B.Sc. In Software Development. Year 4. Semester I. Enterprise Development. Introduction to JSP's.

LIMERICK INSTITUTE OF TECHNOLOGY

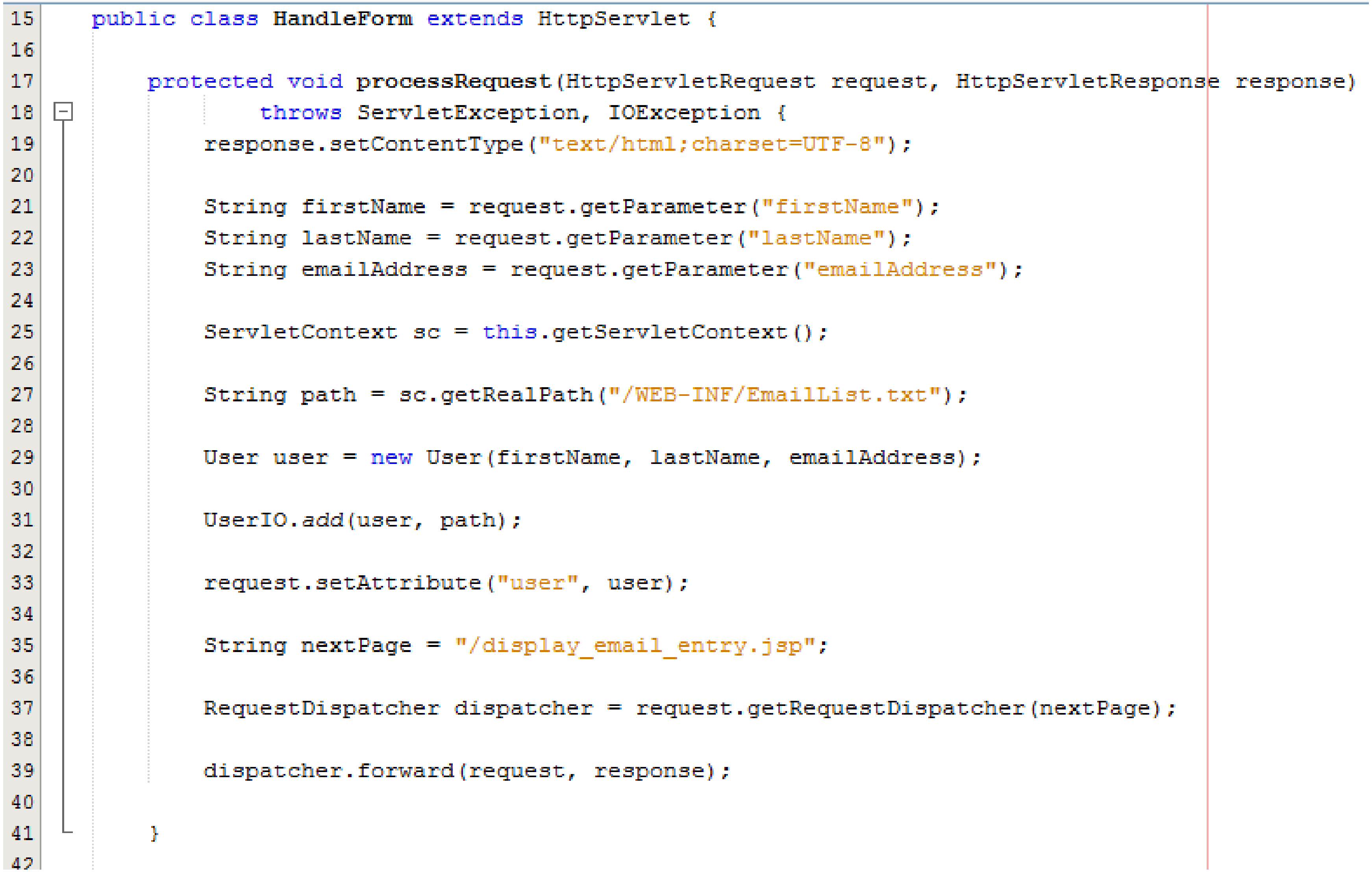
SCHOOL OF SCIENCE, ENGINEERING & I.T.

