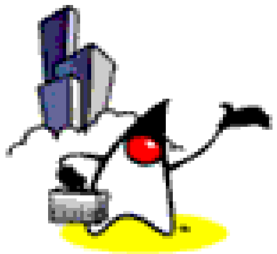
JSP 2.0 (in J2EE 1.4)



Disclaimer & Acknowledgments

- Even though Sang Shin is a full-time employees of Sun Microsystems, the contents here are created as his own personal endeavor and thus does not reflect any official stance of Sun Microsystems.
- Sun Microsystems is not responsible for any inaccuracies in the contents.
- Acknowledgements
  - Many slides are borrowed from “Servlet 2.4 and JSP 2.0 specification” JavaOne 2003 presentation by Mark Roth of Sun Microsystems
  - The slides, speaker notes, and example code of this presentation are created from
    - “Custom Tags” section of Java WSDP 1.2 tutorial written by Stephanie Bodoff of Sun Microsystems

Revision History

- 10/13/2003: version 1: created by Sang Shin
- Things to do
  - Speaker notes still need to be added and polished
  - Some concepts still need to be better explained

Agenda

- Focus of JSP 2.0 technology
- New features in JSP 2.0
  - Expression Language (EL)
  - Simple tag extensions
  - Tag files
  - Improved XML syntax
  - Other features

Focus of JSP 2.0 Technology

- Ease of use – lowers the bar

JSP 1.2

JSP 2.0

User Class	HTML	XML	Java
Tag Library Developer	🧠	🧠	🧠
Advanced Page Author	🧠	🧠	🍼
Basic Page Author	🧠	🍼	—

🍼 = Basic Knowledge 🧠 = Expert

JSP 1.2 Syntax With Scriptlets

```
<%-- Output Shopping Cart --%>
<%@ page import="com.acme.util.*" %>
<%@ taglib prefix="util" uri="http://mytaglib" %>

<html>
   <body>
      <util:getShoppingCart var="cart" />
      <table>
      <%  for( int i = 0; i < cart.size(); i++ ) {
            CartItem item=(CartItem)cart.get(i);
      %>
         <tr>
            <td><%= item.getName() %></td>
            <td><%= item.getPrice() %></td>
         </tr>
      <%  } %>
      </table>
   </body>
</html>
```

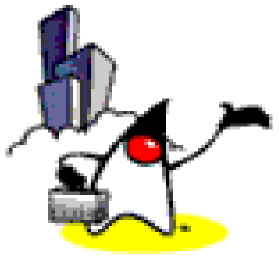
JSP 2.0 Syntax Without Scriptlets

```
<%-- Output Shopping Cart --%>
<%@ taglib prefix="util" uri="http://mytaglib" %>
<%@ taglib prefix="c"
     uri="http://java.sun.com/jsp/jstl/core" %>
<html>
  <body>
    <util:getShoppingCart var="cart" />
    <table>
      <c:forEach var="item" values="${cart}">
        <tr>
          <td>${item.name}</td>
          <td>${item.price}</td>
        </tr>
      </c:forEach>
    </table>
  </body>
</html>
```

JSP 2.0 Improved XML Syntax

```
<!-- Output Shopping Cart -->
<html   xmlns:util="http://mytaglib"
        xmlns:c="http://java.sun.com/jsp/jstl/core">
    <body>
        <util:getShoppingCart var="cart" />
        <table>
            <c:forEach var="item" values="${cart}">
                <tr>
                    <td>${item.name}</td>
                    <td>${item.price}</td>
                </tr>
            </c:forEach>
        </table>
    </body>
</html>
```

Expression Language



Expression Language

- Based on “SPEL” from JSTL 1.0
  - Simplest Possible Expression Language
- Let you access the property values of a JavaBean in a simpler syntax
  - Example: ${item.price}
- Recognized by JSP container in:
  - Template text
  - Attributes of any_standard or custom action
- Support for custom EL functions:
  - Extensible via tag libraries
  - Example: ${fn:allCaps(lastName)}
  - JSTL 1.1 provides 16 standard EL functions

Integrated Expression Language Example

- Using scriptlets:

```
<center>
    <jsp:useBean id="foo" class="FooBean" />
    <%= foo.getBar() %>
</center>
```

