JSTL
(JSP Standard Tag Library)

Что такое JSTL?

- Стандартный набор библиотек действий
- Содержит базовые действия, которые часто используются в JSP
  - циклы и ветвления
  - обработка XML-документов
  - доступ к базе данных
  - форматирование с учетом интернационализации
- В будущих версиях могут быть добавлены новые действия
- Текущая версия — 1.2

Зачем нужна JSTL?

- Чтобы не разрабатывать эти действия самостоятельно и
- Для обеспечения переносимости приложений
- Достаточно выучить JSTL, так как она входит в состав любого Java EE-сервера приложений
  - Поставщики предоставляют оптимизированные реализации JSTL

Библиотеки действий, входящие в JSTL

- Базовые действия (префикс: **c**)
  - Поддержка переменных, управление потоком выполнения, доступ к ресурсам по URL
- Обработка XML (префикс: **x**)
  - Базовые действия, управление потоком выполнения, XSLT-преобразование
- Форматирование (префикс: **fmt**)
  - Локаль, форматирование сообщений, чисел и дат
- Доступ к базе данных (префикс: **sql**)
  - SQL-запросы
- Функции (префикс: **fn**)
  - Размер коллекции, действия со строками

Подключение библиотек действий JSTL

- Базовые действия
  - <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
- Обработка XML
  - <%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
- Форматирование
  - <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
- Доступ к базе данных (SQL)
  - <%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
- Функции
  - <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

Базовые действия

- Работа с переменными
  - <c:set>
  - <c:remove>
- Ветвления
  - <c:if>
  - <c:choose>
    - <c:when>
    - <c:otherwise>
- Циклы
  - <c:forEach>
  - <c:forTokens>
- Доступ по URL
  - <c:import>
    - <c:param>
  - <c:redirect>
    - <c:param>
  - <c:url>
    - <c:param>
- Общего назначения
  - <c:out>
  - <c:catch>

Работа с переменными, <c:set>

- В JSTL поддерживаются только контекстные переменные (scoped variables), то есть атрибуты какого-либо контекста
  - page, request, session, application
- Действие <c:set> устанавливает
  - значение контекстной переменной (атрибут “var”)
  - или ее свойства (атрибуты “target” и “property”)
  - в любом контексте (атрибут “scope”)
- Если переменная еще не существует, то она создается и сохраняется в заданном контексте

Работа с переменными, <c:set>

- Значение можно задать одним из 2 способов
  - <c:set var="bookId" value="${param.BookID}"
    scope="session" />
  - <c:set target="${cust.address}" property="city">
    Владимир
    </c:set>

Работа с переменными, <c:remove>

- Удаление контекстной переменной
  - <c:remove var="cart" scope="session"/>

Ветвления

- Действия управления потоком выполнения устраняют потребность в скриптлетах
- <c:if test=”..”>
  - Тело действия выполняется, если атрибут "test" имеет логическое значение true
- <c:choose>
  - Условное выполнение вложенных действий <c:when> и <c:otherwise>
  - Работает как оператор if-then-else

Пример: <c:if test=”...”>

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

В цикле <c:forEach> отображаются только заказчики из России.

