The JSP Standard Tag Library: JSTL 1.0

A Specification from the JSR-052 Committee

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**
  - Accessing stored attributes
  - Accessing bean properties
  - Accessing collections
  - Implicit objects
- **Looping Tags**
  - Looping a certain number of times
  - Looping over data structures
- **Conditional Evaluation Tags**
  - Single choice
  - Multiple choices
- **Database Access Tags**
- **Other Tags**

JSTL Overview

- **JSTL is not part of the JSP 1.2 Specification**
  - It is a separate specification that requires a separate download
  - Available only in servers that support servlets 2.3 and JSP 1.2 or later. Cannot be retrofitted into JSP 1.1.
- **The JSTL expression language will be part of JSP 2.0**
- **The JSTL Specification is available in PDF**
  - http://jcp.org/aboutJava/communityprocess/final/jsr052/
- **Some JSTL books now available**
  - Most current ones are obsolete (written to beta spec)
  - Recommend *Core JSTL* by David Geary (due 10/02)

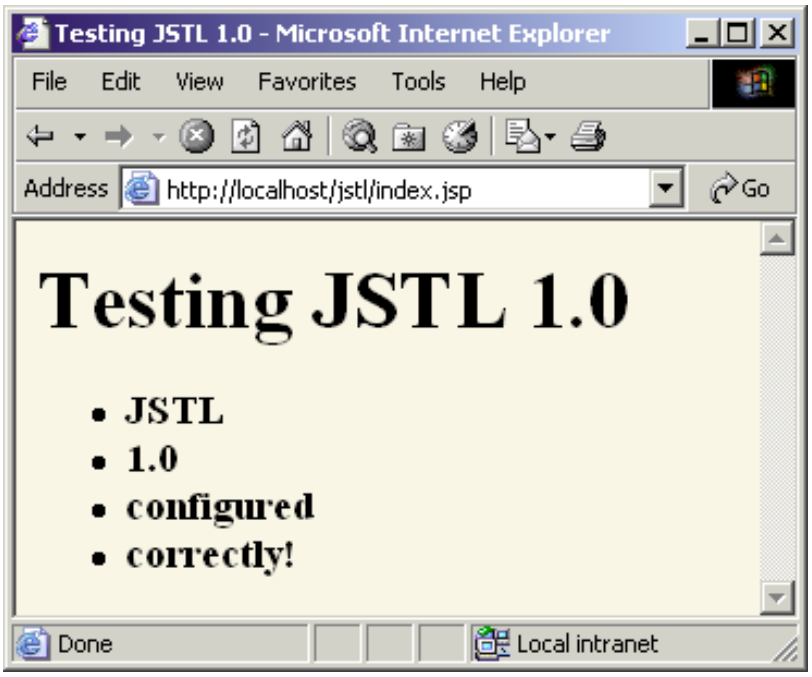
Installing JSTL 1.0

- **Some servers come with JSTL preinstalled**
  - E.g., Caucho Resin
- **Official reference implementation**
  - http://jakarta.apache.org/builds/jakarta-taglibs/releases/standard/
    - Works best with Tomcat 4.0.4 or later
- **JSTL (like JSP) is a specification, not an implementation**
  - Code is portable but not all versions are equal
    - Speed, tools, and development environments vary
- **To install:**
  - Download zip file
  - Unzip into directory of your choice (e.g., C:\jakarta-taglibs).
  - Copy *install_dir*/standard-1.0.1/lib/**jstl.jar** and *install_dir*/standard-1.0.1/lib/**standard.jar** to the WEB-INF/lib directory of your Web application

Testing Installation

```
<%
String[] messages =
  {"JSTL", "1.0", "configured", "correctly!"};
pageContext.setAttribute("messages", messages);
%> …
<H1>Testing JSTL 1.0</H1>
<%@ taglib prefix="c"
           uri="http://java.sun.com/jstl/core" %>
<UL>
<c:forEach var="message" items="${messages}">
  <LI><B><c:out value="${message}"/></B>
</c:forEach>
</UL>
…
```


Installation Test: Successful Result



The JSTL Expression Language

- **Accessed via `${expression}`**
  - Note change from JSTL beta that used `$expression`
- **Similar to JavaScript and XPath**
- **Provides shorthand notation to access:**
  - Attributes of standard servlet objects
  - Bean properties
  - Map, List, and Array elements
- **Is standard part of JSP 2.0**
  - In JSTL, EL can be used only in attributes of JSTL tags
  - In JSP 2.0, the EL can be used anywhere
    - *web.xml* element and page directive attribute let you disable the EL for backward compatibility
  - JSP 2.0 due approx. 1Q 2003

Backdrop: The MVC Architecture

- **Servlet: does real programming**
  - Processes data, does business logic, stores results in beans

```
MyData data = new MyData(…);
request.setAttribute("key", data);
String address
  = "/someDir/PresentationPage.jsp";
RequestDispatcher dispatcher =
  getServletContext().getRequestDispatcher(address);
dispatcher.forward(request, response);
```

- **JSP: presents data only**

```
<jsp:useBean
   id="key"
   class="somepackage.MyData"
   scope="request" />
```

The JSTL EL: Accessing Variables

- **`${name}` means**
  - Search PageContext, HttpServletRequest, HttpSession, and ServletContext (*in order*) for attribute named `name`
  - If a String is expected, convert it to a String via its toString method
  - Accessing data stored in request, session, and servlet context attributes is the approach used in the MVC architecture. JSTL and MVC go hand in hand.
- **`${name1}text${name2}text` concatenates**
  - The following two are equivalent
    - ${foo}blah
    - pageContext.findAttribute("foo").toString() + "blah"
- **Note for following examples**
  - <c:out …> is used to output a value