Department of Information Technology

Introduction

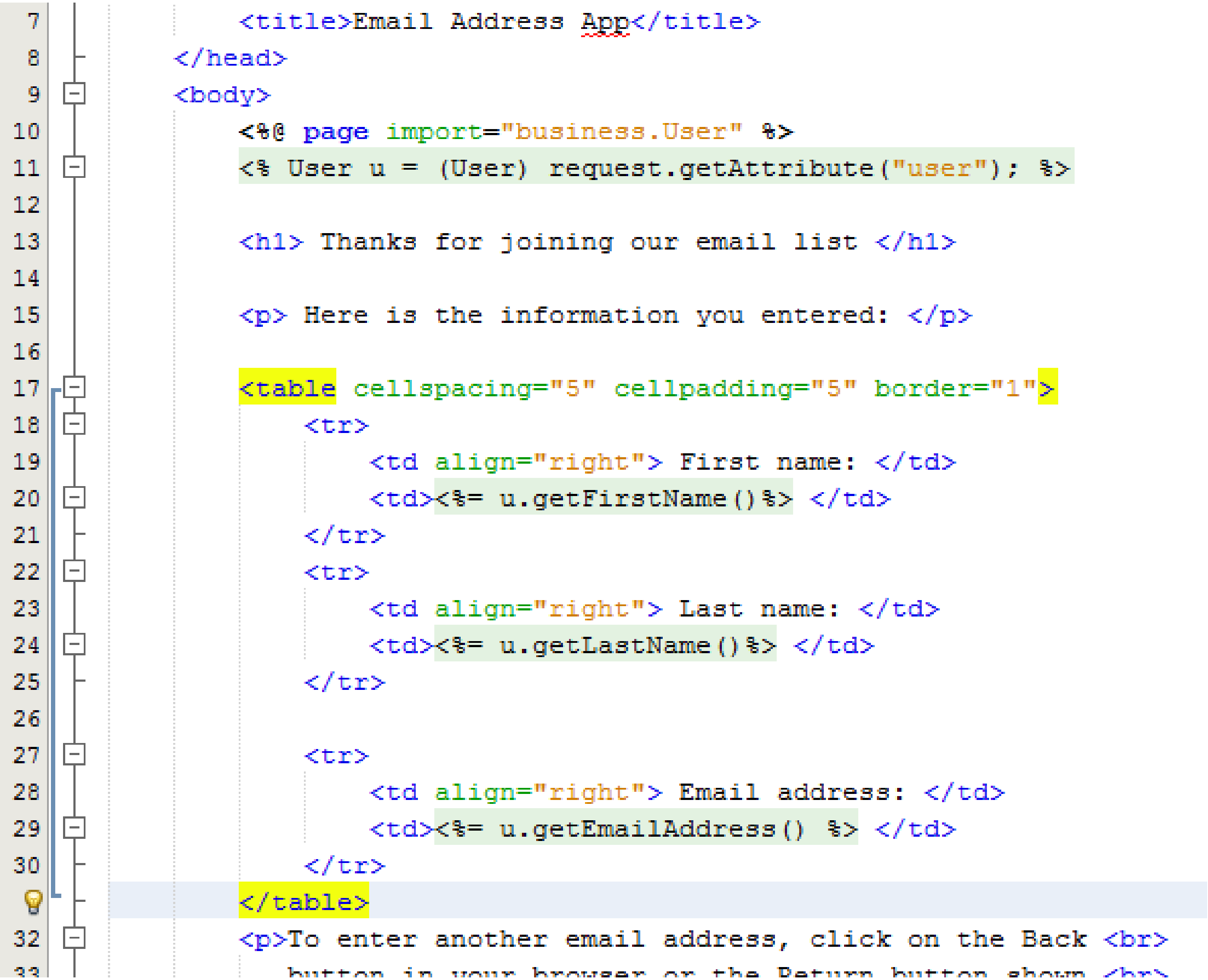
- We are now moving to a situation whereby we will combine what is best of both JSP’s and Servlets into one application.
    - The servlet will handle the processing (the controller).
    - The JSP will handle the presentation (the view).
- Writing the address book application using a Servlet AND a JSP might yield the following.

Code for the reworked Servlet



```java
public class HandleForm extends HttpServlet {

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

        String firstName = request.getParameter("firstName");
        String lastName = request.getParameter("lastName");
        String emailAddress = request.getParameter("emailAddress");

        ServletContext sc = this.getServletContext();

        String path = sc.getRealPath("/WEB-INF/EmailList.txt");

        User user = new User(firstName, lastName, emailAddress);

        UserIO.add(user, path);

        request.setAttribute("user", user);

        String nextPage = "/display_email_entry.jsp";

        RequestDispatcher dispatcher = request.getRequestDispatcher(nextPage);

        dispatcher.forward(request, response);

    }
```