Пример: <c:choose>, <c:when>

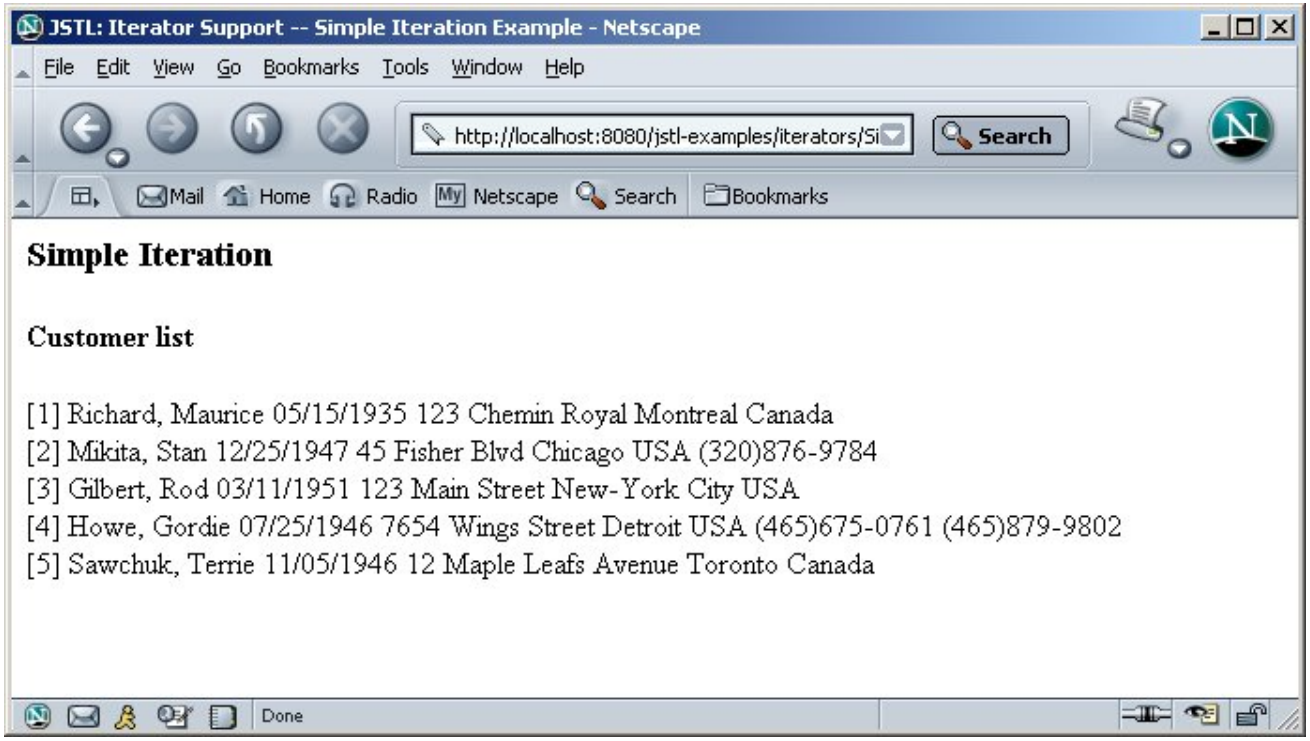
```
<c:forEach var="customer" items="${customers}">
  <c:choose>
    <c:when test="${customer.address.country == 'USA'}">
      <font color="blue">
    </c:when>
    <c:when test="${customer.address.country == 'Canada'}">
      <font color="red">
    </c:when>
    <c:otherwise>
      <font color="green">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

Циклическое действие <c:forEach>

- Позволяет пройти по коллекции объектов
  - items: коллекция объектов
  - var: текущий элемент
  - varStatus: состояние цикла
  - begin, end, step: диапазон и шаг цикла