- Equivalent, using an EL expression:

```
<center>
    ${foo.bar}
</center>
```

Integrated Expression Language Example

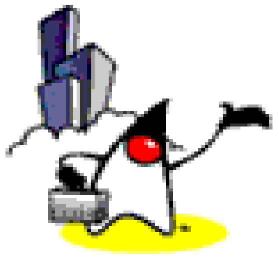
- Using scriptlets:

```
<%  Map m = (Map)pageContext.getAttribute("state" );
    State s = ((State)m.get( "NY" ));
    if( s != null ) {
%>
        <%= s.getCapitol() %>
<%  } %>
```

- Equivalent, using an EL expression:

```
${state["NY"].capitol}
```

Simple Tag Handlers & Extensions



Comparison of Tag Extensions

- Classic tag extensions (JSP 1.2 technology)
  - Complex tag handler API
  - Written only in the Java programming language
  - Created only by tag library developers
- Simple tag extensions & Tag Files (JSP 2.0 technology)
  - Simpler tag handler API
  - Written in the Java programming language or using JSP syntax
  - Created by page authors or tag library developers

Simple Tag Handlers

- Simpler to use than classic tag handlers
- Implement SimpleTag interface
- Usually extend SimpleTagSupport class
- doTag() of SimpleTag interface gets invoked when the end element of the tag is encountered
  - You want to override this method

How is a Simple Tag Handler called from container?

```
ATag t = new ATag();
t.setJSPContext(...);
t.setParent(...);
t.setAttribute1(value1);
t.setAttribute2(value2);
...
t.setJspBody(new JspFragment(...))
t.doTag();
```

JspFragment interface

- Encapsulates a portion of JSP code in an object that can be invoked as many times as needed
- JSP Fragments are defined using JSP syntax
  - as the body of a tag for an invocation to a SimpleTag handler, or
  - as the body of a <jsp:attribute> standard action specifying the value of an attribute that is declared as a fragment, or
  - to be of type JspFragment in the TLD

Simple Tag handler that does not manipulate body

- If a tag handler needs to simply evaluate the body,
  - it gets the body with the getJspBody() method of SimpleTag in the form of JspFragment object
  - and then evaluates the body with the invoke() method
- Invoke() method
  - use invoke(null) when there is no need to manipualte body
  - use invoke(StringWriter writer) otherwise

Example: Simple Tag handler that does not manipulate body

```
public class IfSimpleTag extends SimpleTagSupport {
  private boolean test;
  public void setTest(boolean test) {
    this.test = test;
  }
  public void doTag() throws JspException, IOException {
    if(test){
      getJspBody().invoke(null);
    }
  }
}
```

Simple Tag handler that does manipulate body

- If the tag handler needs to manipulate the body, the tag handler must capture the body in a StringWriter

```
public class SimpleWriter extends SimpleTagSupport {
  public void doTag() throws JspException, IOException {
    StringWriter sw = new StringWriter();
    jspBody.invoke(sw);
    jspContext().getOut().println(sw.toString().toUpperCase());
  }
}
```

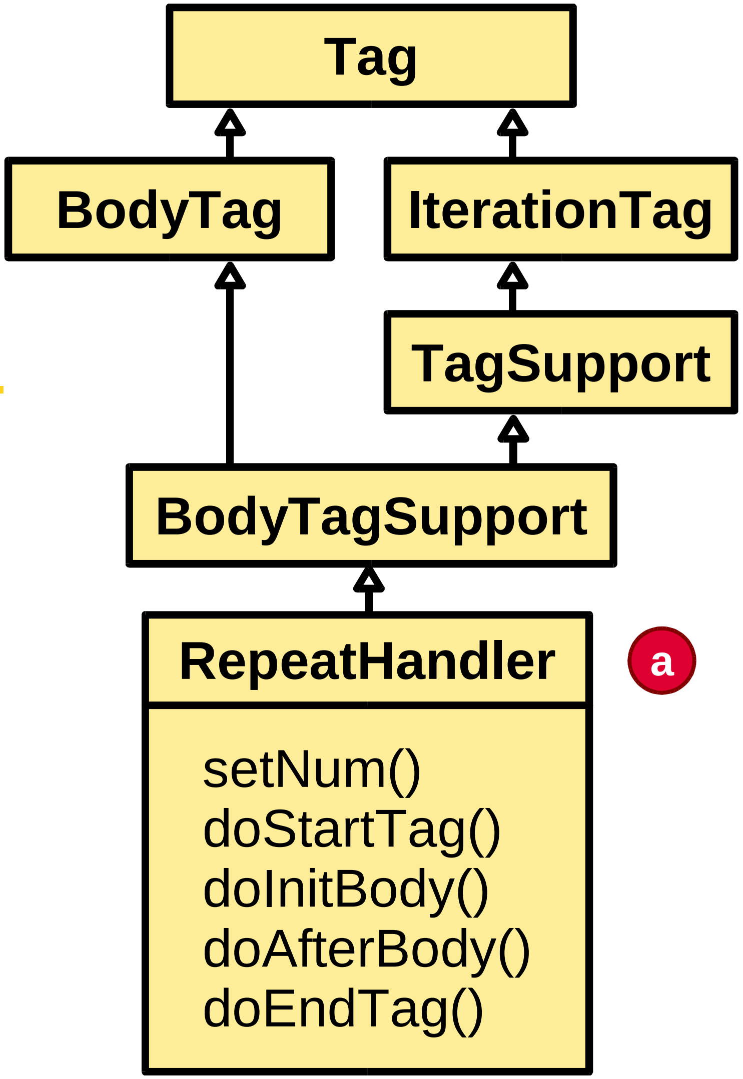
Repeat Tag Implemented as a Classic JSP 1.2 Tag Extension