Example: Accessing JSTL Variables (Servlet)

```
public class SimpleMVCServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    request.setAttribute("attribute1",
                         "First Attribute");
    HttpSession session = request.getSession(true);
    session.setAttribute("attribute2",
                         "Second Attribute");
    ServletContext application = getServletContext();
    application.setAttribute("attribute3",
                             new java.util.Date());
    request.setAttribute("repeated",
                         "HttpServletRequest");
    session.setAttribute("repeated", "HttpSession");
    application.setAttribute("repeated",
                             "ServletContext");
    gotoPage("/el/vars.jsp", request, response);
  }
```

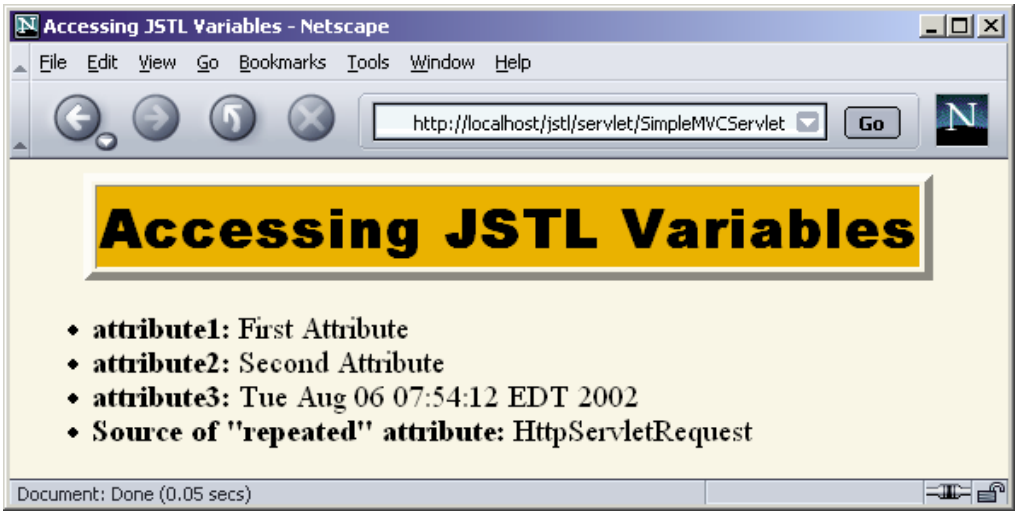
Example: Accessing JSTL Variables (Servlet Continued)

```
private void gotoPage(String address,
                      HttpServletRequest request,
                      HttpServletResponse response)
    throws ServletException, IOException {
  RequestDispatcher dispatcher =
    getServletContext().getRequestDispatcher(address);
  dispatcher.forward(request, response);
}
```


Example: Accessing JSTL Variables (JSP)

```
<%@ taglib prefix="c"
           uri="http://java.sun.com/jstl/core" %>
<UL>
  <LI><B>attribute1:</B> <c:out value="${attribute1}"/>
  <LI><B>attribute2:</B> <c:out value="${attribute2}"/>
  <LI><B>attribute3:</B> <c:out value="${attribute3}"/>
  <LI><B>Source of "repeated" attribute:</B>
      <c:out value="${repeated}"/>
</UL>
```

Example: Accessing JSTL Variables (Results)



The JSTL EL: Accessing Bean Properties with "." and "[]"

- **Simple case: object is simple bean**
  - Use var.propName to mean var.getPropName()
  - var.propName equivalent to var["propName"]
    - First version shorter; second version lets you use run-time values as property names
    - Be sure to use single quotes on outside with second version!
  - Dots or array notation can be chained
    - a.b.c.d means a.getB().getC().getD()
- **Advanced case: object is array, List, or Map**
  - Array: var[i] or var.i equivalent to theObject[i]
    - Where theObject is typecast version of attribute named var
  - List: var[i] or var.i equivalent to theObject.get(i)
  - Map: var.key or var["key"] equivalent to theObject.get("key")

Example: Simple Bean Property Access

```
package beans;

public class Employee {
  private Name name;
  private Company company;

  public Employee(Name name, Company company) {
    setName(name);
    setCompany(company);
  }

  public Name getName() { return(name); }

  public void setName(Name newName) {
    name = newName;
  }
  public Company getCompany() { return(company); }

  public void setCompany(Company newCompany) {
    company = newCompany;
  }
}
```


