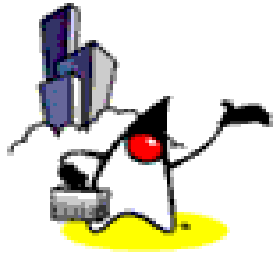
# JSTL (JSP Standard Tag Library)




# Sang Shin

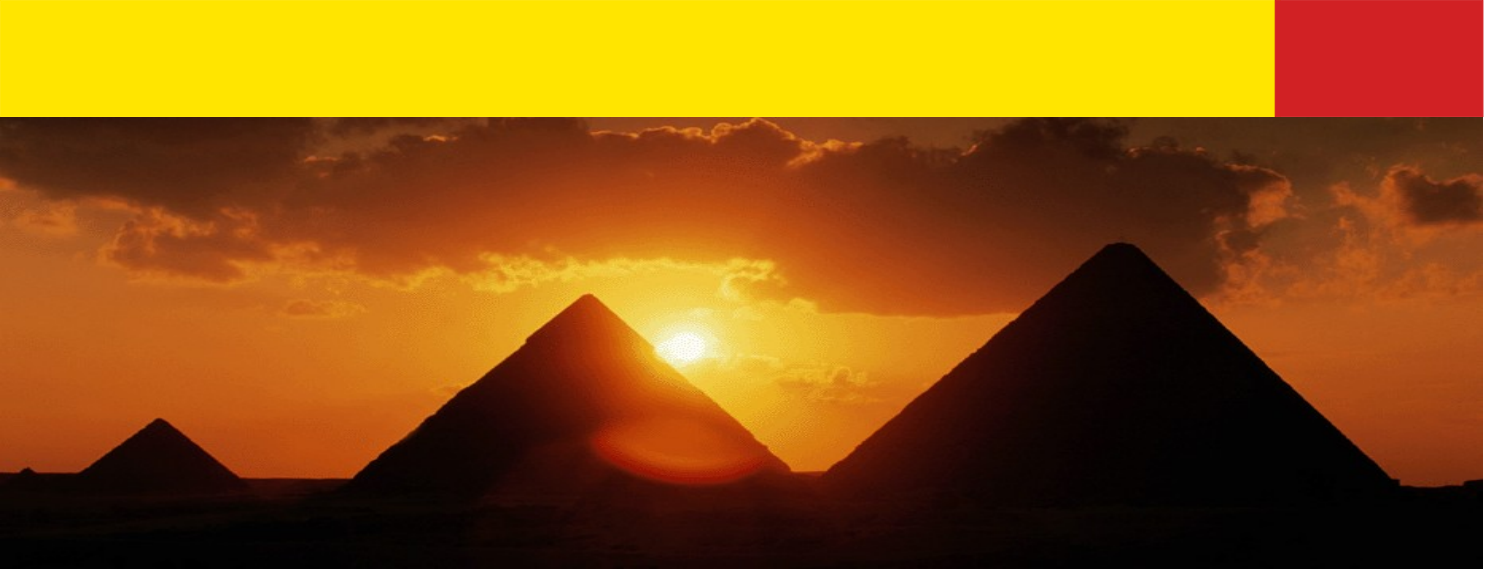
sang.shin@sun.com
www.javapassion.com
Java™ Technology Evangelist
Sun Microsystems, Inc.

# Revision History

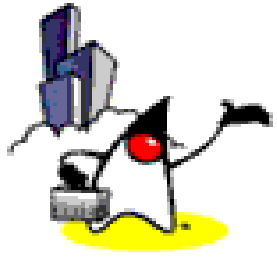
- 11/01/2003: version 1: created by Sang Shin
- Things to do
  - speaker notes need to be polished
  - there are still some topics that are not covered yet

# Agenda

- What is and Why JSTL?
- JSTL Functional areas
  - Core tags
  - Database Access tags
  - XML tags
    - Quick review on XPath
  - Internationalization and Text Formatting tags
  - EL (Expression Language) functions tags



# What is & Why JSTL?



# What is JSTL?

- Standard set of tag libraries
- Encapsulates core functionality common to many JSP applications
  - iteration and conditionals
  - XML
  - database access
  - internationalized formatting
- Likely to evolve to add more commonly used tags in future versions

# Why JSTL?

- You don't have to write them yourself
- You learn and use a single standard set of tag libraries that are already provided by compliant Java EE platforms
- Vendors are likely to provide more optimized implementation
- Portability of your applications are enabled

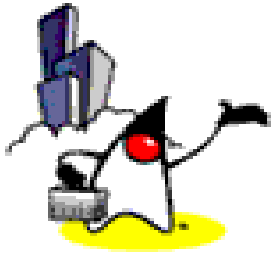
# JSTL Tag Libraries

- Core (prefix: c)
  - Variable support, Flow control, URL management
- XML (prefix: x)
  - Core, Flow control, Transformation
- Internationalization (i18n) (prefix: fmt)
  - Locale, Message formatting, Number and date formatting
- Database (prefix: sql)
  - SQL query and update
- Functions (prefix: fn)
  - Collection length, String manipulation

# Declaration of JSTL Tag Libraries

- Core
  - <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
- XML
  - <%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
- Internationalization (i18n)
  - <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
- Database (SQL)
  - <%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
- Functions
  - <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

# Core Tags
# (Core Actions)



# Core Tags Types (page 1)

- Variable support
  - <c:set>
  - <c:remove>
- Conditional
  - <c:if>
  - <c:choose>
    - <c:when>
    - <c:otherwise>
- Iteration
  - <c:forEach>
  - <c:forTokens>

# Core Tags Types (page 2)

- URL management
  - <c:import>
    - <c:param>
  - <c:redirect>
    - <c:param>
  - <c:url>
    - <c:param>
- General purpose
  - <c:out>
  - <c:catch>

# Variable Support, <c:set>

- Sets the value of an EL variable or the property of an EL variable via "var" attribute
  - in any of the JSP scopes via "scope" attribute
  - page, request, session, application
- If the variable does not already exist, it gets created and saved (in the scope object)
- Variable can be set in 2 different ways
  - <c:set var="foo" scope="session" value="..."/>
    - example: <c:set var="bookId" value="${param.BookID}"/>
  - <c:set var="foo">value to be set</c:set>

# Example: <c:set> definition quoted from ./elsupport/Set.jsp

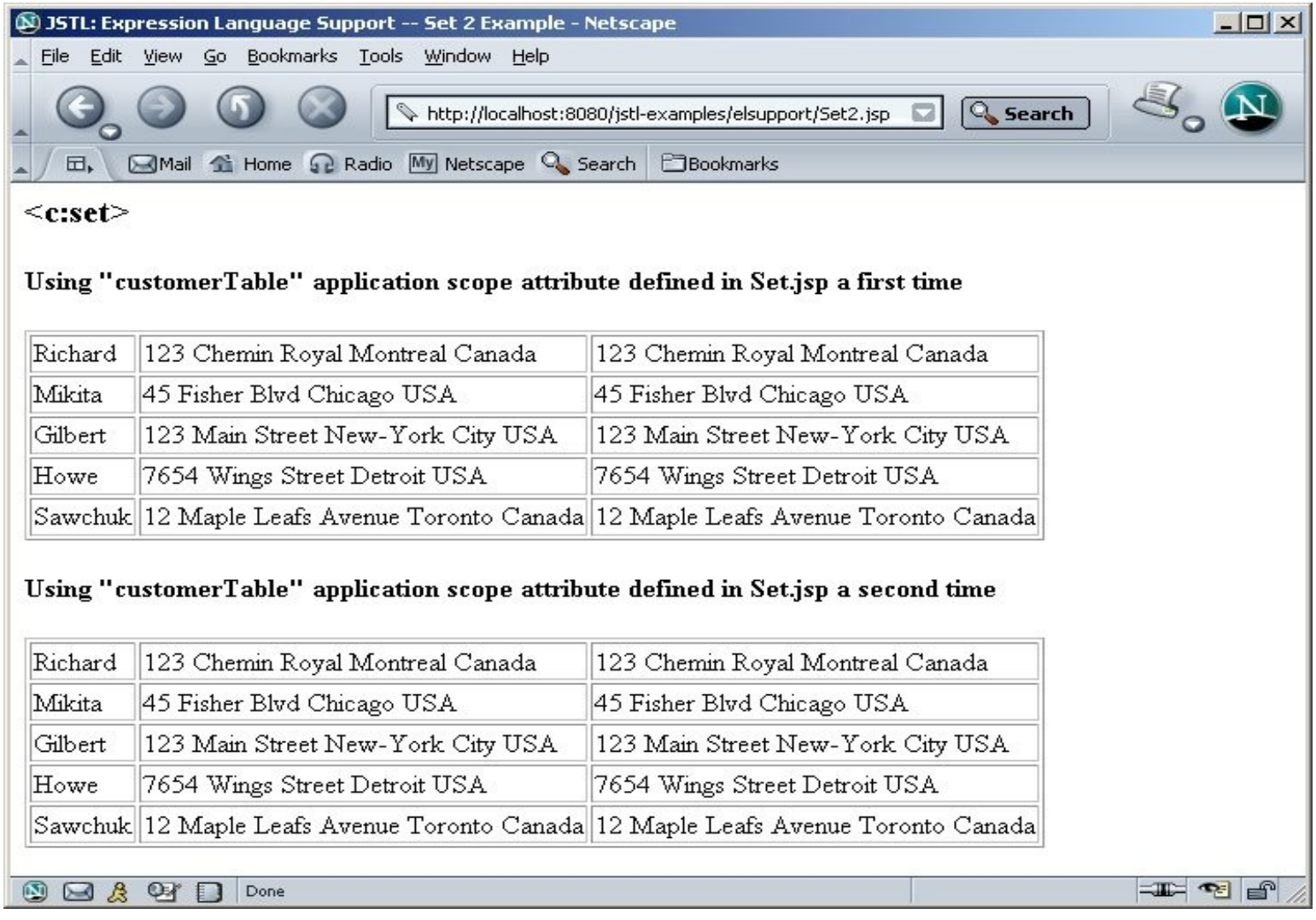
```
<c:set var="customerTable" scope="application">
<table border="1">
    <c:forEach var="customer" items="${customers}">
    <tr>
  <td>${customer.lastName}</td>
  <td><c:out value="${customer.address}" default="no address
    specified"/></td>
  <td>
    <c:out value="${customer.address}">
  <font color="red">no address specified</font>
</c:out>
      </td>
</tr>
  </c:forEach>
</table>
</c:set>
```

The content between <c:set> and /c:set is saved as ${customerTable}.

# Example: How a predefined Variable “customerTable” is used: ./elsupport/Set2.jsp

```
<h4>Using "customerTable" application scope attribute defined in
    Set.jsp a first time</h4>
<c:out value="${customerTable}" escapeXml="false"/>

<h4>Using "customerTable" application scope attribute defined in
    Set.jsp a second time</h4>
<c:out value="${customerTable}" escapeXml="false" />
```

# Example: How a predefined Variable "customerTable" is used: ./elsupport/Set2.jsp



# Variable Support <c:remove>

- Remove an EL variable
  - <c:remove var="cart" scope="session"/>

# Conditional Tags

- Flow control tags eliminate the need for scriptlets
  - Without conditional tags, a page author must generally resort to using scriptlets in JSP page
- <c:if test=”..”>
  - Conditional execution of its body according to value of a test attribute
- <c:choose>
  - Performs conditional block execution by the embedded <c:when> and <c:otherwise> sub tags
  - Works like if-then-else

# Example: <c:if test="...">, ./conditionals/If.jsp

```
<c:forEach var="customer" items="${customers}">
  <c:if test="${customer.address.country == 'USA'}">
    ${customer}<br>
  </c:if>
</c:forEach>
```

Only the customers whose "address.country" property value is "USA" are displayed through <c:forEach> loop.

# Example: <c:choose>, <c:when> ./conditionals/Choose.jsp

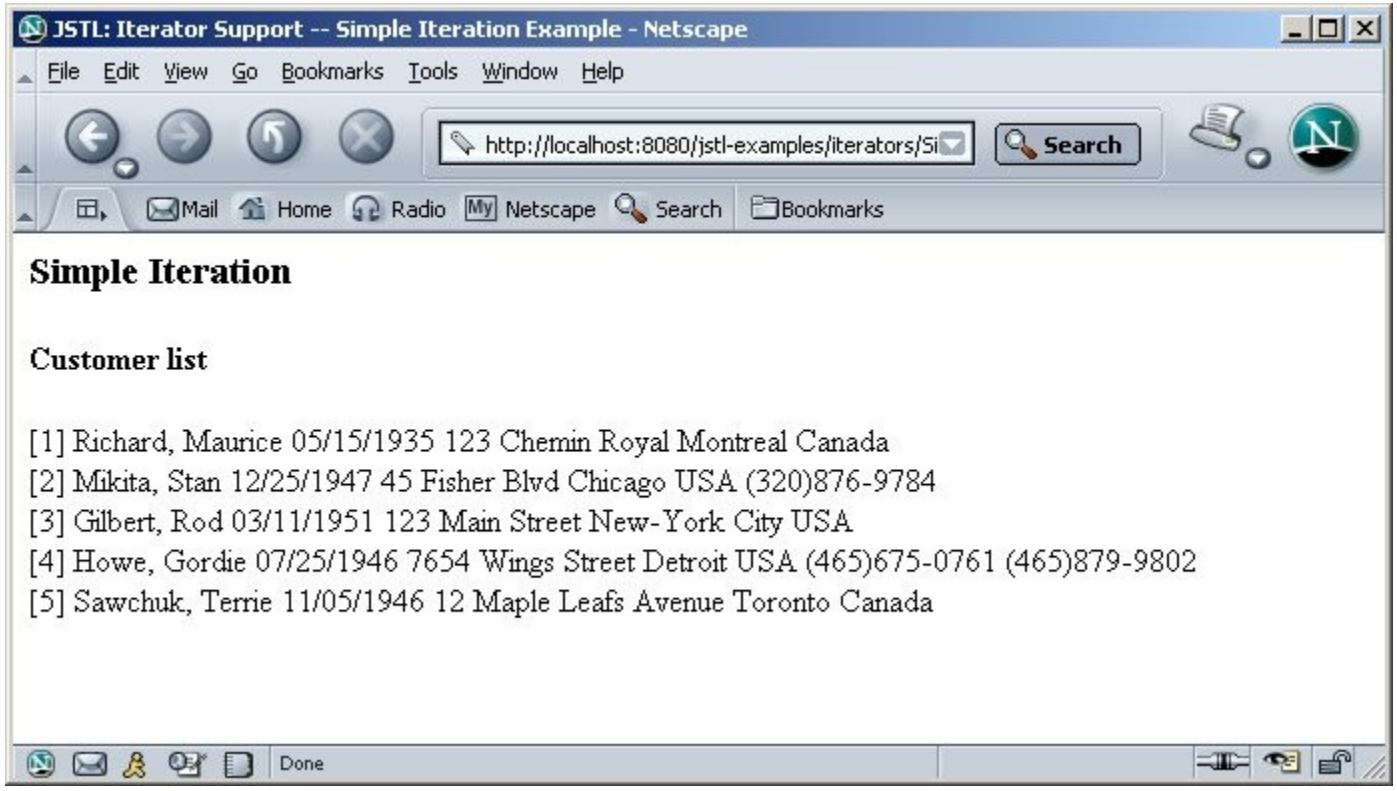
```
<c:forEach var="customer" items="${customers}">
  <c:choose>
    <c:when test="${customer.address.country == 'USA'}">
      <font color="blue">
    </c:when>
    <c:when test="${customer.address.country == 'Canada'}">
      <font color="red">
    </c:when>
    <c:otherwise>
      <font color="green">
    </c:otherwise>
  </c:choose>
  ${customer}</font><br>
</c:forEach>
```

