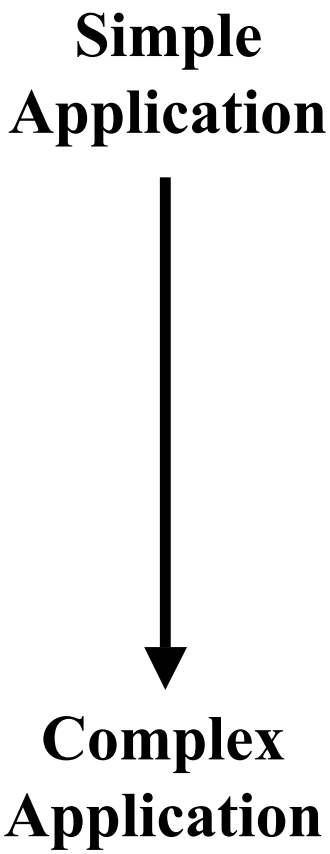
The JSTL Standard Tag Library

***Core* Servlets & JSP book: www.coreservlets.com**
***More* Servlets & JSP book: www.moreservlets.com**
Servlet and JSP Training Courses: courses.coreservlets.com

Agenda

- **Obtaining JSTL documentation and code**
- **The JSTL Expression Language**
- **Looping Tags**
  - Looping a certain number of times
  - Looping over data structures
- **Conditional Evaluation Tags**
  - Simple choice
  - Multiple choices
- **Database access tags**

Uses of JSP Constructs



- Scripting elements calling servlet code directly
- Scripting elements calling servlet code indirectly (by means of utility classes)
- Beans
- Custom tags
- Servlet/JSP combo (MVC), with beans and possibly custom tags

JSTL Overview

- **Full JSTL**
  - Contains many common and useful JSP custom tags
  - Particularly useful when you are using MVC, but the data contains a varying number of entries
  - Based on Struts looping and logic tags
  - Not part of the JSP 1.2, 2.0 or 2.1 specs
    - It is a separate specification that requires a separate download
- **Expression language**
  - The JSP expression language came from JSTL, but is now part of JSP 2.0, JSP 2.1 and JSF

Iteration Tags

Simplest for Page Author

Most Control for Page

- Bean
  - ${cust.withdrawls}
- Custom tag (non-looping)
  - <mytags:withDrawalTable customer="${cust}" border="..."headerStyles="...
- Custom tag (looping)
- JSTL Loop
- JSP Scripting Loop

Looping Tags: Summary

- **Looping with explicit numeric values**

```
<c:forEach var="name" begin="x" end="y" step="z">
  Blah, blah ${name}
  </c:forEach
```

- **Looping over data structures**
  - Can loop down arrays, strings, collections, maps

```
<c:forEach var="name"
 items="array-or-collection">
 Blah, blah ${name}
 </c:forEach>
```

- **Looping down delimited strings**
  - -forTokens

Looping Tags: Motivation

- **JSP without JSTL**

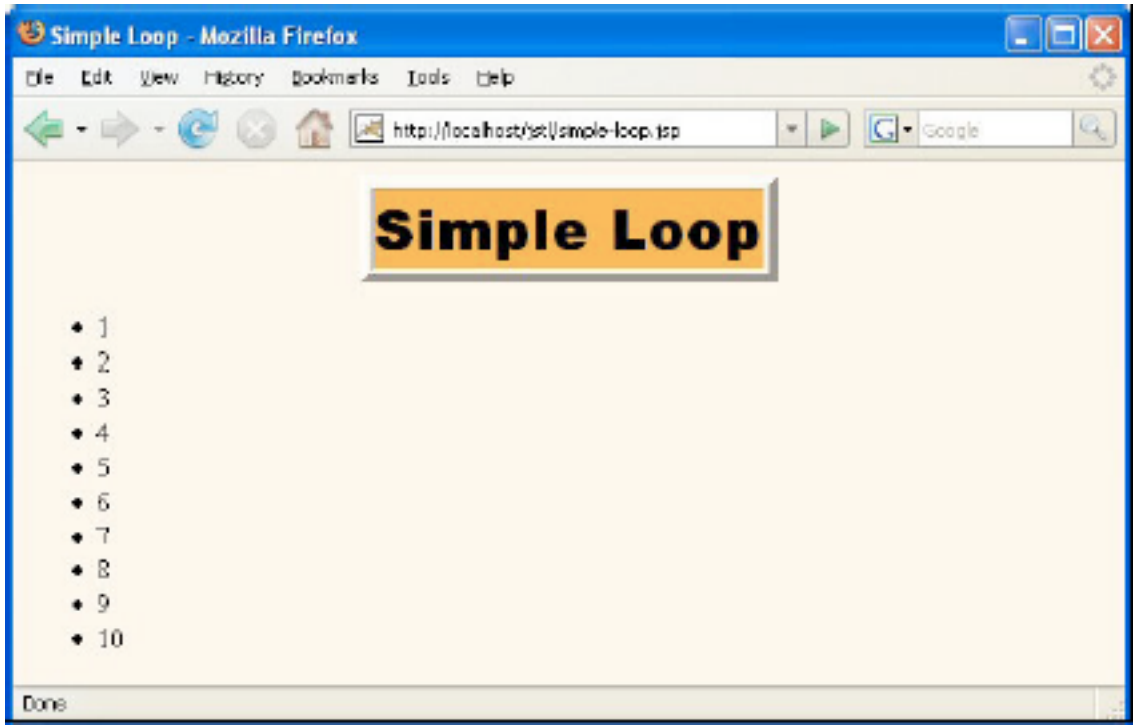
```
<UL>
<%
for(int i=0; i<messages.length; i++) {
String message = messages[i];
%>
<LI><%= message %>
<% } %>
</UL>
```

