Corporate Profile

Session 6:

JSTL

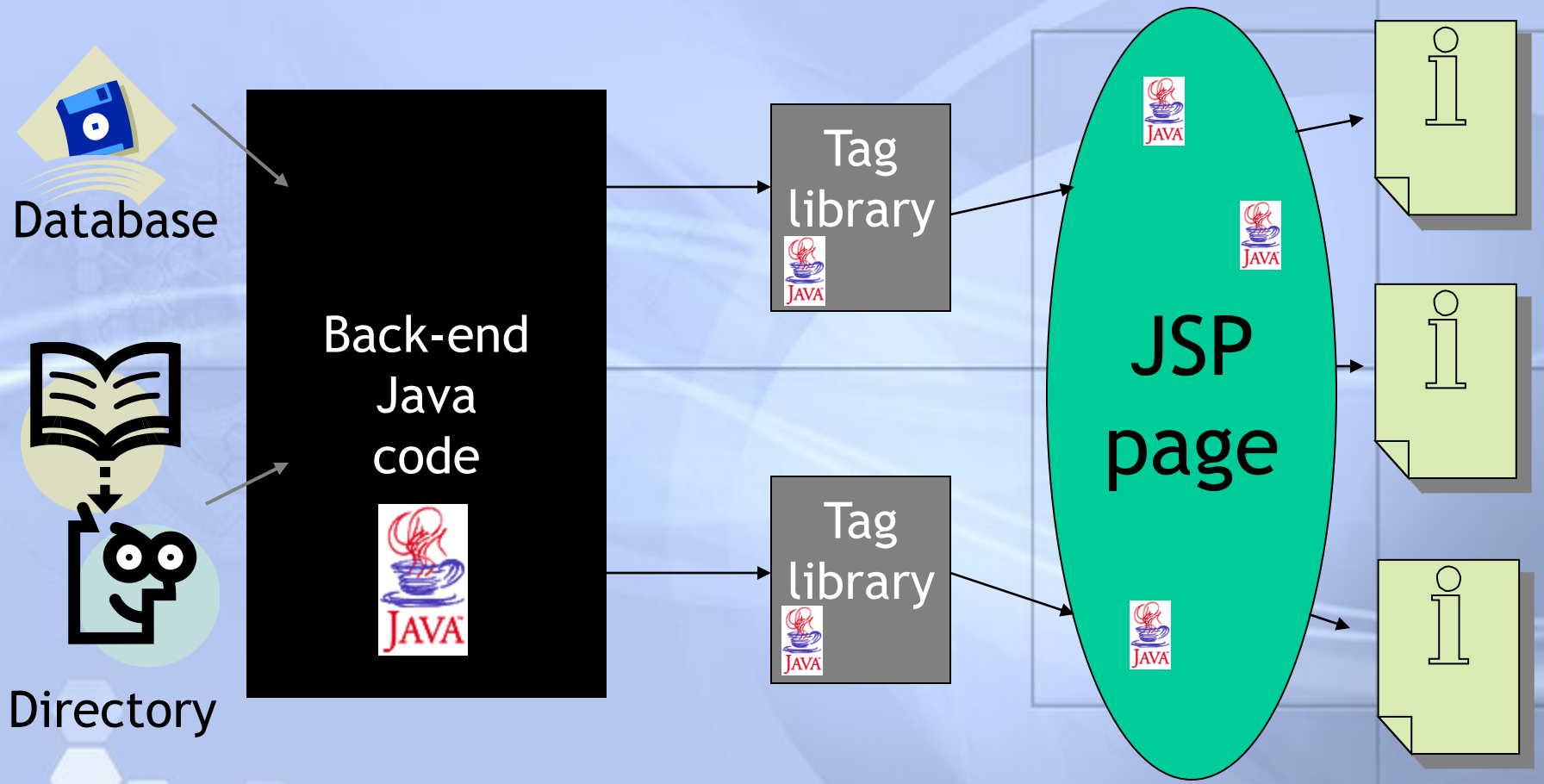
Contents

- What is JSTL?
- JSTL Core Tags
  - Control
  - Output
  - Loop
  - URL
  - Exception
- Examples

What is a Custom Tag Library?

- collection of custom actions (tags) made available in a JSP page
- standardized to be portable between different JSP containers
- make it possible to write JSP pages without the use of scripting elements (embedded java fragments of code)
  - code is easier to maintain
  - logic is separated from presentation
  - web designer role separated from java developer

Tags: Abstracting logic



What is JSTL

- Created by the Java Community Process as the JSP specification itself
- vendors can offer versions of the JSTL optimized for their container
- Includes:
  - Core
  - Functions
  - Relational Database Access
  - XML processing
  - Internationalization (i18n)