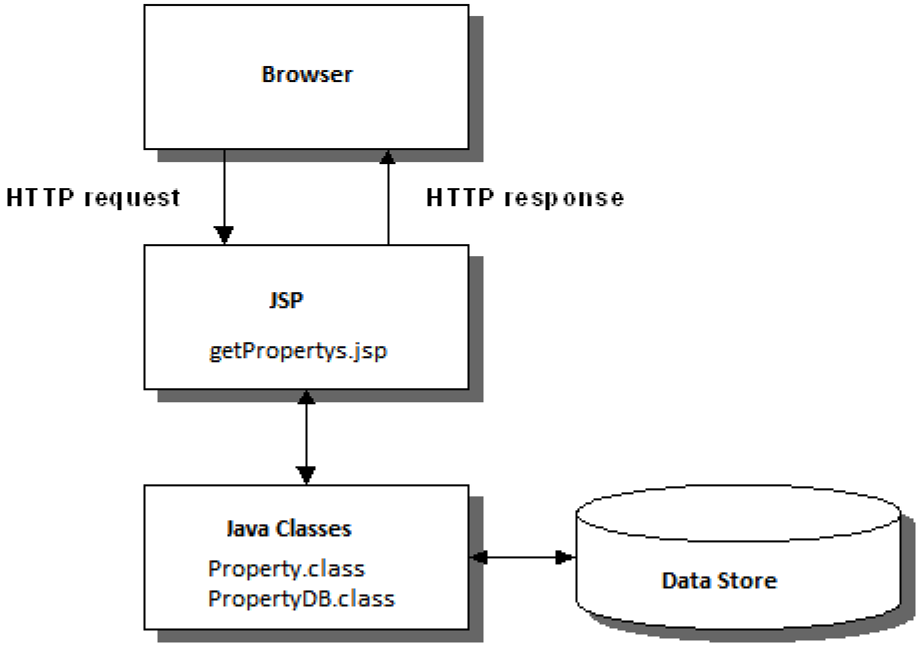
Code for the reworked JSP



```
        <title>Email Address App</title>
    </head>
    <body>
        <%@ page import="business.User" %>
        <% User u = (User) request.getAttribute("user"); %>

        <h1> Thanks for joining our email list </h1>

        <p> Here is the information you entered: </p>

        <table cellspacing="5" cellpadding="5" border="1">
            <tr>
                <td align="right"> First name: </td>
                <td><%= u.getFirstName()%> </td>
            </tr>
            <tr>
                <td align="right"> Last name: </td>
                <td><%= u.getLastName()%> </td>
            </tr>

            <tr>
                <td align="right"> Email address: </td>
                <td><%= u.getEmailAddress() %> </td>
            </tr>
        </table>
        <p>To enter another email address, click on the Back <br>
           button in your browser or the Return button shown <br>
```