Example: Simple Bean Property Access (Continued)

```
package beans;

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 newFirstName) {
    firstName = newFirstName;
  }
  public String getLastName() { return(lastName); }

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

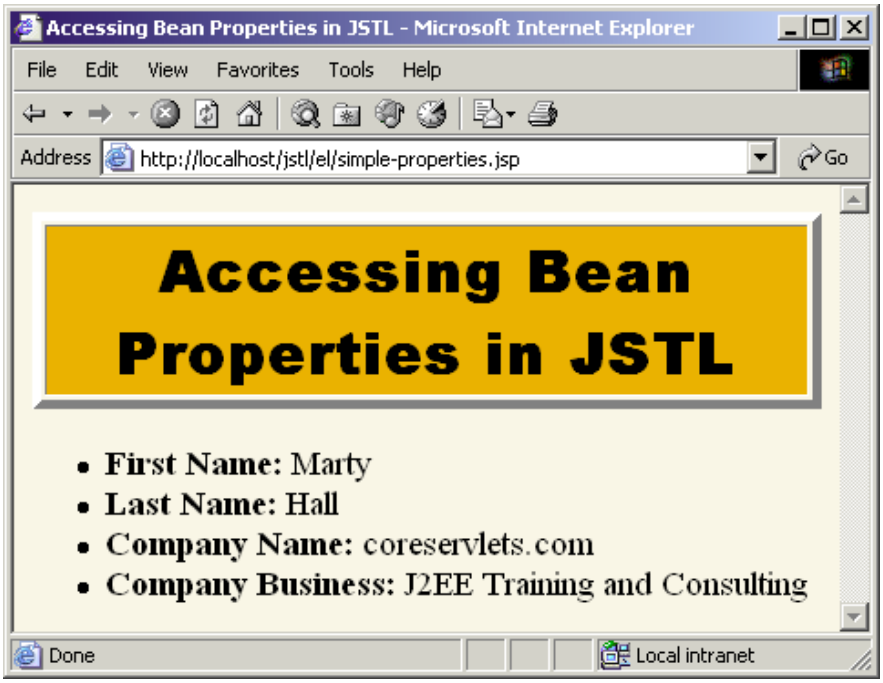
Example: Simple Bean Property Access (Continued)

```
package beans;

public class Company {
  private String companyName;
  private String business;

  public Company(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;
  }
}
```


Example: Simple Bean Property Access (Continued)

```
<%
Name name1 = new Name("Marty", "Hall");
Company company1 = new Company("coreservlets.com",
                               "J2EE Training and Consulting");
Employee employee1 = new Employee(name1, company1);
pageContext.setAttribute("employee", employee1);
%> …
<%@ taglib prefix="c"
           uri="http://java.sun.com/jstl/core" %>
<UL>
  <LI><B>First Name:</B>
      <c:out value="${employee.name.firstName}"/>
  <LI><B>Last Name:</B>
      <c:out value="${employee.name.lastName}"/>
  <LI><B>Company Name:</B>
      <c:out value="${employee.company.companyName}"/>
  <LI><B>Company Business:</B>
      <c:out value="${employee.company.business}"/>
</UL>
…
```

Example: Simple Bean Property Access (Results)



Accessing Bean Properties: Benefit

- Consider previous example:

```
<LI><B>First Name:</B>
    <c:out value="${employee.name.firstName}"/>
```

- Without JSTL, it would be

```
<LI><B>First Name:</B>
<%
Employee employee =
  (Employee)pageContext.findAttribute("employee");
%>
<%= employee.getName().getFirstName() %>
```

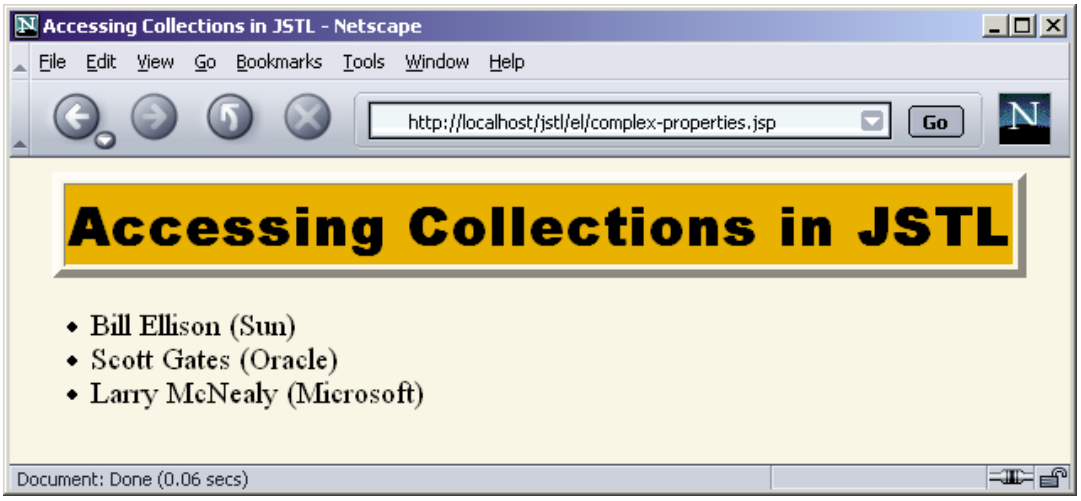
Example: Complex Access

```
<%
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");
pageContext.setAttribute("first", firstNames);
pageContext.setAttribute("last", lastNames);
pageContext.setAttribute("company", companyNames);
%>
```


Example: Complex Access (Continued)