Installing Custom Tag Libraries

- Download **Standard 1.0 Taglib** zip file

  http://jakarta.apache.org/site/downloads/downloads_taglibs.html

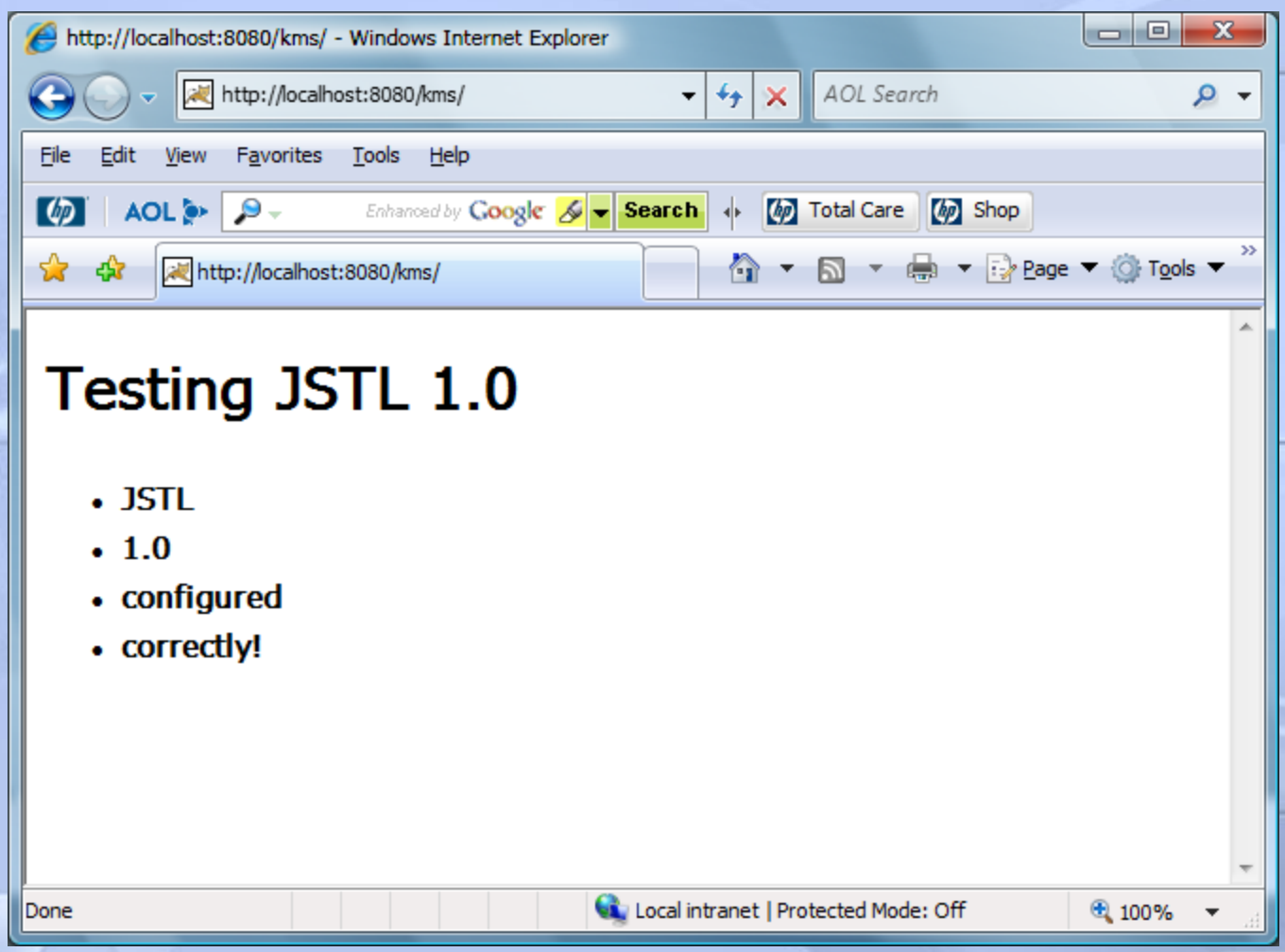
- Unzip into directory of your choice (e.g.,C:\jakarta-taglibs).

- Copy **\lib\jstl.jar** and **\lib\standard.jar** to the **WEB-INF\lib** directory of your Web application

Example : JSP with jstl

```
<html>
    <%
        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>
</html>
```

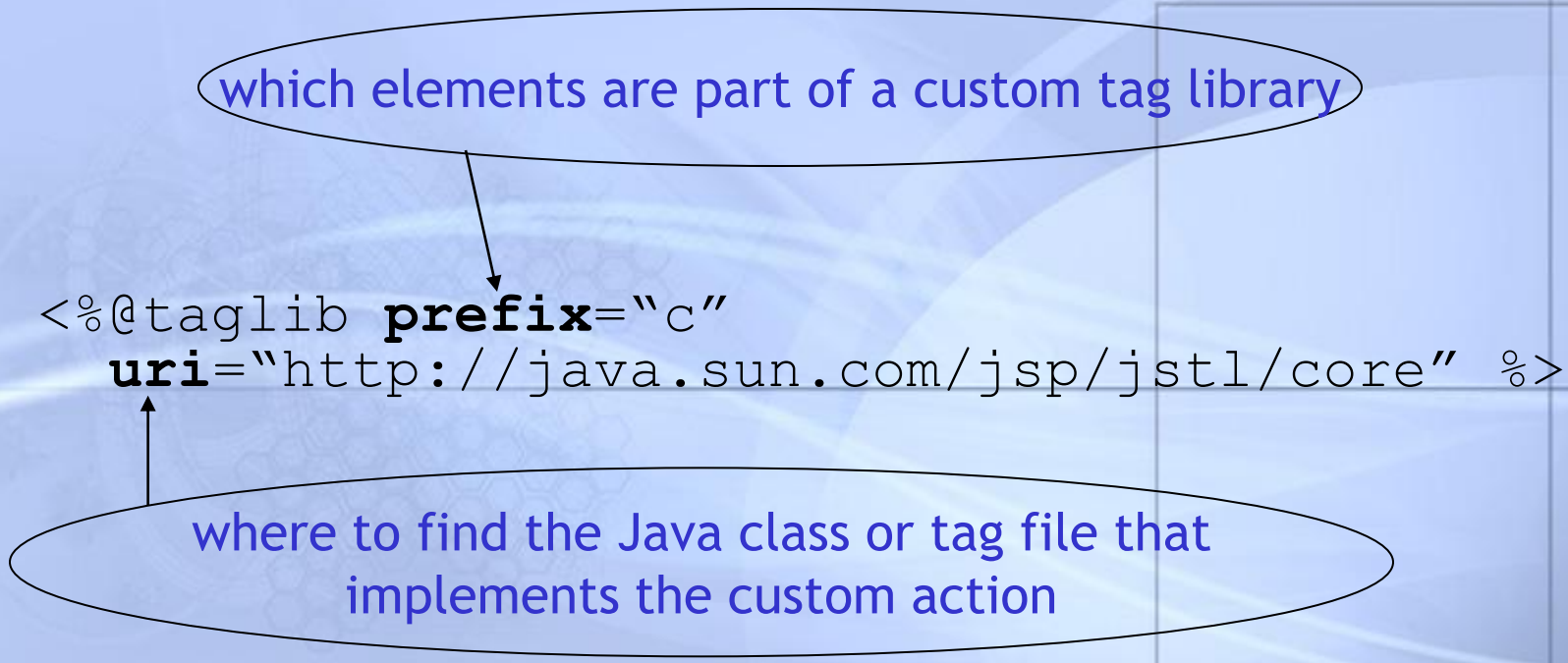
Installation Test



Example : JSP without jstl

```
<UL>
<%
    String[] messages={"JSTL", "1.0","configured", "correctly!"};
    for(int i=0; i<messages.length; i++) {
        String message = messages[i];
%>
<LI><%= message %>
<% } %>
</UL>
```

