B.Sc. In Software Development. Year 4.
Semester I. Enterprise Development.

Using Expression Language (EL) and JSP Standard Tag Library (JSTL) to enhance MVC.

LIMERICK INSTITUTE OF TECHNOLOGY
SCHOOL OF SCIENCE, ENGINEERING & I.T.
Department of Information Technology

Using The JSP Expression Language (EL)

- The JSP *Expression Language* (*EL*) provides a compact syntax that lets you get data from JavaBeans, maps, arrays, and lists that have been stored as attributes of a web application.

Advantages of EL

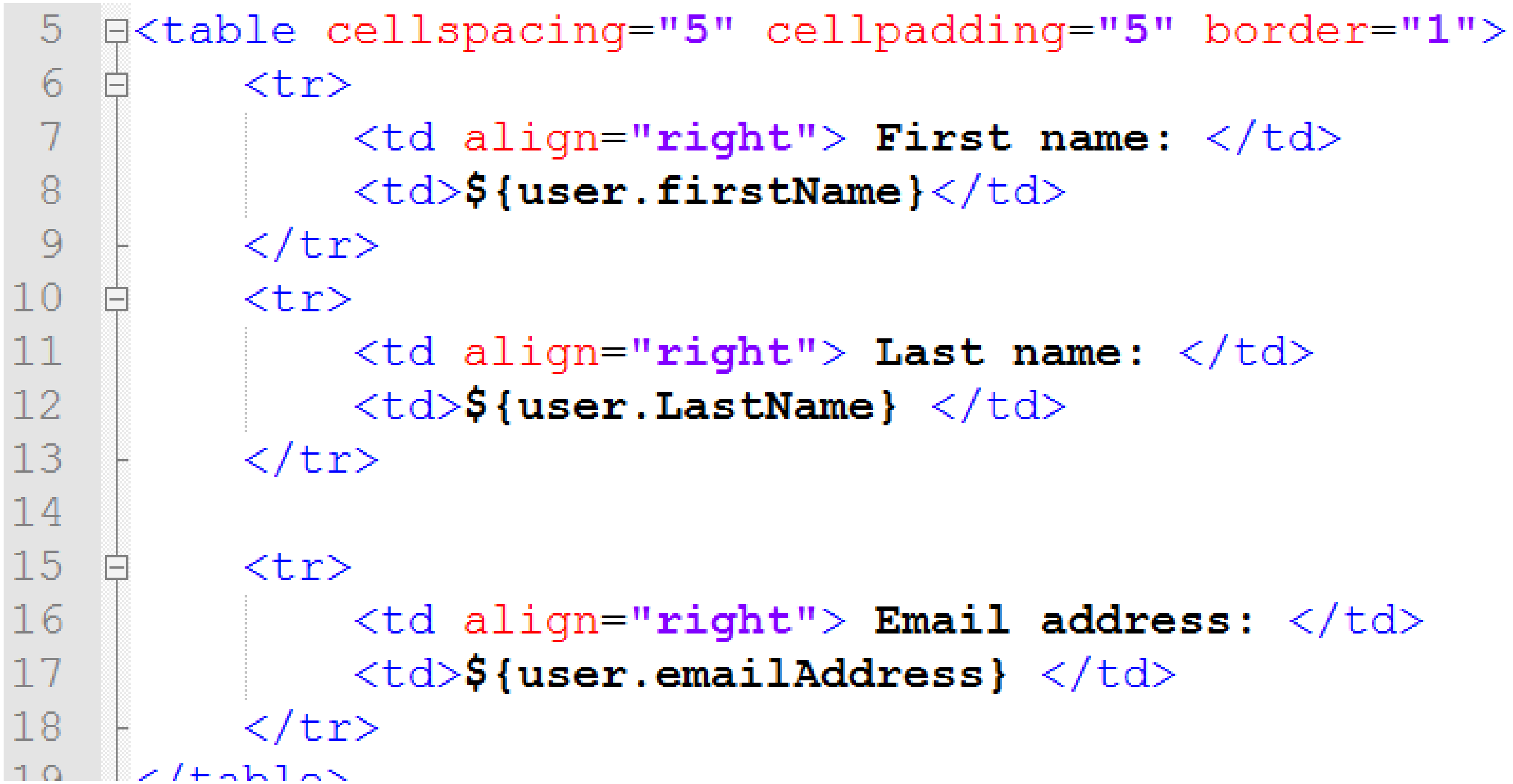
- EL has a more elegant and compact syntax than standard JSP tags.
- EL lets you access nested properties.
- EL lets you access collections such as maps, arrays, and lists.

Disadvantages of EL

- EL doesn’t create a JavaBean if it doesn’t already exist.
- EL doesn’t provide a way to set properties.

Using The JSP Expression Language (EL)

- Example of a JSP that uses EL to access a User object named *user* that has been stored in the session object.



```
<table cellspacing="5" cellpadding="5" border="1">
    <tr>
        <td align="right"> First name: </td>
        <td>${user.firstName}</td>
    </tr>
    <tr>
        <td align="right"> Last name: </td>
        <td>${user.LastName} </td>
    </tr>

    <tr>
        <td align="right"> Email address: </td>
        <td>${user.emailAddress} </td>
    </tr>
</table>
```

Using The JSP Expression Language (EL)

- The same JSP using standard JSP tags.

```
<jsp:useBean id="user" scope="session" class="business.User"></jsp:useBean>

<table cellspacing="5" cellpadding="5" border="1">
    <tr>
        <td align="right"> First name: </td>
        <td><jsp:getProperty property="firstName" name="user"/></td>
    </tr>
    <tr>
        <td align="right"> Last name: </td>
         <td><jsp:getProperty property="lastName" name="user"/></td>
    </tr>

    <tr>
        <td align="right"> Email address: </td>
         <td><jsp:getProperty property="emailAddress" name="user"/></td>
    </tr>
</table>
```

Using The JSP Expression Language (EL)

- An example that accesses an attribute named currentDate.

- **Syntax**

```
${attribute}
```

- **Servlet code**

```
Date currentDate = new Date();
request.setAttribute("currentDate", currentDate);
```

- **JSP code**

```
<p>The current date is ${currentDate}</p>
```

Using The JSP Expression Language (EL)

- An example that accesses the firstName property of an attribute named user

- **Syntax**

```
${attribute.property}
```

- **Servlet code**

```
User user =
    new User(firstName, lastName, emailAddress);
session.setAttribute("user", user);
```

- **JSP code**

```
<p>Hello ${user.firstName}</p>
```

Other Implicit Objects To Use With EL.

Other implicit objects include

paramValues

param

header

headerValues

initParam

cookie

pageContext

param and paramValues

Implicit Object	Type	Description
param	map	Used to get the request parameter value, returns a single value
paramValues	map	Used to get the request param values in an array, useful when request parameter contain multiple values.

```html
<tr>
    <td align="right"> Last name: </td>
    <td><input type="text" name="lastName"> </td>
</tr>

<tr>
    <td align="right"> Primary email address: </td>
    <td><input type="text" name="emailAddress"> </td>
</tr>

                <tr>
    <td align="right"> Alternate email address: </td>
    <td><input type="text" name="emailAddress"> </td>
</tr>
```

HTML

```jsp
Last name: ${param.LastName} <br>
Primary Email address: ${paramValues.emailAddress[0]} <br>
Alternate Email address: ${paramValues.emailAddress[1]} <br>
```

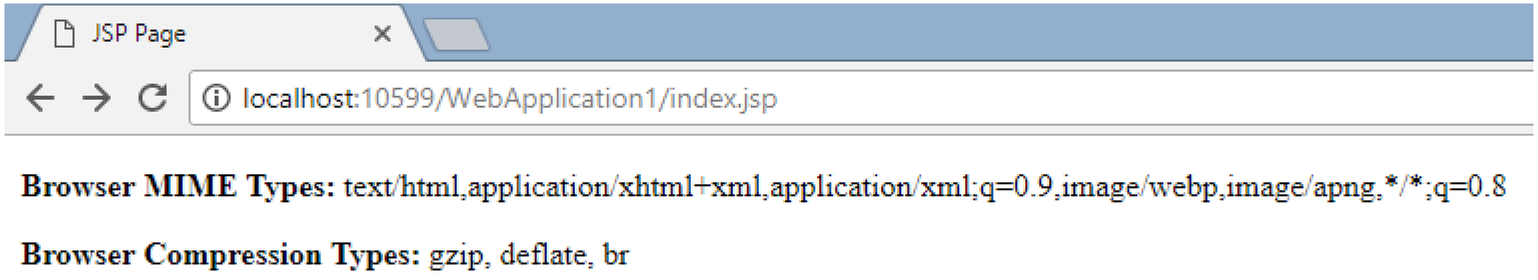
JSP

header and headerValues

Implicit Object	Type	Description
header	map	Used to get request header information.
headerValues	map	Used to get header values in an array.