# Iterator Tag: <c:forEach>

- Allows you to iterate over a collection of objects
  - items: represent the collection of objects
  - var: current item
  - varStatus: iteration status
  - begin, end, step: range and interval
- Collection types
  - java.util.Collection
  - java.util.Map
    - value of var attribute should be of type java.util.Map.Entry

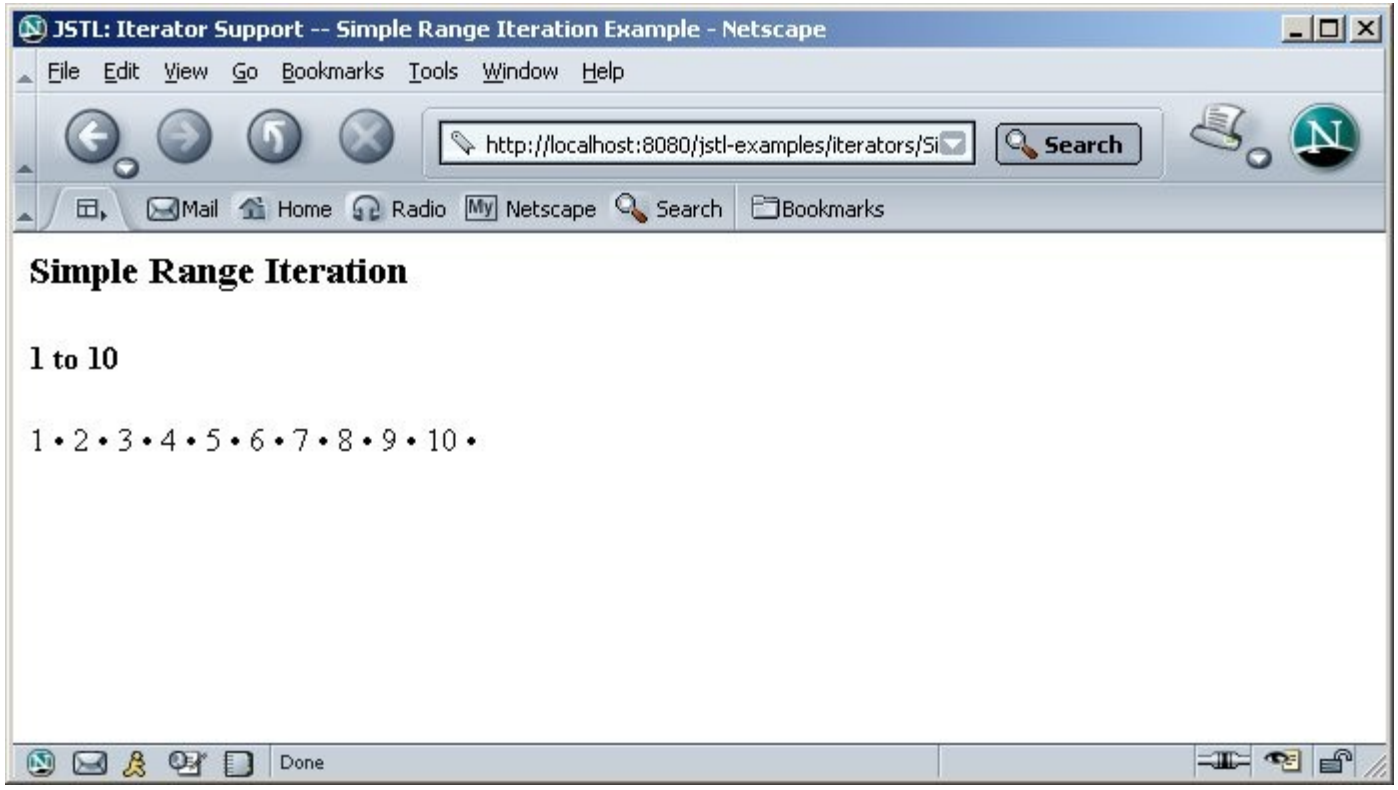
# Example: <c:forEach> quoted from ./iterators/Simple.jsp

```
<c:forEach var="customer" items="${customers}">
  ${customer}<br>
</c:forEach>
```



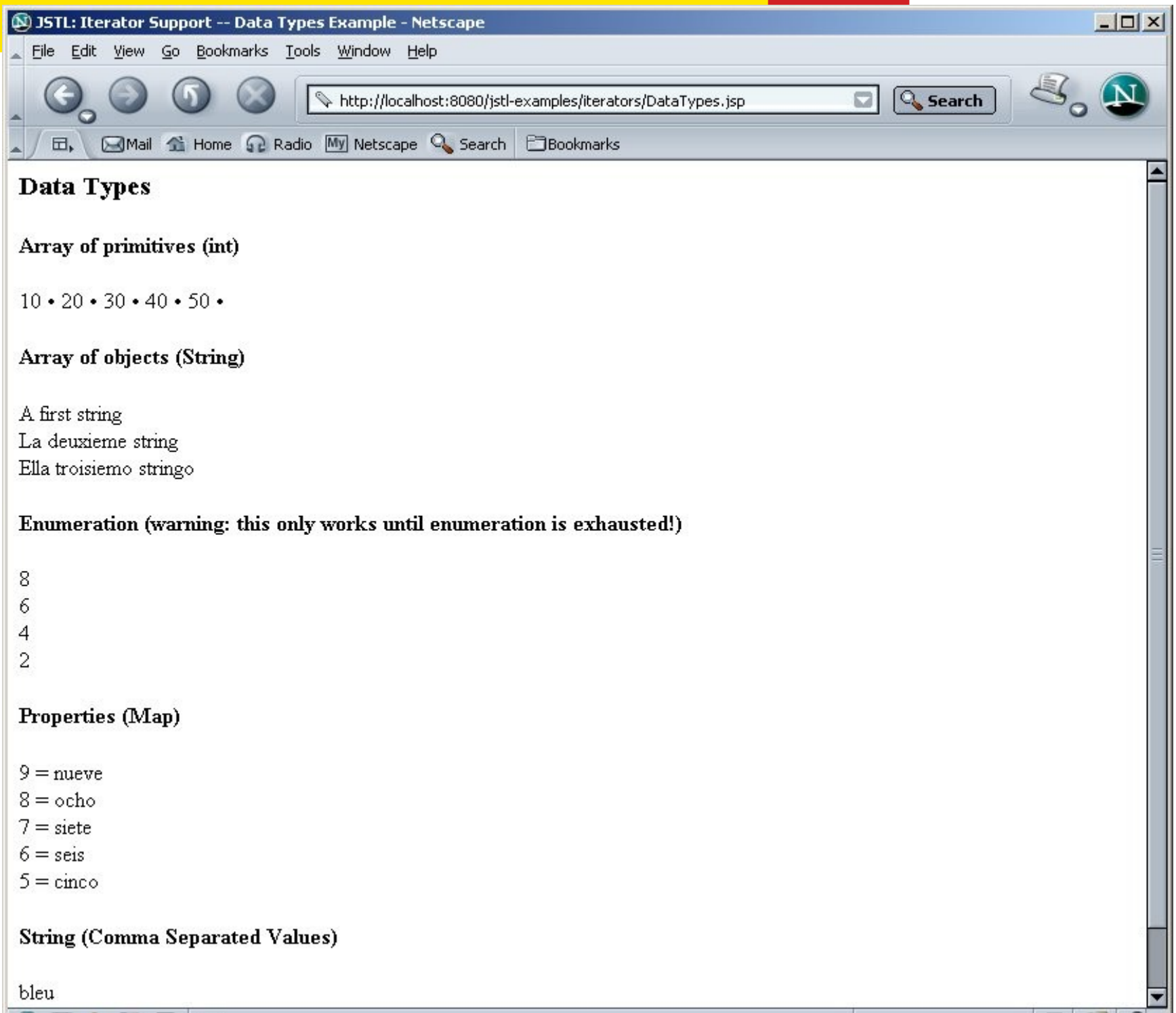
# Example: <c:forEach>, Range quoted from ./iterators/SimpleRange.jsp

```
<c:forEach var="i" begin="1" end="10">
  ${i} •
</c:forEach>
```



# Example: <c:forEach>, Data types quoted from ./iterators/DataTypes.jsp

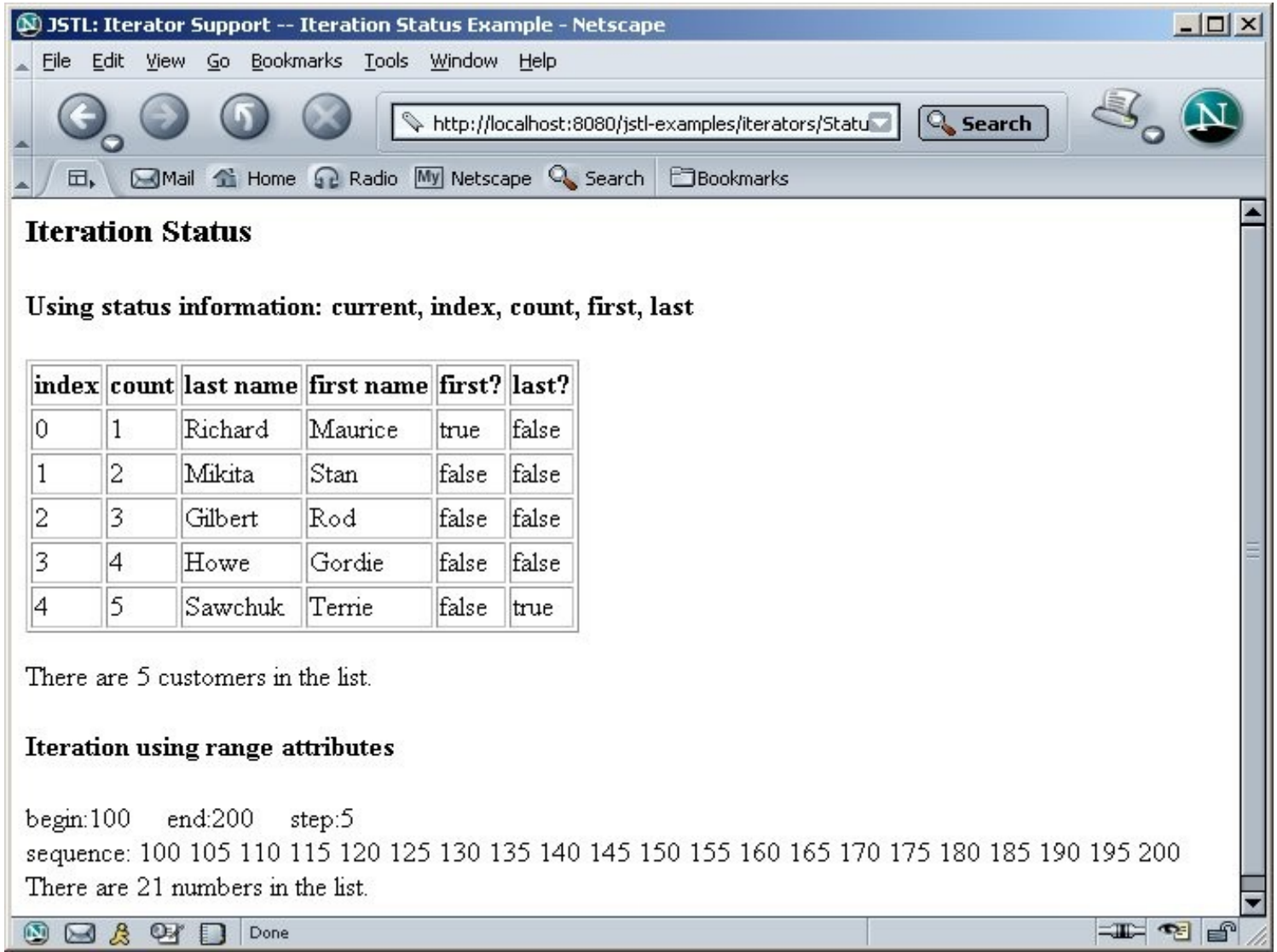
```
<c:forEach var="i" items="${intArray}">
  <c:out value="${i}"/> •
</c:forEach>

<c:forEach var="string" items="${stringArray}">
  <c:out value="${string}"/><br>
</c:forEach>

<c:forEach var="item" items="${enumeration}" begin="2" end="10" step="2">
  <c:out value="${item}"/><br>
</c:forEach>

<c:forEach var="prop" items="${numberMap}" begin="1" end="5">
  <c:out value="${prop.key}"/> = <c:out value="${prop.value}"/><br>
</c:forEach>

<c:forEach var="token" items="bleu,blanc,rouge">
  <c:out value="${token}"/><br>
</c:forEach>
```


JSTL: Iterator Support -- Data Types Example - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jstl-examples/iterators/DataTypes.jsp
Search
Mail Home Radio Netscape Search Bookmarks
Data Types
Array of primitives (int)
10 • 20 • 30 • 40 • 50 •
Array of objects (String)
A first string
La deuxieme string
Ella troisiemo stringo
Enumeration (warning: this only works until enumeration is exhausted!)
8
6
4
2
Properties (Map)
9 = nueve
8 = ocho
7 = siete
6 = seis
5 = cinco
String (Comma Separated Values)
bleu

# Example: <c:forEach>, Iteration status quoted from ./iterators/Status.jsp

```
<c:forEach var="customer" items="${customers}" varStatus="status">
    <tr>
      <td><c:out value="${status.index}"/></td>
      <td><c:out value="${status.count}"/></td>
      <td><c:out value="${status.current.lastName}"/></td>
      <td><c:out value="${status.current.firstName}"/></td>
      <td><c:out value="${status.first}"/></td>
      <td><c:out value="${status.last}"/></td>
    </tr>...
</c:forEach>
<c:forEach var="i" begin="100" end="200" step="5" varStatus="status">
  <c:if test="${status.first}">
    begin:<c:out value="${status.begin}">begin</c:out>
      end:<c:out value="${status.end}">end</c:out>
     step:<c:out value="${status.step}">step</c:out><br>
    sequence:
  </c:if>  ...
</c:forEach>
```