Declaring a Custom Tag Library



JSTL URIs and Default Prefixes

Library	URI	Prefix
Core	http://java.sun.com/jsp/jstl/core	c
XML Processing	http://java.sun.com/jsp/jstl/xml	x
Formatting	http://java.sun.com/jsp/jstl/fmt	fmt
Database Access	http://java.sun.com/jsp/jstl/sql	sql
Functions	http://java.sun.com/jsp/jstl/functions	fn

Declaring a Custom Tag Library

- A URI is a string that tells the container how to locate the TLD file for the library, where it finds the tag file name or java class for all actions in the library
- when the web server is started
  - it locates all TLD files,
  - for each of them gets the default URI
  - create a mapping between the URI and the TLD
- a default URI must be a globally unique string

Using Actions from a Tag Library

- The syntax:

```
<prefix:action-name attr1="value1" attr2="value2">
  action_body
</prefix:action-name>
```

- or (with no body)

```
<prefix: action-name attr1="value1" attr2="value2"/>
```

The Standard Tag Library Core

- actions for control-flow
- URL manipulation
- importing resources
- general-purpose tasks

Core Library Tags - Listing

- **Flow control**
  - <c:if>
  - <c:choose>
    - <c:when>
    - <c:otherwise>
  - <c:forEach>
  - <c:forToken>
- **Variable support**
  - <c:set>
  - <c:remove>
- **URL**
  - <c:param>
  - <c:redirect>
  - <c:import>
  - <c:url>
- **Output**
  - <c:out>
- **Exception handling**
  - <c:catch>

Output Tags

<c:out>

- adds the value of the evaluated expression to the response buffer
- Syntax 1:

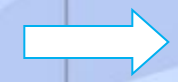
```
<c:out value="expression"
       [excapeXml="true|false"]
       [default="defaultExpression"] />
```

- Syntax 2:

```
<c:out value="expression"
       [excapeXml="true|false"]>
    defaultExpression
</c:out>
```

Output

```
<c:out value="100" />
<c:out value="${price}" />
```



```
${100}
${price}
```

- You want to use <c:out> if
  - *escapeXML* = true
  - *value* is a Java.io.Reader object

Corporate Profile

Flow Control Tags

<c:choose>

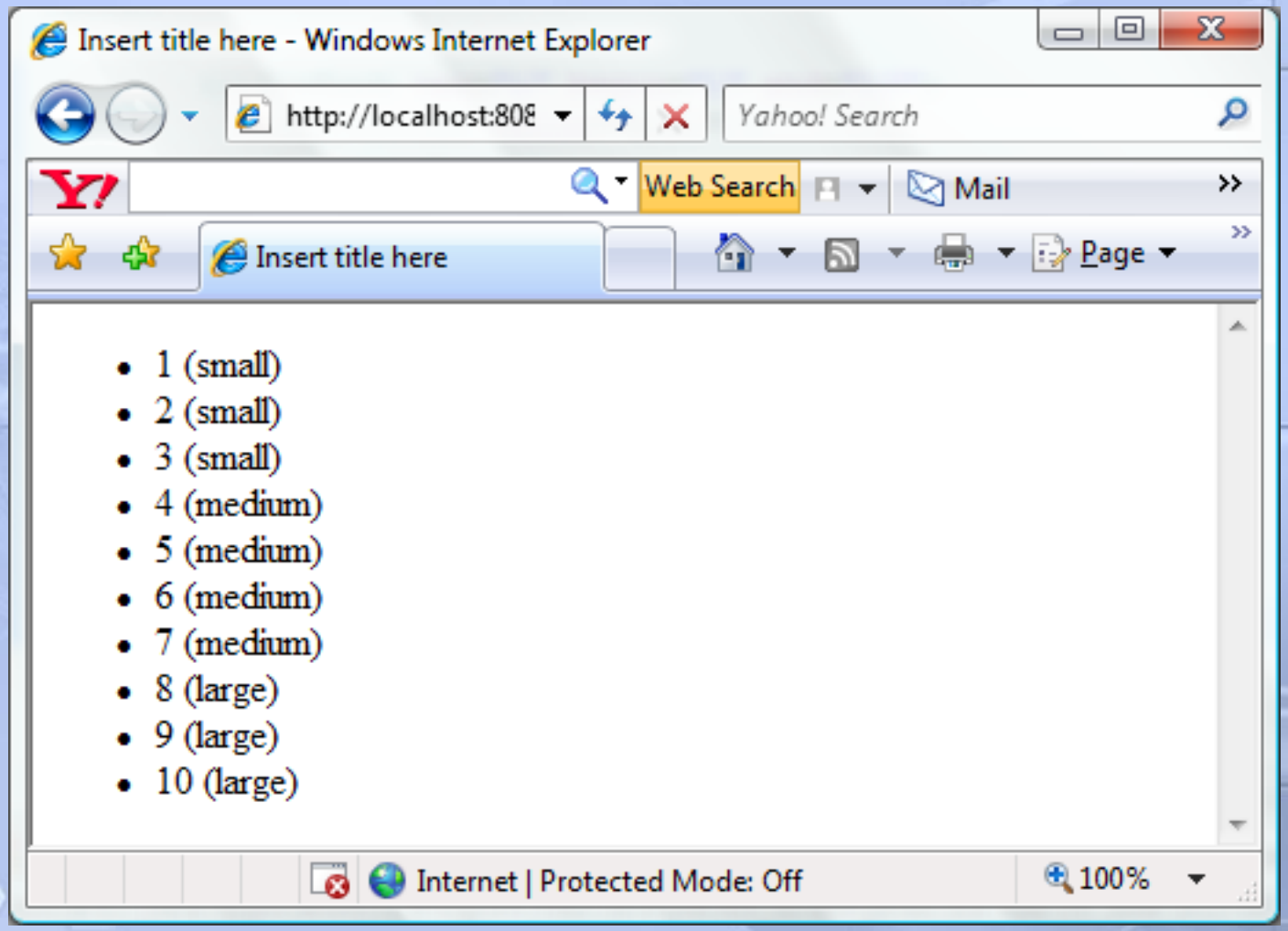
- only the first <c:when> action that evaluates to true is processed
- if no <c:when> evaluates to true <c:otherwise> is processed, if exists
- Syntax:

```
<c:choose>
   1 or more <c:when> tags and optionally a
  <c:otherwise> tag
</c:choose>
```