JSP

```
<p><b>Browser MIME Types:</b> ${header.accept}</p>
<p><b>Browser Compression Types:</b>  ${header["accept-encoding"]}  </p>
```



Browser MIME Types: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8

Browser Compression Types: gzip, deflate, br

cookie

Implicit Object	Type	Description
cookie	map	Used to get the cookie value in the JSP

Servlet

```
Cookie c = new Cookie("emailAddress", request.getParameter("emailAddress"));

c.setMaxAge(60 * 60);

response.addCookie(c);
```

JSP

```
Email Address in Cookie is : ${cookie.emailAddress.value} <br>
```

initParam

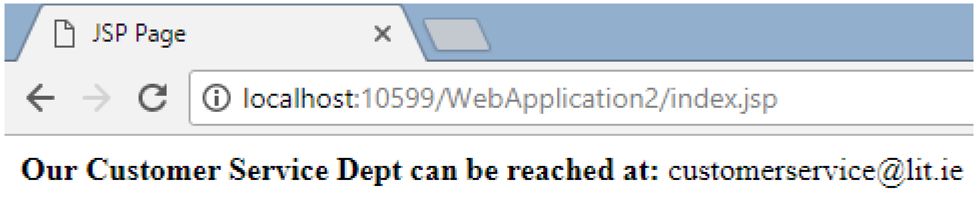
Implicit Object	Type	Description
initParam	map	Used to get the context init params (from web.xml)

web.xml

```
<context-param>
    <param-name>customerServiceEmail</param-name>
    <param-value>customerservice@lit.ie</param-value>
</context-param>
```

JSP

```
<b>Our Customer Service Dept can be reached at: </b> ${initParam.customerServiceEmail}
```



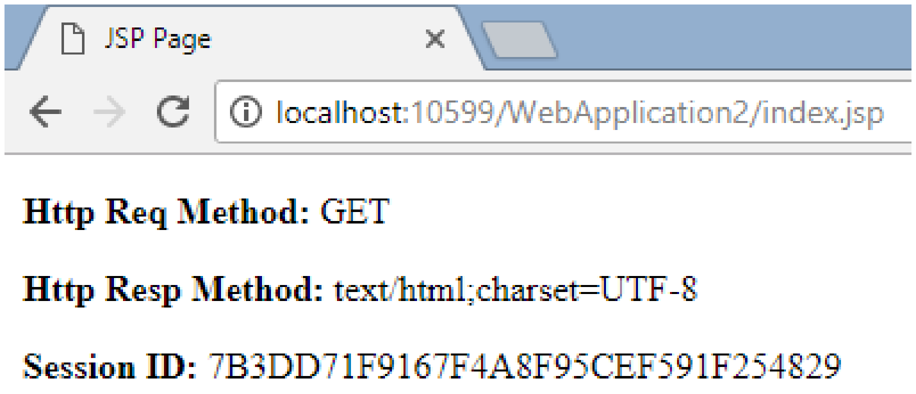
Our Customer Service Dept can be reached at: customerservice@lit.ie

Other Implicit Objects To Use With EL: pageContext

Implicit Object	Type	Description
pageContext	map	The JSP PageContext object for the current page

JSP

```
<p><b>Http Req Method:</b> ${pageContext.request.method}</p>
<p><b>Http Resp Method:</b> ${pageContext.response.contentType}</p>
<p><b>Session ID:</b> ${pageContext.session.id}</p>
```



Http Req Method: GET

Http Resp Method: text/html;charset=UTF-8

Session ID: 7B3DD71F9167F4A8F95CEF591F254829

Arithmetic EL Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/ *or* div	Division
% *or* mod	Modulus

Example	Result
${10 + 2}	12
${10 – 2}	8
${10 * 2}	20
${10 / 2} ${10 div 2}	5

Arithmetic EL Operators

Example	Result
${10 % 2} ${10 mod 2}	0
${(10 + 2) * 4}	48
${x + 2}	Assuming x = 10, then the result is = 12

Relational EL Operators

Operator	Description
== *or* eq	Equal to
!= *or* ne	Not equal to
< *or* lt	Less than
> *or* gt	Greater than
<= *or* le	Less than or equal to
>= *or* ge	Greater than or equal to

Example	Result
${“test” == “test”} ${“test” eq “test”} ${5 == 5}	true
${5 != 5} ${5 ne 5}	false

Relational EL Operators

Example	Result
${4 < 5} ${4 lt 5}	true
${4 > 5} ${4 gt 5}	false
${user.firstName == null}	true if firstName is null
${user.firstName == “Tom”}	True if firstName has a value of “Tom”
${flag == true}	True if flag is true (otherwise false)

Logical EL Operators

Operator	Description
&& *or* and	and
!! *or* or	or
! *or* not	not

Example	Result
${“test” == “test” && 4 > 3} ${“test” == “test” and 4 > 3} ${“test” != “test” || 4 < 3}	true
${!true} ${not true}	false

Other EL Operators

Syntax	Description
empty x	Returns true if the value of x is null or equal to an empty string.
X ? y : z	If x evaluates to true, returns y, otherwise returns z.

Example	Result
${empty firstName}	True if firstName returns a null value or an empty string.
${true ? “s1” : “s2”}	s1
${false ? “s1” : “s2”}	s2

The JSTL Library

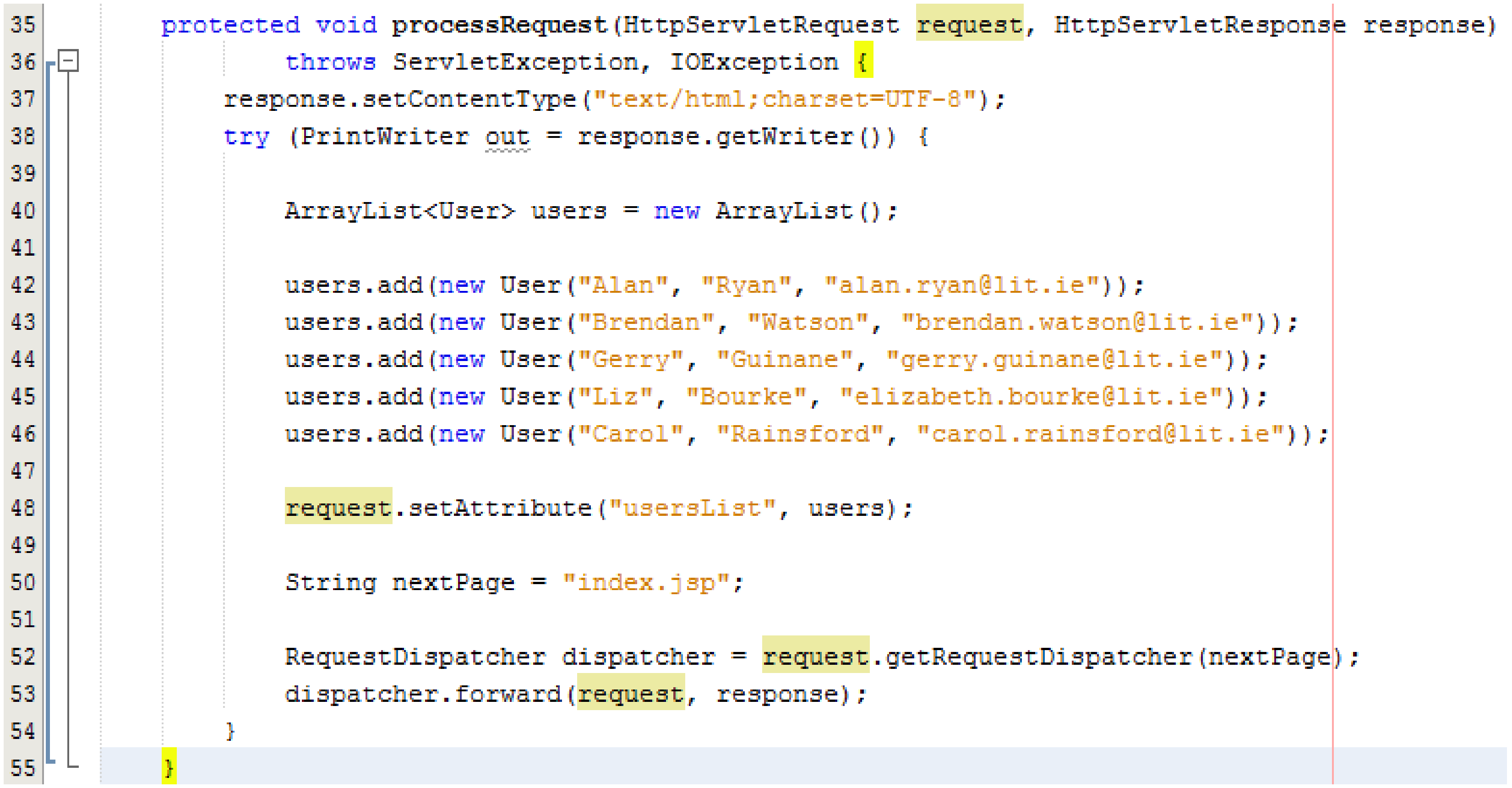
- Use the JSTL (JSP Standard Tag Library) core library in combination with EL to remove all Java code from your JSPs.
- There are five core JSTL libraries.