- Типы коллекций
  - java.util.Collection
  - java.util.Map
    - Значение переменной var будет типа java.util.Map.Entry
  - Массив (любого типа)
  - Итератор (java.util.Iterator или java.util.Enumeration)
  - String — список значений, разделенных запятыми

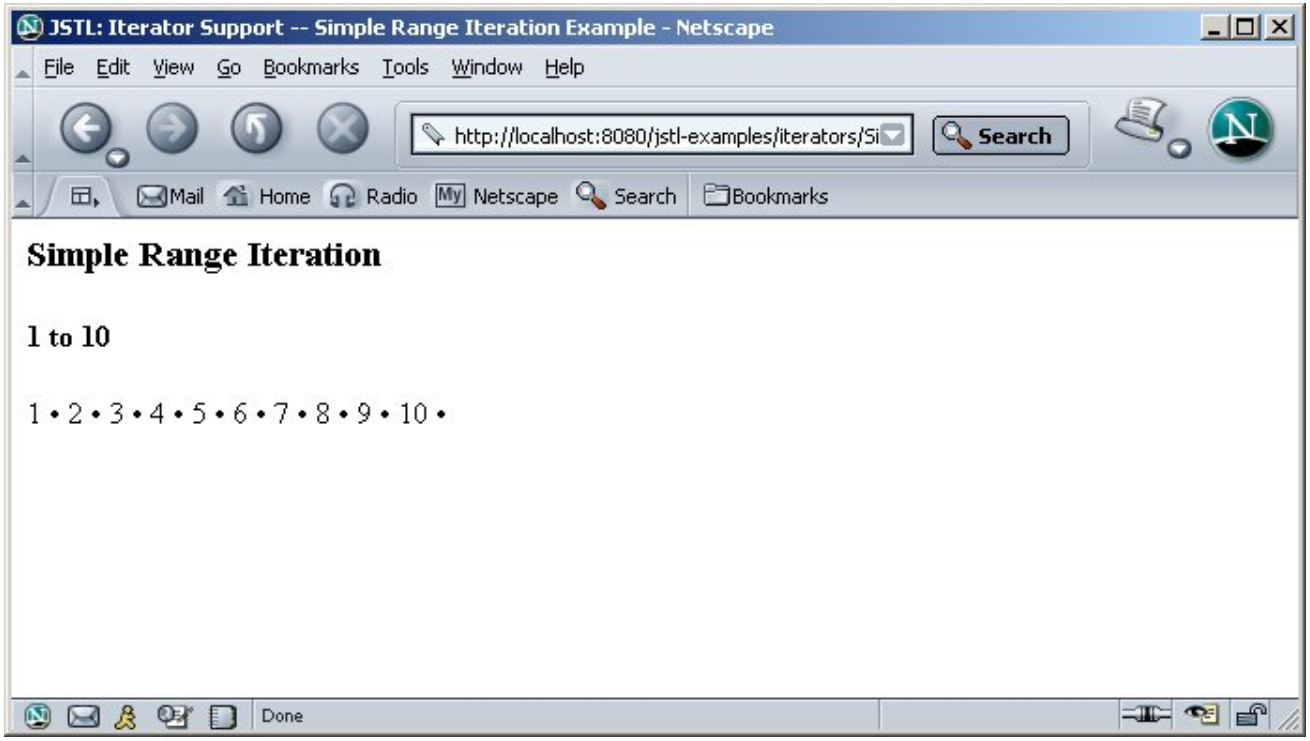
Пример: <c:forEach>

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



Пример: <c:forEach>, диапазон

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



Пример: <c:forEach>, типы коллекций

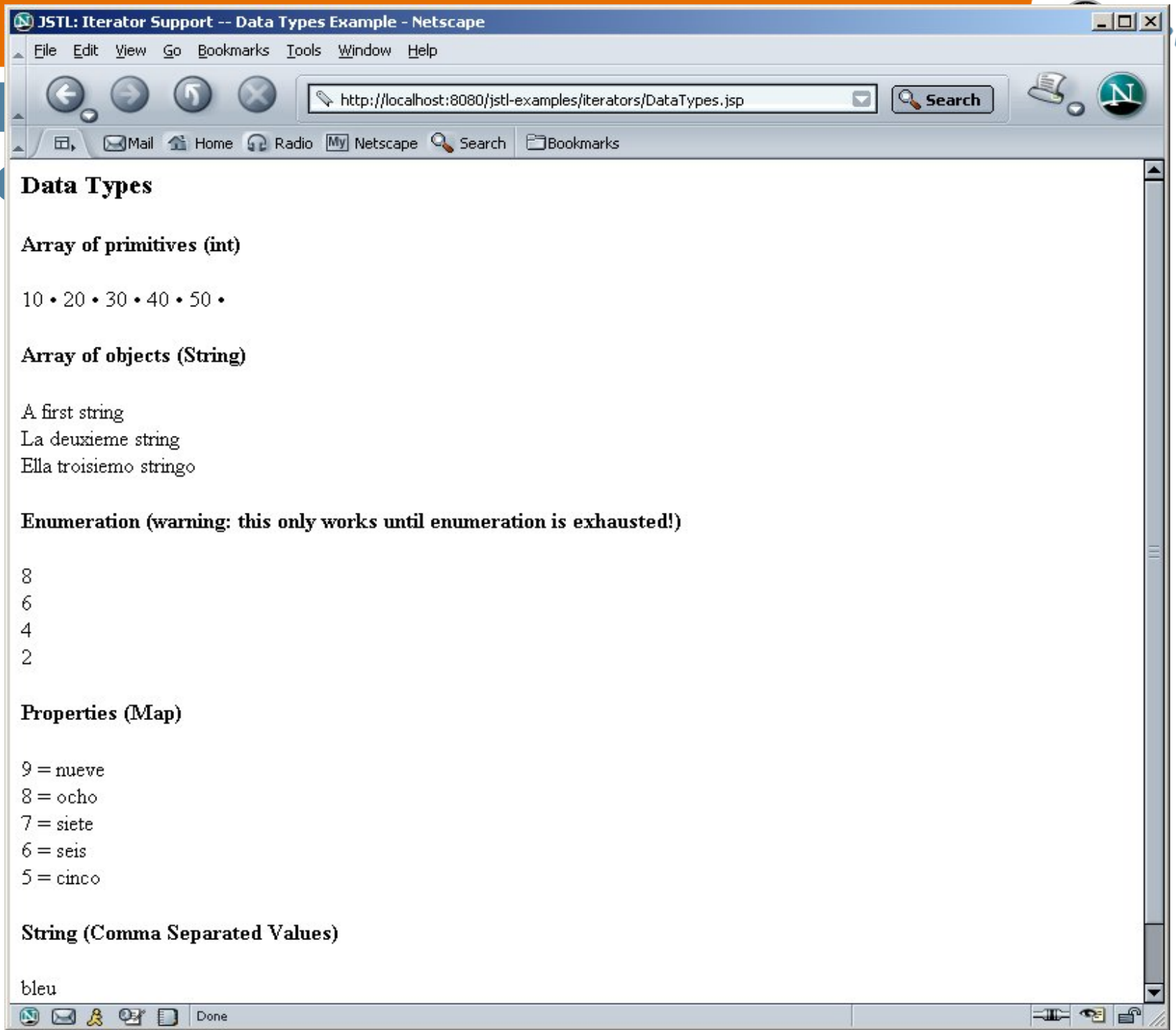
```
<c:forEach var="i" items="${intArray}">
  <c:out value="${i}"/> •
</c:forEach>