- **JSP with JSTL**

```
<UL>
<c:forEach var="message" items="${messages}”
<LI>${message}
</c:forEach>
</UL>
```

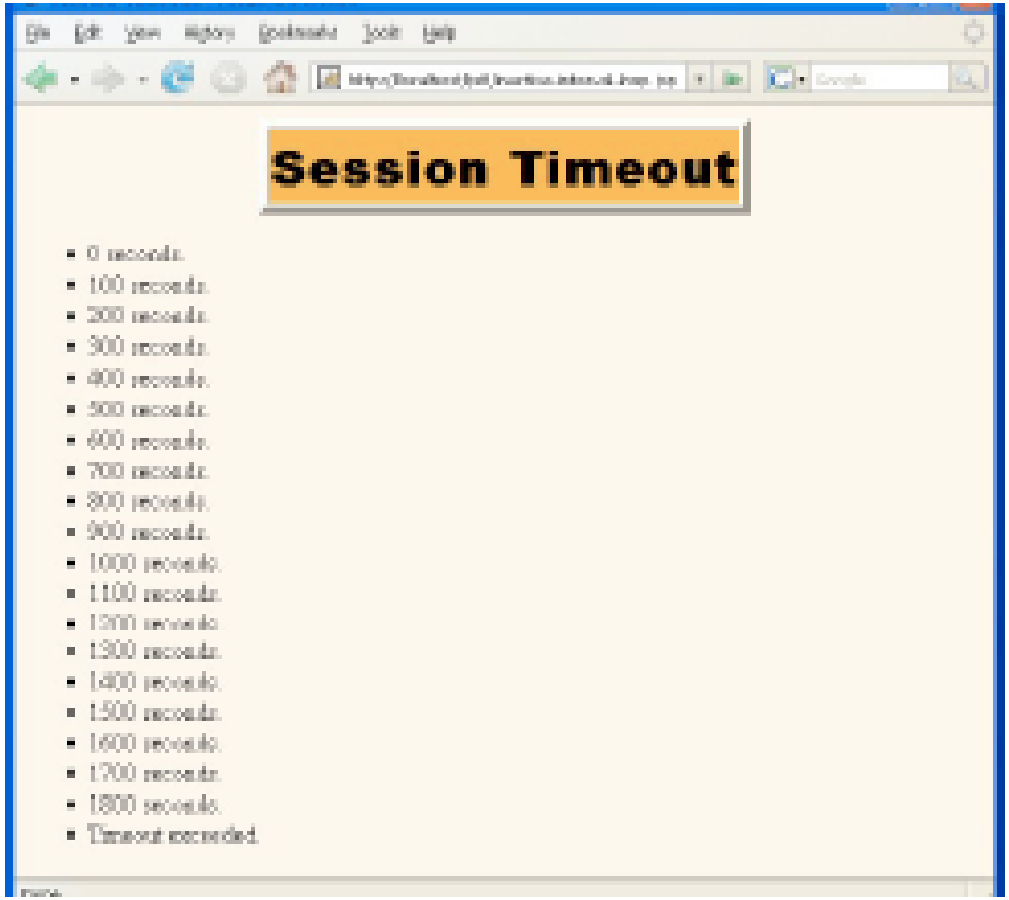
Looping with Simple Numeric Values

```
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/
core"%>
<UL>
<c:forEach var="i" begin="1"
end="10">
<LI>${i}
</c:forEach>
</UL>
```