Name	Prefix	Description
Core	c	Contains core tags for tasks such as looping and selection statements.
Formatting	fmt	Contains tags for formatting numbers, times and dates. Also I18N support.
SQL	sql	Contains tags for working with SQL queries.
XML	x	Contains tags for working with XML documents
Functions	fn	Contains tags for manipulating tags.

JSTL: for each

- The for each tag is used to loop through items that are stored in most collections (Lists, ArrayLists, Vectors) including arrays.

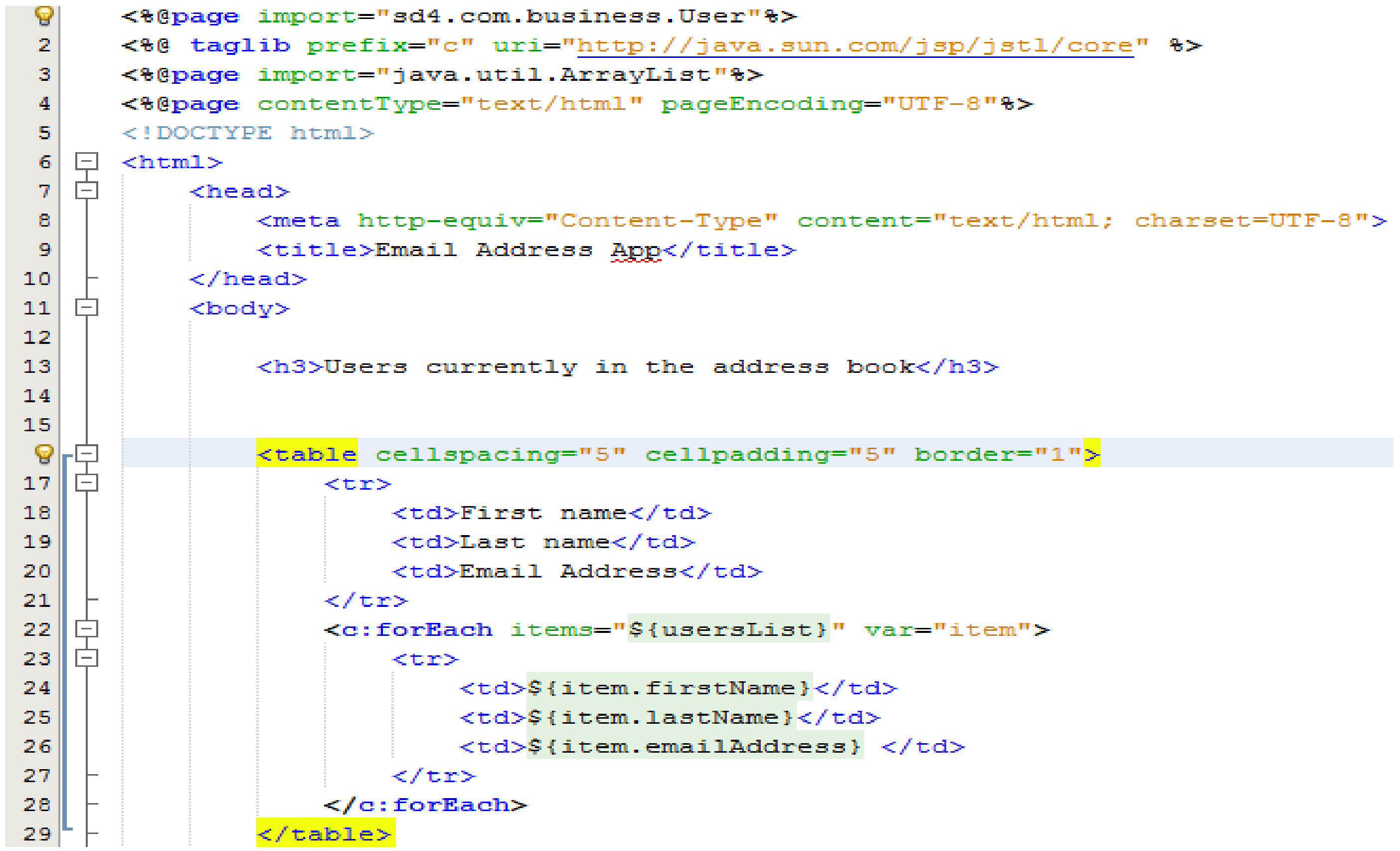
Servlet



```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {

        ArrayList<User> users = new ArrayList();

        users.add(new User("Alan", "Ryan", "alan.ryan@lit.ie"));
        users.add(new User("Brendan", "Watson", "brendan.watson@lit.ie"));
        users.add(new User("Gerry", "Guinane", "gerry.guinane@lit.ie"));
        users.add(new User("Liz", "Bourke", "elizabeth.bourke@lit.ie"));
        users.add(new User("Carol", "Rainsford", "carol.rainsford@lit.ie"));

        request.setAttribute("usersList", users);

        String nextPage = "index.jsp";

        RequestDispatcher dispatcher = request.getRequestDispatcher(nextPage);
        dispatcher.forward(request, response);
    }
}
```

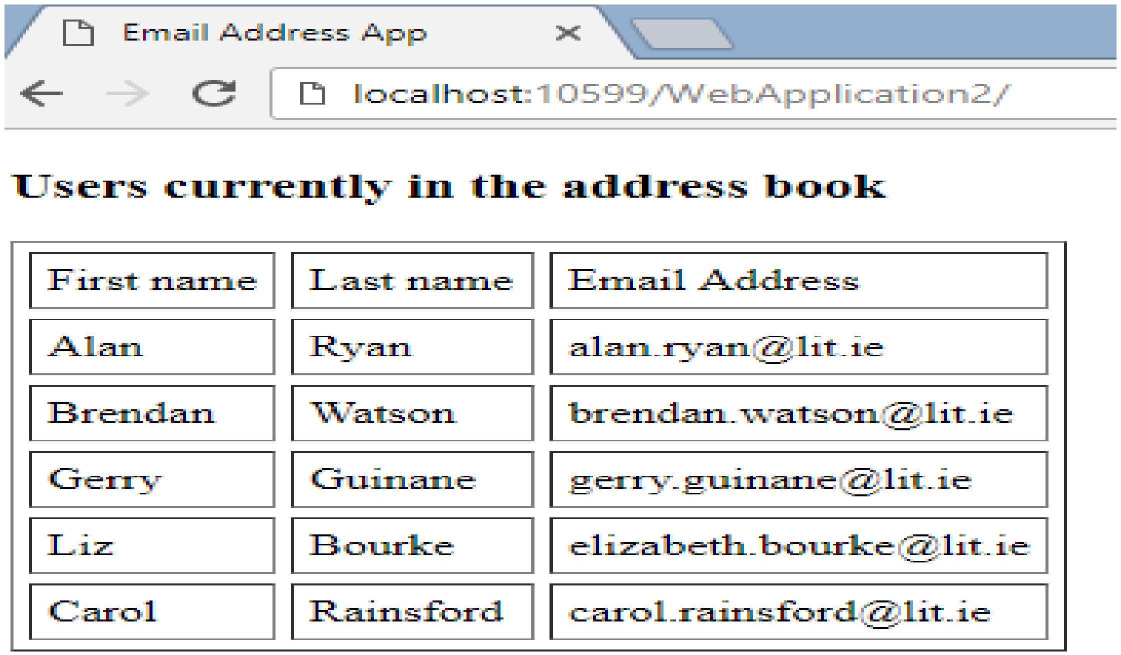
JSTL: for each

JSP



```
<%@page import="sd4.com.business.User"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@page import="java.util.ArrayList"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Email Address App</title>
    </head>
    <body>

        <h3>Users currently in the address book</h3>


        <table cellspacing="5" cellpadding="5" border="1">
            <tr>
                <td>First name</td>
                <td>Last name</td>
                <td>Email Address</td>
            </tr>
            <c:forEach items="${usersList}" var="item">
                <tr>
                    <td>${item.firstName}</td>
                    <td>${item.lastName}</td>
                    <td>${item.emailAddress} </td>
                </tr>
            </c:forEach>
        </table>
```