Example **<c:choose>**

```
<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>
```

Example <c:choose>



<c:forEach>

- evaluates its body once for each element in a collection
  - java.util.Collection
  - java.util.Iterator
  - java.util.Enumeration
  - java.util.Map
  - array of Objects or primitive types
- Syntax:

```
<c:forEach items="collection" [var="var"]
  [varStatus="varStatus"] [begin="startIndex"]
  [end="endIndex"] [step="increment"]>
  JSP elements
</c:forEach>
```

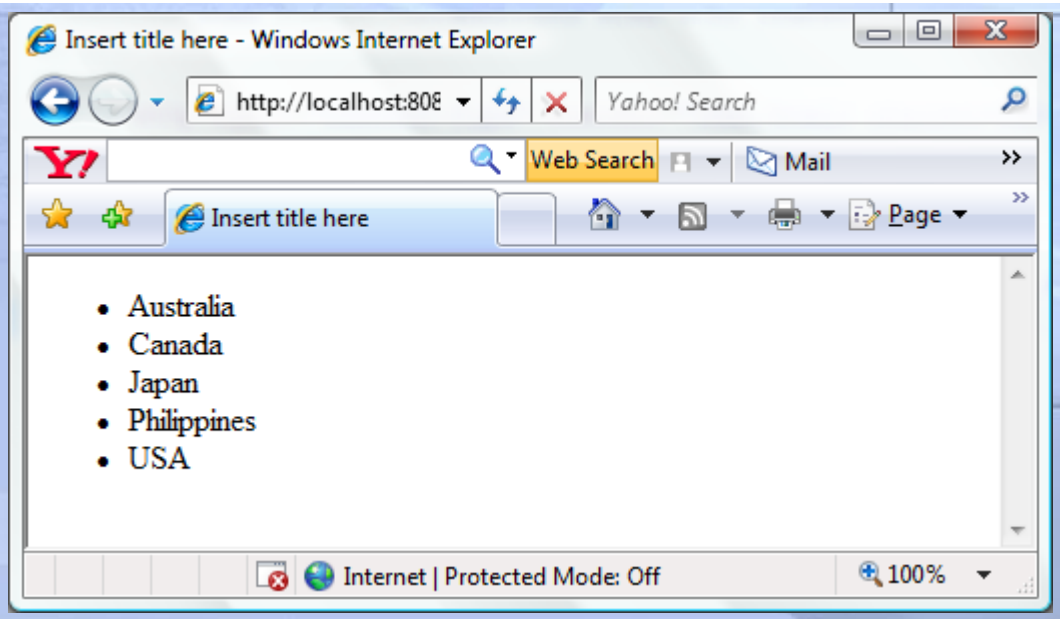
Example **forEach**

```
<c:forEach var="i" begin="1" end="5" step="1">
    <c:out value="${i}"/> <br/>
</c:forEach>
```

```
<%
        String arr[]={"Monday","Tuesday","Wednesday"};
        Vector V=new Vector();
        V.addAll(Arrays.asList(arr));
        pageContext.setAttribute("vec", V);
%>

<c:forEach var="day" items="${vec}">
        <c:out value="${day}"/> <br/>
</c:forEach>
```

Example

```
<UL>
<c:forEach var="country" items="Australia,Canada,Japan,Philippines,USA">
    <LI> <c:out value="${country}"/>
</c:forEach>
</UL>
```



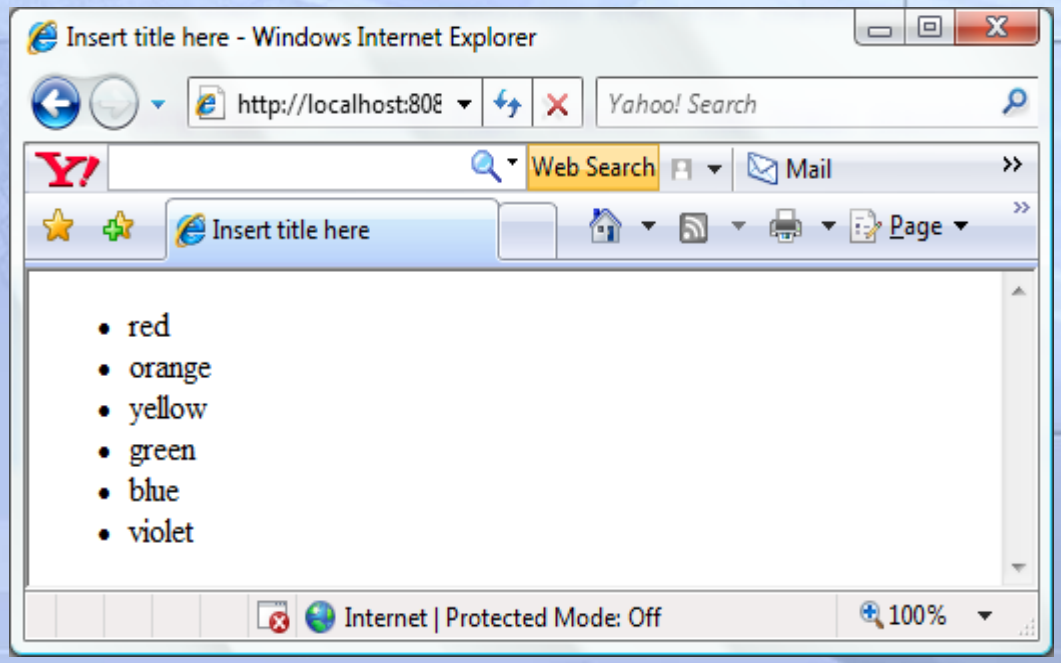
<c:forTokens>

- evaluates its body once for each token n a string delimited by one of the delimiter characters
- Syntax:

```
<c:forTokens items="stringOfTokens"
  delims="delimiters" [var="var"]
  [varStatus="varStatus"] [begin="startIndex"]
  [end="endIndex"] [step="increment"]>
  JSP elements
</c:forTokens>
```