# Example: <c:forEach>, Iteration status quoted from ./iterators/Status.jsp



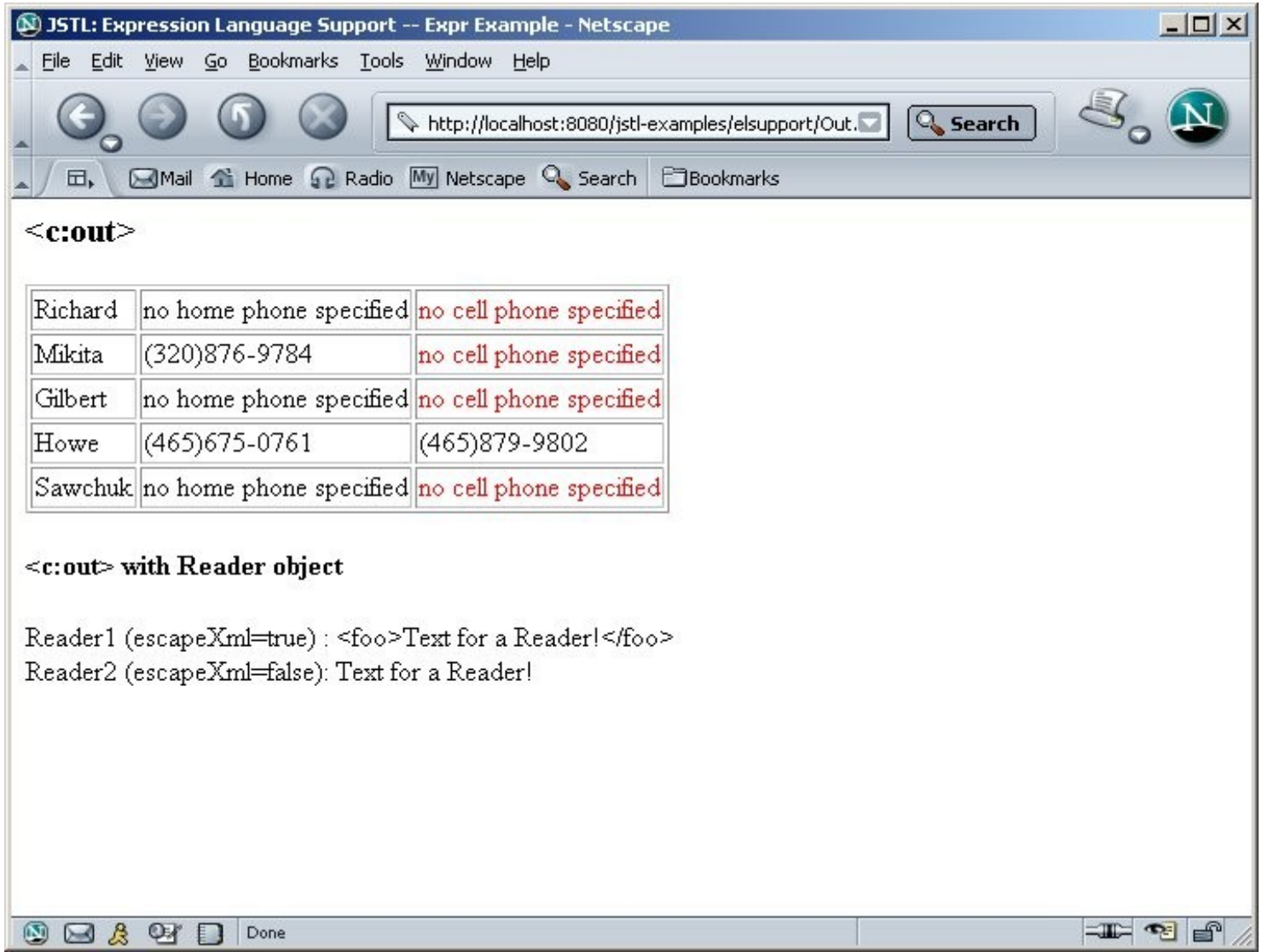
# Example: <c:forToken>

```
<c:forTokens var="token" items="one,two,three" delims=",">
<c:out value="${token}"/>
</c:forTokens>
```

# Example: <c:out> quoted from /elsupport/Out.jsp

```
<table border="1">
  <c:forEach var="customer" items="${customers}">
    <tr>
      <td><c:out value="${customer.lastName}"/></td>
      <td><c:out value="${customer.phoneHome}" default="no
    home phone specified"/></td>
      <td>
        <c:out value="${customer.phoneCell}" escapeXml="false">
          <font color="red">no cell phone specified</font>
        </c:out>
      </td>
    </tr>
  </c:forEach>
</table>
```

# Example: <c:out> quoted from /elsupport/Out.jsp



# URL Import: <c:import>

- More generic way to access URL-based resources (than <jsp:include>)
  - Absolute URL: for accessing resources outside of Web application
  - Relative URL: for accessing resources inside of the same Web application
- More efficient (than <jsp:include>)
  - No buffering
- <c:param> tag can be used to specify parameters (like <jsp:param>)

# Example: <c:import> Absolute URL quoted from /import/Absolute.jsp

```
<blockquote>
<ex:escapeHtml>
  <c:import url="http://www.cnn.com/cnn.rss"/>
</ex:escapeHtml>
</blockquote>
```

# Example: <c:import>  with <c:param>

```
<c:import url="header.jsp">
    <c:param name="pageTitle" value="newInstance.com"/>
    <c:param name="pageSlogan" value=" " />
</c:import>
```

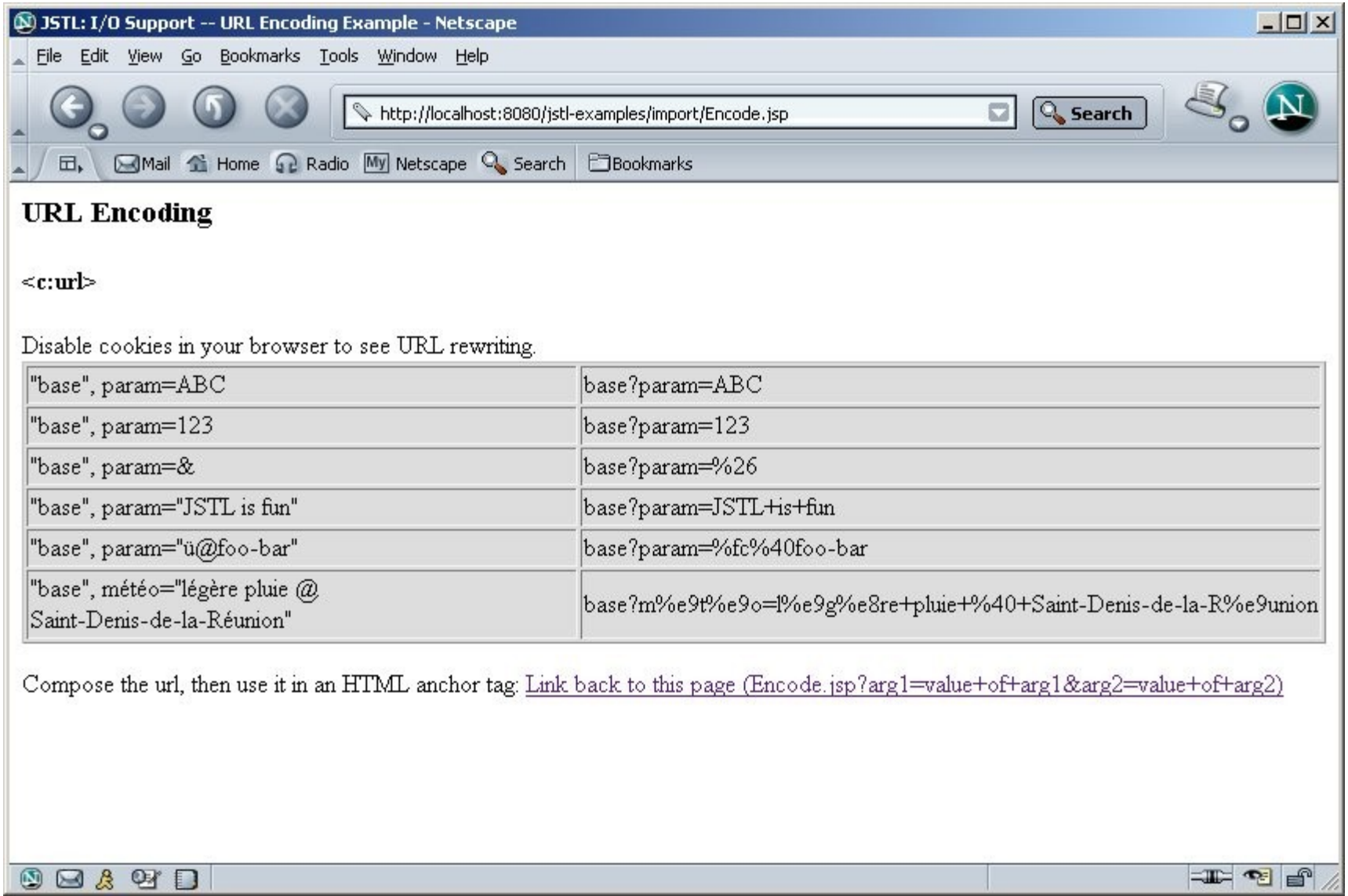
# URL Rewriting: <c:url>

- Used for URL rewriting
  - All the URL's that are returned from a JSP page (to a browser) have session ID if Cookie is disabled on the browser
- Can take param subtags for including parameters in the returned URL

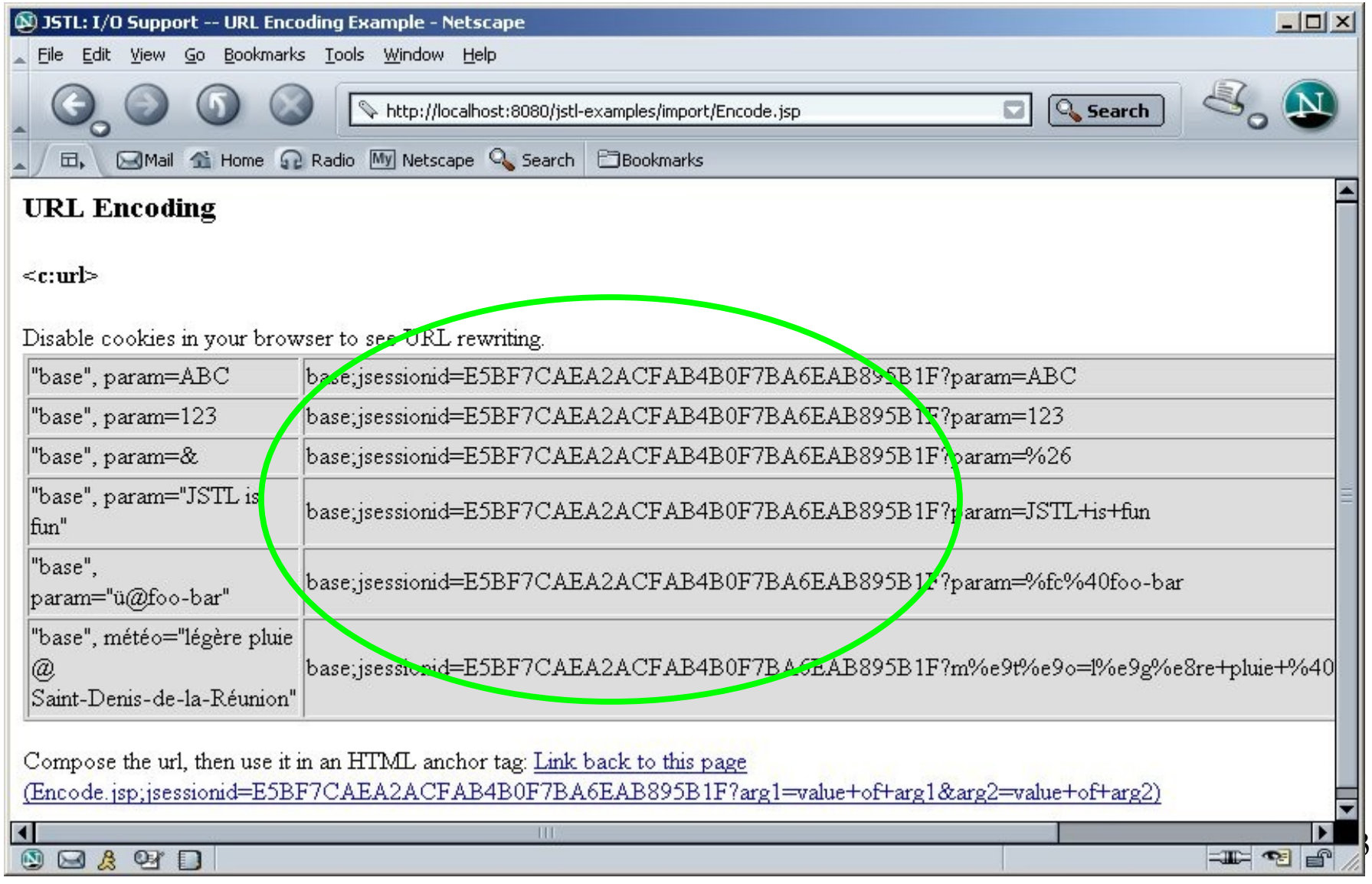
# Example: <c:url>

```
<table border="1" bgcolor="#dddddd">
 <tr>
  <td>"base", param=ABC</td>
  <td>
      <c:url value="base">
        <c:param name="param" value="ABC"/>
      </c:url>
  </td>
 </tr>
 <tr>
  <td>"base", param=123</td>
  <td>
        <c:url value="base">
          <c:param name="param" value="123"/>
        </c:url>
  </td>
 </tr>
 <tr>
```

# Example: <c:url> - Cookie enabled quoted from /import/Encode.jsp



# Example: <c:url> - Cookie disabled



# Redirection: <c:redirect>

- Sends an HTTP redirect to the client
- Takes <c:param> subtags for including parameters in the returned URL

# <c:out>

- Evaluates an expression and outputs the result of the evaluation to the current JspWriter object
- If the result of the evaluation is a java.io.Reader object, data is first read from the Reader object and then written into the current JspWriter object
  - improved performance
- Syntax
  - <c:out value="value" [escapeXml="{true|false}"] [default="defaultValue"] />
  - If escapeXml is true, escape character conversion

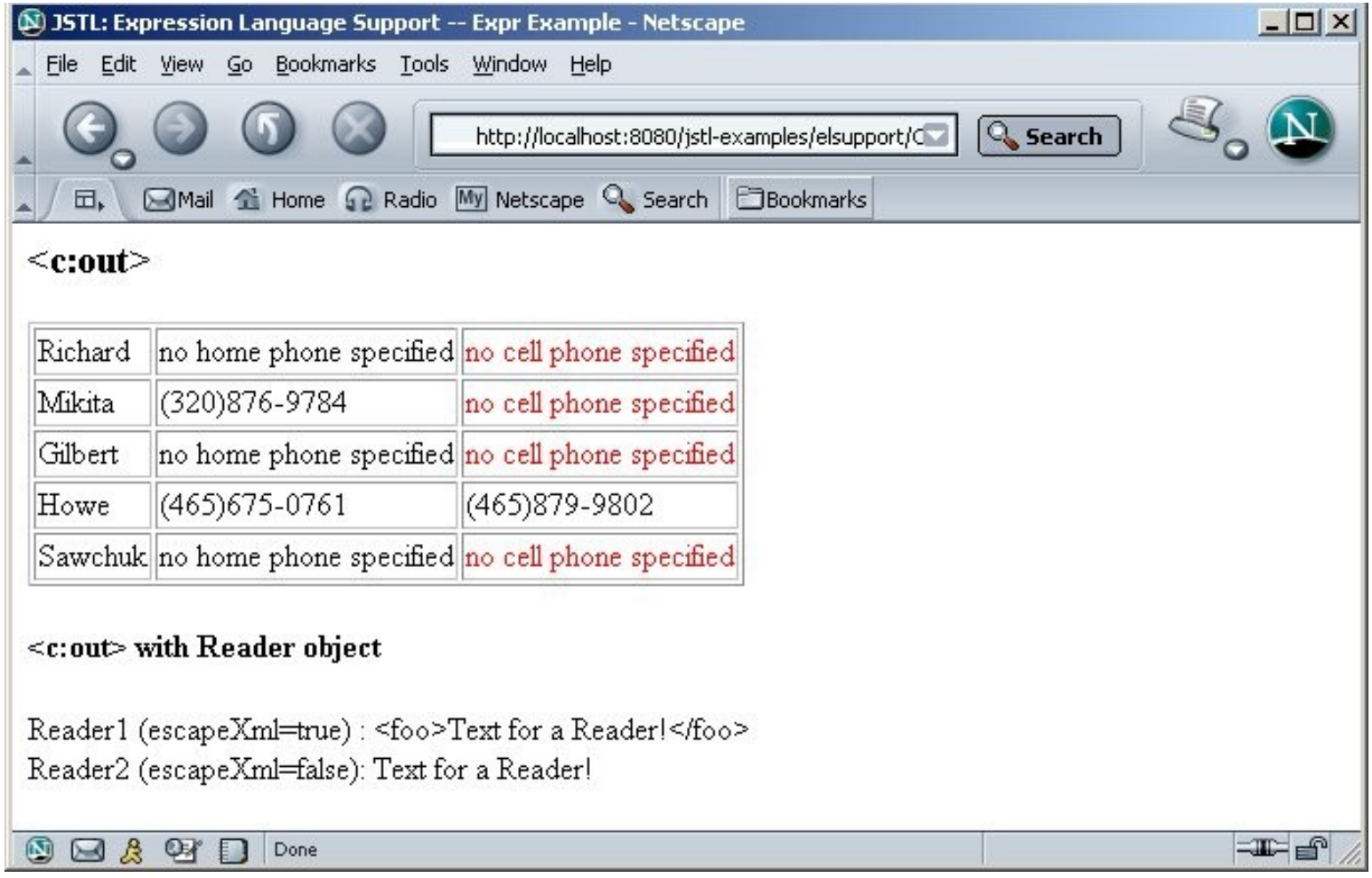
# Example: <c:out> quoted from /elsupport/Out.jsp

```
<table border="1">
  <c:forEach var="customer" items="${customers}">
    <tr>
      <td><c:out value="${customer.lastName}"/></td>
      <td><c:out value="${customer.phoneHome}" default="no
    home phone specified"/></td>
      <td>
        <c:out value="${customer.phoneCell}" escapeXml="false">
          <font color="red">no cell phone specified</font>
        </c:out>
      </td>
    </tr>
  </c:forEach>
</table>
```