JSTL: for each

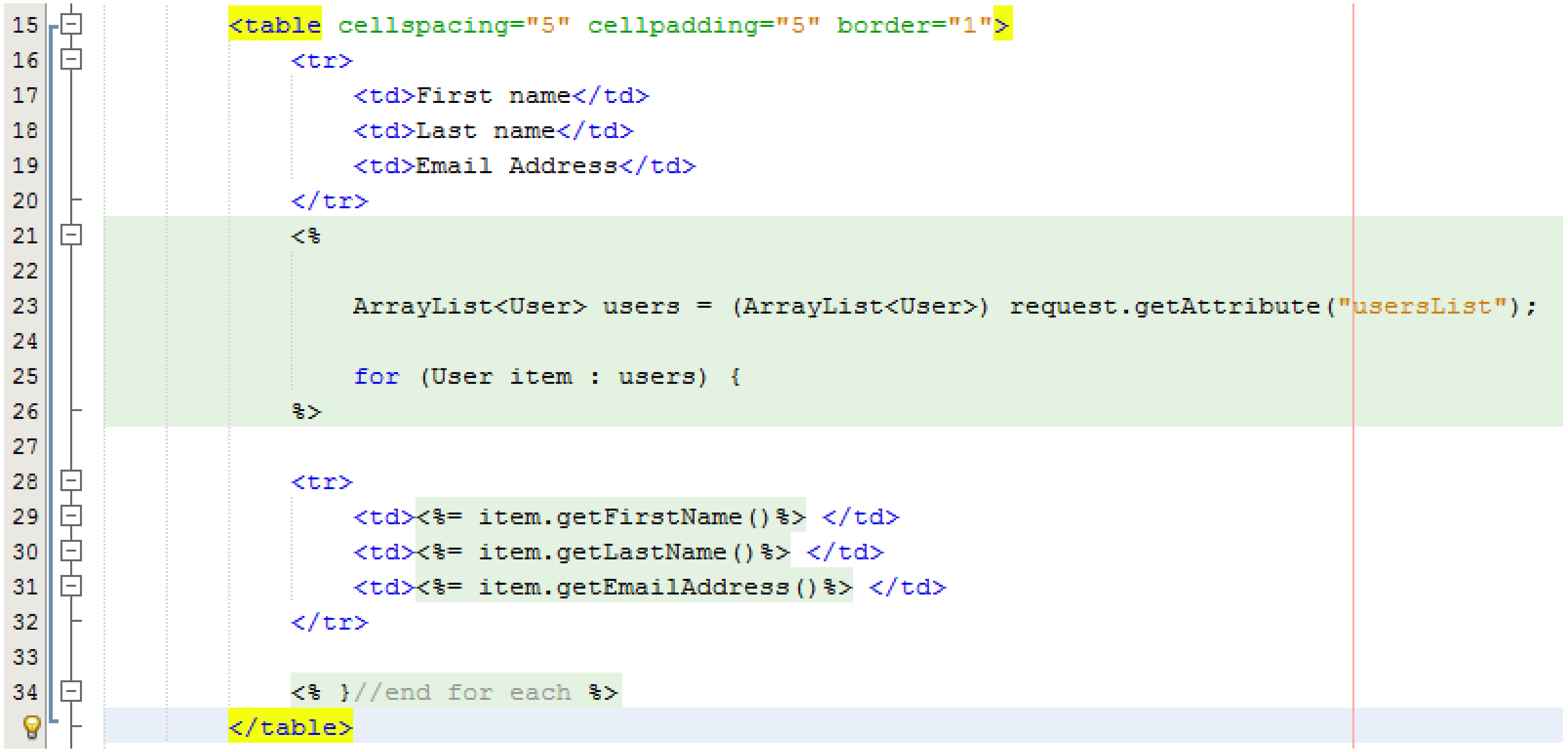


Users currently in the address book

First name	Last name	Email Address
Alan	Ryan	alan.ryan@lit.ie
Brendan	Watson	brendan.watson@lit.ie
Gerry	Guinane	gerry.guinane@lit.ie
Liz	Bourke	elizabeth.bourke@lit.ie
Carol	Rainsford	carol.rainsford@lit.ie

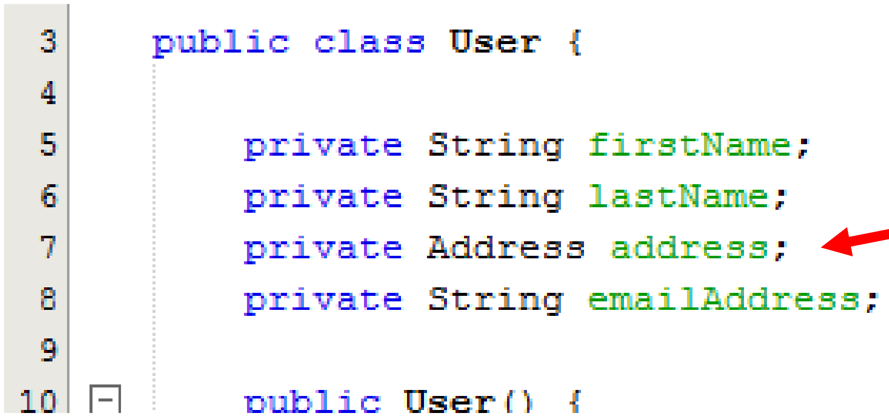
JSTL: for each

Without using JSTL and EL



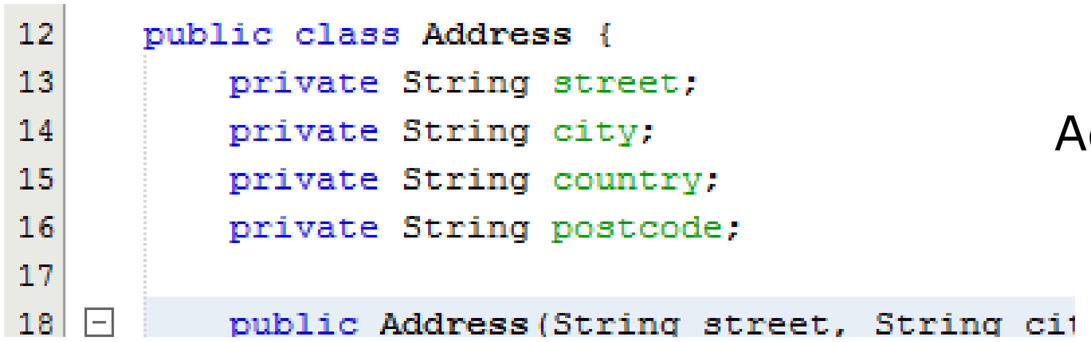
```
<table cellspacing="5" cellpadding="5" border="1">
    <tr>
        <td>First name</td>
        <td>Last name</td>
        <td>Email Address</td>
    </tr>
    <%

        ArrayList<User> users = (ArrayList<User>) request.getAttribute("usersList");

        for (User item : users) {
    %>

    <tr>
        <td><%= item.getFirstName()%> </td>
        <td><%= item.getLastName()%> </td>
        <td><%= item.getEmailAddress()%> </td>
    </tr>

    <% }//end for each %>
</table>
```

JSTL: for each. Drilling down for properties



```
public class User {

    private String firstName;
    private String lastName;
    private Address address;
    private String emailAddress;

    public User() {
```

Property added to User class

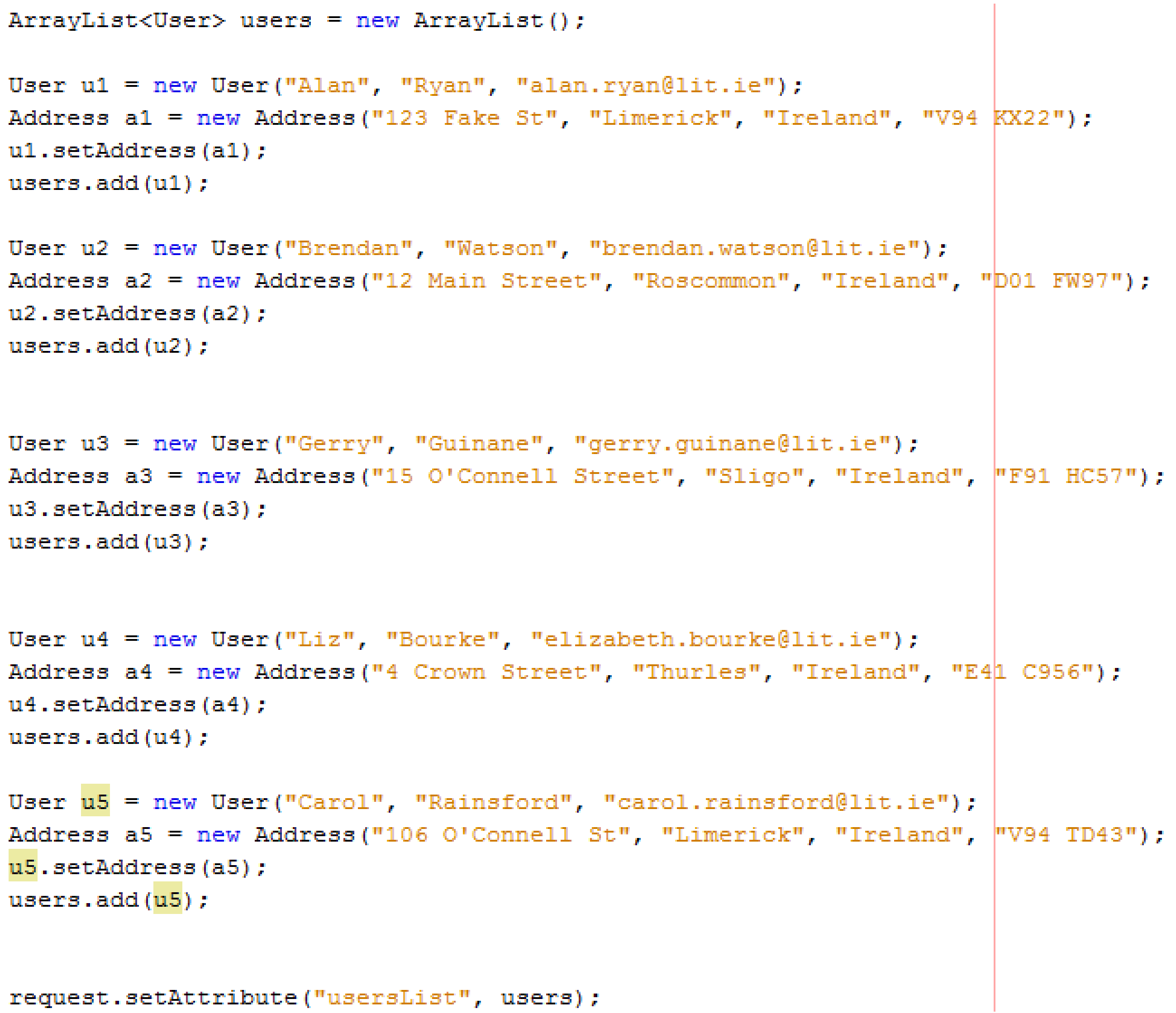


```
public class Address {
    private String street;
    private String city;
    private String country;
    private String postcode;

    public Address(String street, String cit
```