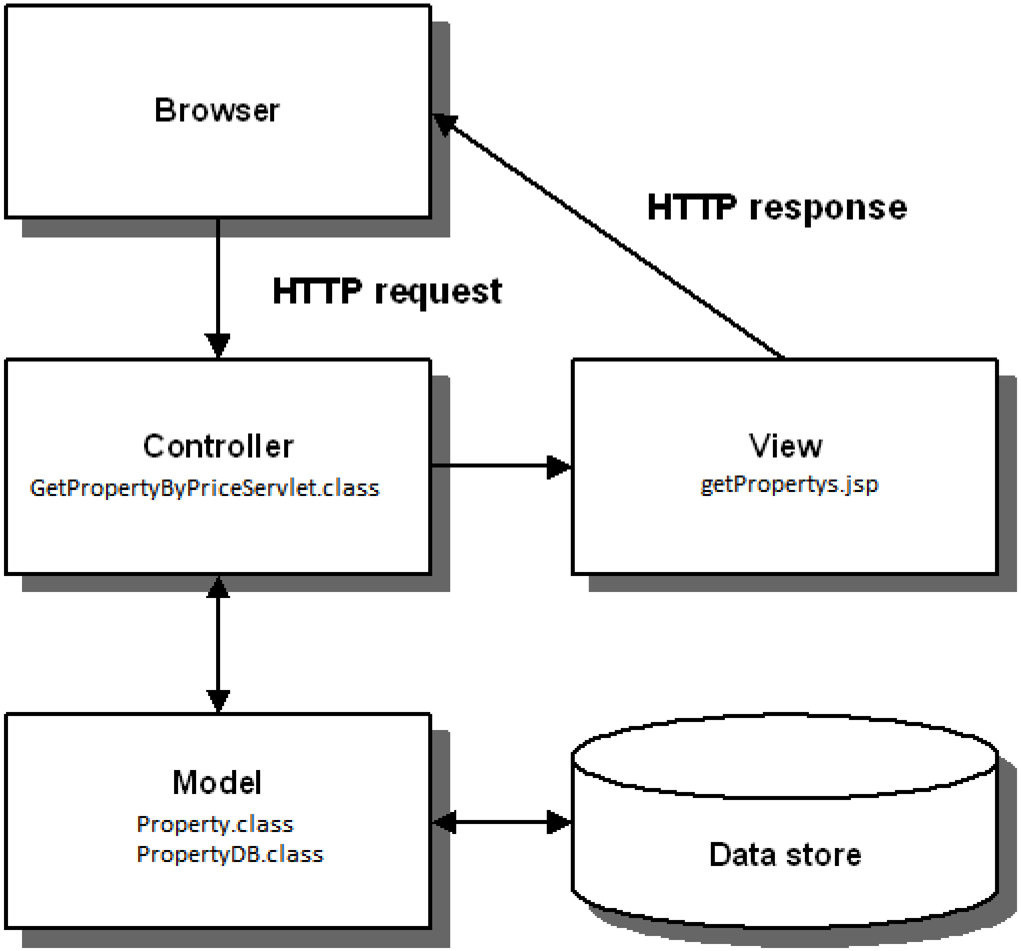
Introduction

- Up until now, your web applications have consisted of HTML and JSP pages.
- The JSP is responsible for handling both the request and response aspect of the application.



Model 1 Architecture

Introduction



Model 2 Architecture

Introduction

- The MVC pattern is commonly used to structure web applications that have significant processing requirements.
- That makes them easier to code and maintain.
- This pattern is also known as the Model 2 architecture.
- In the MVC pattern, the model consists of business objects like a User/Customer/Property object, the view consists of HTML pages and JSPs, and the controller consists of servlets.
- Usually, the methods of data classes like the UserDB class are used to read and write business objects like the User object to and from the data store

Forwarding Requests and Redirecting Responses

- When you use the MVC pattern your servlets often need to forward a request object to a JSP or another servlet.
- But first you sometimes need to store a business object (a Property, a Product, a User etc).

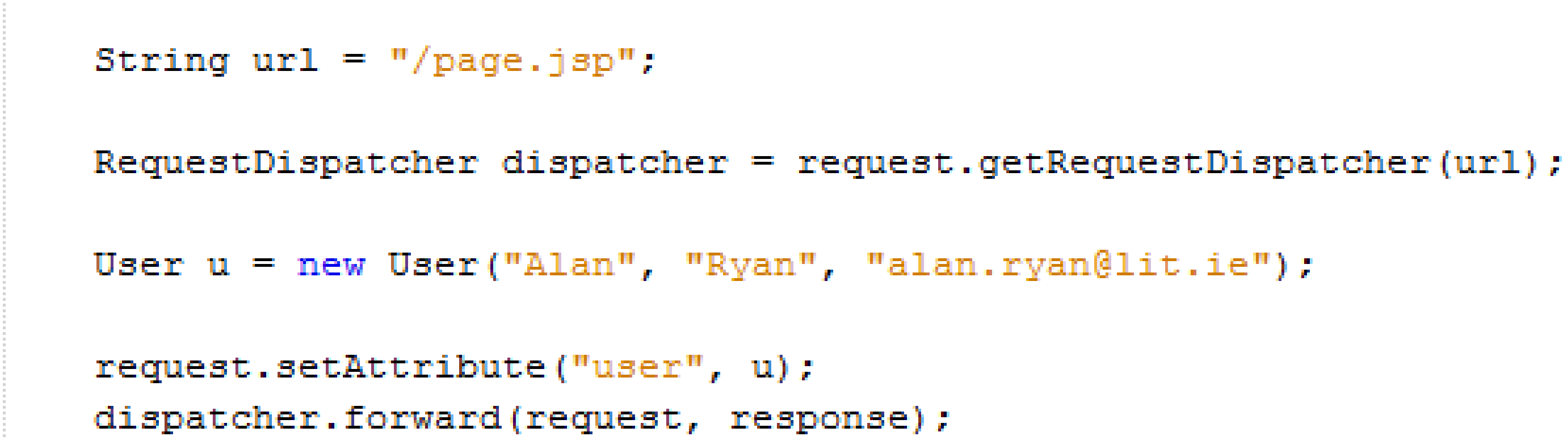
Method	Description
setAttribute(String name, Object o)	Binds ‘o’ to the request under the specified ‘name’
getAttribute(String name)	Retrieves an Object from the request.