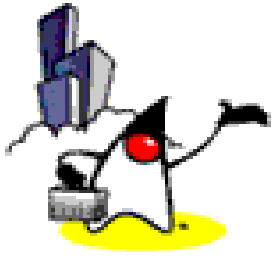
# Example: <c:out> quoted from /elsupport/Out.jsp

```
<h4><c:out> with Reader object</h4>
<%
java.io.Reader reader1 = new java.io.StringReader("<foo>Text for
    a Reader!</foo>");
pageContext.setAttribute("myReader1", reader1);
java.io.Reader reader2 = new java.io.StringReader("<foo>Text for
    a Reader!</foo>");
pageContext.setAttribute("myReader2", reader2);
%>
Reader1 (escapeXml=true) : <c:out value="${myReader1}"/><br>
Reader2 (escapeXml=false): <c:out value="${myReader2}"
    escapeXml="false"/><br>
```

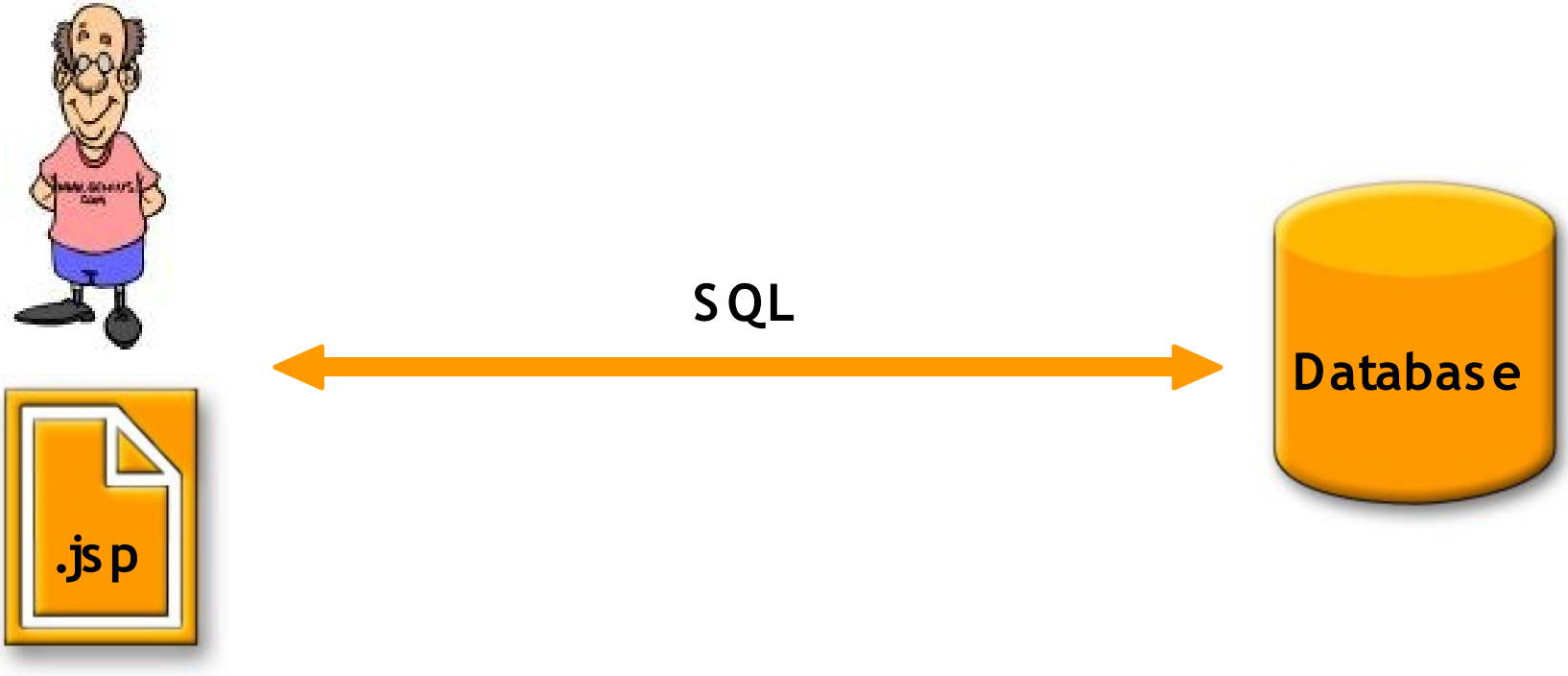
# Example: <c:out> quoted from /elsupport/Out.jsp



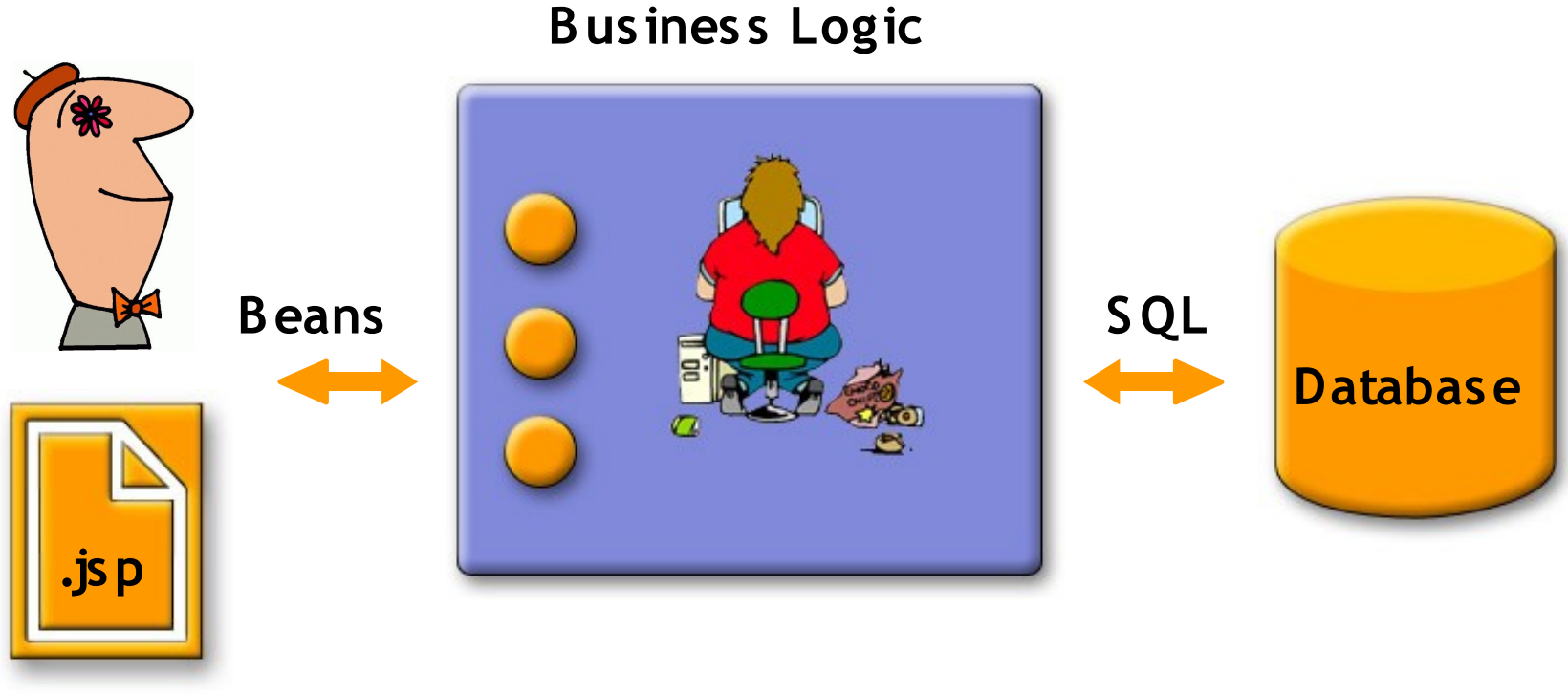
# Database Access Tags (SQL Tags)



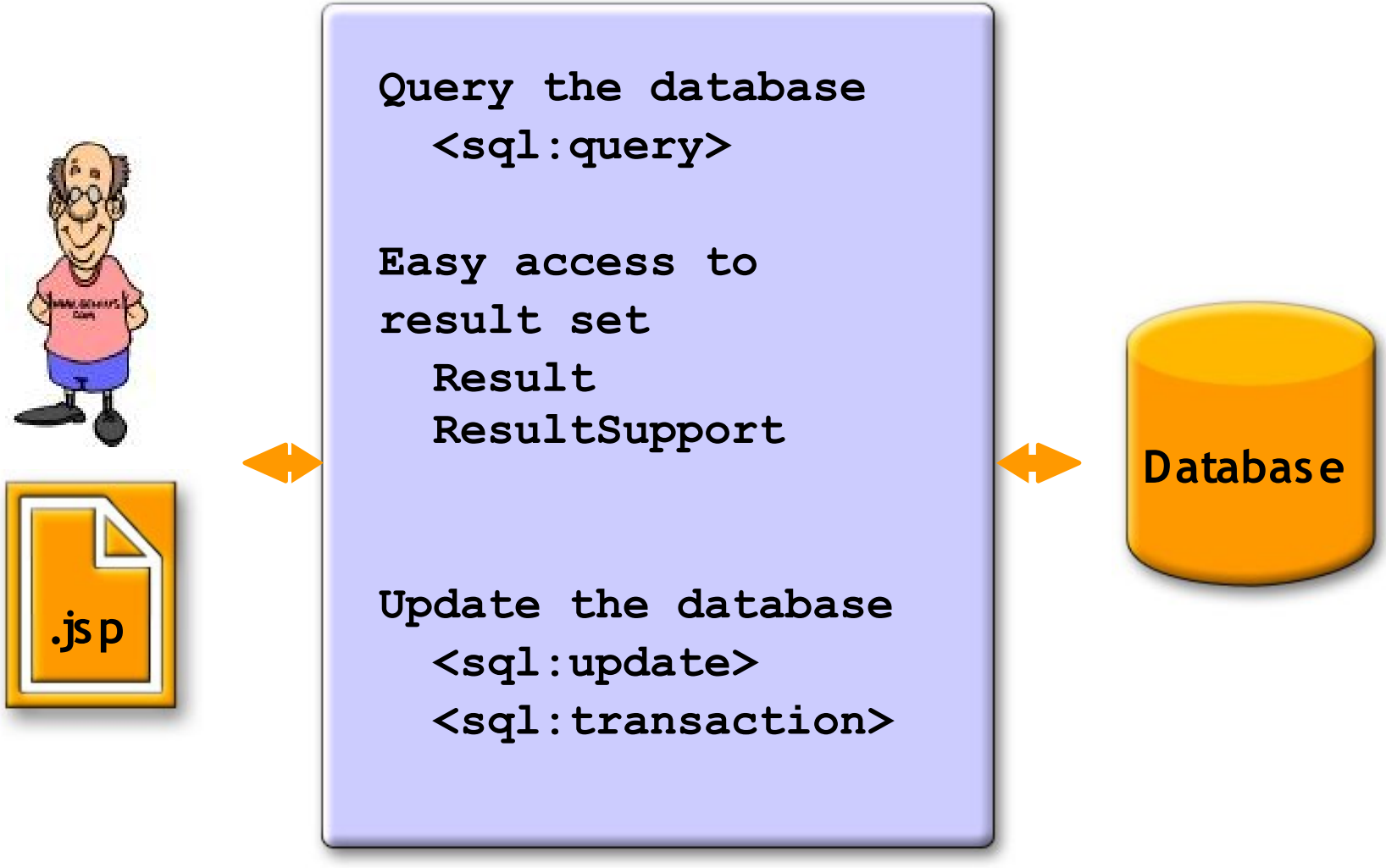
# RAD/Prototyping/Simple Apps



# MVC Architecture



# SQL Tags


Query the database
<sql:query>
Easy access to
result set
Result
ResultSupport
Update the database
<sql:update>
<sql:transaction>
.jsp
Database

# DataSource

- All DB actions operate on a DataSource
- Different ways to access a DataSource
  - Object provided by application logic
  - Object provided by <sql:dataSource> action