Simple Loop - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/jstl/simple-loop.jsp
Google
Simple Loop
1
2
3
4
5
6
7
8
9
10
Done

Looping with a Designated Step Size

```
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core"%>
<UL>
<c:forEach
  var="seconds"
  begin="0"
  end="${pageContext.session.maxInactiveInterval}"
  step="100">
 <LI>${seconds} seconds.
</c:forEach>
 <LI>Timeout exceeded.
</UL>
```


Session Timeout
0 seconds.
100 seconds.
200 seconds.
300 seconds.
400 seconds.
500 seconds.
600 seconds.
700 seconds.
800 seconds.
900 seconds.
1000 seconds.
1100 seconds.
1200 seconds.
1300 seconds.
1400 seconds.
1500 seconds.
1600 seconds.
1700 seconds.
1800 seconds.
Timeout exceeded.

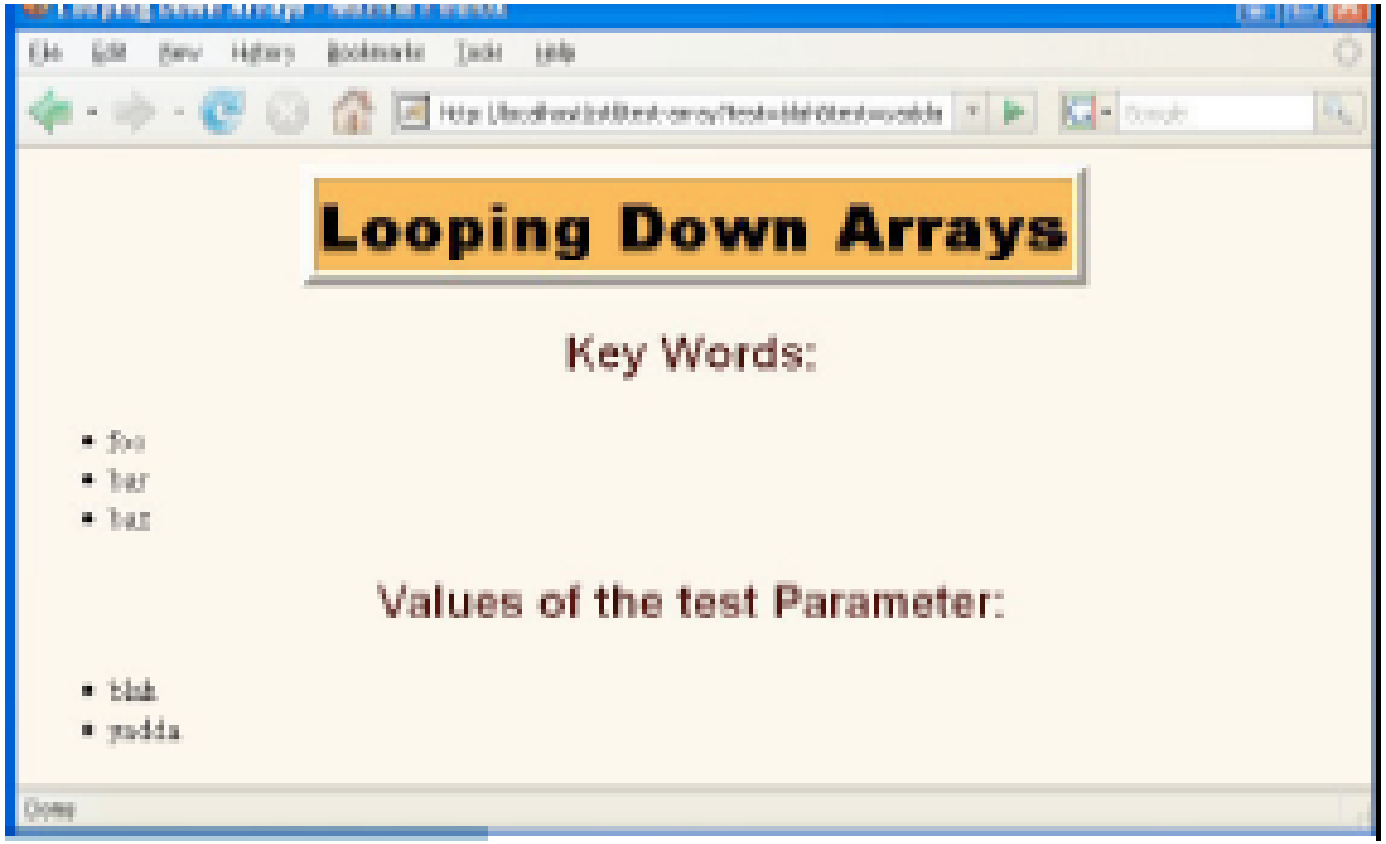
Looping Down Arrays (Servlet)

```
public class ArrayServlet extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
String[] words = { "foo", "bar", "baz"};
request.setAttribute("words", words);
String address = "/WEB-INF/results/array-loop.jsp";
RequestDispatcher dispatcher =
request.getRequestDispatcher(address);
dispatcher.forward(request, response);
}
}
```