<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>

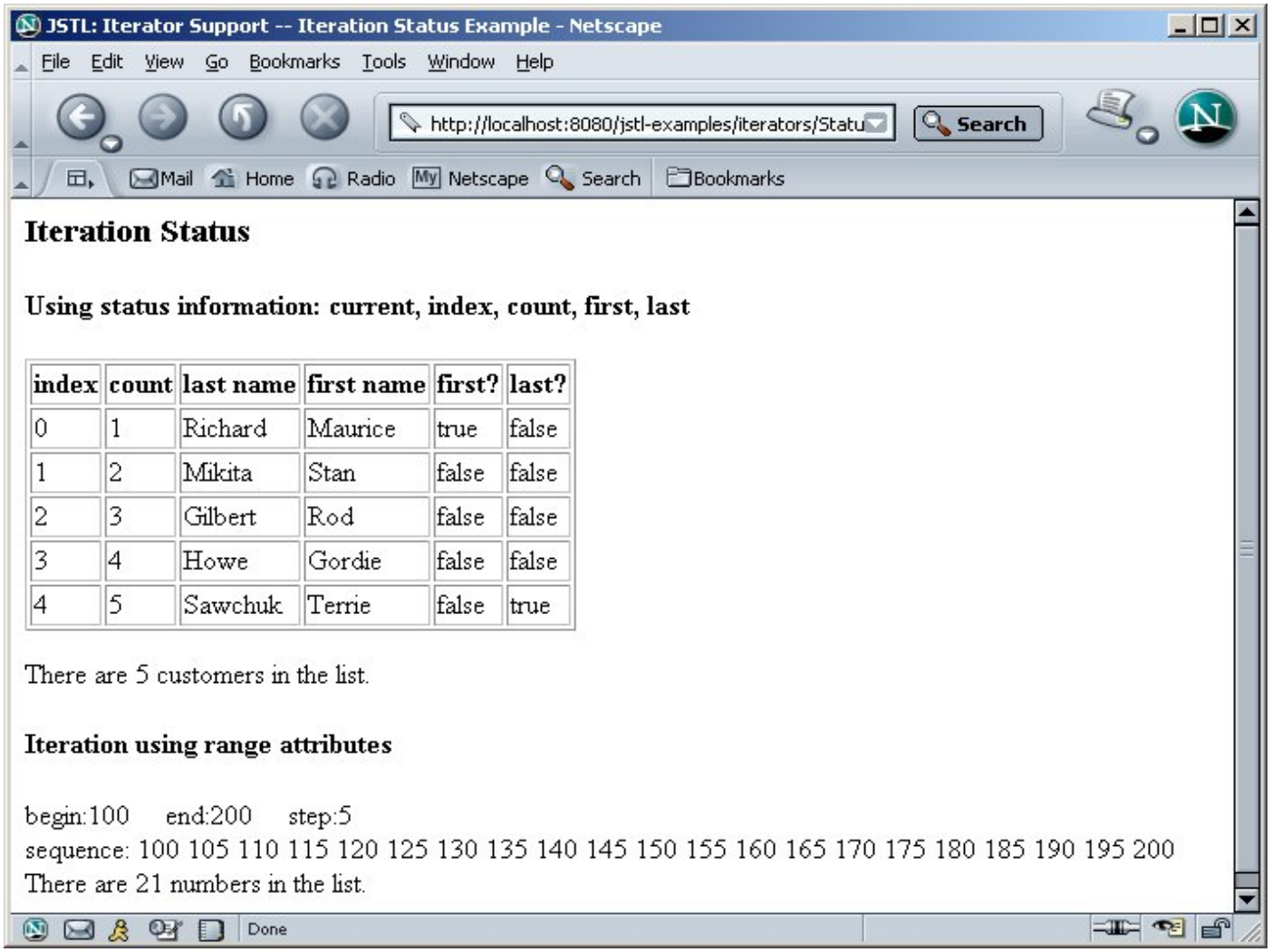
<c:forEach var="token" items="bleu,blanc,rouge">
  <c:out value="${token}"/><br>
</c:forEach>
```


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
Done

Пример: <c:forEach>, состояние цикла