Example : forTokens

```
<c:forTokens   var="color"
                    items="(red (orange) yellow)(green)((blue) violet)"
                    delims="()">
    <LI><c:out value="${color}"/>
</c:forTokens>
```



<c:if>

- evaluates its body only if the specified expression is true
- Syntax1:

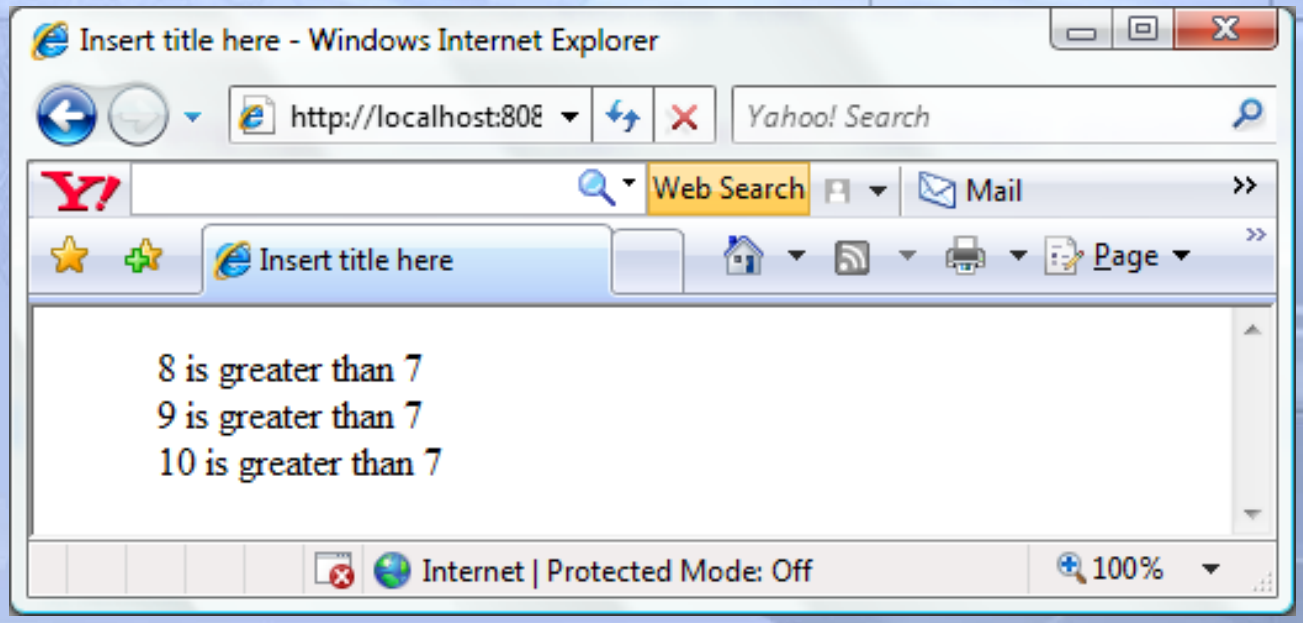
```
<c:if test="booleanExpression" var="var"
  [scope="page|request|session|application"] />
```

- Syntax2:

```
<c:if test="booleanExpression" >
  JSP elements
</c:if>
```

Example : if

```
<c:forEach var="i" begin="1" end="10">
    <c:if test="${i>7}">
        <c:out value="${i} is greater than 7" /> <br/>
    </c:if>
</c:forEach>
```



Corporate Profile

ExceptionTag

<c:catch>

- catches an exception thrown by JSP elements in its body
- the exception can optionally be saved as a page scope variable
- Syntax:

```
<c:catch>
   JSP elements
</c:catch>
```

Corporate Profile

URL Tags

<c:url>

- applies encoding and conversion rules for a relative or absolute URL
- Rules:
  - URL encoding of parameters specified in <c:param> tags
  - context-relative path to server-relative path
  - add session Id path parameter for context- or page-relative path
- Syntax:

```
<c:url   url="url"   [context="externalContextPath"]
  [var="var"]
  scope="page|request|session|application" >
  <c:param> actions
</c:url>
```

<c:import>

- imports the content of an internal or external resource
- like <jsp:include> for internal resources
- however, allows the import of external resources as well (from a different application OR different web container)
- Syntax:

```
<c:import url="url" [context="externalContext"]
  [var="var"]
  [scope="page|request|session|application"]
  [charEncoding="charEncoding"]>
   Optional <c:param> actions
</c:import>
```

Example: <c:import> & <c:url>

```
<c:import url="/books.xml" var="something" />
<x:parse    doc="${something}"
            var="booklist"
            scope="application" />
```

```
<c:url var="url" value="/catalog" >
    <c:param name="Add" value="${bookId}" />
</c:url>
<a href="${url}">Get book</a>
```

<c:param>

- nested action in <c:import> <c:redirect> <c:url> to add a request parameter
- Syntax 1:

```
<c:param name="parameterName"
  value="parameterValue" />
```

- Syntax 2:

```
<c:param name="parameterName">
  parameterValue
</c:param>
```

<c:redirect>

- sends a redirect response to a client telling it to make a new request for the specified resource
- Syntax 1:

```
<c:redirect url="url"
  [context="externalContextPath"] />
```

- Syntax 2:

```
<c:redirect url="url"
  [context="externalContextPath"] > <c:param>
  tags
</c:redirect>
```