```
<H1>Accessing Collections in JSTL</H1>
<%@ taglib prefix="c"
           uri="http://java.sun.com/jstl/core" %>
<UL>
  <LI><c:out value=
      '${first[0]} ${last[0]} (${company["Ellison"]})'/>
  <LI><c:out value=
      '${first[1]} ${last[1]} (${company["Gates"]})'/>
  <LI><c:out value=
      '${first[2]} ${last[2]} (${company["McNealy"]})'/>
</UL>
```

Example: Complex Access (Results)



The JSTL EL: Implicit Objects

- **pageScope**
  - pageScope.name returns pageContext.getAttribute("name")
- **requestScope**
  - pageScope.name returns request.getAttribute("name")
- **sessionScope**
  - sessionScope.name returns session.getAttribute("name")
- **applicationScope**
  - applicationScope.name returns application.getAttribute("name")
- **pageContext**
  - The PageContext object; properties interpreted normally

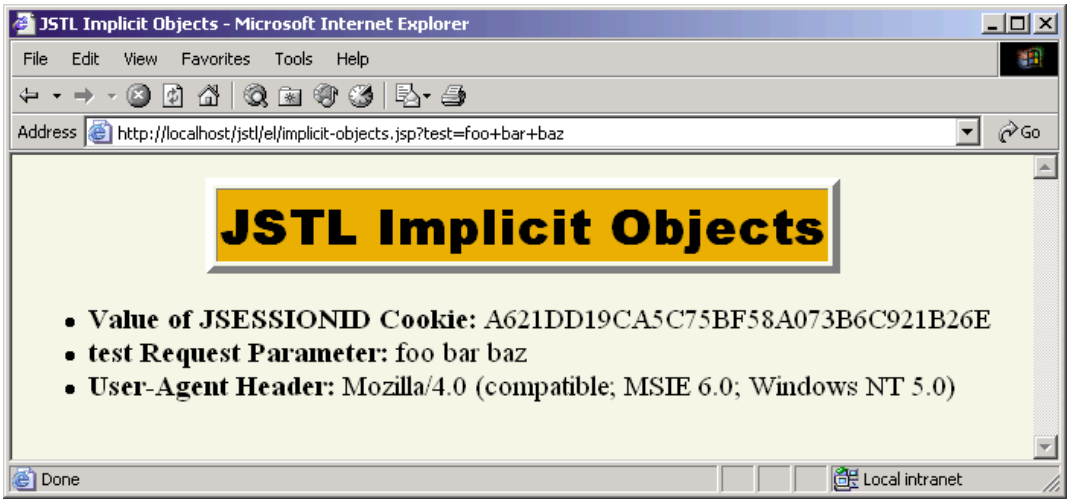
The JSTL EL: Implicit Objects (Continued)

- **param and paramValues**
  - param.name returns request.getParameter("name")
  - paramValues.name returns array of parameters
- **header and headerValues**
  - header.name returns request.getHeader("name")
  - headerValues.name returns array of headers
- **cookie**
  - cookie.name returns the element of request.getCookies whose name is `name`
  - Returns Cookie object; use cookie.name.value for value
- **initParam**
  - initParam.name returns getServletContext().getInitParameter("name")

Implicit Objects: Example