- **Note**
  - url-pattern in web.xml is /test-array

Looping Down Arrays

```
<%@ taglib prefix="c"
           uri="http://java.sun.com/jsp/jstl/core"%>
<H2>Key Words:</H2>
<UL>
<c:forEach var="word"
           items="${words}">
 <LI>${word}
</c:forEach>
</UL>
<H2>Values of the test Parameter:</H2>
<UL>
<c:forEach var="val"
           items="${paramValues.test}">
<LI>${val}
</c:forEach>
   </UL>
Note
```


Looping Down Arrays
Key Words:
foo
bar
baz
Values of the test Parameter:
blah
yadda

Note for Older Servers

- **Examples in this lecture assume JSP 2.0+**
  - JSTL also runs in older servers, but you have to replace direct output of ${foo} with <c:out value="$foo"/>
- **Previous example (JSP 2.0+)**

```
<UL>
<c:forEach var="word" items="${words}">
  <LI>${word}
 </c:forEach>
</UL>
```

- **Previous example (JSP 1.2)**

```
<UL>
<c:forEach var="word" items="${words}">
  <LI><c:out value="${word}"/>
</c:forEach>
</UL>
```

${foo} vs <c:out value="${foo}"/>

- **c:out runs on older servers**
  - Oracle 9i, BEA WebLogic 8.x, IBM WebSphere 5.x
- **c:out escapes HTML (XML) characters**
  - <c:out value="<h1>"/> outputs <h1>
    - Very important if value being output comes from end user.
    - Disable with <c:out value="…" escapeXml="false"/>
- **c:out lets you supply a default value**
  - <c:out value="${foo}" default="explicit value"/> or <c:out value="${foo}" default="${calculatedValue}"/>
  - The default is output when ${foo} evaluates to null

Looping Down Collections:Notes

- **Can loop down collections other than arrays**
  - "items" can refer to an array, collection (List, Map, Set, etc.), comma-separated String, Iterator or Enumeration
- **Can access sub-elements of local variable**
  - <c:forEach var="bean" items="${collectionOfBeans}">
    ${bean.property}
    <c:forEach>
  - <c:forEach var="collection"
    items="${collectionOfCollections}">
    ${collection[part]}
    <c:forEach>
  - For details on accessing collections and sub-elements, see JSP EL

Looping Down Arrays: Accessing SubElements (Servlet)

```
public class ArrayServlet2 extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    Name[] names =
      { new Name("Bill", "Gates"),
        new Name("Larry", "Ellison"),
        new Name("Sam", "Palmisano"),
        new Name("Scott", "McNealy"),
        new Name("Eric", "Schmidt"),
        new Name("Jeff", "Bezos") };
    request.setAttribute("names", names);
    String[][] sales =
      { {"2005", "12,459", "15,622"},
        {"2006", "18,123", "17,789"},
        {"2007", "21,444", "23,555"} };
    request.setAttribute("sales", sales);
    String address = "/WEB-INF/results/array-loop2.jsp";
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
  }
}
```

Note: url-pattern is /test-array2

Looping Down Arrays: Accessing SubElements (Bean)

```
public class Name {
  private String firstName;
  private String lastName;

  public Name(String firstName, String lastName) {
    setFirstName(firstName);
    setLastName(lastName);
  }

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

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

  public String getLastName() { return(lastName); }

  public void setLastName(String lastName) {
    this.lastName = lastName;
  }
}
```