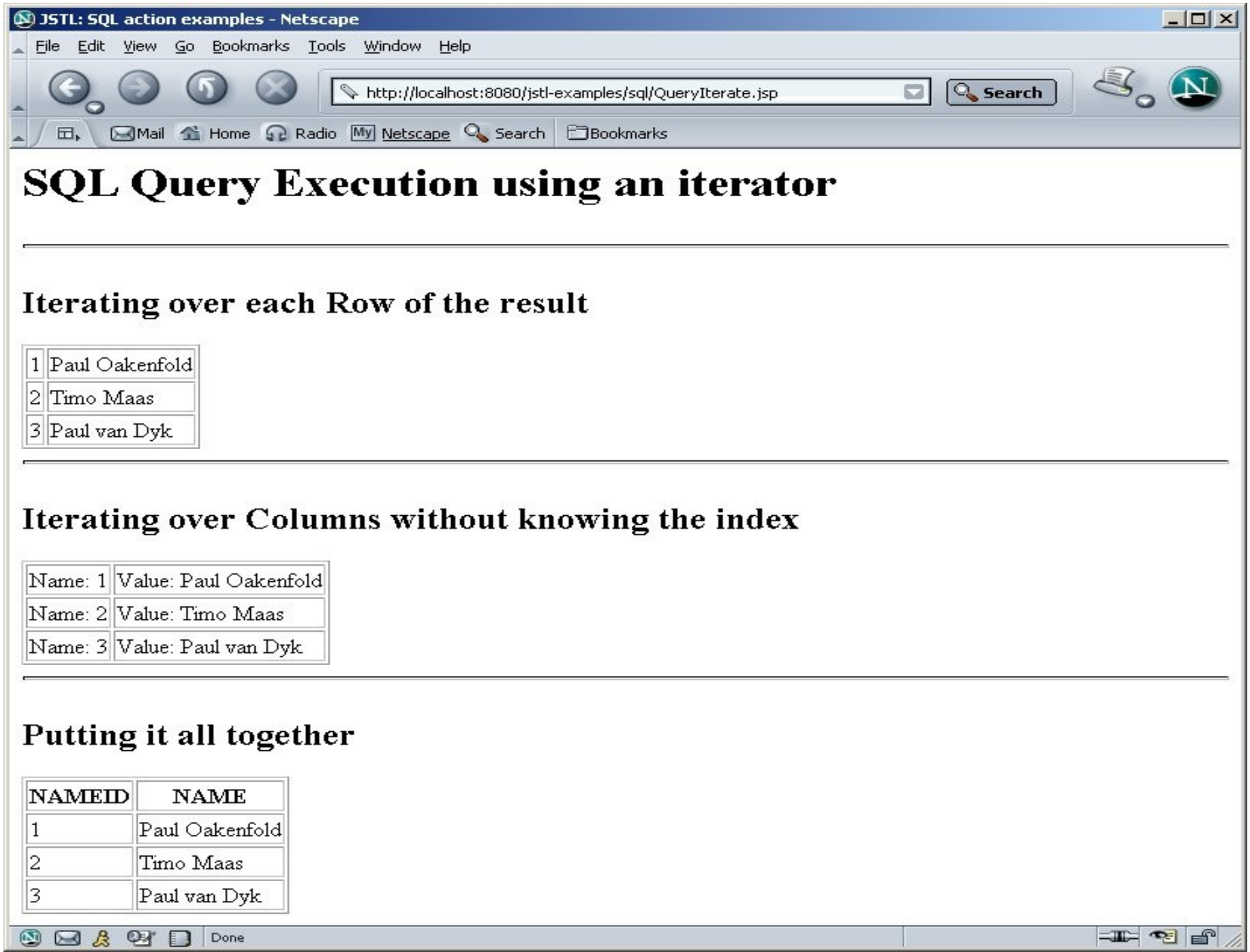
```
<sql:dataSource var="dataSource"
    driver="org.gjt.mm.mysql.Driver"
    url="jdbc:..."/>
<sql:query dataSource="${dataSource}" .../>
```

# Example: <sql:setDataSource> for setting a table using PointBase

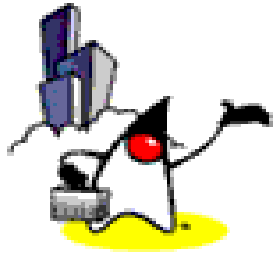
```
<sql:setDataSource
  var="example"
  driver="com.pointbase.jdbc.jdbcUniversalDriver"
  url="jdbc:pointbase:server://localhost:1092/jstlsample;create=true"
/>
```

# Example: <sql:transaction> & <sql:update> from /sql/QueryDirect.jsp

```
<sql:transaction dataSource="${example}">

  <sql:update var="newTable">
    create table mytable (
      nameid int primary key,
      name varchar(80)
    )
  </sql:update>

  <sql:update var="updateCount">
    INSERT INTO mytable VALUES (1,'Paul Oakenfold')
  </sql:update>
  <sql:update var="updateCount">
    INSERT INTO mytable VALUES (2,'Timo Maas')
  </sql:update>
  ...

  <sql:query var="deejays">
    SELECT * FROM mytable
  </sql:query>

</sql:transaction>
```


JSTL: SQL action examples - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jstl-examples/sql/QueryIterate.jsp
Search
Mail Home Radio Netscape Search Bookmarks
SQL Query Execution using an iterator
Iterating over each Row of the result
1 Paul Oakenfold
2 Timo Maas
3 Paul van Dyk
Iterating over Columns without knowing the index
Name: 1 Value: Paul Oakenfold
Name: 2 Value: Timo Maas
Name: 3 Value: Paul van Dyk
Putting it all together
NAMEID NAME
1 Paul Oakenfold
2 Timo Maas
3 Paul van Dyk
Done

# XML Tags

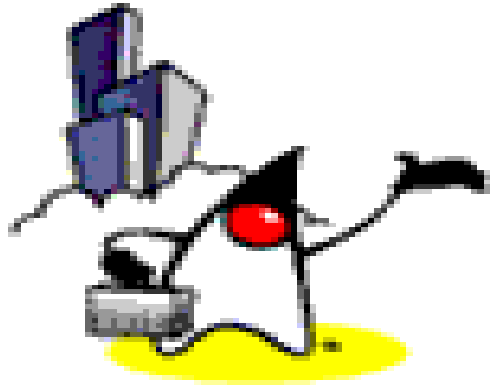


# XML Tags

- Flow control
  - <x:choose>, <x:when>, <x:if>, <x:otherwise>
- Iteration
  - <x:forEach>
- General purpose
  - <x:out>
  - <x:set>
- Parsing and Transformation
  - <x:parse>
  - <x:transform> with <x:param> subtags

# XML Tags

- Used to access information stored in XML document
- Access description is specified in XPath expression as a value of select attribute
  - <x:set var="d" select="$a//d"/>
  - <x:out select="$d/e"/>
- Flow control, Iteration, General purpose XML tags work similarly as corresponding tags in Core tags



# Quick XPath Review (Start)

# Example XML Document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<games>
  <country id="Luxembourg">
    <athlete>
      <name>Lux 1</name>
      <sport>swimming</sport>
      <age>23</age>
      <gender>M</gender>
    </athlete>
  </country>
  <country id="Denmark">
    <athlete>
      <name>Den 1</name>
      <sport>cycling</sport>
      <age>18</age>
      <gender>F</gender>
    </athlete>
    <athlete>
      <name>Den 2</name>
      <sport>sailing</sport>
      <age>27</age>
      <gender>M</gender>
    </athlete>
  </country>
</games>
```

# What is XPath?

- XPath is an Expression Language for referencing particular parts of XML document
- XPath expression uses a tree model to represent a XML document
  - XPath expression /games/country/athlete evaluates to a node-set that contains all nodes corresponding to the athletes of all countries in the games XML document

# XPath Expression Result Data Types: 4 Data Types

- Node set
  - Type we will spend most time with
- Boolean
- Number
- String

# Node Set, Location Path, Location Step, Predicate

- A node-set is a collection of zero or more nodes from XML document
- A node-set is a return type from location path expression
- A location path expression is composed of location steps
- A Location step can be qualified with a predicate
  - /games/country/athlete[sport=”sailing”]
  - /games/country[@id=”Demark”]/athlete

# Examples of Node Set

- /games/country/athlete
  - all athlete elements which has country parent element which in turn has games parent element
- /games/country[1]/athlete[2]
  - the 2nd athlete element under 1st country element
- /games/country/athlete[sport=”sailing”]
  - all athlete elements whose child element sport has string-value sailing
- /games/country[@id=”Demark”]/athlete
  - all athlete elements whose parent element country has id attribute value Denmark

# Examples of Node Set

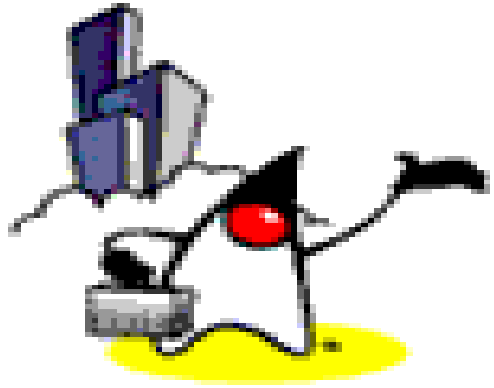
- /games/country/*
  - all child elements under /games/country
- /games/country//sport
  - all sport elements in a subtree that begins with /games/country

# XPath Type Coercion (Conversion)

- XPath specification defines rules on how node-set, boolean, number, string are to be converted to each other
- Node-set is converted to
  - boolean: true if note-set is not empty, false otherwise
  - string: the string value of the first node in the node-set
    - the reason why <x:out select="$doc//sport"/> results in “swimming”
  - number: node-set is first coerced to a string, which is then coerced to a number

# XPath Functions

- XPath expression can contain functions
- Example
  - count(node-set): returns number of nodes in a node-set
    - count(/games/country) returns 2 since there are 2 country nodes in the node-set
  - id(object): selects a node with the specified id
  - last(): returns size of the current node-set
  - string functions
    - string substring(/games/country, 1, 3)
  - boolean functions
    - boolean not(/games/country)



# Quick XPath Review (End)

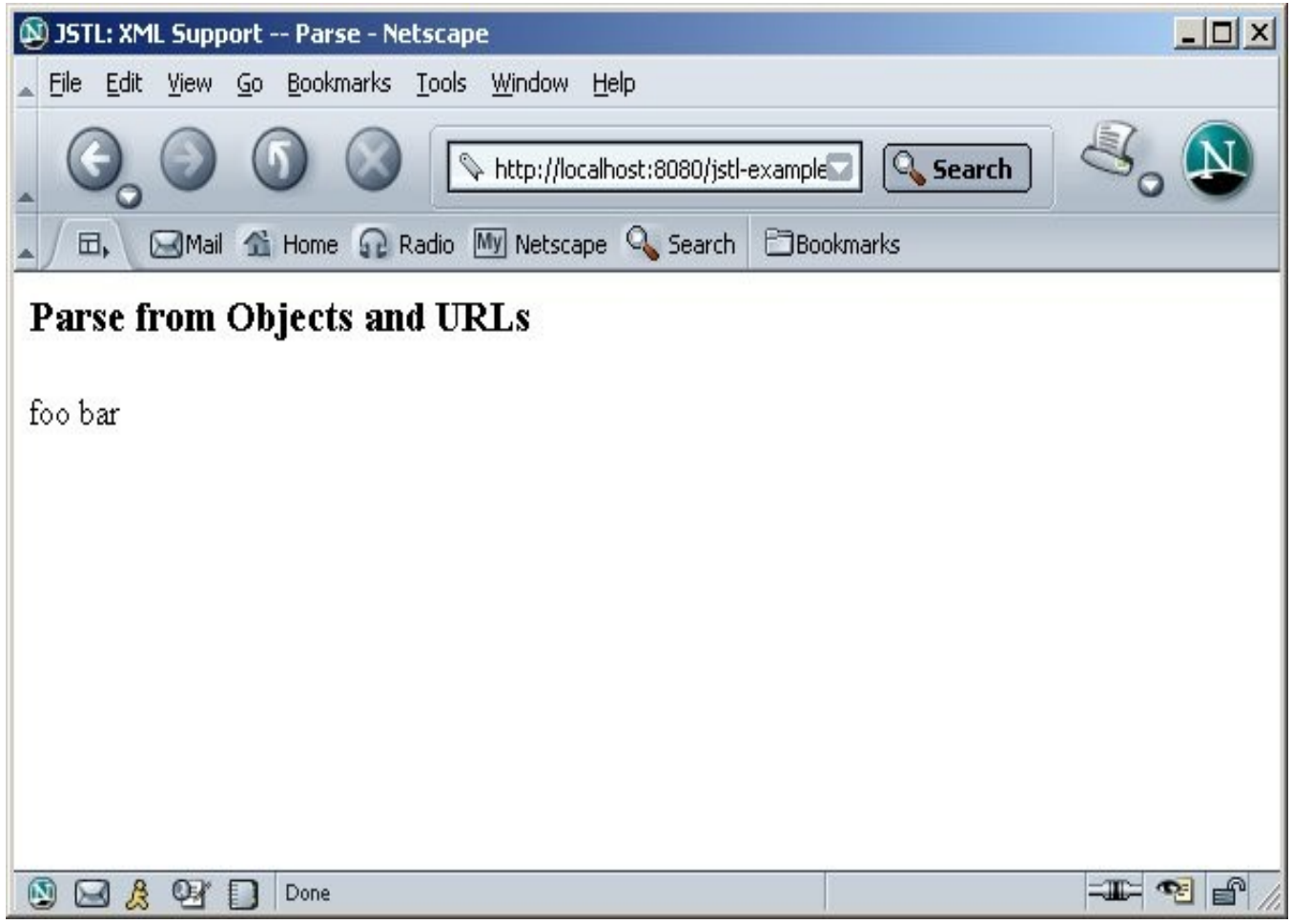
# <x:parse>

- Parse XML document into a scoped variable

# Example: <x:parse> from /xml/Parse.jsp

```
<c:set var="xmlText">
  <a>
      <b>
          <c>
            foo
          </c>
      </b>
      <d>
            bar
      </d>
  </a>
</c:set>

<x:parse var="a" doc="${xmlText}" />

<x:out select="$a//c"/>
<x:out select="$a/a/d"/>
```

# Example: <x:parse> from /xml/Parse.jsp



# <x:out>

- Works like <c:out> tag
- <x:out> tag converts node-set type to a String type
  - the string value of the first node in the node-set
  - The string value of an element is the concatenation of all descendent text nodes, no matter how deep
  - Example element

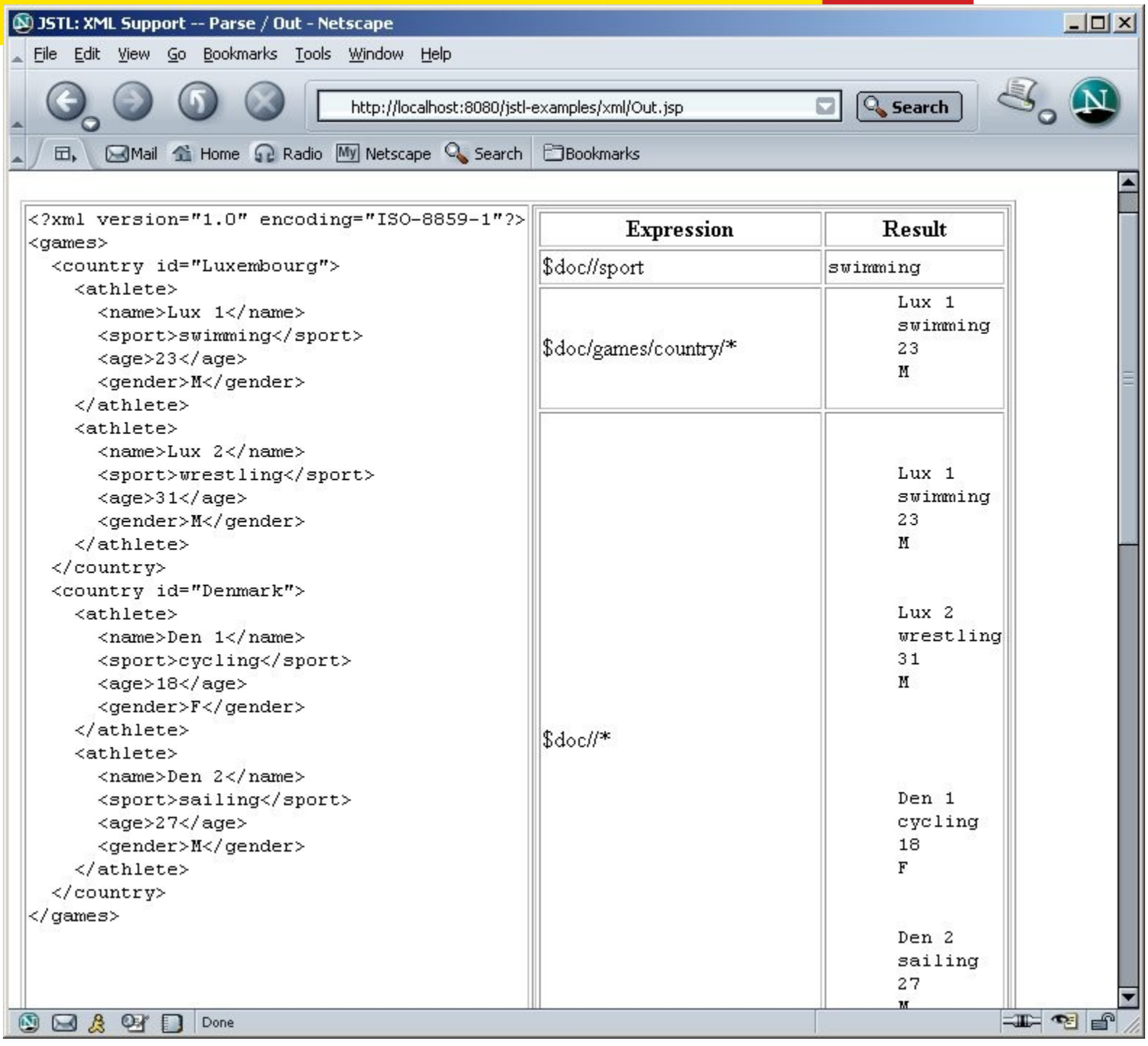
```
<athlete>
  <name>Lux 1</name>
  <sport>swimming</sport>
  <age>23</age>
  <gender>M</gender>
</athlete>
```

String value