```
<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>
```

Пример: <c:forEach>, состояние цикла



Iteration Status

Using status information: current, index, count, first, last

index	count	last name	first name	first?	last?
0	1	Richard	Maurice	true	false
1	2	Mikita	Stan	false	false
2	3	Gilbert	Rod	false	false
3	4	Howe	Gordie	false	false
4	5	Sawchuk	Terrie	false	true

There are 5 customers in the list.

Iteration using range attributes

begin:100 end:200 step:5
sequence: 100 105 110 115 120 125 130 135 140 145 150 155 160 165 170 175 180 185 190 195 200
There are 21 numbers in the list.

Циклическое действие <c:forTokens>

- Выделение подстрок из заданной строки согласно установленным разделителям
  - items: исходная строка
  - var: текущая подстрока
  - delims: символы-разделители
  - varStatus: состояние цикла
  - begin, end, step: диапазон и шаг цикла
- Пример
  - ```
    <c:forTokens var="token" items="one,two;three" delims=",;">
        <c:out value="${token}"/>
    </c:forTokens>
    ```
    ```

Включение ресурсов по URL: <c:import>

- Позволяет включить произвольный ресурс по URL (в отличие от <jsp:include>)
  - Абсолютный URL: для доступа к внешним ресурсам
  - Относительный URL: для доступа к ресурсам в том же или в указанном веб-приложении
- Производительнее, чем <jsp:include>
  - Нет промежуточной буферизации
- Для передачи параметров используется вложенное действие <c:param> (аналогично <jsp:param>)

Примеры: <c:import>

```
<%-- импорт ресурса с абсолютным URL --%>
<c:import url="http://acme.com/exec/customers?country=Japan"/>

<%-- импорт ресурса с относительным URL в текущем веб-приложении --%>
<c:import url="/copyright.html"/>

<%-- импорт ресурса с относительным URL в другом веб-приложении --%>
<c:import url="/logo.html" context="/master"/>
```

Переписывание и кодирование URL: <c:url>

- Используется для поддержки HTTP-сессий
  - Если в браузере выключены куки, то во всех URL будет передаваться идентификатор сессии
- К относительным URL (начинающимся с /) автоматически добавляется корень веб-приложения
- Можно сохранить URL в переменную
- Можно указать относительный URL в другом веб-приложении
- Для добавления параметров в URL используются вложенные действия <c:param>