Forwarding Requests and Redirecting Responses

- The methods on the previous slide are used in conjunction with a RequestDispatcher.
- A RequestDispatcher allows the 'including' of content in a request/response or 'forwarding' a request/response to a resource (JSP/HTML page or another servlet.).
- To forward request/response objects you need a RequestDispatcher object.

Forwarding Requests and Redirecting Responses: Example 1: Forward a single object

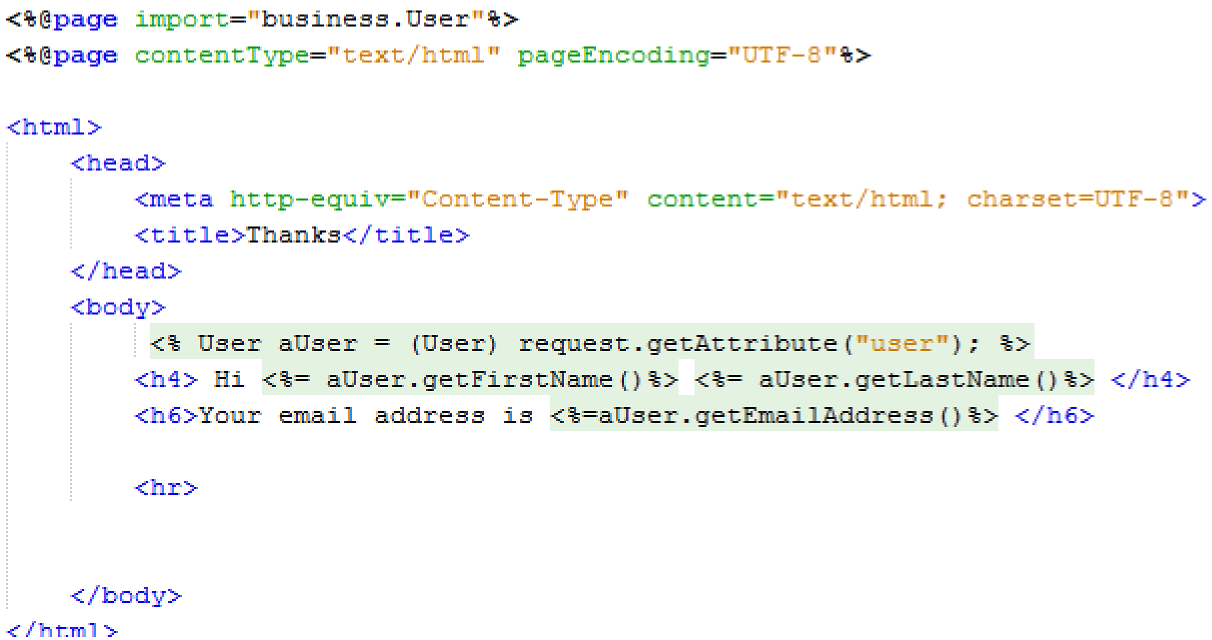
In the Servlet prepare a User object.



```
String url = "/page.jsp";

RequestDispatcher dispatcher = request.getRequestDispatcher(url);

User u = new User("Alan", "Ryan", "alan.ryan@lit.ie");

request.setAttribute("user", u);
dispatcher.forward(request, response);
```

Forwarding Requests and Redirecting Responses: Example 1: Forward a single object