Address class

JSTL: for each. Drilling down for properties

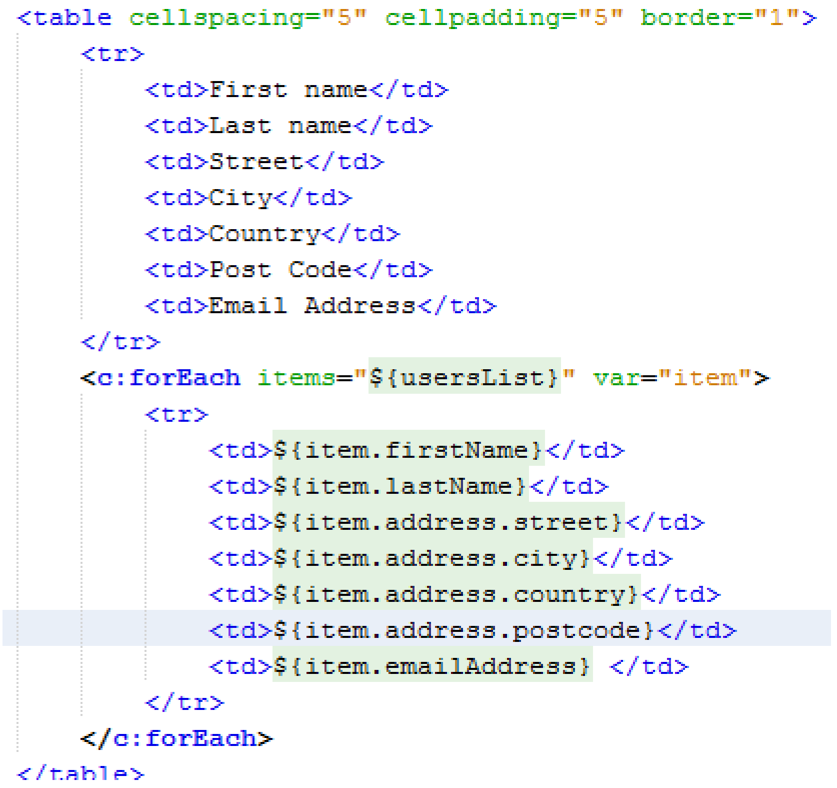
Partial listing for the Servlet



```java
ArrayList<User> users = new ArrayList();

User u1 = new User("Alan", "Ryan", "alan.ryan@lit.ie");
Address a1 = new Address("123 Fake St", "Limerick", "Ireland", "V94 KX22");
u1.setAddress(a1);
users.add(u1);

User u2 = new User("Brendan", "Watson", "brendan.watson@lit.ie");
Address a2 = new Address("12 Main Street", "Roscommon", "Ireland", "D01 FW97");
u2.setAddress(a2);
users.add(u2);


User u3 = new User("Gerry", "Guinane", "gerry.guinane@lit.ie");
Address a3 = new Address("15 O'Connell Street", "Sligo", "Ireland", "F91 HC57");
u3.setAddress(a3);
users.add(u3);


User u4 = new User("Liz", "Bourke", "elizabeth.bourke@lit.ie");
Address a4 = new Address("4 Crown Street", "Thurles", "Ireland", "E41 C956");
u4.setAddress(a4);
users.add(u4);

User u5 = new User("Carol", "Rainsford", "carol.rainsford@lit.ie");
Address a5 = new Address("106 O'Connell St", "Limerick", "Ireland", "V94 TD43");
u5.setAddress(a5);
users.add(u5);


request.setAttribute("usersList", users);
```

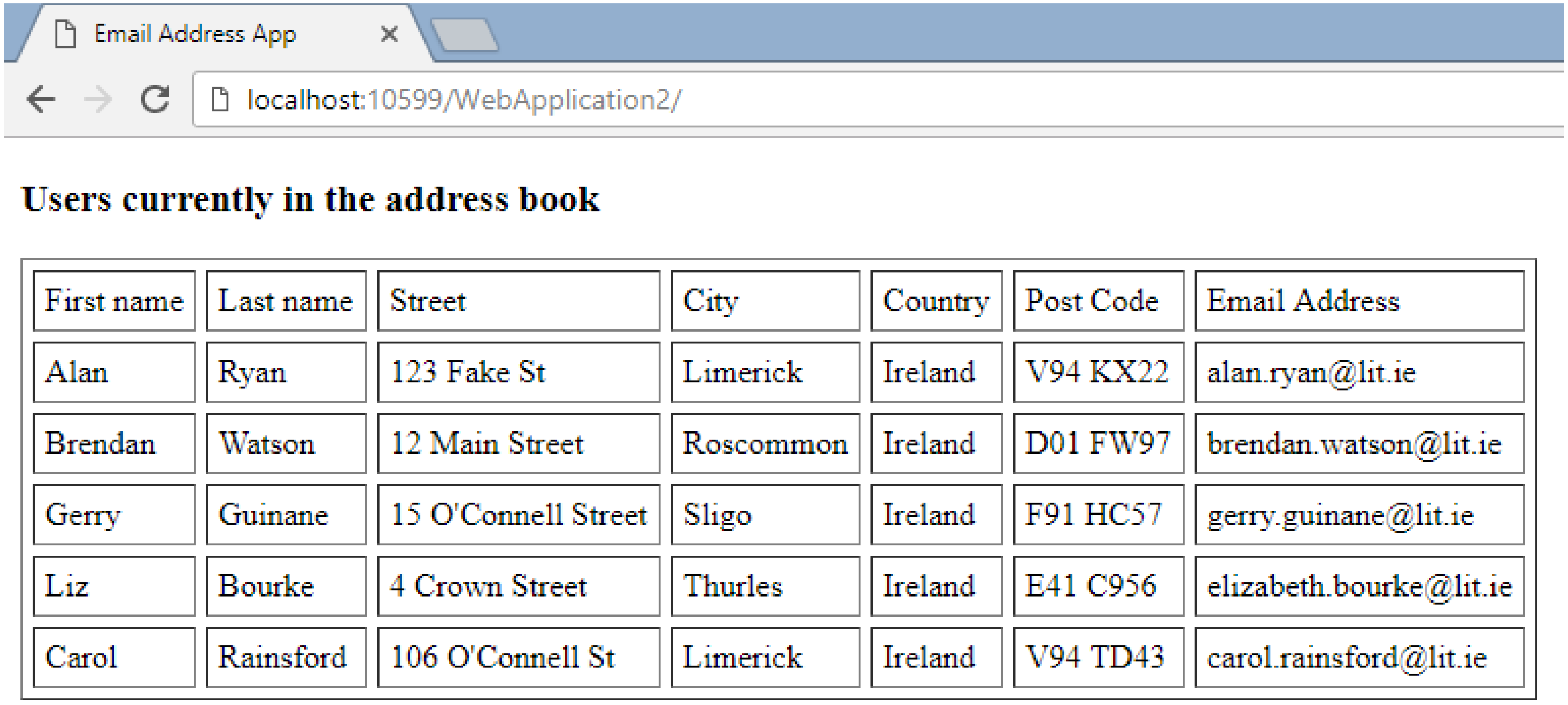
JSTL: for each. Drilling down for properties

Partial listing for the JSP



```
<table cellspacing="5" cellpadding="5" border="1">
    <tr>
        <td>First name</td>
        <td>Last name</td>
        <td>Street</td>
        <td>City</td>
        <td>Country</td>
        <td>Post Code</td>
        <td>Email Address</td>
    </tr>
    <c:forEach items="${usersList}" var="item">
        <tr>
            <td>${item.firstName}</td>
            <td>${item.lastName}</td>
            <td>${item.address.street}</td>
            <td>${item.address.city}</td>
            <td>${item.address.country}</td>
            <td>${item.address.postcode}</td>
            <td>${item.emailAddress} </td>
        </tr>
    </c:forEach>
</table>
```

