

# What's New in the Servlet and JSP Specifications

---

Bryan Basham  
Sun Microsystems, Inc  
[bryan.basham@sun.com](mailto:bryan.basham@sun.com)



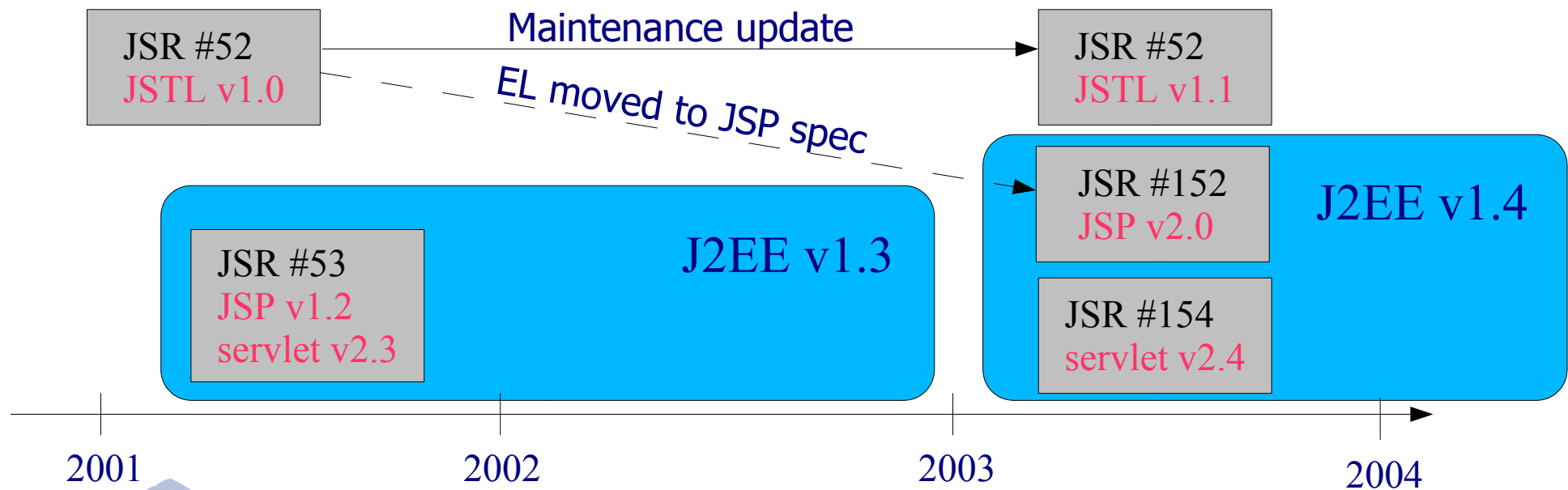
# Topics Covered

---

- **History**
- Servlet Spec Changes
- JSP Spec Changes: General
- JSP Spec Changes: Expression Language
- JSP Spec Changes: Tag Alternatives
- JSP Standard Tag Library

# History

- The Web component specifications have been managed by the JCP for two generations.
- The EL was developed in JSTL, but moved to the JSP v2.0 specification.





# Topics Covered

---

- History
- Servlet Spec Changes
- JSP Spec Changes: General
- JSP Spec Changes: Expression Language
- JSP Spec Changes: Tag Alternatives
- JSP Standard Tag Library Changes



# Overview of Servlet Changes

---

- Servlet spec is mature and stable
- Mostly refining the existing servlet container model
- Major changes:
  - Web event model
  - Filters and Request Dispatchers
  - SingleThreadModel deprecated
  - Deployment Descriptor



# Web Event Model

---

- Application scale:

- The `ServletContextListener` interface
- The `ServletContextAttributeListener` interface

- Session scale:

- The `HttpSessionListener` interface
- The `HttpSessionAttributeListener` interface

- Request scale:

- The `ServletRequestListener` interface
- The `ServletRequestAttributeListener` interface