Looping Down Arrays: Accessing SubElements (JSP)

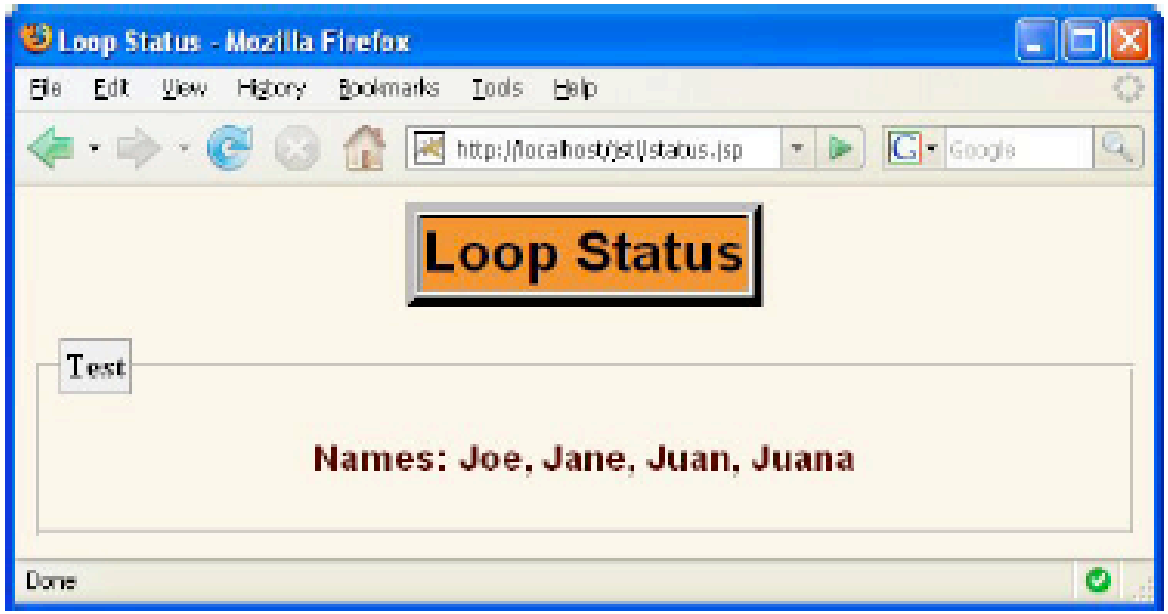
```
<H2>Attendees at the coreservlets.com Party</H2>
<UL>
<c:forEach var="name" items="${names}">
  <LI>${name.firstName} ${name.lastName}
</c:forEach>
</UL>
<H2>Comparing Apples and Oranges</H2>
<TABLE BORDER="1">
  <TR><TH>Year</TH>
      <TH>Apples Sold</TH>
      <TH>Oranges Sold</TH></TR>
<c:forEach var="row" items="${sales}">
  <TR><c:forEach var="col" items="${row}">
        <TD>${col}</TD>
      </c:forEach>
</c:forEach>
</TABLE>
```

Looping Down Arrays: Accessing SubElements (Result)

Looping Down Arrays: Accessing SubElements (Result)