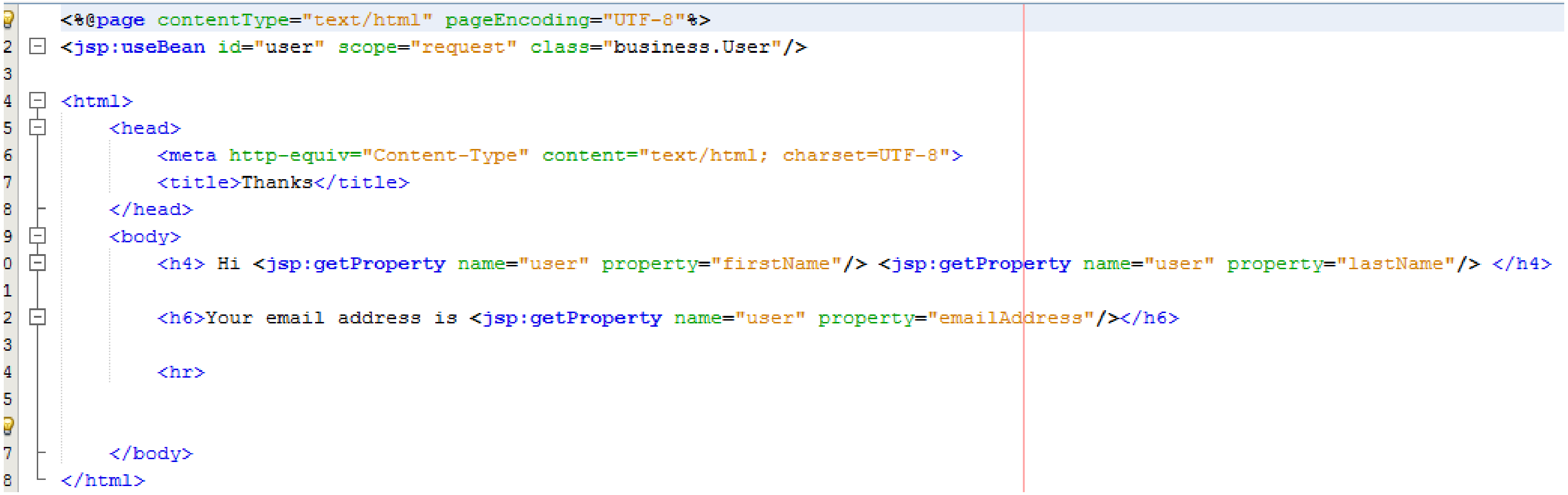
In the JSP (page.jsp) there are two types of syntax to access the user object.



```
<%@page import="business.User"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Thanks</title>
    </head>
    <body>
         <% User aUser = (User) request.getAttribute("user"); %>
        <h4> Hi <%= aUser.getFirstName()%> <%= aUser.getLastName()%> </h4>
        <h6>Your email address is <%=aUser.getEmailAddress()%> </h6>

        <hr>


    </body>
</html>
```

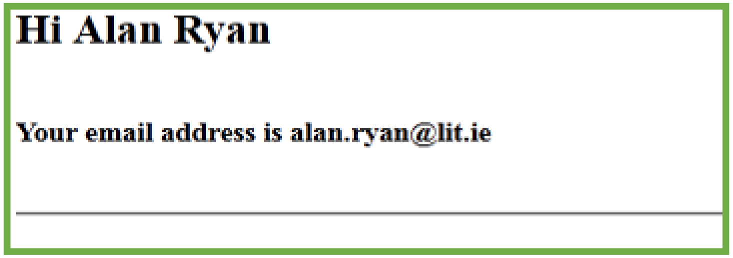
Syntax 1

Forwarding Requests and Redirecting Responses: Example 1: Forward a single object



```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<jsp:useBean id="user" scope="request" class="business.User"/>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Thanks</title>
    </head>
    <body>
        <h4> Hi <jsp:getProperty name="user" property="firstName"/> <jsp:getProperty name="user" property="lastName"/> </h4>

        <h6>Your email address is <jsp:getProperty name="user" property="emailAddress"/></h6>

        <hr>


    </body>
</html>
```