Usage

```
<%@ taglib prefix="my"
        uri="/mytags" %>
<my:repeat num="3">
   tag body
</my:repeat>
```

Implementation

```
int doStartTag() {
   this.count = this.num;
   return Tag.EVAL_BODY_INCLUDE;        b
}

int doAfterBody() {
   this.count--;                c
   return (this.count > 0) ?
     Tag.EVAL_BODY_AGAIN :
     Tag.SKIP_BODY;             d
}
```



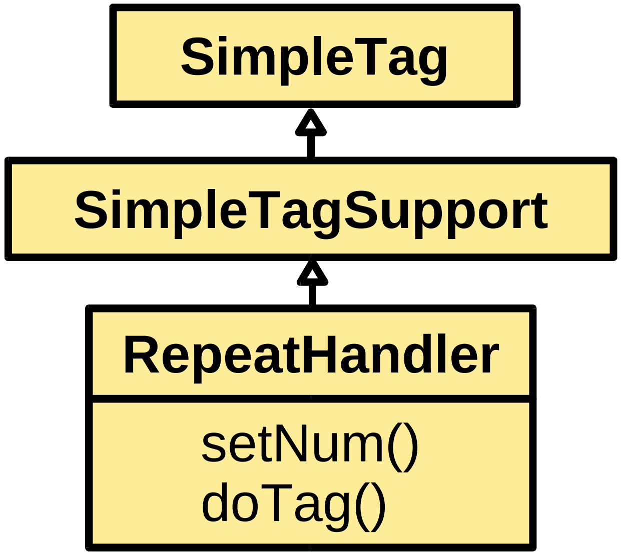
Repeat Tag Implemented as a Simple JSP 2.0 Tag Extension

Usage

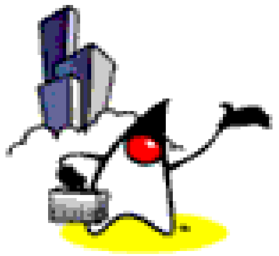
```
<%@ taglib prefix="my"
        uri="/mytags" %>
<my:repeat num="3">
    tag body
</my:repeat>
```

Implementation

```
void doTag() {
    for( int i = 0; i < num; i++ ) {
        getJspBody().invoke( null );
    }
}
```



Tag Files



Tag Files

- A source file that contains a fragment of JSP code that is reusable as a custom tag
- Allow you to create custom tags using JSP syntax
  - Empowers page authors
  - Faster round-trip development
- Get translated into a tag handler and then compiled automatically by container
  - JSP : Servlet :: Tag File : Tag Handler
- No TLD file is required

Tag Files

- Simple yet flexible packaging
  - Just drop a .tag file in /WEB-INF/tags/
  - Implicit tag library automatically generated
  - Or, write a .tld for added flexibility
  - Or, package in a JAR with a .tld

Declaring a tag library: tagdir attribute in taglib directive

- Identifies the location of the tag files
- Value of it must start with /WEB-INF/tags/
- Syntax
  - <%@ taglib prefix="tt" tagdir=/WEB-INF/tags/dir %>

Directives Used In a Tag file

- taglib
- include
- tag
  - Similar to the JSP page's page directive, but applies to tag files
- attribute
  - Declares attributes of the custom tag defined in the tag file
- variable
  - Declares an EL variable exposed by the tag to the calling page