```
<UL>
  <LI><B>Value of JSESSIONID Cookie:</B>
      <c:out value="${cookie.JSESSIONID.value}"/>
  <LI><B>test Request Parameter:</B>
      <c:out value="${param.test}"/>
  <LI><B>User-Agent Header:</B>
      <%-- Using [] syntax because of dash in name!--%>
      <c:out value='${header["User-Agent"]}'/>
</UL>
```

Implicit Objects: Results



The JSP Expression Language: Operators

- **Arithmetic Operators**
  - +, -, *, /, div, %, mod
- **Relational Operators**
  - ==, eq, !=, ne, <, lt, >, gt, <=, le, >=, ge
- **Logical Operators**
  - &&, and, ||, or, !, not
- **Empty Test**
  - empty (tests for null, empty String, empty array, empty List, empty Map)
- **Purpose of Operators**
  - To simplify conditional expressions
  - Examples postponed until conditional evaluation tags

Looping Tags: Summary

- **Looping with explicit numeric values**

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

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

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

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

Looping Tags: Motivation

- **JSP without JSTL**

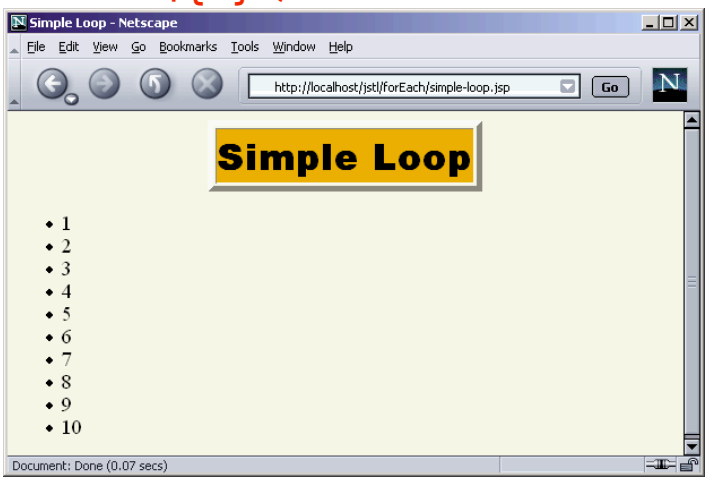
```
<UL>
<%
for(int i=0; i<messages.length; i++) {
  String message = messages[i];
%>
<LI><%= message %>
<% } %>
</UL>
```

- **JSP with JSTL**

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

Looping with Simple Numeric Values

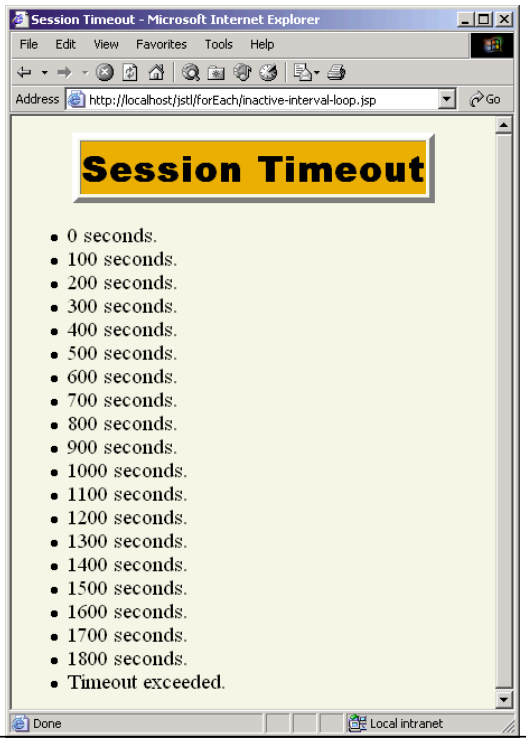
```
<%@ taglib prefix="c"
           uri="http://java.sun.com/jstl/core" %>
<UL>
<c:forEach var="i" begin="1" end="10">
  <LI><c:out value="${i}"/>
</c:forEach>
</UL>
```



Looping with a Designated Step Size

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

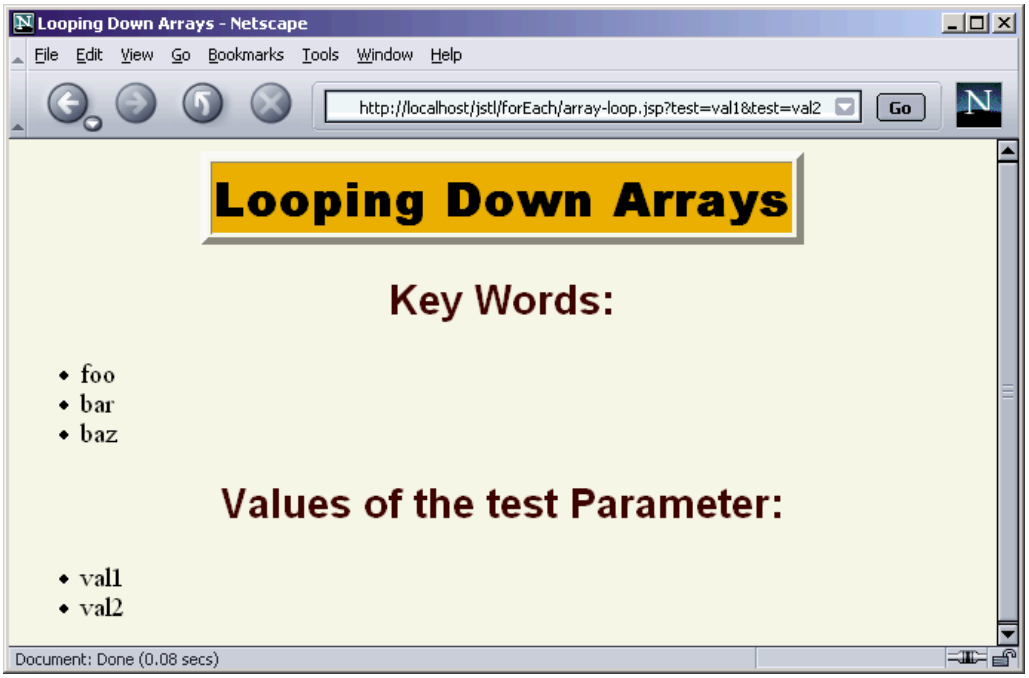
Looping with a Designated Step Size (Results)



Looping Down Arrays

```
<% String[] words = { "foo", "bar", "baz"};
   pageContext.setAttribute("words", words); %>
<%@ taglib prefix="c"
           uri="http://java.sun.com/jstl/core" %>
<H2>Key Words:</H2>
<UL>
<c:forEach var="word"
           items="${words}">
  <LI><c:out value="${word}"/>
</c:forEach>
</UL>
<H2>Values of the test Parameter:</H2>
<UL>
<c:forEach var="val"
           items="${paramValues.test}">
  <LI><c:out value="${val}"/>
</c:forEach>
</UL>
```

Looping Down Arrays (Results)



Looping Down Comma-Delimited Strings

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

Looping Down Comma-Delimited Strings (Results)

Looping Down Arbitrarily-Delimited Strings

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

Conditional Evaluation Tags

- One choice: `if`

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

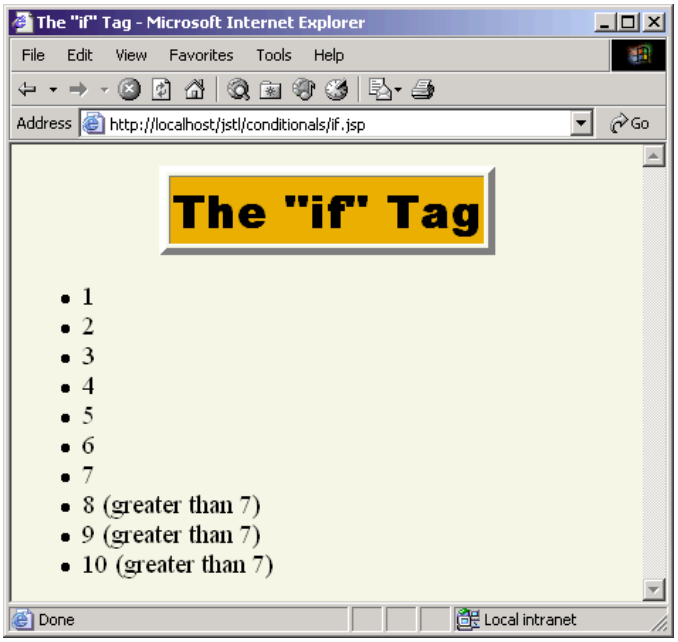
- Lots of choices: `choose`