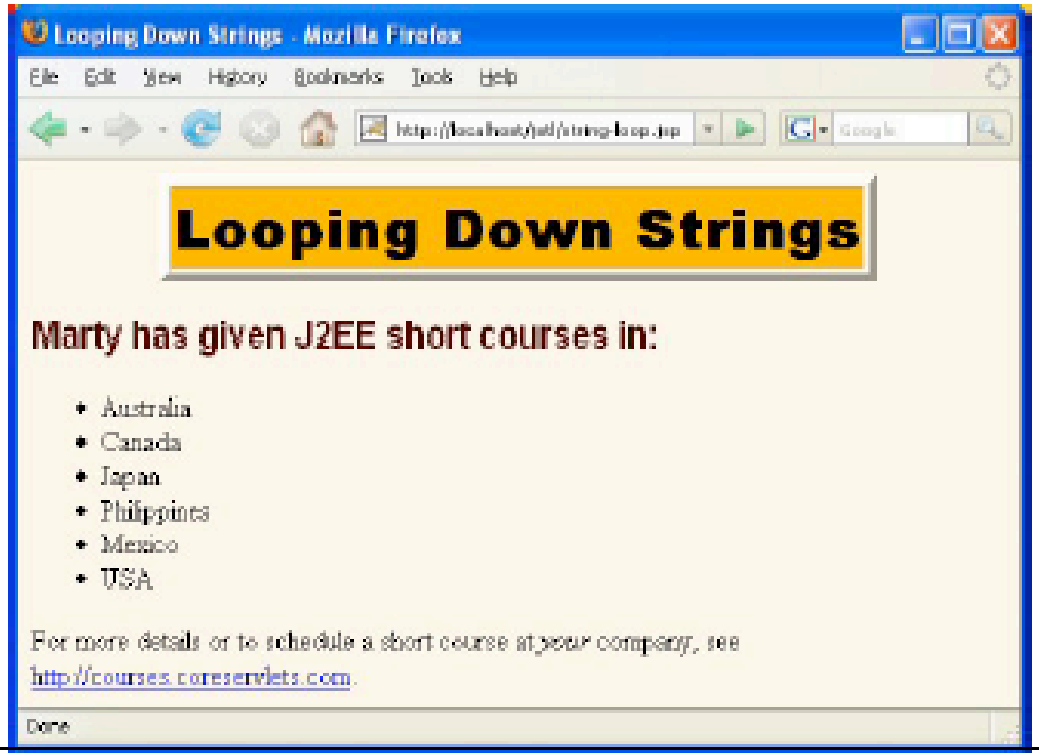
Loop Status: Example

```
<%@ taglib prefix="c"
          uri="http://java.sun.com/jsp/jstl/core" %>
<% String[] names = {"Joe", "Jane", "Juan", "Juana"};
   request.setAttribute("names", names); %>
<h2>Names:
<c:forEach var="name" items="${names}" varStatus="status">
  ${name}<c:if test="${!status.last}">,</c:if>
</c:forEach>
</h2>
```



Looping Down Comma-Delimited Strings

```
<%@ taglib prefix="c"
           uri="http://java.sun.com/jsp/jstl/core"%>
<UL>
<c:forEach
  var="country"
  items="Australia,Canada,Japan,Philippines,Mexico,USA">
  <LI>${country}
</c:forEach>
</UL>
```



Looping Down Arbitrarily-Delimited Strings

```
<%@ taglib prefix="c"
           uri="http://java.sun.com/jsp/jstl/core" %>
<UL>
<c:forTokens var="color"
     items="(red (orange) yellow)(green)((blue) violet)"
     delims="()">
  <LI>${color}
</c:forTokens>
</UL>
```

- **Point:**

  forTokens built on forEach: you can build your own custom tags based on JSTL tags

Logic Tags

Conditional Evaluation Tags

- **One choice: `if`**

```
<c:if test="${someTest}">
  Content
</c:if>
```

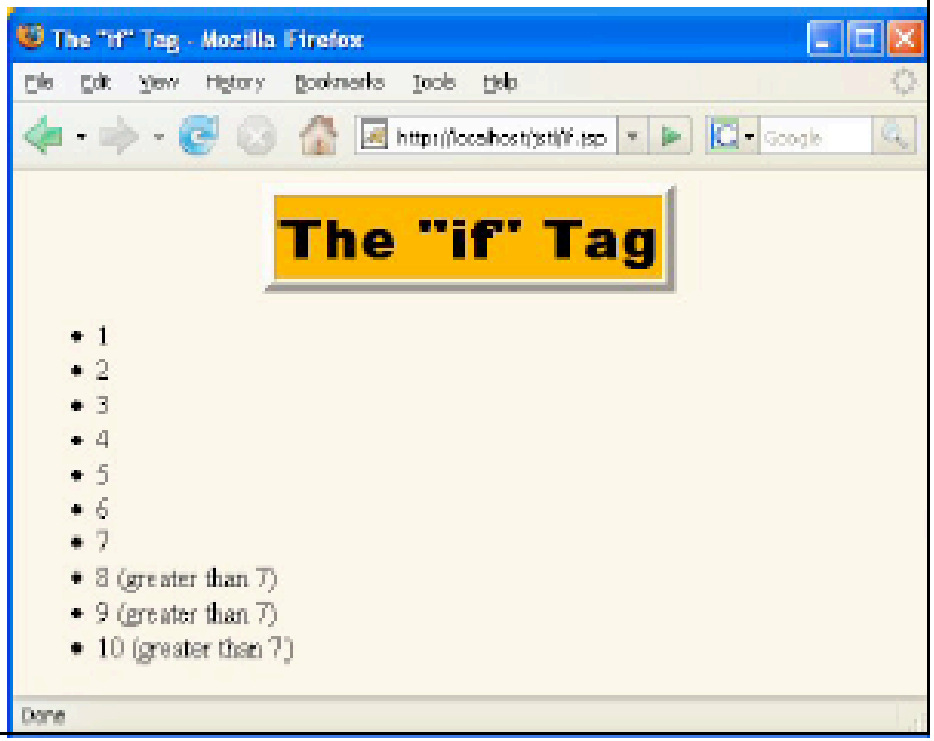
- **Lots of choices: `choose`**

```
<c:choose>
  <c:when test="test1">Content1</c:when>
  <c:when test="test2">Content2</c:when>
  ...
  <c:when test="testN">ContentN</c:when>
  <c:otherwise>Default Content</c:otherwise>
</c:choose>
```