Пример: <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" var="myUrl">
         <c:param name="param" value="123"/>
       </c:url>
       ${myUrl}
 </td>
</tr>
<tr>
```

Пример: <c:url>

- Куки включены:

URL Encoding

<c:url>

Disable cookies in your browser to see URL rewriting.

"base", param=ABC	base?param=ABC
"base", param=123	base?param=123
"base", param=&	base?param=%26
"base", param="JSTL is fun"	base?param=JSTL+is+fun

- Куки выключены:

Disable cookies in your browser to see URL rewriting.

"base", param=ABC	base;jsessionid=E5BF7CAEA2ACFAB4B0F7BA6EAB895B1F?param=ABC
"base", param=123	base;jsessionid=E5BF7CAEA2ACFAB4B0F7BA6EAB895B1F?param=123
"base", param=&	base;jsessionid=E5BF7CAEA2ACFAB4B0F7BA6EAB895B1F?param=%26
"base", param="JSTL is fun"	base;jsessionid=E5BF7CAEA2ACFAB4B0F7BA6EAB895B1F?param=JSTL+is+fun

Перенаправление: <c:redirect>

- Отправка клиенту HTTP-ответа с перенаправлением запроса
- Для добавления параметров в возвращаемый URL используются вложенные действия <c:param>
- Пример:
  - `<c:redirect url="http://acme.com/register"/>`

<c:out>

- Вычисляет выражение и выводит результат в текущий объект JspWriter
- Если результат - объект java.io.Reader, данные из него копируются в текущий объект JspWriter
  - Улучшенная производительность
- <c:out value="value" [escapeXml="{true|false}"] [default="defaultValue"] />
  - Если escapeXml == true, то символы <, >, ', ", & преобразуются в соответствующие сущности (например, < преобразуется в <)
    - Защита от script injection
  - Значение default выводится, если value == null

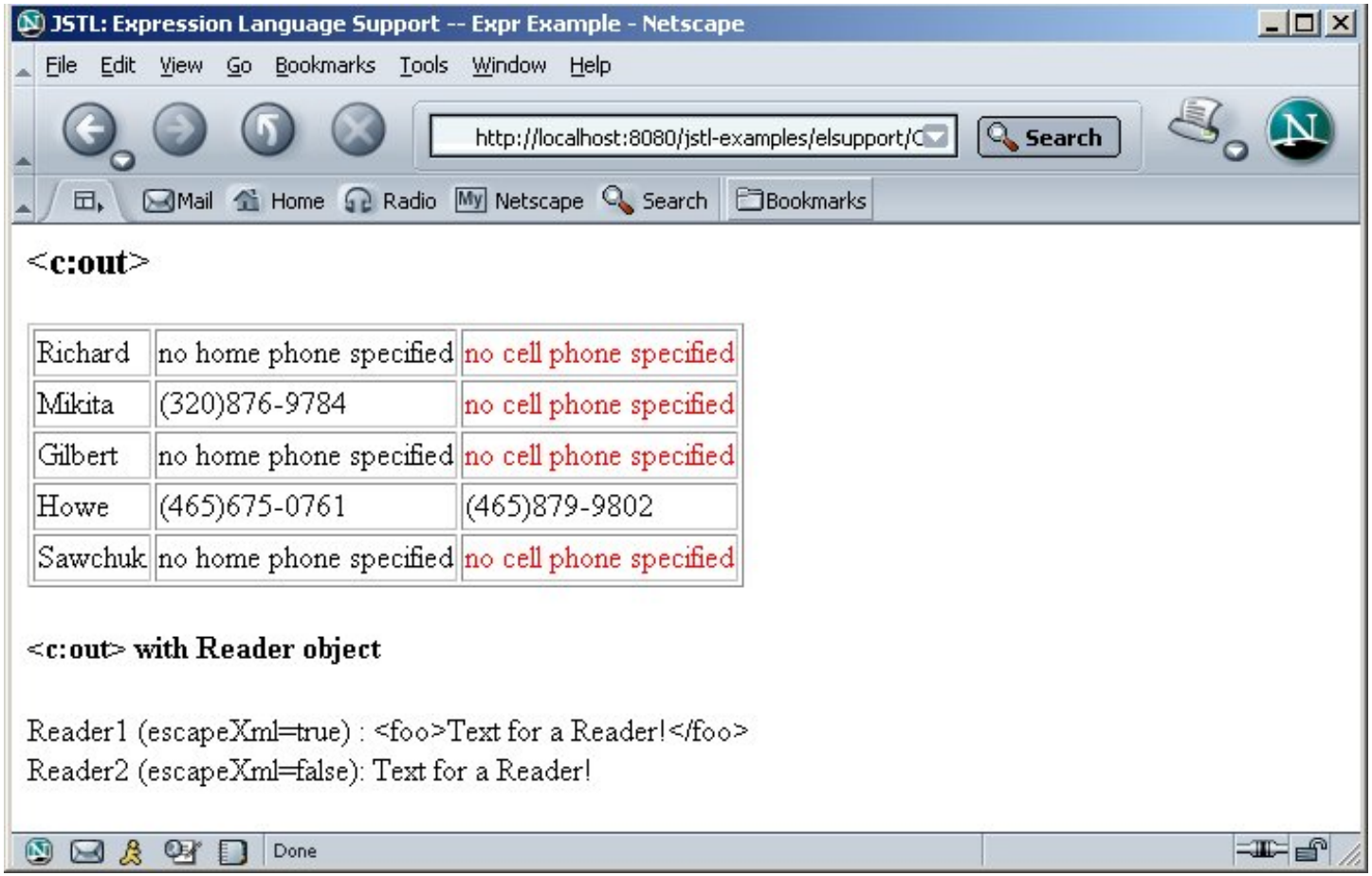
Пример: <c:out>