```
<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>
```


The "if" Tag

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

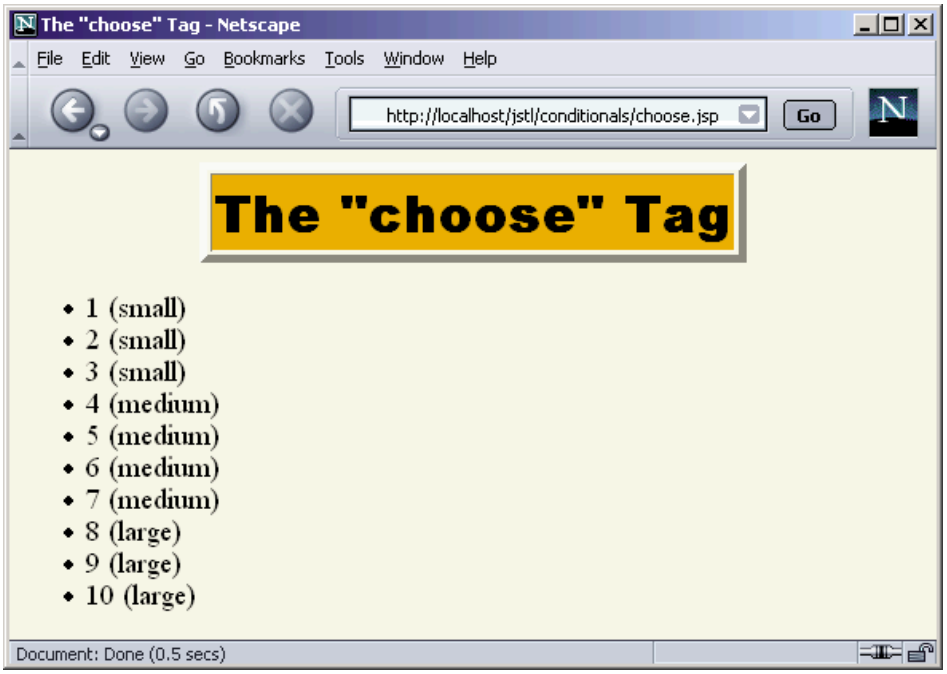
The "if" Tag (Results)



The "choose" Tag

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

The "choose" Tag (Results)

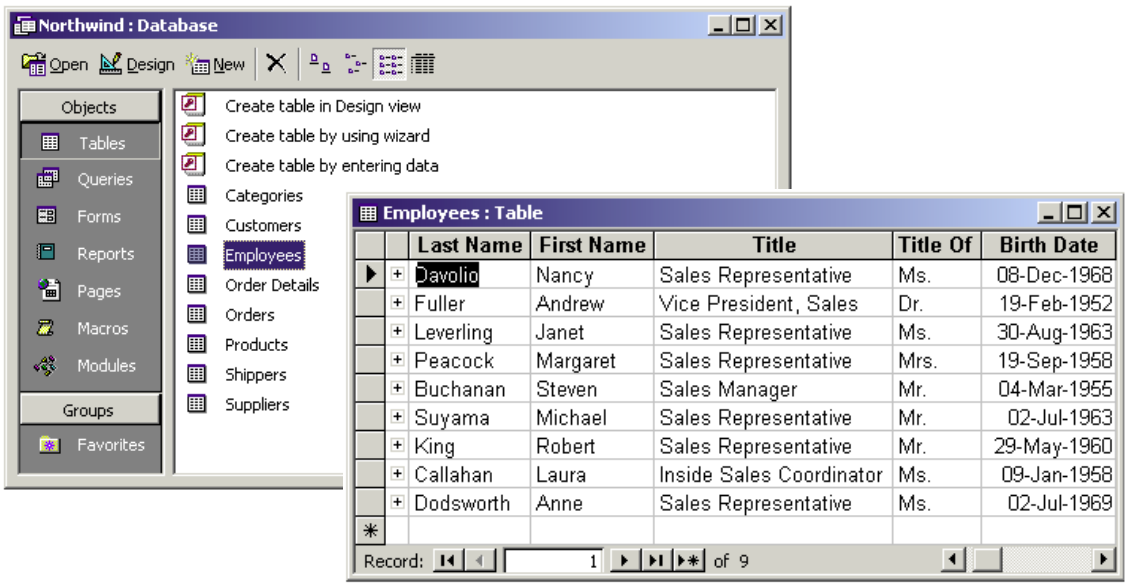


Database Access Tags

- **<sql:setDataSource>**
  - Specifies data source (can also be set by config settings)
- **<sql:query>**
  - Queries database and stores ResultSet in variable
  - Warning: this usage violates rule of keeping business logic out of presentation layer. Instead, do database access in servlet and pass results to JSP via MVC.
- **<sql:update>**
- **<sql:param>**
- **<sql:dateParam>**
- **<sql:transaction>**
  - Performs the enclosed <sql:query> and <sql:update> actions as a single transaction

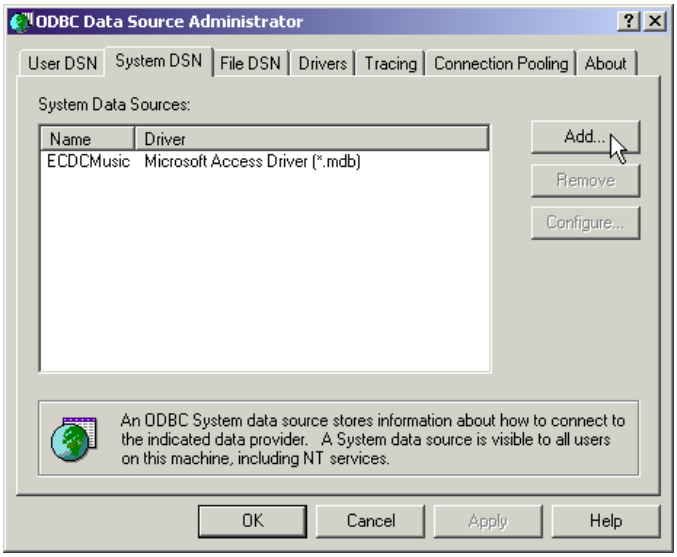
Aside: The Microsoft Access Northwind Database

- **Database that comes preinstalled with Microsoft Office**



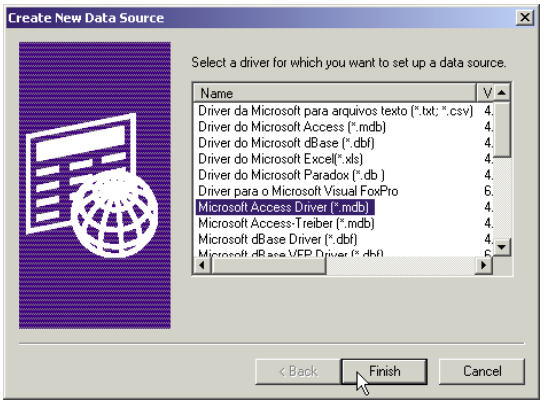