JSTL: for each. Drilling down for properties


Email Address App

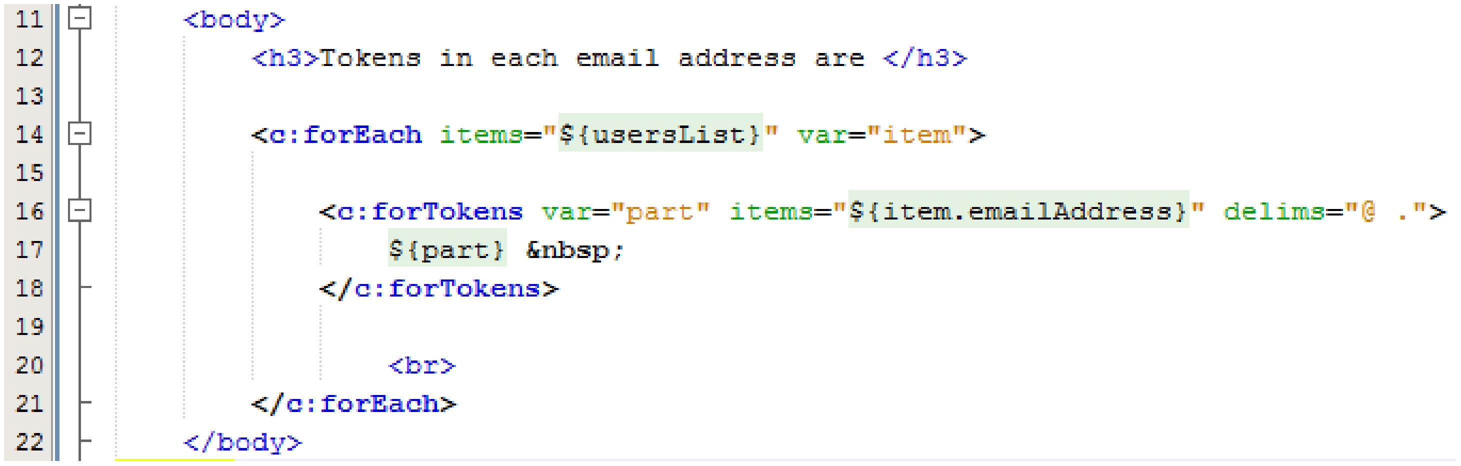
localhost:10599/WebApplication2/

Users currently in the address book

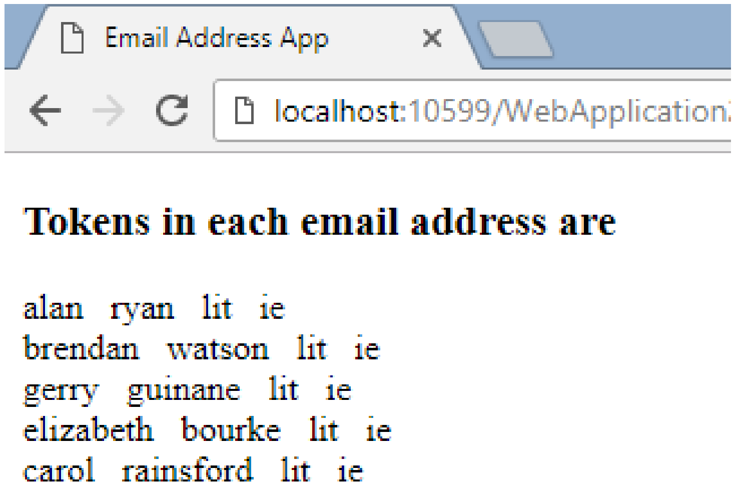
First name	Last name	Street	City	Country	Post Code	Email Address
Alan	Ryan	123 Fake St	Limerick	Ireland	V94 KX22	alan.ryan@lit.ie
Brendan	Watson	12 Main Street	Roscommon	Ireland	D01 FW97	brendan.watson@lit.ie
Gerry	Guinane	15 O'Connell Street	Sligo	Ireland	F91 HC57	gerry.guinane@lit.ie
Liz	Bourke	4 Crown Street	Thurles	Ireland	E41 C956	elizabeth.bourke@lit.ie
Carol	Rainsford	106 O'Connell St	Limerick	Ireland	V94 TD43	carol.rainsford@lit.ie

JSTL: for tokens

- This tag is used to loop through items that are stored in a string as long as the items in the string are separated by one or more delimiters.



```
<body>
    <h3>Tokens in each email address are </h3>

    <c:forEach items="${usersList}" var="item">

        <c:forTokens var="part" items="${item.emailAddress}" delims="@ .">
            ${part} &nbsp;
        </c:forTokens>

            <br>
    </c:forEach>
</body>
```



Tokens in each email address are

```
alan  ryan  lit  ie
brendan  watson  lit  ie
gerry  guinane  lit  ie
elizabeth  bourke  lit  ie
carol  rainsford  lit  ie
```

JSTL: four more attributes for looping

- When working with collections, the servlet code typically creates a collection and passes it to the JSP so the collection can be passed to the user.
- The JSP then uses the forEach tag to loop through the collection and display it to the user.
- However, there may be times when the JSP will need to do some additional processing.
  - For example, the JSP may need to know whether the item is the first or last item so it can apply special formatting to that item.
  - Alternatively, the JSP may need to know the item number so that it can apply shading to alternating items.

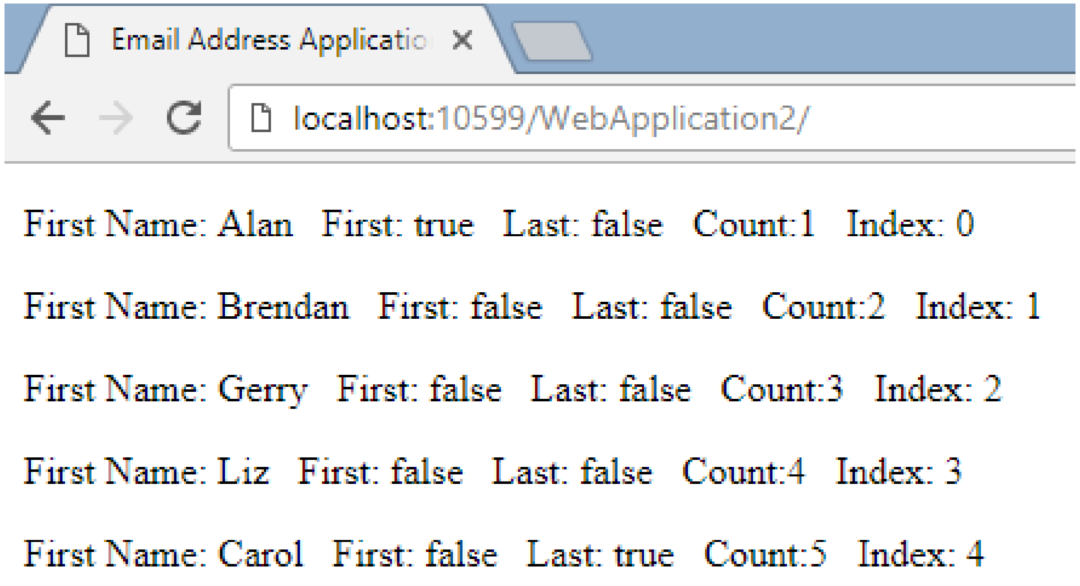
JSTL: four more attributes for looping

Attribute	Description
begin	The first index for the loop
end	The last index for the loop
step	Specifies the amount to increment the index each time through the loop
varStatus	Specifies the name of a variable that can be used to get information about the status of the loop (provides the first, last, index and count properties)

JSTL: four more attributes for looping

Partial JSP listing

```
  <c:forEach items="${usersList}" var="user" begin="0" end = "4" step="1" varStatus="status">

  <p>First Name: ${user.firstName}  First: ${status.first} Last: ${status.last}
                                Count:${status.count} Index: ${status.index} </p>

</c:forEach>
```