```
Lux 1
swimming
23
M
```

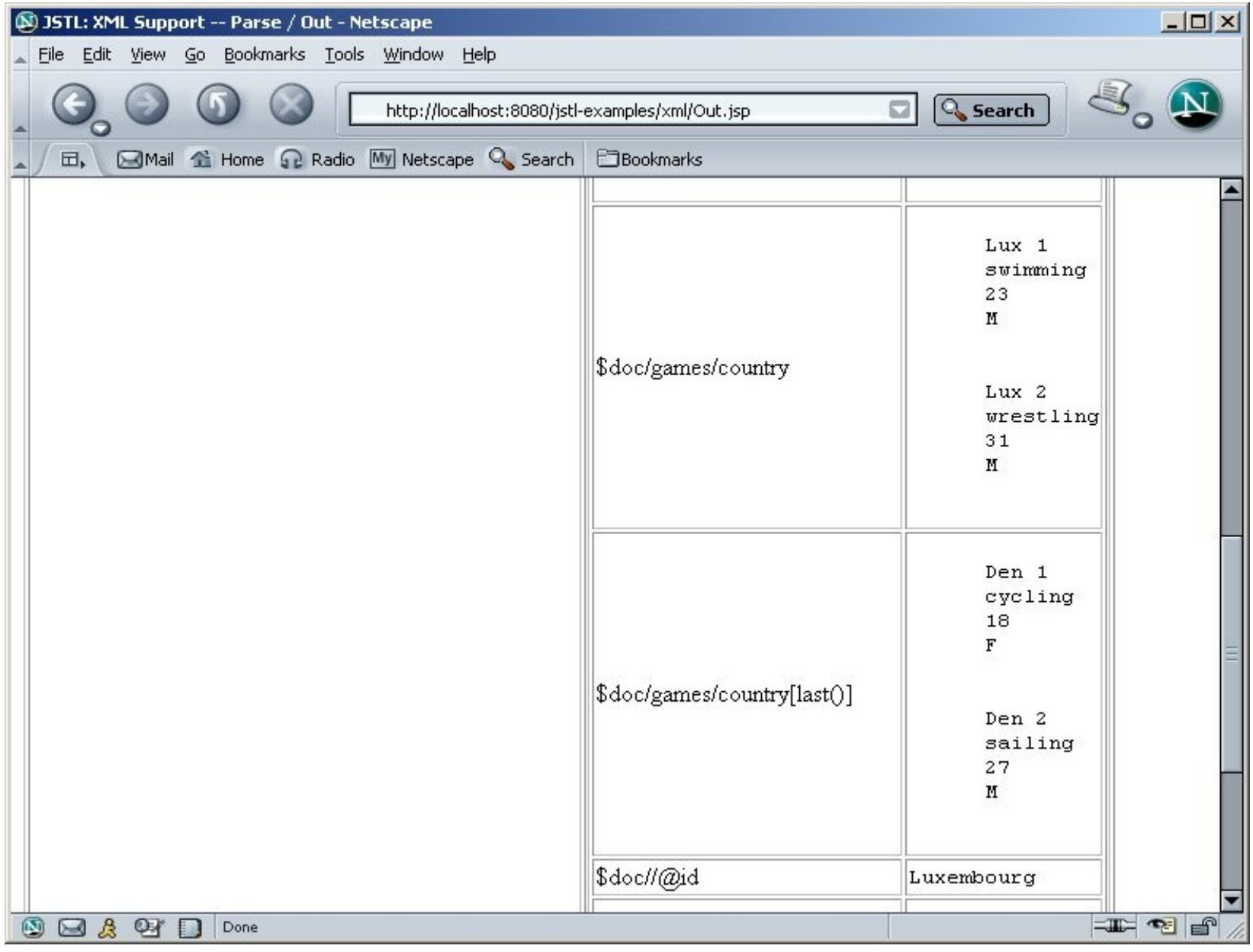
# Example: <x:out> from /xml/Out.jsp

```
<tr>
  <td>$doc//sport</td>
  <td><pre><x:out select="$doc//sport"/></pre></td>
</tr>
<tr>
  <td>$doc/games/country/*</td>
  <td><pre><x:out select="$doc/games/country/*"/></pre></td>
</tr>
<tr>
  <td>$doc//*</td>
  <td><pre><x:out select="$doc//*"/></pre></td>
</tr>
<tr>
  <td>$doc/games/country</td>
  <td><pre><x:out select="$doc/games/country"/></pre></td>
</tr>
```

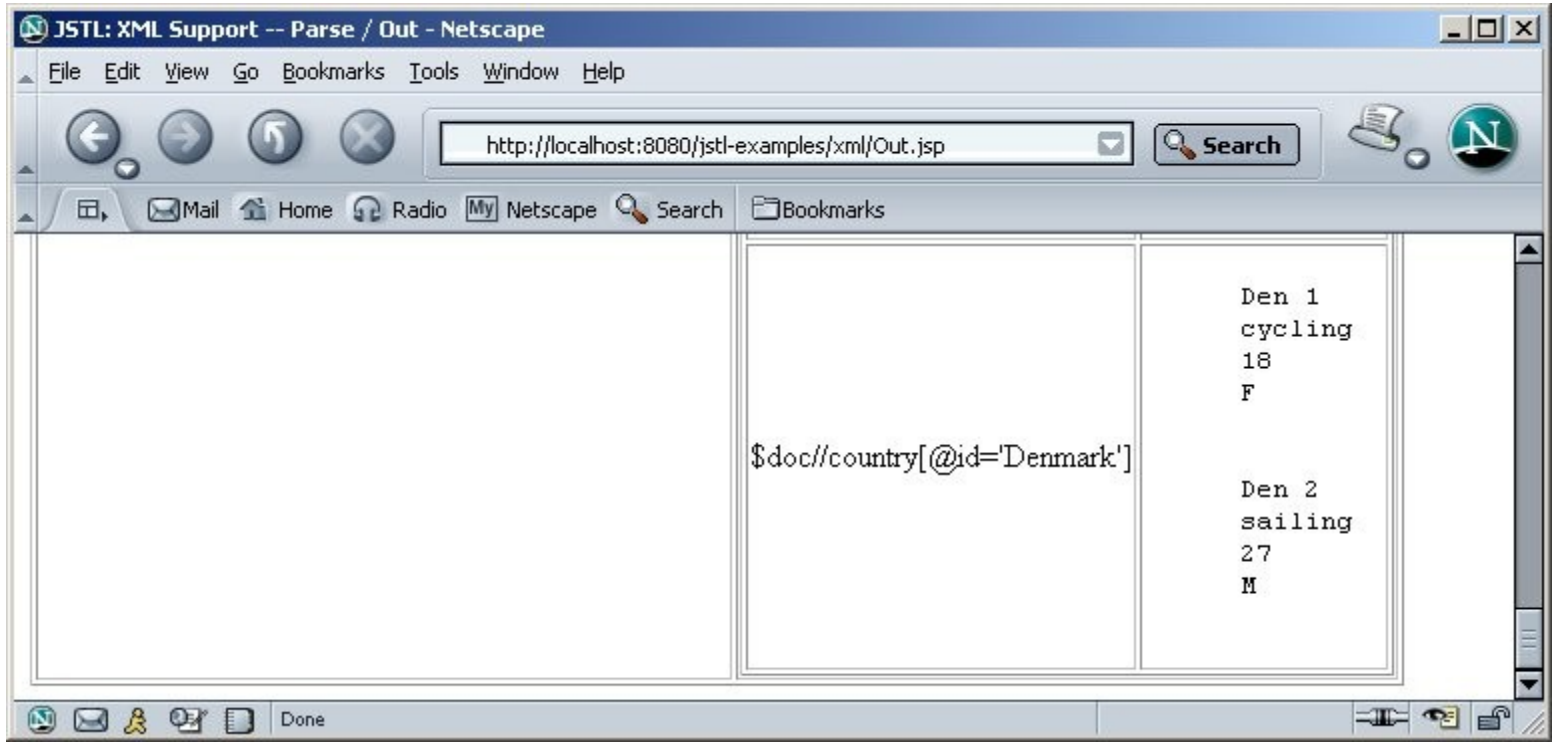

JSTL: XML Support -- Parse / Out - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jstl-examples/xml/Out.jsp
Search
Mail Home Radio Netscape Search Bookmarks
<?xml version="1.0" encoding="ISO-8859-1"?>
<games>
<country id="Luxembourg">
<athlete>
<name>Lux 1</name>
<sport>swimming</sport>
<age>23</age>
<gender>M</gender>
</athlete>
<athlete>
<name>Lux 2</name>
<sport>wrestling</sport>
<age>31</age>
<gender>M</gender>
</athlete>
</country>
<country id="Denmark">
<athlete>
<name>Den 1</name>
<sport>cycling</sport>
<age>18</age>
<gender>F</gender>
</athlete>
<athlete>
<name>Den 2</name>
<sport>sailing</sport>
<age>27</age>
<gender>M</gender>
</athlete>
</country>
</games>
Expression
Result
$doc//sport
swimming
$doc/games/country/*
Lux 1
swimming
23
M
$doc//*
Lux 1
swimming
23
M
Lux 2
wrestling
31
M
Den 1
cycling
18
F
Den 2
sailing
27
M
Done

# Example: <x:out> from /xml/Out.jsp

```
      <tr>
        <td>$doc/games/country[last()]</td>
        <td><pre><x:out select="$doc/games/country[last()]"/></pre></td>
      </tr>
      <tr>
        <td>$doc//@id</td>
        <td><pre><x:out select="$doc//@id"/></pre></td>
      </tr>
      <tr>
        <td>$doc//country[@id='Denmark']</td>
        <td><pre><x:out select="$doc//country[@id='Denmark']"/></pre></td>
      </tr>
    </table>
  </td>
</tr>
```


JSTL: XML Support -- Parse / Out - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jstl-examples/xml/Out.jsp
Search
Mail Home Radio Netscape Search Bookmarks
$doc/games/country
Lux 1
swimming
23
M
Lux 2
wrestling
31
M
$doc/games/country[last()]
Den 1
cycling
18
F
Den 2
sailing
27
M
$doc//@id
Luxembourg
Done

# Example: <x:out> from /xml/Out.jsp



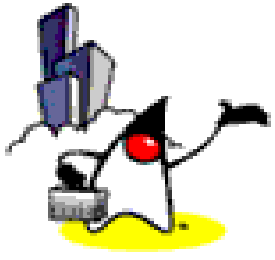
# Access to built-in scope variables in XPath expression

- $foo
- $param:
- $header:
- $cookie:
- $initParam:
- $pageScope:
- $requestScope:
- $sessionScope:
- $applicationScope:

# Example: Access to built-in scope variables

- $sessionScope:profile
  - The session-scoped EL variable named profile
- $initParam:mycom.productId
  - The String value of the mycom.productId context parameter

# EL Functions

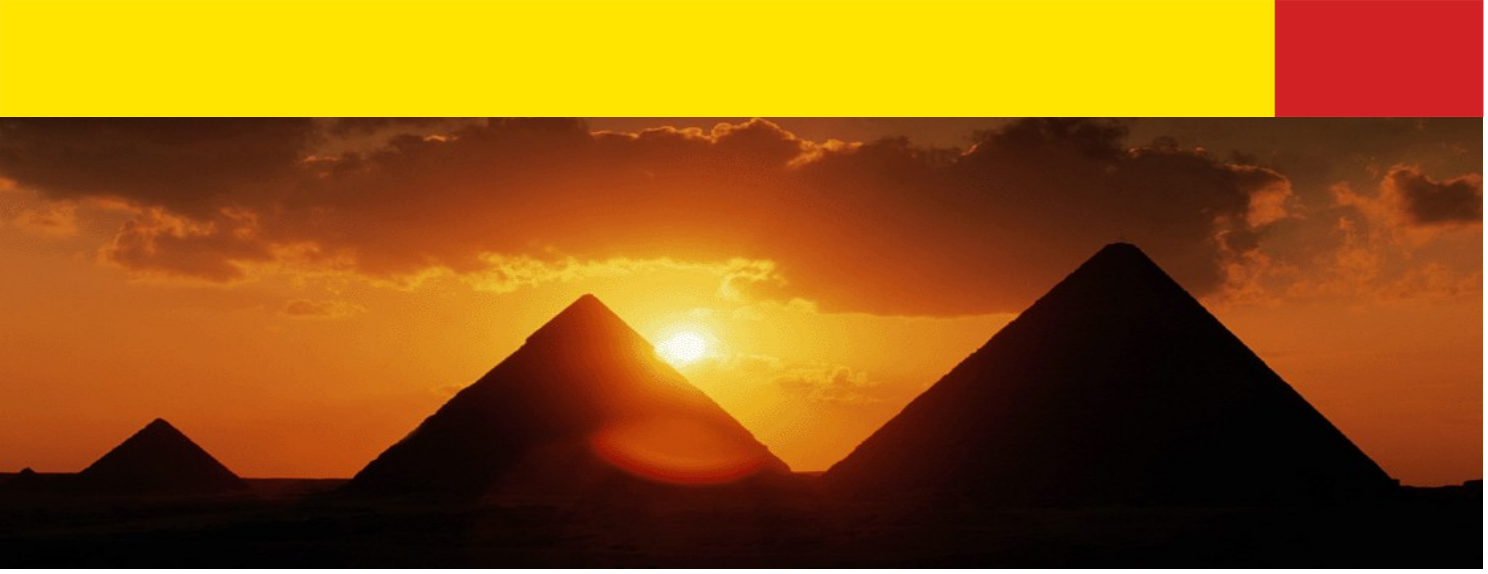


# EL Functions in JSTL 1.1

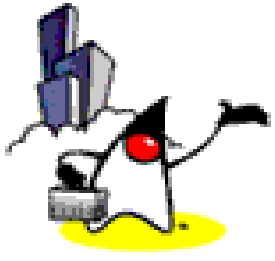
- <fn:length> Length of collection of string
- <fn:toUpperCase>, <fn:toLowerCase> Change the capitalization of a string
- <fn:substring>, <fn:substringBefore>, <fn:substringAfter> Get a subset of a string
- <fn:trim> Trim a string
- <fn:replace> Replace characters in a string
- <fn:indexOf>, <fn:startsWith>, <fn:endsWith contains>, <fn:containsIgnoreCase> Check if a string contains another string
- <fn:split>, <fn:join> Split a string into an array,and join a collection into a string
- <fn:escapeXml> Escape XML characters in the string

# Example: EL Functions

```
<%-- truncate name to 30 chars and display it in uppercase --%>
${fn:toUpperCase(fn:substring(name, 0, 30))}