Using Microsoft Access via ODBC

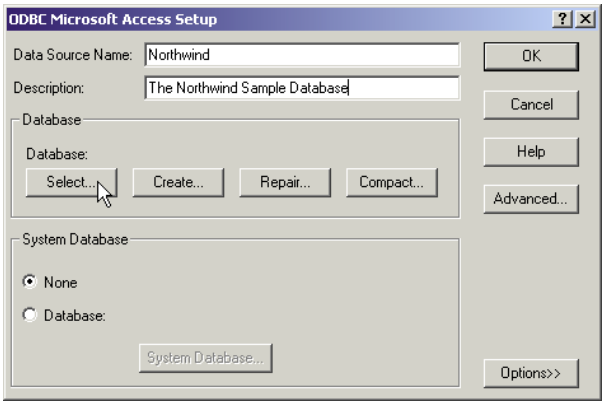
- **Click Start, Settings, Control Panel, Administrative Tools, Data Sources, System DSN, and select Add**



Using Microsoft Access via ODBC (Continued)

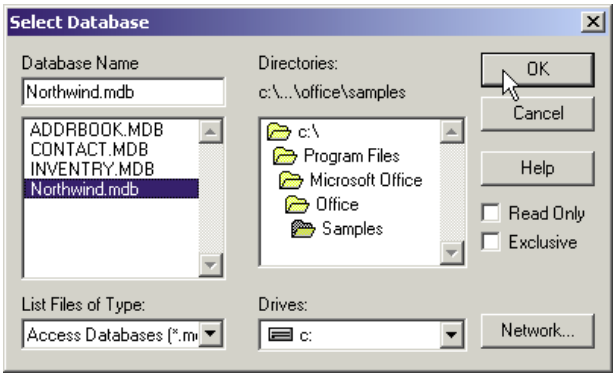
- **Select Microsoft Access Driver, Finish, type a name under Data Source Name, and hit Select**

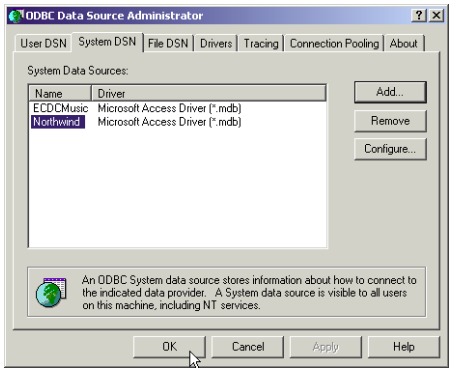




Using Microsoft Access via ODBC (Continued)

- **Navigate to the Samples directory of MS Office, select Northwind.mdb, hit OK, then hit OK in following two windows**





Using Microsoft Access via ODBC (Continued)

- **Driver**
  - sun.jdbc.odbc.JdbcOdbcDriver
    - Comes bundled with JDK
- **URL**
  - jdbc:odbc:Northwind
    - Local access only; for testing. Not for serious applications.
- **Username**
  - Empty string
- **Password**
  - Empty string

sql:setDataSource

- **You can set a data source globally via configuration settings or application-scoped variables.**
  - Preferred approach in real applications
- **Or, you can set it on a specific page**