Corporate Profile

Variable Support

<c:set>

- sets a scoped variable or a property of a target object to the value of a given expression
- The target object must be of type java.util.Map or a Java Bean with a matching setter method
- Syntax 1:

```
<c:set value="expression" target="beanOrMap"
  property="propertyName" />
```

- Syntax 2:

```
<c:set value="expression" var="var"
  scope="page|request|session|application" />
```

<c:remove>

- removes a scoped variable
- if no scope is specified the variable is removed from the first scope it is specified
- does nothing if the variable is not found
- Syntax 1:

```
<c:remove var="var"
  [scope="page|request|session|application"] />
```

JSTL Usage Examples

- ShoppingCart.jsp
- Login.jsp and Members.jsp

Branch Tags

```
<c:if test="${!cart.notEmpty}"> The cart is empty.</c:if>
```

```
<c:choose>
    <c:when test="${!cart.notEmpty}">
        The cart is emtpy.
    </c:when>
    <c:otherwise>
        <%-- do something --%>
    </c:otherwise>
</c:choose>
```

Loop Tags

```
<%-- iterator style --%>
<c:forEach items="${cart.items}" var="i">
      ${i} <br>
</c:forEach>

<%-- for loop style --%>
<c:forEach begin="0" end="${cart.size}" step="1"
var="i">
      ${cart.items[i]}
</c:forEach>
```

Set and Remove Scope Variables

In Login.jsp

```
<c:set var="authorized" value="true" scope="session"/>
```

In CheckLogin.jsp

```
    <c:if test="${empty sessionScope.authorized}">
        <c:redirect url="Login.jsp" />
    </c:if>
```

In Logout.jsp

```
    <c:remove var="authorized" scope="session" />
```

JSTL Implicit Scope 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

JSTL Implicit Scope Objects

- **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")*

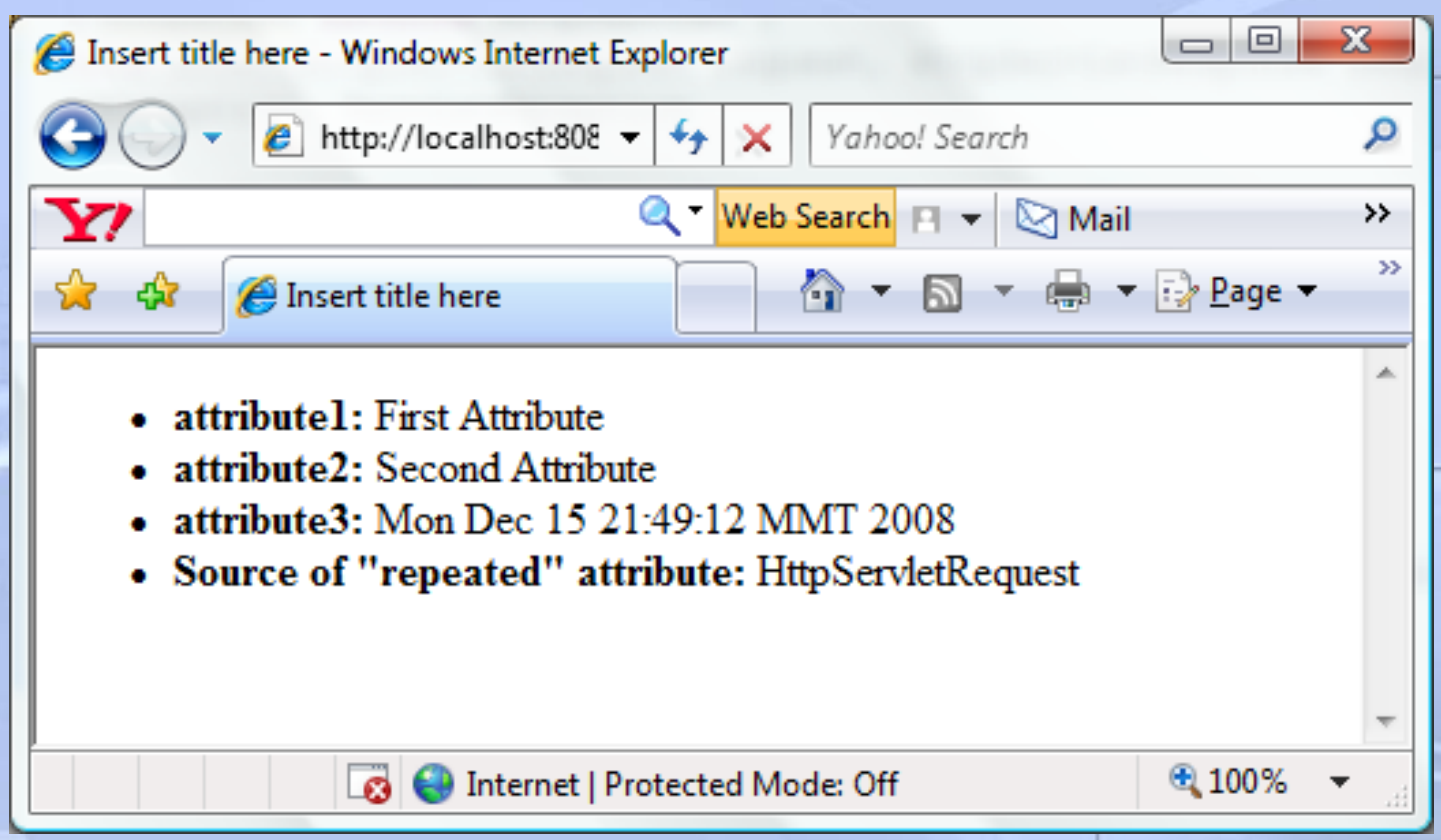
Example : Scope Test

```
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");

RequestDispatcher dispatcher =getServletContext().getRequestDispatcher
                                        ("/ScopeJSP.jsp");
dispatcher.forward(request, response);
```

Example : Scope Test

```
<body>
<%@ 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>
</body>
```

