address.city}`
 - There is no direct way to get this nested property using the JSP scripting elements!

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

Example 16.6

- **BeanProperties.java**
 - Sets up a NameBean, EmployeeBean and CompanyBean
- **bean-properties.jsp**
 - Contains JSP 2.0 EL expressions to extract the nested property values
- **Run example 16.6.BeanProps**

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

Equivalence of Dot Notation and Array Access Notation

- One can replace:
`${name.property}`
- with:
`${name["property"]}`
- Array form is typically used only for collections (more to come on this)
 - Array form is actually more flexible
 - The property string can be computed at request time!
 - Dot notation has to be statically set at page compile time

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

Accessing Collections

- Arrays, Lists and Maps all support [] access
 - Recall that List and Map are interfaces in Java
 - So there are many implementing classes that can be used
 - Arrays are also “special” types of objects in Java
 - They get special treatment due to the [] operator
 - Still, an array is a full-fledged object in Java
- The following are interchangeable
 - `${stateCapital["wisconsin"]}`
 - `${stateCapital.wisconsin}`

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

Example 16.7

- **Collections.java**
 - Populates a `String[]`, an `ArrayList` and a `HashMap`
- **collections.jsp**
 - Extracts data from these containers
 - And messes up the names of company heads
- **Run example 16.7.Collections**

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

Implicit Objects

- **The EL provides several predefined objects:**
 - `pageContext`
 - `param` and `paramValues`
 - `header` and `headerValues`
 - `cookie`
 - `initParam`
 - `pageScope`, `requestScope`, `sessionScope` and `applicationScope`

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

pageContext

- Refers to the `PageContext` of the current `JavaServer page`
- Has `request`, `response`, `session`, `out` and `servletContext` properties
- Thus `PageContext` has corresponding "getter" methods
 - `getRequest()`, `getResponse()`, `getSession()`, `getOut()` and `getServletContext()`
- Just use EL dot notation on properties
 - `${pageContext.session.id}`

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

param and paramValues

- These objects allow access to the individual request parameters (`param`) or the array of parameter values (`paramValues`)
- For instance, to get the value of the debug request parameter, use
 - `${param.debug}`

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

header and headerValues

- These objects allow access to the individual request headers (`header`) or the array of header values (`headerValues`)
- For instance, to get the value of the `Accept-Encoding` request header, use
 - `${header["Accept-Encoding"]}`

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

cookie

- Allows access to named cookies
- However, the return value is the `Cookie` object, not the cookie value
 - To get the value, just use the `.value` property
 - For example:
 - `${cookie.userID.value}`
 - `${cookie["userID"].value}`
 - Both return the value of the `userID` cookie (or `""`)

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

initParam

- **We're getting repetitive here**
 - To get the value of the defaultColor init parameter
 - `${initParam.defaultColor}`

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

Selecting a Specific Scope

- **What if the same name is used in more than one scope?**
- **If you know the scope then use the appropriate predefined object**
- **Instead of using**
`${name}`
- USE**
 - `${requestScope.name}`
 - `${sessionScope.name}`
 - `${applicationScope.name}`

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

Example 16.8

- **implicit-objects.jsp**
 - Accesses some of the HTTP info available in JSPs
- **Run example 16.8.ImplicitObjs**

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

EL Operators

- **Arithmetic**
 - +, -, *, /
 - div, mod
- **Relational**
 - <, <=, >, >=, ==, !=
 - lt, le, gt, ge, eq, ne
- **Logical**
 - &&, ||, !
 - and, or, not
 - empty
 - true if argument is null or an empty string, array, List, Map, or Collection

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

Example 16.9

- **operators.jsp**
 - Builds a table of results using the mentioned operators
- **Run example 16.9.Operators**

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

Conditionally Evaluating EL Expr.

- **Uses the ternary (teritary?) operator**
 - `${test ? expr1 : expr2}`
 - Returns the result of evaluating expr1 if test returns true or the result of evaluating expr2 if test returns false
- **Run example 16.10.Conditionals**

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

What about iteration?

- **Check the JSTL 1.0 link on our course web page for more**
 - The JSTL “core” package article
 - Iteration
 - Internationalization and Localization
 - (I18N and L10N)
 - XML parsing and XSL translation
 - SQL database access
 - more

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

Summary

- **The EL greatly simplifies JSP syntax**
- **Use of the EL is optional**
 - Classic JSP syntax will always be available
- **Use of the EL requires a servlet container that supports Servlets 2.4 spec. or higher**
 - But that’s easily controlled on the server-side of things
- **Velocity and PHP have had their impact on the J2EE**
 - As have many other popular web technologies
 - JDK 1.5 (Tiger) is heavily influenced by C# and .NET (Microsoft)

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



© 2003 Marty Hall

Questions?

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