Email Address Applicatio

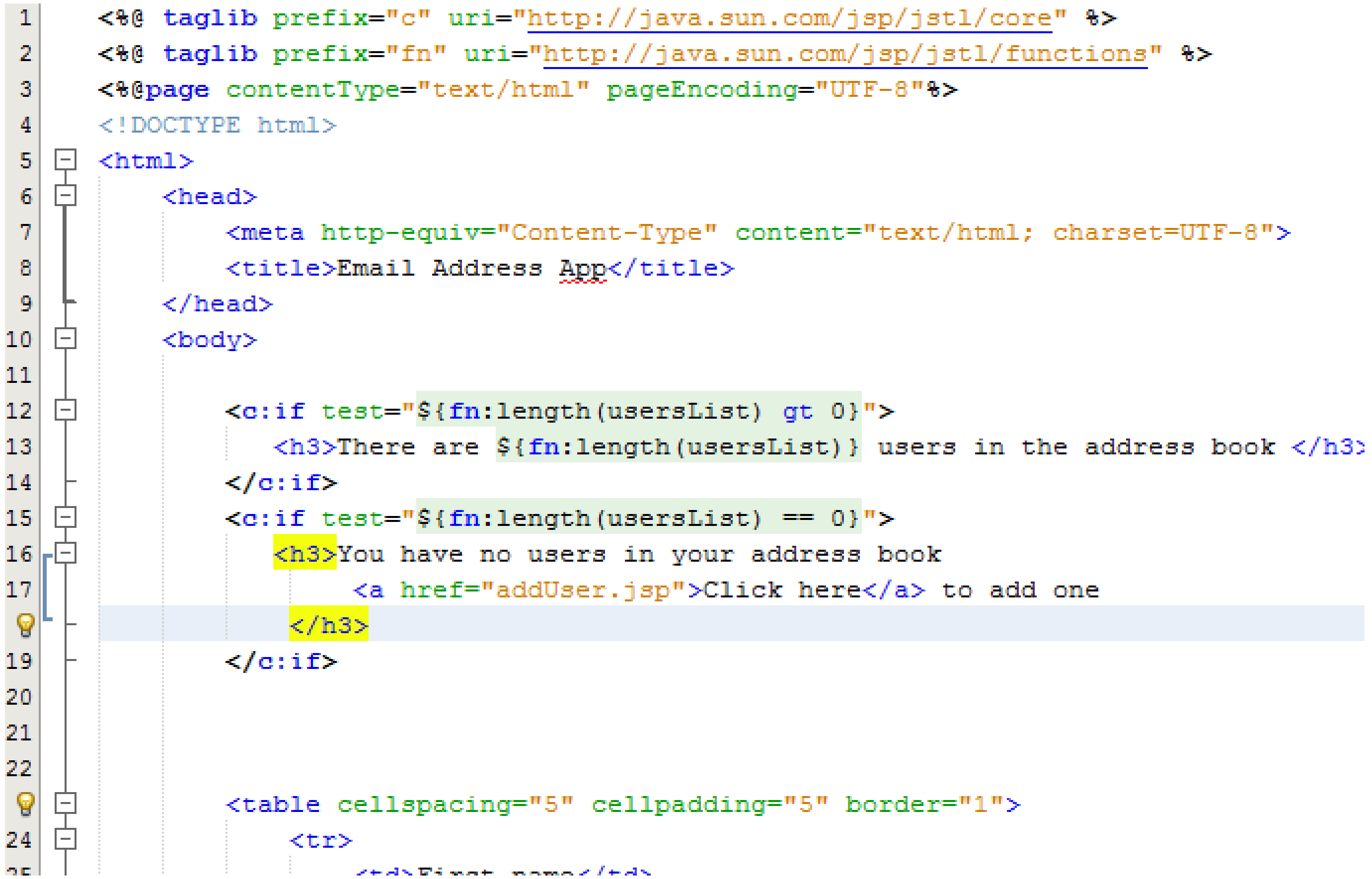
localhost:10599/WebApplication2/

```
First Name: Alan   First: true   Last: false   Count:1   Index: 0

First Name: Brendan   First: false   Last: false   Count:2   Index: 1

First Name: Gerry   First: false   Last: false   Count:3   Index: 2

First Name: Liz   First: false   Last: false   Count:4   Index: 3

First Name: Carol   First: false   Last: true   Count:5   Index: 4
```

JSTL: the if tag

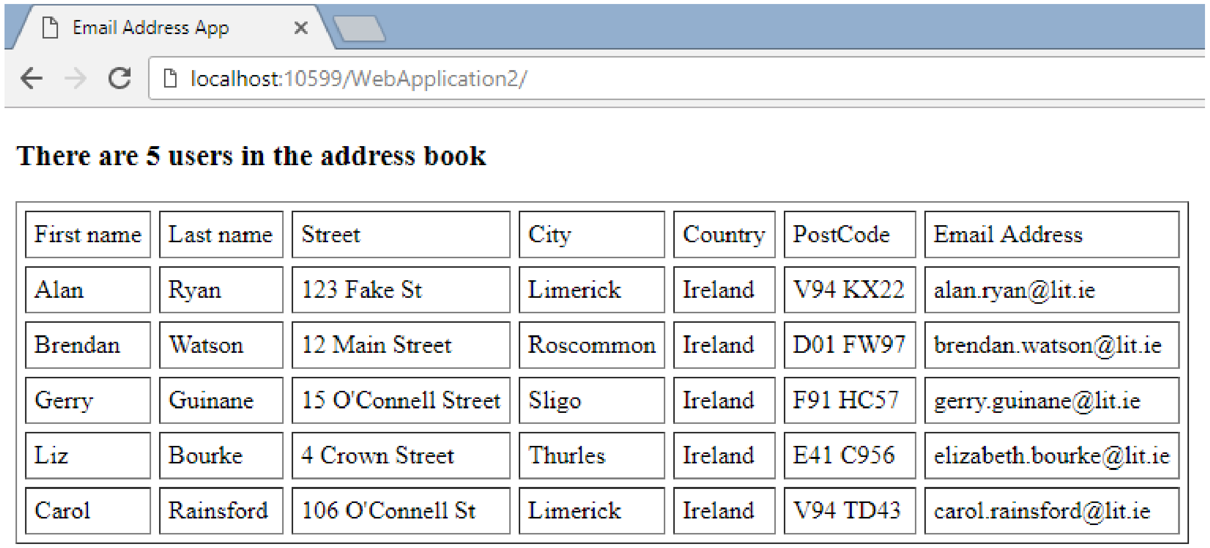
- When coding a JSP you may need to perform conditional processing to change the appearance of the page depending on the values of the attributes that are available to the page.

JSP



```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Email Address App</title>
    </head>
    <body>

        <c:if test="${fn:length(usersList) gt 0}">
           <h3>There are ${fn:length(usersList)} users in the address book </h3
        </c:if>
        <c:if test="${fn:length(usersList) == 0}">
           <h3>You have no users in your address book
                <a href="addUser.jsp">Click here</a> to add one
            </h3>
        </c:if>



        <table cellspacing="5" cellpadding="5" border="1">
            <tr>
                <td>First name</td>
```