```
<%@ taglib prefix="sql"
           uri="http://java.sun.com/jstl/sql" %>
<sql:setDataSource
  driver="sun.jdbc.odbc.JdbcOdbcDriver"
  url="jdbc:odbc:Northwind"
  user=""
  password=""/>
```

sql:query

- **Form 1: use the sql attribute**

```
<sql:query var="results" sql="SELECT * FROM …"/>
```

- **Form 2: put the query in the body of the tag**

```
<sql:query var="results">
  SELECT * FROM …
</sql:query>
```

- **Options**
  - dataSource
  - maxRows
  - startRow
- **Caution**
  - Embedding SQL directly in JSP may be hard to maintain.

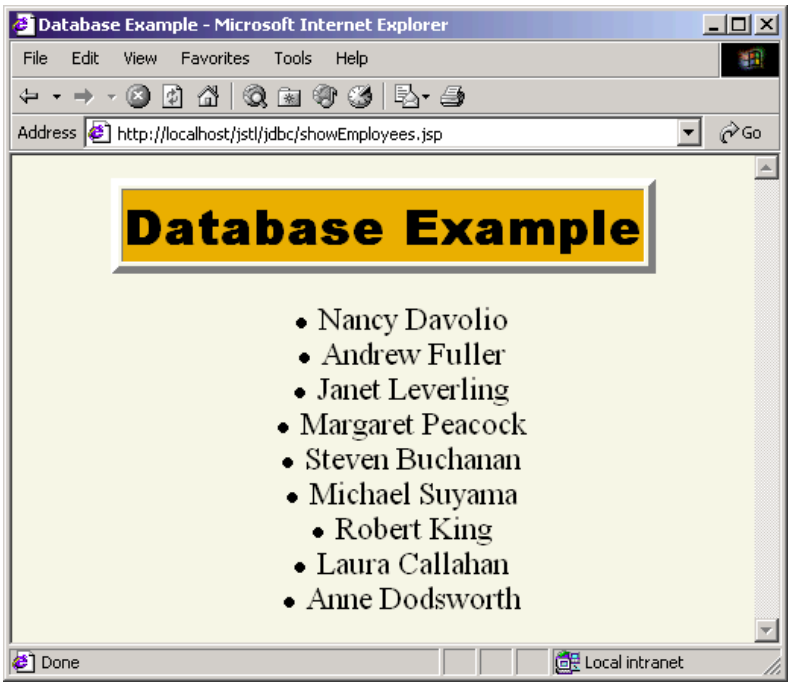
Simple Example

```
<%@ taglib prefix="c"
           uri="http://java.sun.com/jstl/core" %>
<%@ taglib prefix="sql"
           uri="http://java.sun.com/jstl/sql" %>
<sql:setDataSource
  driver="sun.jdbc.odbc.JdbcOdbcDriver"
  url="jdbc:odbc:Northwind"
  user=""
  password=""/>
```

Simple Example (Continued)

```
<sql:query var="employees">
  SELECT * FROM employees
</sql:query>

<UL>
<c:forEach var="row" items="${employees.rows}">
  <LI><c:out value="${row.firstname}"/>
      <c:out value="${row.lastname}"/>
</c:forEach>
</UL>
```


Simple Example: Results



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>

Formatting Tags

- **<fmt:formatNumber>**
  - Formats a numeric value as a number, currency value, or percent, in a locale-specific manner
- **<fmt:parseNumber>**
  - Reads string representations of number, currency value, or percent
- **<fmt:formatDate>**
- **<fmt:parseDate>**
- **<fmt:timeZone>**
- **<fmt:setTimeZone>**

Internationalization (I18N) Tags

- **<fmt:setLocale>**
  - Sets Locale
- **<fmt:setBundle>**
- **<fmt:bundle>**
- **<fmt:message>**
  - Retrieves localized message from a resource bundle
- **<fmt:param>**

XML Manipulation Tags

- **Core**
  - <x:parse>
- **XPath**
  - <x:if>
  - <x:choose>
  - <x:when>
  - <x:otherwise>
  - <x:forEach>
- **XSLT**
  - <x:transform>
  - <x:param>

Summary

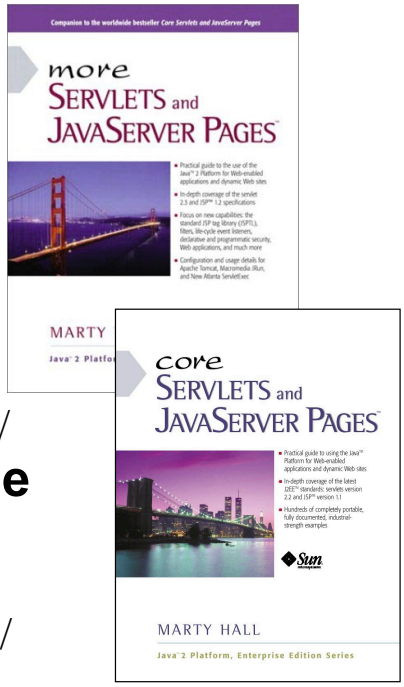
- **JSTL is standardized, but not a standard part of JSP 1.2**
  - It must be downloaded and installed separately
- **Supports a concise expression language**
  - Lets you access bean properties and implicit objects in shorthand manner
  - Is standard part of JSP 2.0
- **Looping tags**
  - Explicit numeric values
  - Arrays, collections, maps, strings, etc.
- **Conditional evaluation tags**
  - Single options
  - Multiple options

Summary (Continued)

- **Database Access Tags**
  - sql:setDataSource to specify a database
  - sql:query to perform query
  - Loop down results using iteration tags
- **Other tags**
  - Handling URLS
  - Internationalizing pages
  - Formatting strings
  - Manipulating XML

More Information

- ***More Servlets & JSP***
  - http://www.moreservlets.com
  - Site includes info on servlet and JSP training courses
- ***Core Servlets & JSP***
  - Prequel to *More Servlets & JSP*
  - http://www.coreservlets.com
- **Servlet home page**
  - http://java.sun.com/products/servlet/
- **JavaServer Pages home page**
  - http://java.sun.com/products/jsp/
- **JSTL docs**
  - http://jakarta.apache.org/taglibs/doc/standard-doc/intro.html



Questions?

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