# Filters and Request Dispatch

- In servlet v2.3 spec, filters were not applied to calls to a RequestDispatcher
- Now, you can configure a filter for use by RDs:

```
<filter-mapping>  
  <filter-name>Logging Filter</filter-name>  
  <url-pattern>/products/*</url-pattern>  
  <dispatcher>FORWARD</dispatcher>  
  <dispatcher>REQUEST</dispatcher>  
</filter-mapping>
```



# SingleThreadModel

---

- The STM interface has been deprecated.
- This means that **you** must guarantee the thread-safety of your servlets and JSP pages.
- Suggestions:
  - Use the `synchronize` keyword to control access to critical (shared) data
  - Local variables and request attributes are thread-safe; all other attributes should be controlled
  - Use thread-safe, shared components (like ConnPool)



# Deployment Descriptor

---

- The `web.xml` now is defined by an XML Schema.  
*(was a DTD)*
- JSP configuration has been expanded.  
*(discussed later)*
- The DD is extensible to include application-specific XML content.



# Topics Covered

---

- History
- Servlet Spec Changes
- JSP Spec Changes: General
- JSP Spec Changes: Expression Language
- JSP Spec Changes: Tag Alternatives
- JSP Standard Tag Library Changes



# General JSP Changes

---

- Large changes intended for ease of use and greater flexibility of choice (EL, tag alternatives)
- Other significant changes:
  - The `jsp-config` tag structure in the `web.xml`
  - Simplified JSP Document structure

# JSP Configuration

## Example:

Taglib declarations grouped into jsp-config

```
<web-app>
  <jsp-config>
    <taglib>
      <taglib-uri>http://java.sun.com/jsp/jstl/core</taglib-uri>
      <taglib-location>/WEB-INF/tld/c.tld</taglib-location>
    </taglib>
    <taglib>
      <taglib-uri>http://myCorp.com/tags/html</taglib-uri>
      <taglib-location>/WEB-INF/tags/html/h.tld</taglib-location>
    </taglib>
    <jsp-property-group>
      <display-name>JSP Configuration</display-name>
      <url-pattern>*.jsp</url-pattern>
      <el-ignored>>false</el-ignored>
      <scripting-invalid>>false</scripting-invalid>
    </jsp-property-group>
  </jsp-config>
</web-app>
```

Property groups define translation configuration

# JSP Documents

- For generating XML content
- Easier than in past specifications:

```
<table size='${3}'>
  <c:forEach
    xmlns:c="http://java.sun.com/jsp/jstl/core"
    var="counter" begin="1" end="${3}">
    <row>${counter}</row>
  </c:forEach>
</table>
```

Taglib namespace

EL syntax simplifies XML coding



# Topics Covered

---

- History
- Servlet Spec Changes
- JSP Spec Changes: General
- JSP Spec Changes: Expression Language
- JSP Spec Changes: Tag Alternatives
- JSP Standard Tag Library Changes



# Refactoring Example

---

## ■ Before EL:

```
<jsp:useBean id="customer" scope="session" type="com.my.Customer" />
```

Welcome, `<jsp:getProperty name="customer" property="firstName" />`, to our web site...

## ■ Using EL:

Welcome, `${sessionScope.customer.firstName}`, to our web site...

## ■ Or simply:

Welcome, `${customer.firstName}`, to our web site...

# Definition of the EL

- Syntax: `${ expr1 }`
- Similar to: `<%= expr2 %>`
- But, the syntax of `expr2` is Java
- Whereas, the syntax of `expr1` is EL (not Java)

# Definition of the EL *(Continued)*

- Expressions can be used in:
  - Template text: Welcome, `${customer.firstName}`...
  - Tag attributes: `<c:out value="${customer.firstName}"/>`
- Expressions cannot be used when:
  - In non-RT tag attributes, such as `jsp:useBean/id`
  - In the body of a tag with `tagdependent` body type
  - EL processing has been disabled
    - In deployment descriptor
    - Using `<%@ page isElIgnored="true" %>`