“attribute” Directive attributes

- description
- name
- required
- rtexprvalue
- type
- fragment (default is false)
  - if true, container fixes
    - the rtexprvalue attribute at true
    - the type attribute at javax.servlet.jsp.tagext.JspFragment
  - Otherwise, it is a normal attribute to be evaluated by the container prior to being passed to the tag handler

Example 1: Simple Attribute (shipDate.tag)

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<%@ attribute name="shipping" required="true" %>

<jsp:useBean id="now" class="java.util.Date" />
<jsp:useBean id="shipDate" class="java.util.Date" />
<c:choose>
  <c:when test="${shipping == 'QuickShip'}">
    <c:set var="days" value="2" />
  </c:when>
  <c:when test="${shipping == 'NormalShip'}">
    <c:set var="days" value="5" />
  </c:when>
  <c:when test="${shipping == 'SaverShip'}">
    <c:set var="days" value="7" />
  </c:when>
</c:choose>
<jsp:setProperty name="shipDate" property="time"
  value="${now.time + 86400000 * days}" />
<fmt:formatDate value="${shipDate}" type="date"
  dateStyle="full"/>.<br><br>
```

Example 1: Simple Attribute (bookreceipt.jsp - calling page)

```
<%@ taglib prefix="sc" tagdir="/WEB-INF/tags" %>
<h3><fmt:message key="ThankYou"/> ${param.cardname}.</h3><br>
<fmt:message key="With"/>
<em><fmt:message key="${param.shipping}"/></em>,
<fmt:message key="ShipDateLC"/>
<sc:shipDate shipping="${param.shipping}" />
<c:remove var="cart" scope="session" />
<c:url var="url" value="/bookstore" />
<strong><a href="${url}"><fmt:message
   key="ContinueShopping"/></a>&nbsp;&nbsp;&nbsp;</strong>
```

“variable” Directive

- Declares EL variables
- EL variables
  - EL variables emulate OUT (from tag file to calling page) type while Tag attributes emulate IN (from calling page to tag file) type
  - not initialized by the calling page
  - set by the tag file

“variable” Directive attributes

- description
- name-given|name-from-attribute
  - Defines an EL variable to be used in the page invoking this tag
- alias
- variable-class
- declare
- scope

Evaluating Fragments passed to Tag Files

- Web container passes two types of fragments
  - fragment attribute
  - tag body
- Fragments are evaluated by tag handler not container
- Within a tag file,
  - use jsp:invoke to evaluate a frament attribute
  - use jsp:doBody to evaluate a tag file body

Evaluating Fragments passed to Tag Files

- Result of evaluation
  - is sent to the response or
  - is stored in an EL variable for later manipulatipn
    - var: type String
    - varReader: type java.io.Reader
  - scope attribute (optional) indicates the scope of resulting variable
    - page (default)
    - request
    - session
    - application

Example 2: Simple and Fragment Attributes and Variables (catalog.tag - 1)

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

<%@ attribute name="bookDB" required="true" type="database.BookDB" %>
<%@ attribute name="color" required="true" %>

<%@ variable name-given="price" %>
<%@ variable name-given="salePrice" %>

<%@ attribute name="normalPrice" fragment="true" %>
<%@ attribute name="onSale" fragment="true" %>
```

Example 2: Simple and Fragment Attributes and Variables (catalog.tag - 2)

```
<center>
<table>
<c:forEach var="book" begin="0" items="${bookDB.books}">
  <tr>
  <c:set var="bookId" value="${book.bookId}" />
  <td bgcolor="${color}">
     <c:url var="url" value="/bookdetails" >
       <c:param name="bookId" value="${bookId}" />
     </c:url>
     <a href="${url}"><strong>${book.title}&nbsp;</strong></a></td>
  <td bgcolor="${color}" rowspan=2>

  <c:set var="salePrice" value="${book.price * .85}" />
  <c:set var="price" value="${book.price}" />

  <c:choose>
     <c:when test="${book.onSale}" >
       <jsp:invoke fragment="onSale" />
     </c:when>
     <c:otherwise>
       <jsp:invoke fragment="normalPrice" />
     </c:otherwise>
   </c:choose>  &nbsp;</td>
```

Attributes passed from calling page

Variables

Attributes with fragments

Example 2: Simple and Fragment Attributes and Variables (catalog.tag - 3)

```
  <td bgcolor="${color}" rowspan=2>
  <c:url var="url" value="/bookcatalog" >
    <c:param name="Add" value="${bookId}" />
  </c:url>
  <p><strong><a href="${url}">&nbsp;<fmt:message
     key="CartAdd"/>&nbsp;</a></td></tr>

  <tr>
  <td bgcolor="#ffffff">
  &nbsp;&nbsp;<fmt:message key="By"/> <em>${book.firstName}&nbsp;$
     {book.surname}</em></td></tr>
</c:forEach>

</table>
</center>
```