JSTL: the if tag



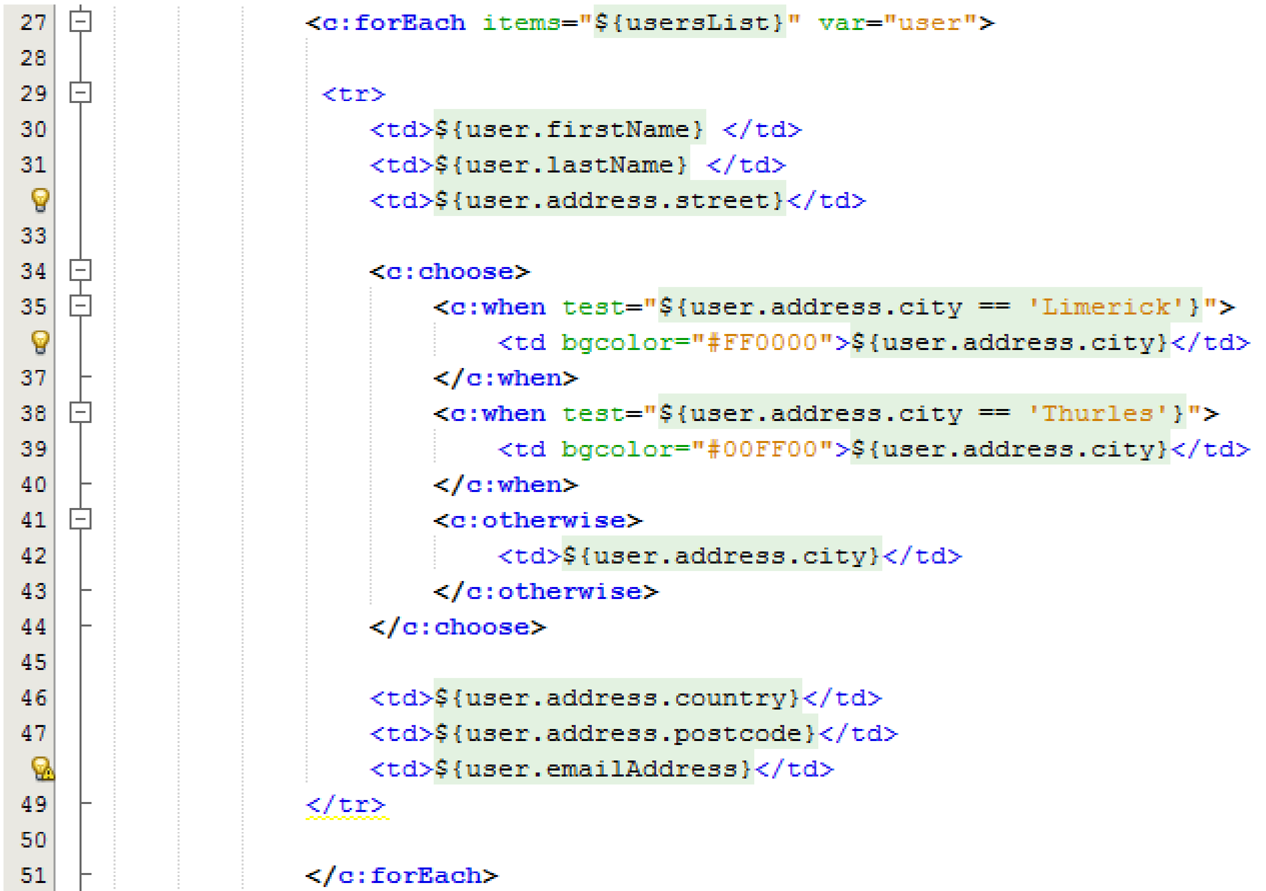
There are 5 users in the address book

First name	Last name	Street	City	Country	PostCode	Email Address
Alan	Ryan	123 Fake St	Limerick	Ireland	V94 KX22	alan.ryan@lit.ie
Brendan	Watson	12 Main Street	Roscommon	Ireland	D01 FW97	brendan.watson@lit.ie
Gerry	Guinane	15 O'Connell Street	Sligo	Ireland	F91 HC57	gerry.guinane@lit.ie
Liz	Bourke	4 Crown Street	Thurles	Ireland	E41 C956	elizabeth.bourke@lit.ie
Carol	Rainsford	106 O'Connell St	Limerick	Ireland	V94 TD43	carol.rainsford@lit.ie

- If necessary you can use the `var` and `scope` attributes to expose the `boolean` condition in the test attribute as a variable with the specified scope.
  - This means you can reuse the `boolean` condition in other if statements.
- If tags can be nested and can be nested within a `forEach` or a `forTokens` tag.

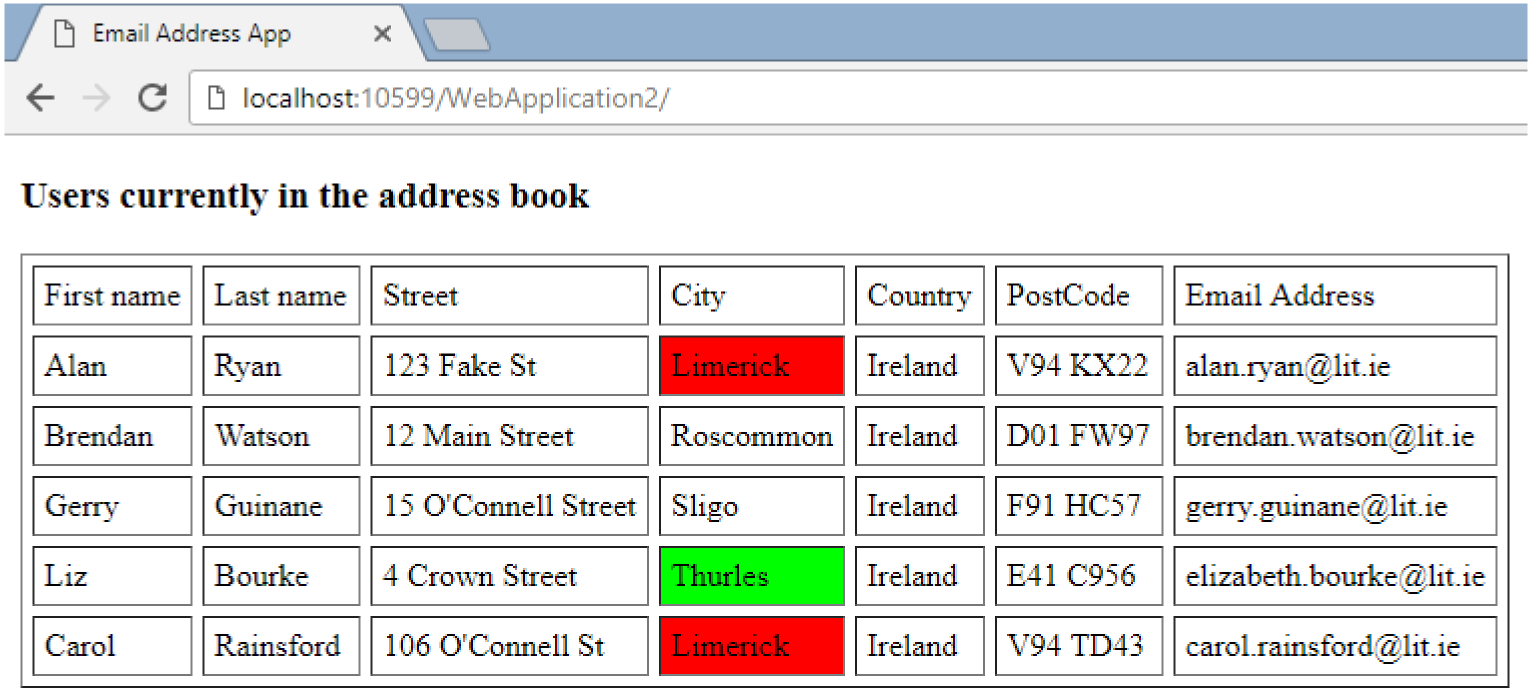
JSTL: the choose tag

- The equivalent of an if/else statement. To begin, code the opening and closing *choose* tags.
- Within those tags you can code one or more *when* tags (and an optional otherwise tag).



```
<c:forEach items="${usersList}" var="user">

 <tr>
    <td>${user.firstName} </td>
    <td>${user.lastName} </td>
    <td>${user.address.street}</td>

    <c:choose>
        <c:when test="${user.address.city == 'Limerick'}">
            <td bgcolor="#FF0000">${user.address.city}</td>
        </c:when>
        <c:when test="${user.address.city == 'Thurles'}">
            <td bgcolor="#00FF00">${user.address.city}</td>
        </c:when>
        <c:otherwise>
            <td>${user.address.city}</td>
        </c:otherwise>
    </c:choose>

    <td>${user.address.country}</td>
    <td>${user.address.postcode}</td>
    <td>${user.emailAddress}</td>
</tr>

</c:forEach>
```

JSTL: the choose tag



Email Address App

localhost:10599/WebApplication2/

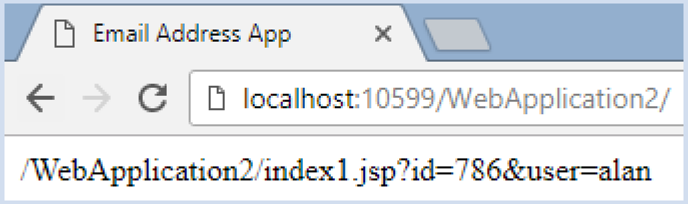
Users currently in the address book

First name	Last name	Street	City	Country	PostCode	Email Address
Alan	Ryan	123 Fake St	🟥 Limerick	Ireland	V94 KX22	alan.ryan@lit.ie
Brendan	Watson	12 Main Street	Roscommon	Ireland	D01 FW97	brendan.watson@lit.ie
Gerry	Guinane	15 O'Connell Street	Sligo	Ireland	F91 HC57	gerry.guinane@lit.ie
Liz	Bourke	4 Crown Street	🟩 Thurles	Ireland	E41 C956	elizabeth.bourke@lit.ie
Carol	Rainsford	106 O'Connell St	🟥 Limerick	Ireland	V94 TD43	carol.rainsford@lit.ie

Other JSTL Tags

Tag	Description
out	Displays the result of an expression. Similar to the way **<%= %>** works. The difference here is that **<c:out>** tag lets you use the **"."** notation to access properties. For example, to access user.address.street, use the following: **<c:out value = “user.address.street"/>**.
set	Sets the value of an attribute in scope.
remove	Removes an attribute from scope
catch	Used to catch exceptions that may occur at runtime
redirect	Redirects the browser to a new URL. **<c:redirect url="http://www.lit.ie"/>**

Other JSTL Tags

Tag	Description
param	The **<c:param>** tag allows URL request parameter(s) to be specified with a URL and also does the necessary URL encoding required. Within a **<c:param>** tag, the name attribute indicates the parameter name, and the value attribute indicates the parameter value **<c:url value="/index1.jsp" var="url">** ** <c:param name="id" value="786"/>** ** <c:param name="user" value="alan"/>** **</c:url>** **${url}**  localhost:10599/WebApplication2/ /WebApplication2/index1.jsp?id=786&user=alan

Other JSTL Tags

A comprehensive list of JSTL functions can be found here.

With a short tutorial found here.

References

Murach, J., (2014) *Murachs Java Servlets JSP*, 3rd edn. Mike Murach and Associates, Inc.

Jendrock E, Cervera-Navarro R, Evans I, Hasse K, Markito W (2014) *The Java EE 7 Tutorial*, 5th edn. Addison-Wesley Professional.

http://docs.oracle.com/javaee/6/tutorial/doc/

https://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm

https://www.javatpoint.com/jstl