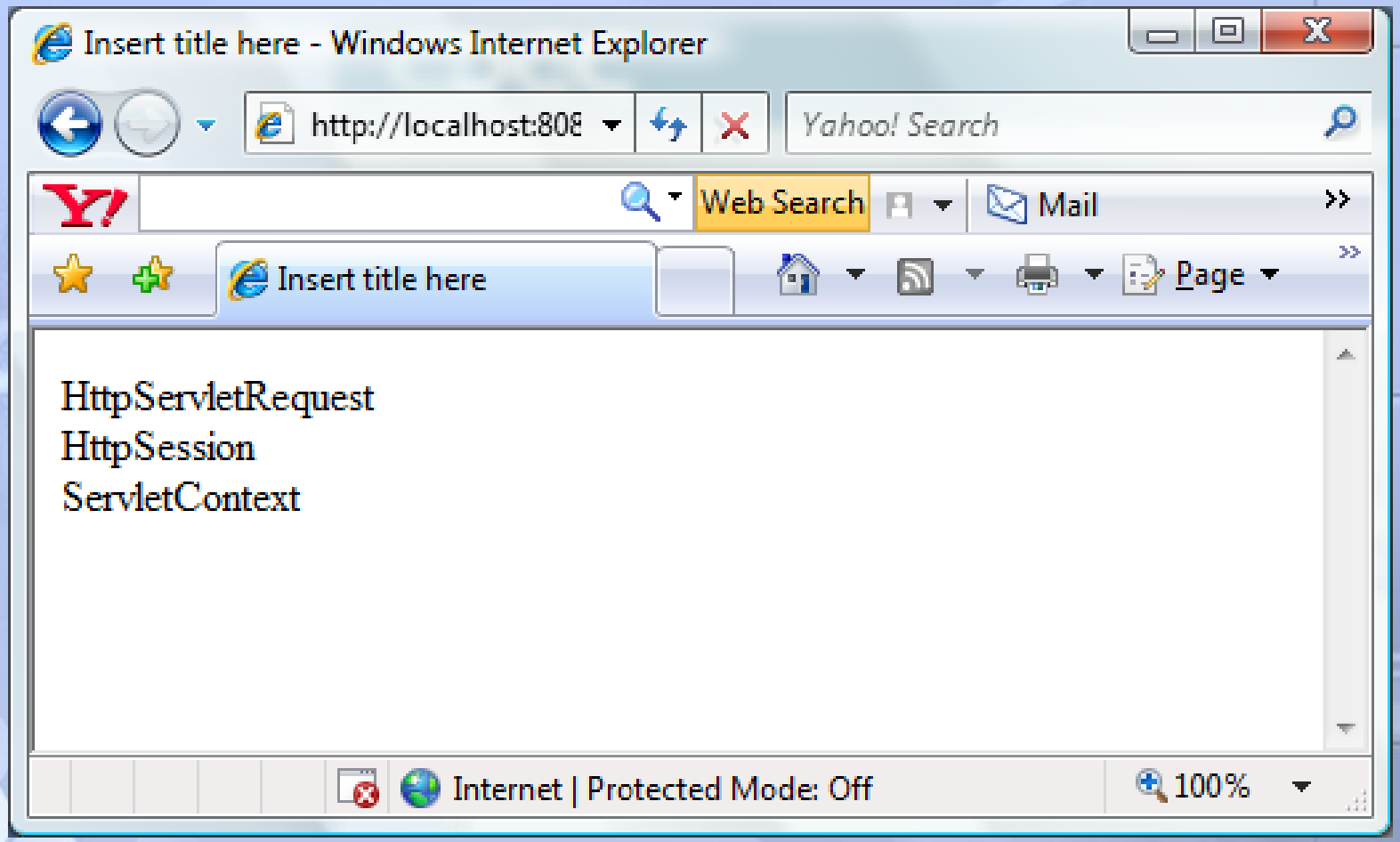
Example


Insert title here - Windows Internet Explorer
http://localhost:808
Yahoo! Search
Web Search
Mail
Insert title here
Page
attribute1: First Attribute
attribute2: Second Attribute
attribute3: Mon Dec 15 21:49:12 MMT 2008
Source of "repeated" attribute: HttpServletRequest
Internet | Protected Mode: Off
100%

Example : Scope Test

```
<body>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<UL>
<LI><c:out value="${repeated}"></c:out>
<LI><c:out value="${sessionScope.repeated}"></c:out>
<LI><c:out value="${applicationScope.repeated}"></c:out>
</UL>
</body>
```

Example


Insert title here - Windows Internet Explorer
http://localhost:808
Yahoo! Search
Web Search
Mail
Insert title here
Page
HttpServletRequest
HttpSession
ServletContext
Internet | Protected Mode: Off
100%

Function Tags

Standard Syntax:

```
<%@ taglib prefix="fn"
 uri="http://java.sun.com/jsp/jstl/functions"
%>
```

XML Syntax:

```
<anyxmlelement
xmlns:fn="http://java.sun.com/jsp/jstl/functions" />
```