<%-- Display the text value prior to the first '*' character --%>
${fn:substringBefore(text, '*')}

<%-- Scoped variable "name" may contain whitespaces at the
beginning or end. Trim it first, otherwise we end up with +'s in the URL
--%>
<c:url var="myUrl" value="${base}/cust/${fn:trim(name)}"/>

<%-- Display the text in between brackets --%>
${fn:substring(text, fn:indexOf(text, '(')+1, fn:indexOf(text, ')'))}

<%-- Display the name if it contains the search string --%>
<c:if test="${fn:containsIgnoreCase(name, searchString)}">
  Found name: ${name}
</c:if>

<%-- Display the last 10 characters of the text value --%>
${fn:substring(text, fn:length(text)-10)}

<%-- Display text value with bullets instead of '-' --%>
${fn:replace(text, '-', '&#149;')}
```

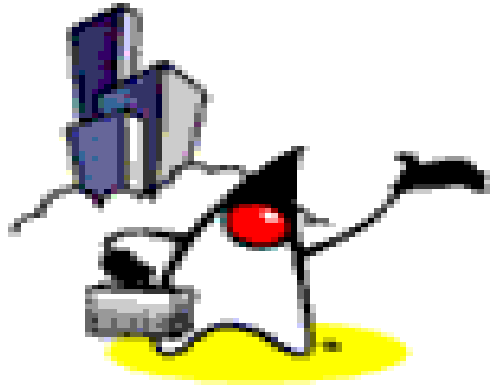


# Internationalization (i18n) &Text Formatting Tags



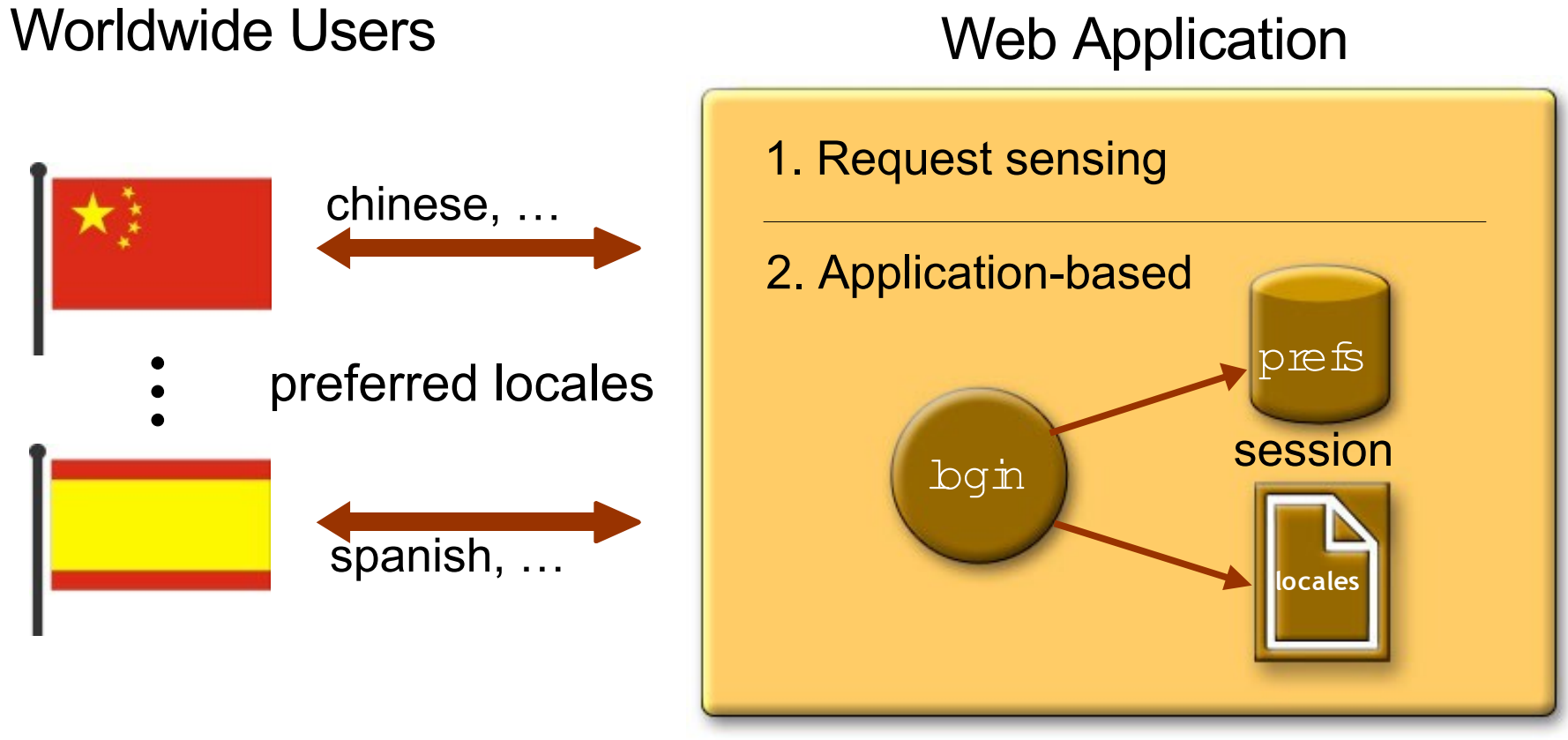
# I18N and Formatting Tags

- Setting locale
  - <fmt:setLocale>
  - <fmt:requestEncoding>
- Messaging
  - <fmt:bundle>
  - <fmt:message> with <fmt:param> subtag
  - <fmt:setBundle>
- Number and Date formatting
  - <fmt:formatNumber>, <fmt:parseNumber>
  - <fmt:formatDate>, <fmt:parseDate>
  - <fmt:setTimeZone>, <fmt:timeZone >

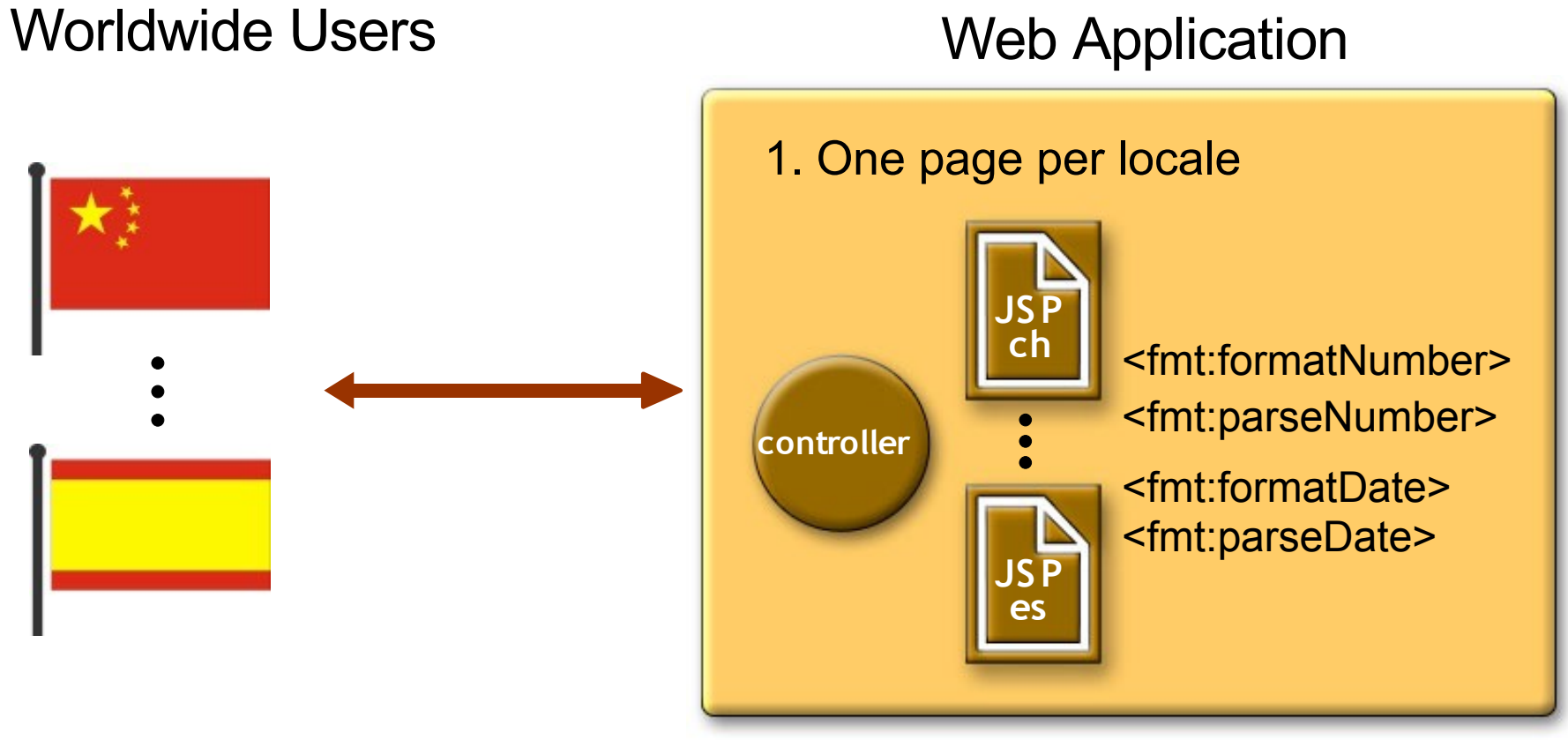


# Quick I18N Review (Start)

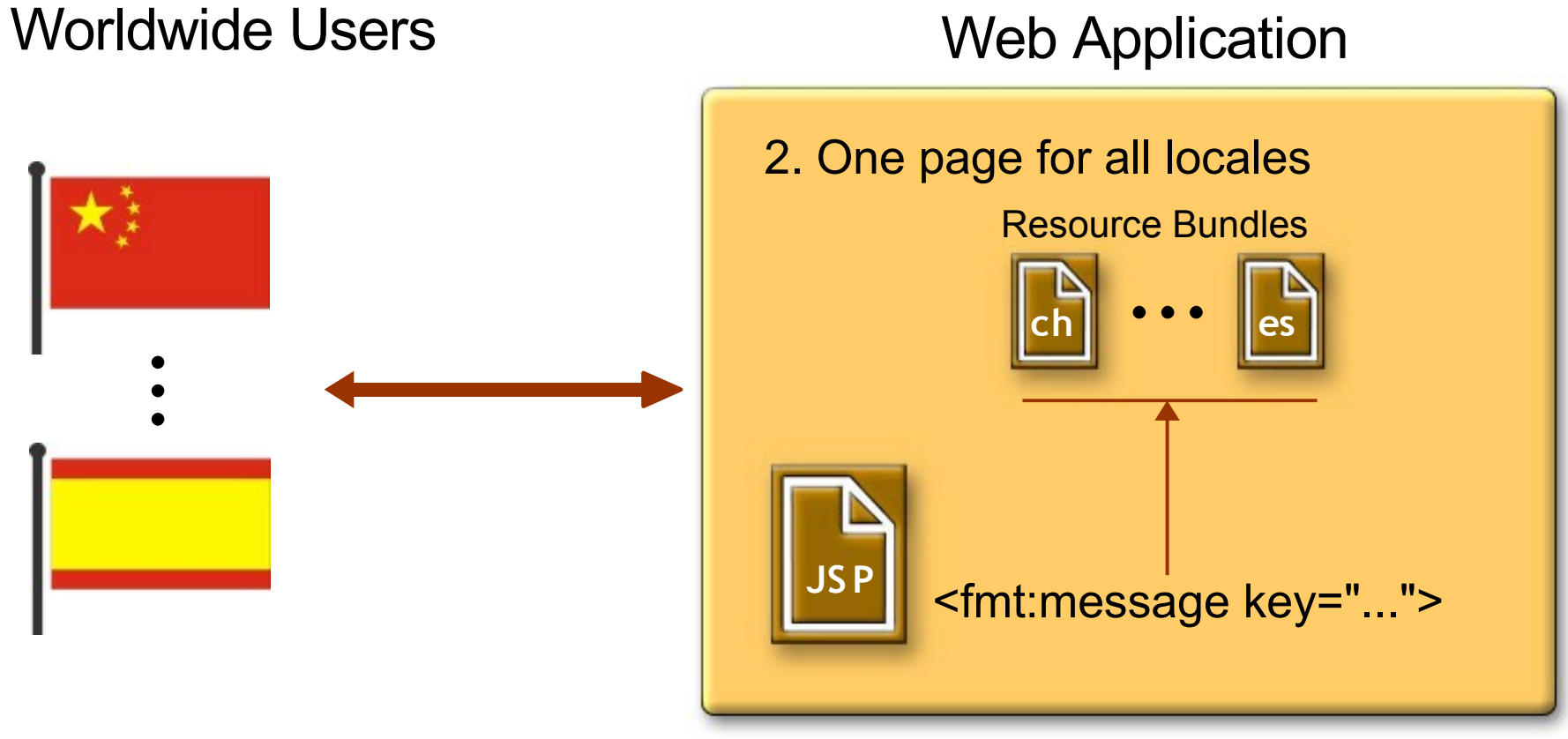
# How Locale Is Set in Web app

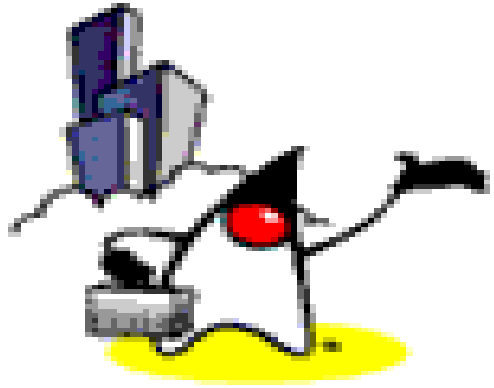


# I18N Architecture: Option 1



# I18N Architecture: Option 2





# Quick I18N Review (End)

# Setting Locales

- <fmt:setLocale>
  - Override client-specified locale for a page
- <fmt:requestEncoding>
  - Set the request's character encoding, in order to be able to correctly decode request parameter values whose encoding is different from ISO-8859-1

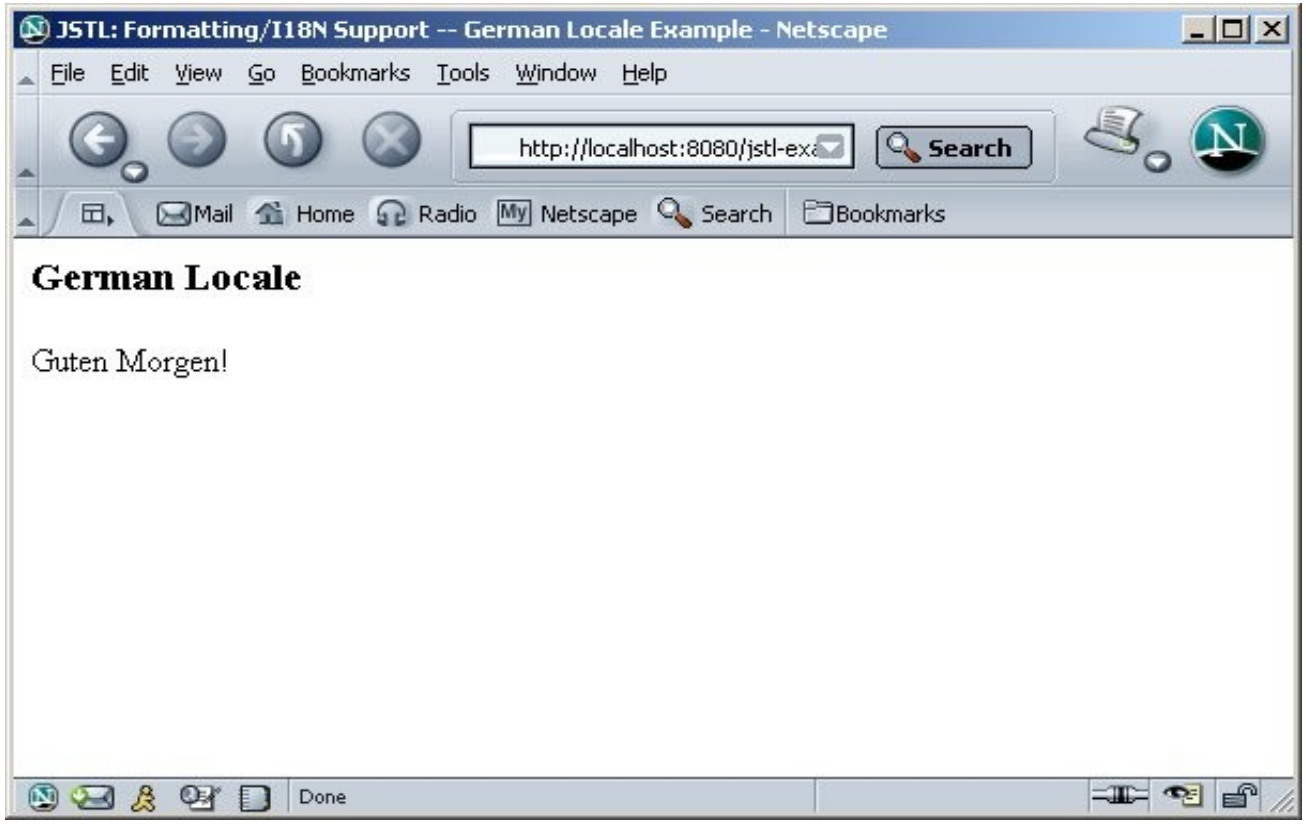
# Messaging Tags

- <fmt:bundle>
  - specify a resource bundle for a page
- <fmt:message key=”..”>
  - used to output localized strings
  - <fmt:param> subtag provides a single argument (for parametric replacement) to the compound message or pattern in its parent message tag

# Example: quoted from ./fmt/GermanLocale.jsp