```
<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>
```

Пример: <c:out>

```
<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>
```

Пример: <c:out>


JSTL: Expression Language Support -- Expr Example - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jstl-examples/elsupport/C
Search
Mail Home Radio Netscape Search Bookmarks
<c:out>
Richard | no home phone specified | no cell phone specified
Mikita | (320)876-9784 | no cell phone specified
Gilbert | no home phone specified | no cell phone specified
Howe | (465)675-0761 | (465)879-9802
Sawchuk | no home phone specified | no cell phone specified
<c:out> with Reader object
Reader1 (escapeXml=true) : <foo>Text for a Reader!</foo>
Reader2 (escapeXml=false): Text for a Reader!
Done

Стандартный набор EL-функций

- <fn:length> Размер коллекции или длина строки
- <fn:toUpperCase>, <fn:toLowerCase> Изменение регистра строки
- <fn:substring>, <fn:substringBefore>, <fn:substringAfter> Получение подстроки
- <fn:trim> Удаление начальных и конечных пробельных символов из строки
- <fn:replace> Замена символов в строке
- <fn:indexOf>, <fn:startsWith>, <fn:endsWith contains>, <fn:containsIgnoreCase> Проверка наличия подстроки в строке
- <fn:split>, <fn:join> Разбивка строки на массив подстрок и соединение массива подстрок в строку
- <fn:escapeXml> Преобразование служебных символов XML в сущности

Примеры EL-функций

```
<%-- получение первых 30 символов строки и перевод их в верхний регистр --%>
${fn:toUpperCase(fn:substring(name, 0, 30))}
<%-- значение переменной text до первого символа '*' --%>
${fn:substringBefore(text, '*')}
<%-- удаление лишних пробелов из контекстной переменной custId, чтобы URL не
содержал лишних символов '+' --%>
<c:url var="myUrl" value="${base}/cust">
    <c:param name="custId" value="${fn:trim(custId)}"/>
</c:url>
<%-- вывод текста, заключенного в скобках --%>
${fn:substring(text, fn:indexOf(text, '(')+1, fn:indexOf(text, ')'))}
<%-- вывод переменной name, если она содержит искомую строку --%>
<c:if test="${fn:containsIgnoreCase(name, searchString)}">
    Found name: ${name}
</c:if>
<%-- вывод последних 10 символов переменной text --%>
${fn:substring(text, fn:length(text)-10)}
<%-- вывод значения переменной text с точками вместо '-' --%>
${fn:replace(text, '-', '&#149;')}
```

Поддержка отложенных EL-выражений в JSTL

- В действии <c:set>

```
<c:set var="d" value="#{handler.everythingDisabled}"/>
...
<h:inputText id="i1" disabled="#{d}"/>
<h:inputText id="i2" disabled="#{d}"/>
```

- В действиях <c:forEach> и <c:forTokens>
  - Позволяет формировать таблицы для ввода/вывода данных, аналогичные <h:dataTable>

```
<c:forEach items="#{bestScoresBean.bestScores}" var="item">
    <tr>
        <td><h:inputText value="#{item.guessCount}"/></td>
        <td><h:inputText value="#{item.playerName}"/></td>
        <td><h:outputText value="#{item.date}"/></td>
    </tr>
</c:forEach>
```