JSTL Functions

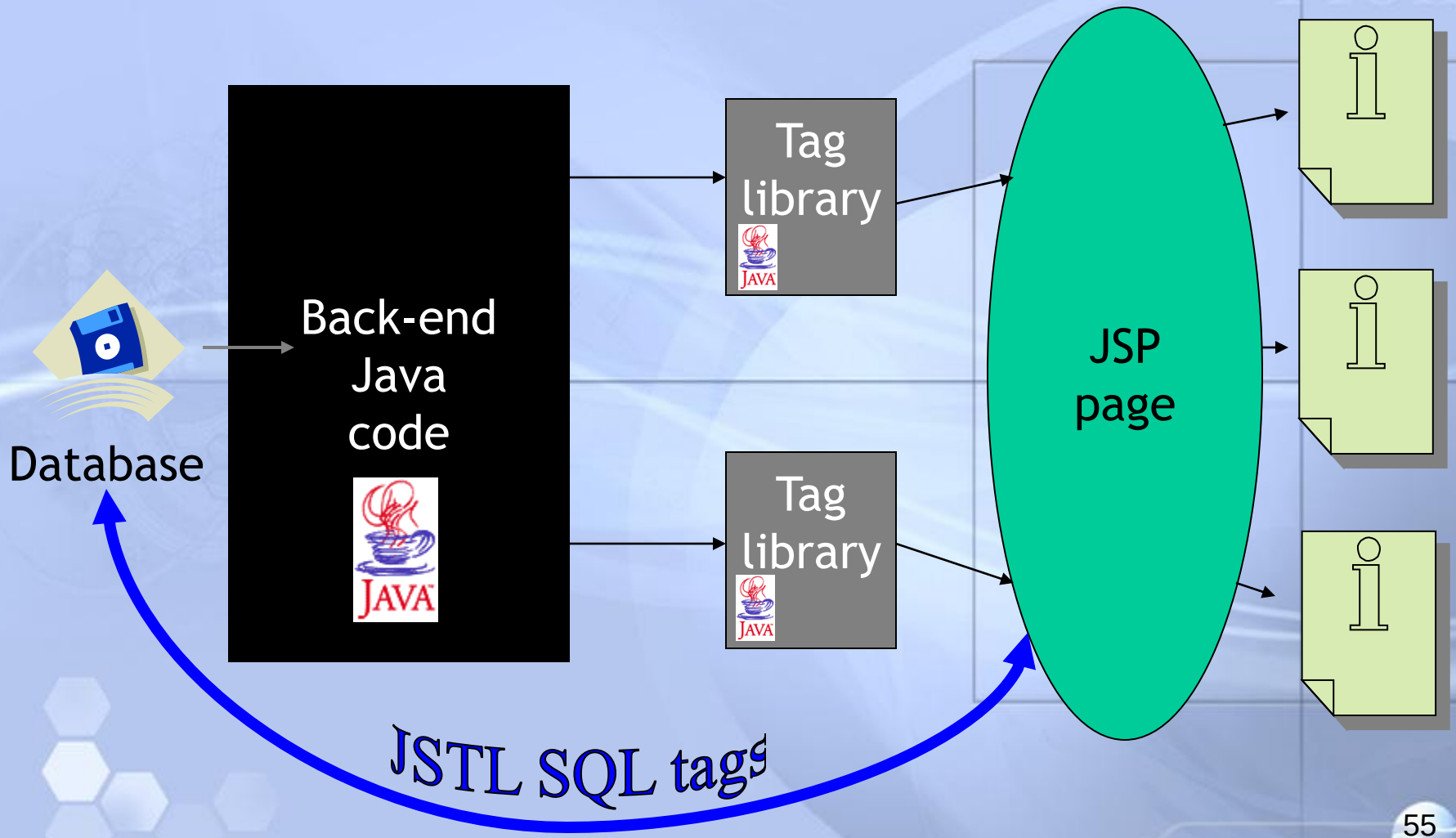
- fn:length()
- fn:contains()
- fn:containsIgnoreCase()
- fn:startWith()
- fn:endsWith()
- fn:indexOf()
- fn:replace()
- fn:trim()
- fn:toUpperCase()
- fn:toLowerCase()
- fn:substring()
- fn:substringAfter()
- fn:substringBefore()
- fn:split()
- fn:join()
- fn:escapeXML()

Example Syntax

```
<c:if test="${fn:contains(name, searchString)}">

${fn:length(shoppingCart.products)}

${fn:indexOf(name, "-")}
```

http://java.sun.com/products/jsp/jstl/1.1/docs/tlddocs/fn/tld-summary.html

SQL tags: the database



JSTL SQL

- sql:transaction
- sql:query
- sql:update
- sql:param
- sql:dateParam
- sql:setDataSource

http://java.sun.com/products/jsp/jstl/1.1/docs/tlddocs/index.html

Example: HelloSQL.jsp

- Data source
- Query
- Results display

sql:setDataSource

- `var` – data source name. Only needed when you have multiple db sources.
- `scope` – scope of the data source
- `driver` – "com.mysql.jdbc.Driver"
- `url`
- `user`
- `password`
- `dataSource`

sql:setDataSource

```
<%@ taglib prefix="sql”    uri="http://java.sun.com/jstl/sql" %>

<sql:setDataSource      driver="sun.jdbc.odbc.JdbcOdbcDriver"
                        url="jdbc:odbc:dataSourceName"
                        user=""
                        password=""/>
```

sql:query

- `var` – name of the result set
- `scope` – scope of the result set
- `sql` – query statement
- `dateSource` – name of the data source
- `startRow`
- `maxRows` – max number of rows in the result set

sql:query Result Set

- javax.servlet.jsp.jstl.sql.Result
  - SortedMap[] getRows()
  - Object[][] getRowsByIndex()
  - String[] getColumnNames()
  - int getRowCount()
  - boolean isLimitedByMaxRows()

sql:query example 1

```
<sql:query var="results" sql="select * from Student"/>


<c:forEach items="${results.rows}" var="row">
    <c:forEach items="${row}” var="col”>
        <c:out value="${col.key}" />
        <c:out value="${col.value}" /> <br/>
    </c:forEach>
  </c:forEach>
```

sql:query example 2

```
<sql:query var="results">
  select * from items where price > 2.00
</sql:query>

<table>
  <c:forEach items="${results.rowsByIndex}" var="row">
  <tr>
    <c:forEach items="${row}" var="col">
        <td>${col}</td>
    </c:forEach>
  </tr>
  </c:forEach>
</table>
```

sql:query example 3

Place holder and <sql:param>

```
<sql:query var="results">

    select * from items where
        price < ? and quantity > ?

  <sql:param value="2.00"/>
  <sql:param value="2"/>

</sql:query>
```

sql:update

- `var` – name of the result variable. Int
  - – number of rows affected by the update
  - – 0 if the update statement doesn't return anything
- `scope`
- `sql`
- `dateSource` – name of the data source

sql:update example

```
<c:if test="${! empty param.setPrice}">

    <sql:update var="r">
        update items set price = ? where name = ?
        <sql:param value="${param.price}"/>
        <sql:param value="${param.name}"/>
    </sql:update>

</c:if>
```

JSTL SQL vs. JDBC

- JSTL SQL
  - Simple application
  - Small relation
  - Straight-forward operations
  - In the final
- JDBC
  - Everything else

Thank You!