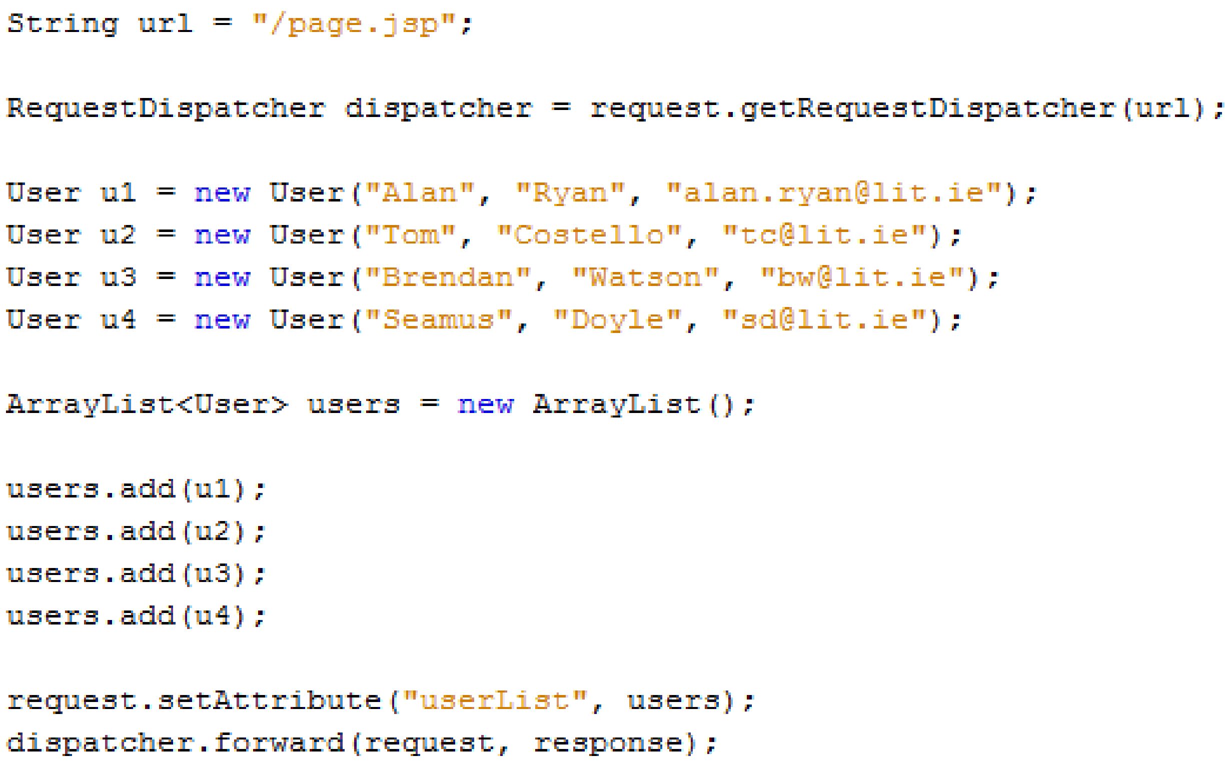
Syntax 2



```
Hi Alan Ryan

Your email address is alan.ryan@lit.ie
```

Forwarding Requests and Redirecting Responses: Example 2: Forward an ArrayList

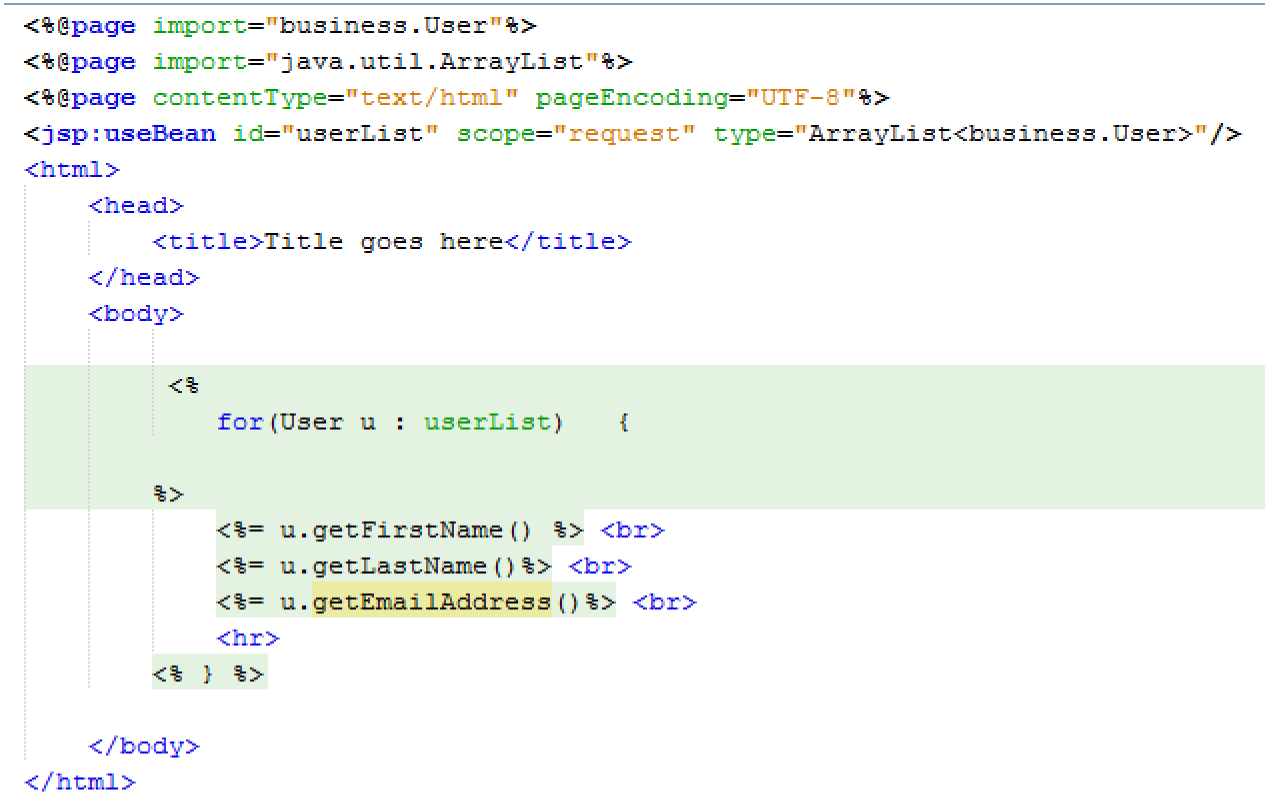
In the Servlet prepare an ArrayList of user objects.



```
String url = "/page.jsp";

RequestDispatcher dispatcher = request.getRequestDispatcher(url);

User u1 = new User("Alan", "Ryan", "alan.ryan@lit.ie");
User u2 = new User("Tom", "Costello", "tc@lit.ie");
User u3 = new User("Brendan", "Watson", "bw@lit.ie");
User u4 = new User("Seamus", "Doyle", "sd@lit.ie");

ArrayList<User> users = new ArrayList();

users.add(u1);
users.add(u2);
users.add(u3);
users.add(u4);

request.setAttribute("userList", users);
dispatcher.forward(request, response);
```