```
<fmt:setLocale value="de"/>
<fmt:bundle
   basename="org.apache.taglibs.standard.examples.i18n.R
   esources">
 <fmt:message>
   greetingMorning
 </fmt:message>
</fmt:bundle>
```

# Example: <fmt:setLocale>




http://localhost:8080/webapps-jstl/format/GermanLocale.jsp

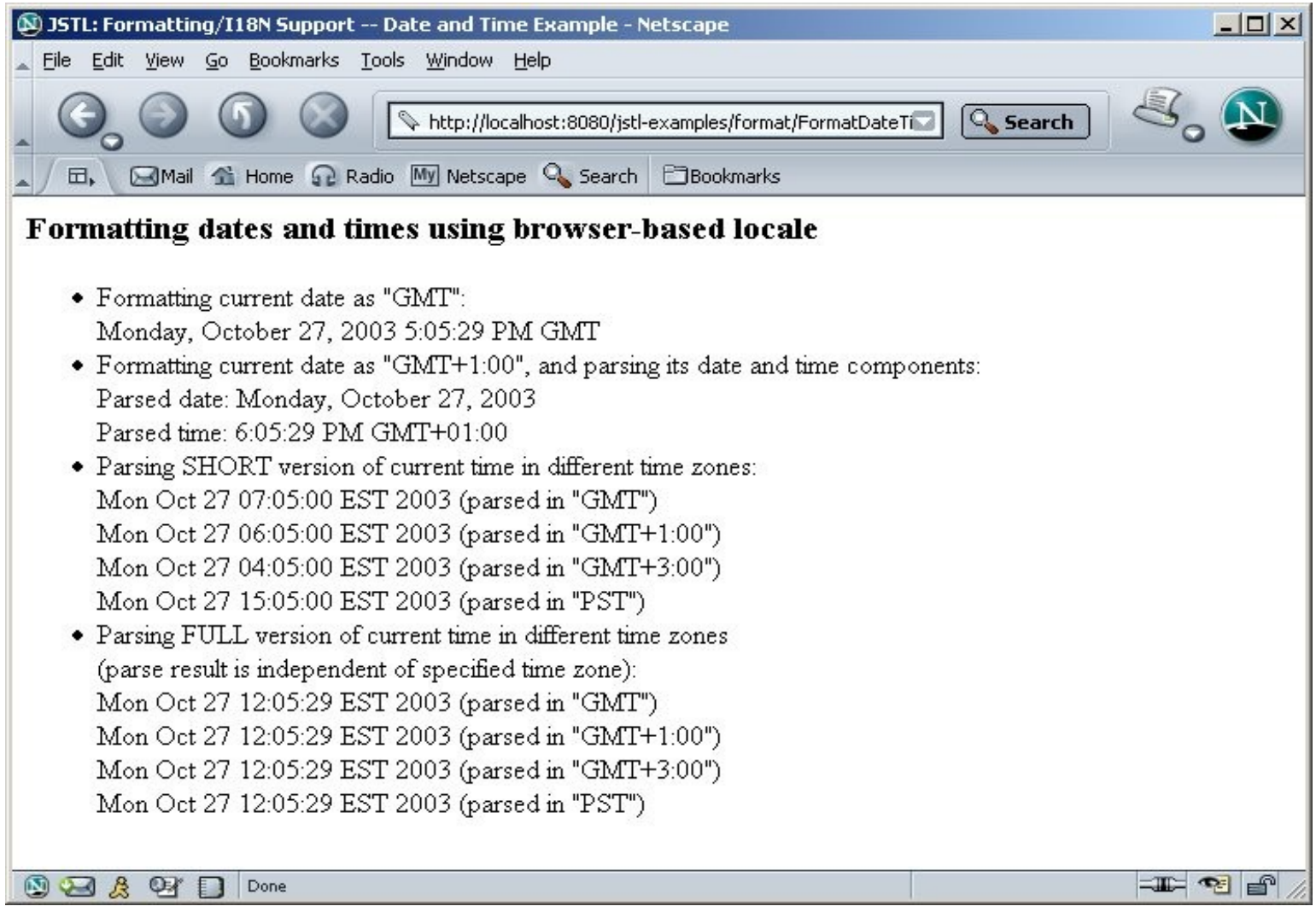
# Formatting Tags

- <fmt:formatNumber>, <fmt:formatDate>
  - used to output localized numbers and dates
- <fmt:parseNumber>, <fmt:parseDate>
  - used to parse localized numbers and dates
- <fmt:setTimeZone>, <fmt:timeZone >
  - used to set and get timezone

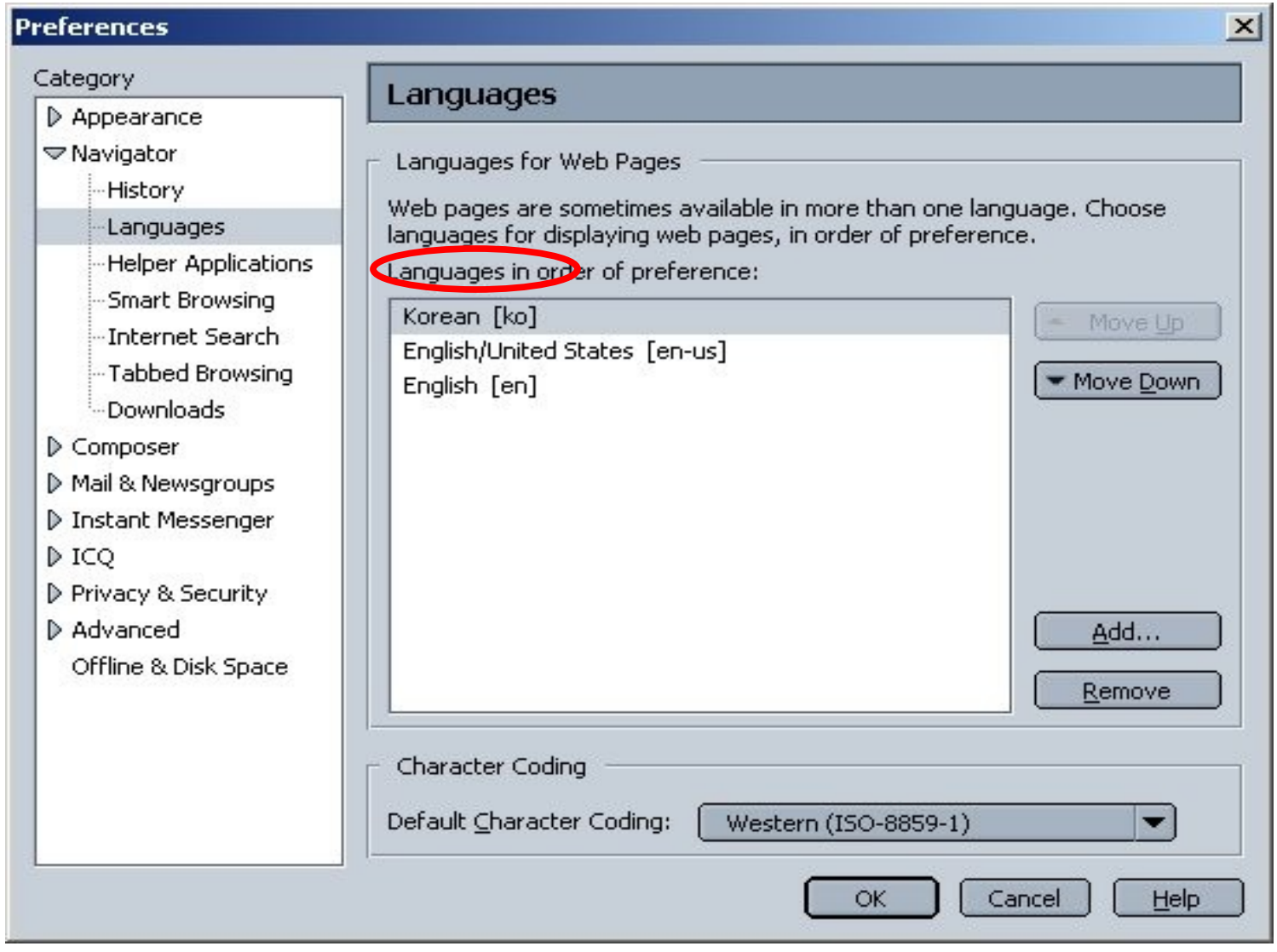
# Example:
# quoted from ./format/FormatDateTime.jsp

```
<jsp:useBean id="now" class="java.util.Date" />
<fmt:setLocale value="en-US" />

<ul>
 <li> Formatting current date as "GMT":<br>
  <fmt:timeZone value="GMT">
   <fmt:formatDate value="${now}" type="both" dateStyle="full" timeStyle="full"/>
  </fmt:timeZone>

 <li> Formatting current date as "GMT+1:00", and parsing
      its date and time components:<br>
  <fmt:timeZone value="GMT+1:00">
   <fmt:formatDate value="${now}" type="both" dateStyle="full"
                   timeStyle="full" var="formatted"/>
   <fmt:parseDate value="${formatted}" type="both" dateStyle="full"
                  timeStyle="full" timeZone="PST" var="parsedDateTime"/>
   Parsed date: <fmt:formatDate value="${parsedDateTime}" type="date"
                                dateStyle="full"/><br>
   Parsed time: <fmt:formatDate value="${parsedDateTime}" type="time"
                                timeStyle="full"/>
  </fmt:timeZone>
```

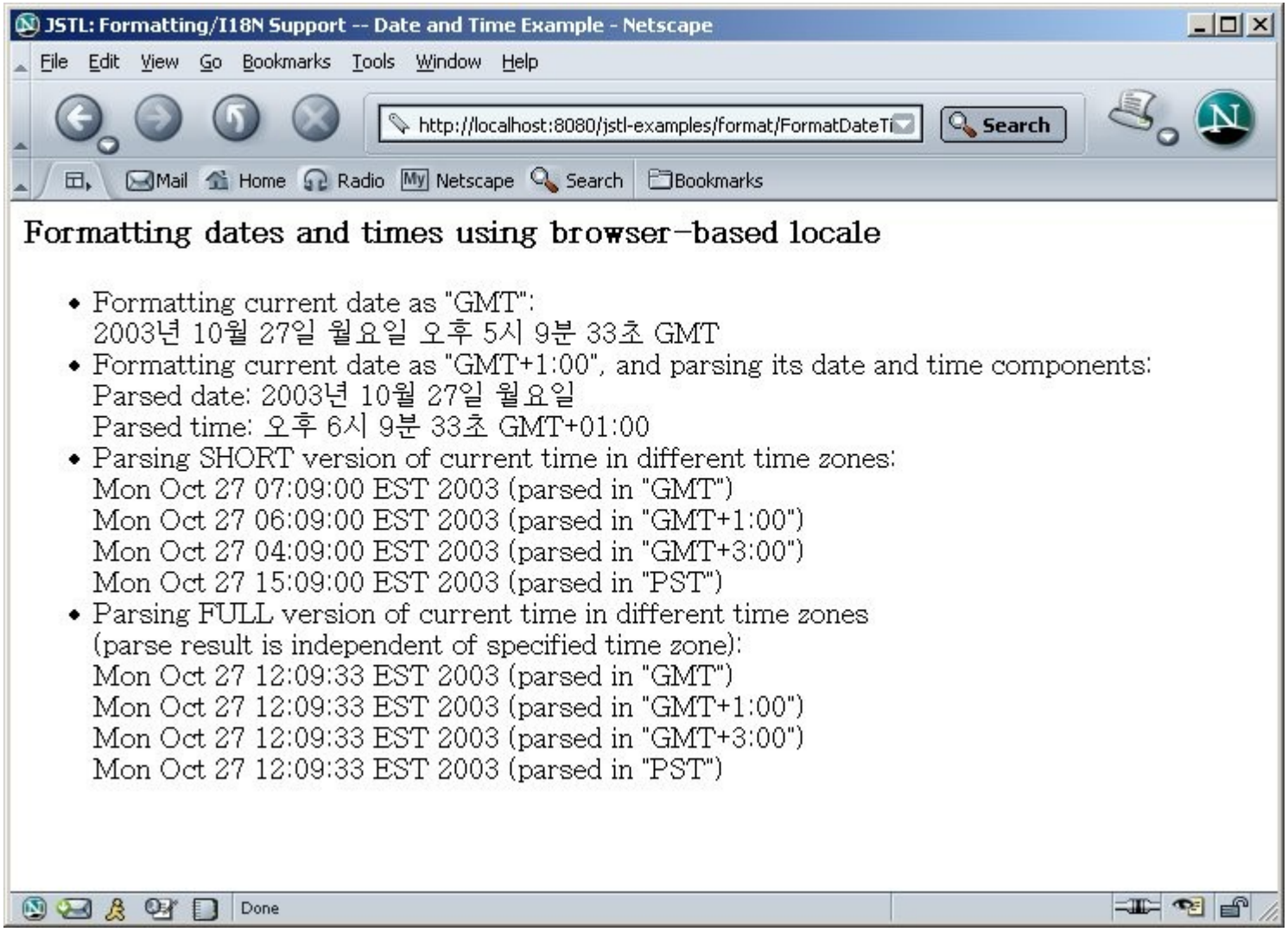
# Example: using browser locale en-us quoted from ./format/FormatDateTime.jsp



# Change Browser Locale preference to Korean



# Example: using browser locale ko quoted from ./format/FormatDateTime.jsp



# Passion!