Example 2: Simple and Fragment Attributes and Variables (bookcatalog.jsp - 1)

```
<%@ taglib prefix="sc" tagdir="/WEB-INF/tags" %>

<jsp:useBean id="bookDB" class="database.BookDB" scope="page" >
  <jsp:setProperty name="bookDB" property="database"
     value="${bookDBAO}" />
</jsp:useBean>


<c:if test="${!empty param.Add}">
  <c:set var="bid" value="${param.Add}"/>
  <jsp:setProperty name="bookDB" property="bookId" value="${bid}" />
  <c:set var="addedBook" value="${bookDB.bookDetails}" />
    <p><h3><font color="red" size="+2">
    <fmt:message key="CartAdded1"/> <em>${addedBook.title}</em>
     <fmt:message key="CartAdded2"/></font></h3>
  </c:forEach>
</c:if>
```

Example 2: Simple and Fragment Attributes and Variables (bookcatalog.jsp - 2)

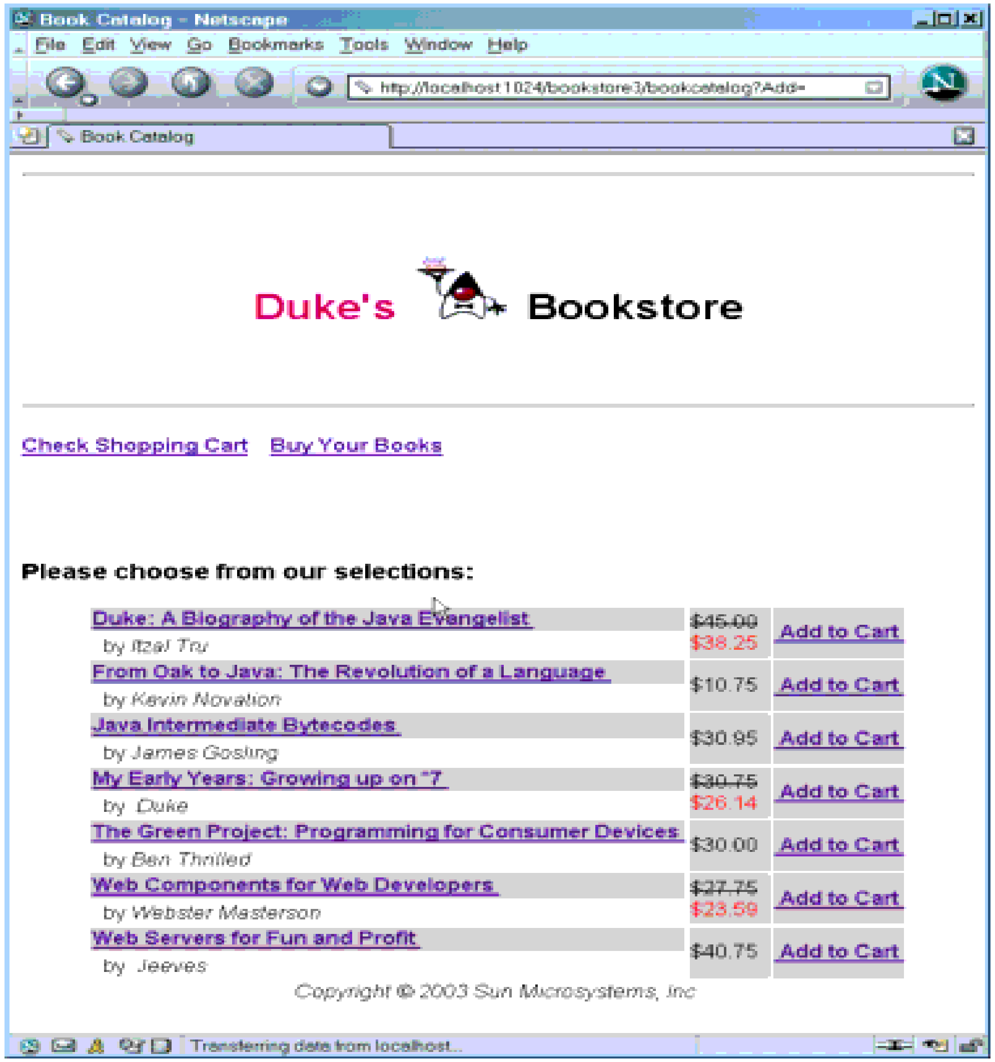
```
<c:if test="${sessionScope.cart.numberOfItems > 0}">
  <c:url var="url" value="/bookshowcart" >
    <c:param name="Clear" value="0" />
    <c:param name="Remove" value="0" />
  </c:url>
<p><strong><a href="${url}"><fmt:message
     key="CartCheck"/></a>&nbsp;&nbsp;&nbsp;
    <c:url var="url" value="/bookcashier" />
    <a href="${url}"><fmt:message key="Buy"/></a></p></strong>
</c:if>

<br>&nbsp;
<br>&nbsp;
<h3><fmt:message key="Choose"/></h3>
```