# Implicit Object Examples

---

- Retrieve the request URI:

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

- Retrieve a property of an attribute of the session:

```
${sessionScope.customer.firstName}
```

- Alternate syntax:

```
${sessionScope['customer'].firstName}
```

- Retrieve the `userName` parameter:

```
${param['userName']}
```

- Alternate syntax:

```
${param.userName}
```



# Expressions

---

## ■ Standard binary operations:

- **Arithmetic:** +, -, \*, /, div, %, and mod
- **Relational:** ==, eq, !=, ne, <, lt, <=, le, and so on
- **Logical:** &&, and, ||, or, !, not, and empty

## ■ Examples:

```
<c:if test="{ (b*b-4*a*c) < 0 }">complex</c:if>
```

```
<c:forEach var="product" items="{sessionScope.productList}">
<tr>
  <td>{product.name}</td>
  <td>{product.unitCost * orders[product.ID].quantity}</td>
</tr>
</c:forEach>
```

# EL Function Syntax

- Syntax: Taglib prefix function name

FunctionInvocation ::=

[Identifier ':'] Identifier '(' ParameterList ')'

ParameterList ::= Expression | Expression ',' ParameterList

- Examples:

There are `${fn:length(athletes[country])}` athletes representing `${country}`.

```
<c:url var="myUrl" value="${base}/cust">
  <c:param name="custId" value="${fn:trim(custId)}"/>
</c:url>
```



# Topics Covered

---

- History
- Servlet Spec Changes
- JSP Spec Changes: General
- JSP Spec Changes: Expression Language
- JSP Spec Changes: Tag Alternatives
- JSP Standard Tag Library Changes



# Overview of Tag Alternatives

---

- Classic Tag Handlers
- Simple Tag Handlers
- Tag Files



# Classic Tag Handlers

---

- **Event methods:** `doStartTag`, `doAfterBody`, and `doEndTag`
- **Assumptions:**
  - Scripting elements can appear in the body
- **Uses:**
  - The Classic version is the richest; you can do anything
  - Best for non-presentation actions



# Simple Tag Handlers

---

- Event methods: `doTag`
- Assumptions:
  - Scripting elements **cannot** appear in the body
- Uses:
  - Same as Classic, but easier to program



# Tag Files

---

- Assumptions:
  - Scripting elements **cannot** appear in the body
- Uses:
  - Tag development for non-Java programmers
- Example: (`/WEB-INF/tags/html/head.tag`)

```
<%@ tag body-content="scriptless" %>  
<%@ attribute name="headingLevel" required="true" %>  
<h${headingLevel}><jsp:doBody></h${headingLevel}>
```



# Topics Covered

---

- History
- Servlet Spec Changes
- JSP Spec Changes: General
- JSP Spec Changes: Expression Language
- JSP Spec Changes: Tag Alternatives
- JSP Standard Tag Library Changes

# Alignment with JSP v2.0

- JSTL v1.0 was intended to be used in a JSP v1.2 container.
  - Two versions of each tag library.
  - JSTL contain its own EL parser/evaluator.
- JSTL v1.1 is intended to be used in a JSP v2.0 container.
  - One version of each tag library.
  - JSP Container EL is used.
  - Added functions for EL use.



# JSTL Pre-Defined Functions

---

- Length function: `length`
- String compare: `contains`, `containsIgnore`, `endsWith`, **and** `startsWith`
- String search: `indexOf`
- String modification: `replace`, `substring`, `substringAfter`, `substringBefore`, `toLowerCase`, `toUpperCase`, `trim`, **and** `escapeXml`
- String splitting: `split` **and** `join`



# Topics Covered

---

- History
- Servlet Spec Changes
- JSP Spec Changes: General
- JSP Spec Changes: Expression Language
- JSP Spec Changes: Tag Alternatives
- JSP Standard Tag Library Changes
- Questions?