- **Caution: resist use of business logic!**

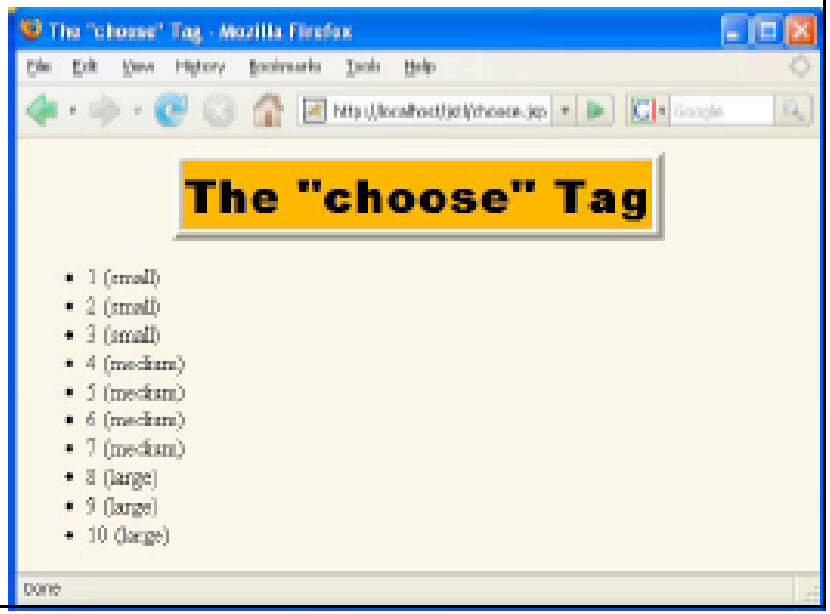
The "if" Tag

```
<%@ taglib prefix="c"
           uri="http://java.sun.com/jsp/jstl/core"%>
<UL>
<c:forEach var="i" begin="1" end="10">
  <LI>${i}
      <c:if test="${i > 7}">
      (greater than 7)
      </c:if>
</c:forEach>
</UL>
```



The "choose" Tag

```
<%@ taglib prefix="c"
           uri="http://java.sun.com/jsp/jstl/core"%>
<UL>
<c:forEach var="i" begin="1" end="10">
  <LI>${i}
    <c:choose>
      <c:when test="${i < 4}">
        (small)
      </c:when>
      <c:when test="${i < 8}">
        (medium)
      </c:when>
      <c:otherwise>
        (large)
      </c:otherwise>
    </c:choose>
</c:forEach>
</UL>
```



Other Tags

URL-Handling Tags

- **<c:import>**
  - Read content from arbitrary URLs
    - Insert into page
    - Store in variable
    - Or make accessible via a reader
  - Unlike <jsp:include>, not restricted to own system
- **<c:redirect>**
  - Redirects response to specified URL
- **<c:param>**
  - Encodes a request parameter and adds it to a URL
  - May be used within body of <c:import> or <c:redirect>

Summary

- **JSTL overview**
  - JSTL is similar to the old Struts looping and logic tags, but better
  - JSTL is standardized, but not a standard part of JSP
    - You have to add JAR files to WEB-INF/lib
    - It is supposed to be included with all JSF implementations
- **Supports a concise expression language**
  - Lets you access bean properties and implicit objects
  - EL is standard part of JSP 2.0, JSP 2.1, and JSF
- **Looping tags**
  - Explicit numeric values
  - Arrays, collections, maps, strings, etc.
- **Conditional evaluation tags**
  - Single options
  - Multiple options

Questions?