Example 2: Simple and Fragment Attributes and Variables (bookcatalog.jsp - 3)

```
<sc:catalog bookDB ="${bookDB}" color="#cccccc">
  <jsp:attribute name="normalPrice">
    <fmt:formatNumber value="${price}" type="currency"/>
  </jsp:attribute>
  <jsp:attribute name="onSale">
    <strike><fmt:formatNumber value="${price}"
     type="currency"/></strike><br/>
    <font color="red"><fmt:formatNumber value="${salePrice}"
     type="currency"/></font>
  </jsp:attribute>
</sc:catalog>
```

onSale fragment

Example 2: Simple and Fragment Attributes and Variables (bookcatalog.jsp - 3)


Book Catalog - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:1024/bookstore3/bookcatalog?Add=
Book Catalog
Duke's Bookstore
Transferring data from localhost...

Check Shopping Cart Buy Your Books

Please choose from our selections:

Duke: A Biography of the Java Evangelist by *Itzal Tru*	~~$45.00~~ $38.25	Add to Cart
From Oak to Java: The Revolution of a Language by *Kevin Novation*	$10.75	Add to Cart
Java Intermediate Bytecodes by *James Gosling*	$30.95	Add to Cart
My Early Years: Growing up on *7 by *Duke*	~~$30.75~~ $26.14	Add to Cart
The Green Project: Programming for Consumer Devices by *Ben Thrilled*	$30.00	Add to Cart
Web Components for Web Developers by *Webster Masterson*	~~$27.75~~ $23.59	Add to Cart
Web Servers for Fun and Profit by *Jeeves*	$40.75	Add to Cart

Copyright © 2003 Sun Microsystems, Inc

Example 3:Repeat Tag Implemented as a Tag File

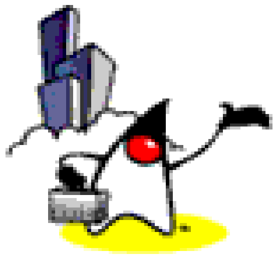
Usage

```
<%@ taglib prefix="my"
        tagdir="/WEB-INF/tags/" %>
<my:repeat num="3">
   tag body
</my:repeat>
```

Implementation

```
<%-- /WEB-INF/tags/repeat.tag --%>
<%@ attribute name="num" %>
<%@ taglib prefix="c"
          uri="http://java.sun.com/jsp/jstl/core" %>

<c:forEach begin="1" end="${num}">
   <jsp:doBody />
</c:forEach>
```

Improved XML Syntax



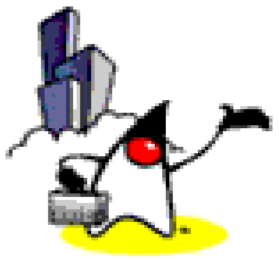
Improved XML Syntax: JSPX and TAGX

- Finally, no more need for **<jsp:root>**!
- JSP technology as an XML namespace
- Ideal for outputting XML content like SVG:

```
<!-- images/hello.svg -->
<svg xmlns:c="http://java.sun.com/jsp/jstl/core"
     width="8.5in" height="11.0in">
   <c:forEach var="y" begin="3" end="10">
      <text style="fill:blue;" y="${y*15}">
         "Hello, ${firstName} ${lastName}!"
      </text>
   </c:forEach>
</svg>
```

- JSP documents (.jspx) or Tag files (.tagx)

Other Features



Other Features...

- Central configuration via url-patterns
  - Map extensions other than .jsp
  - Enable / disable scriptlets or EL globally
  - Control page encoding globally
  - Preludes / codas
- Portable debugging support through JSR-45
- Dynamic attributes
- Enhanced I18N support
- Fragment attributes

Passion!