Forwarding Requests and Redirecting Responses: Example 2: Forward an ArrayList

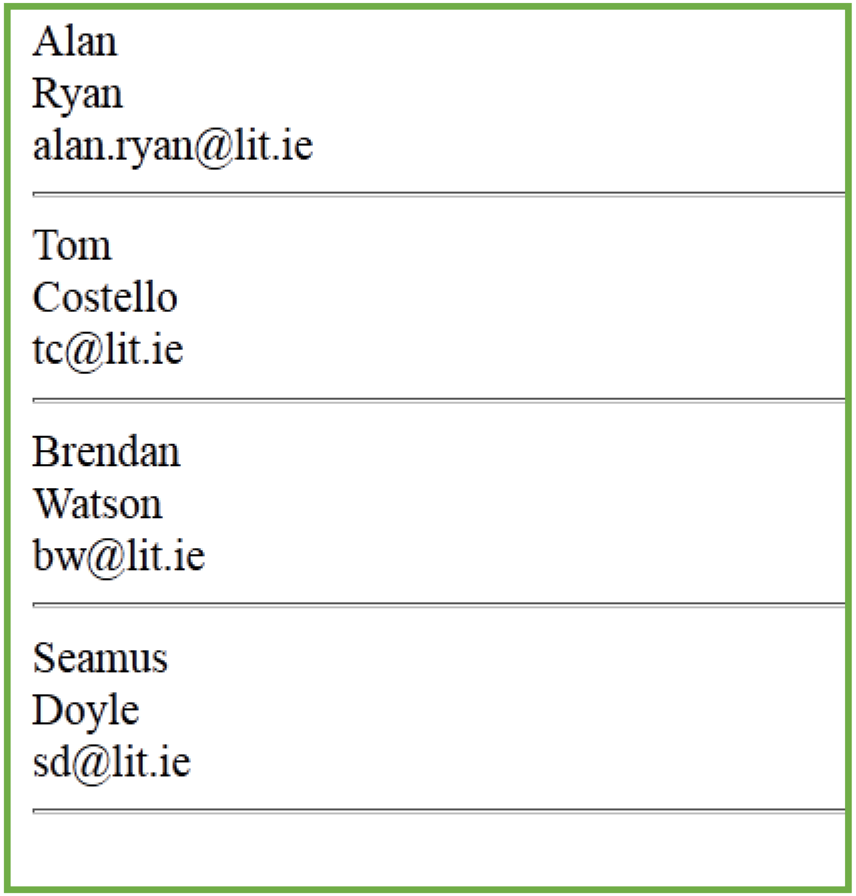
In the JSP (page.jsp) iterate over the ArrayList.



```
<%@page import="business.User"%>
<%@page import="java.util.ArrayList"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<jsp:useBean id="userList" scope="request" type="ArrayList<business.User>"/>
<html>
    <head>
        <title>Title goes here</title>
    </head>
    <body>

         <%
            for(User u : userList)    {

        %>
            <%= u.getFirstName() %> <br>
            <%= u.getLastName()%> <br>
            <%= u.getEmailAddress()%> <br>
            <hr>
        <% } %>

    </body>
</html>
```

Forwarding Requests and Redirecting Responses: Example 2: Forward an ArrayList

The JSP generates the following.



Alan
Ryan
alan.ryan@lit.ie

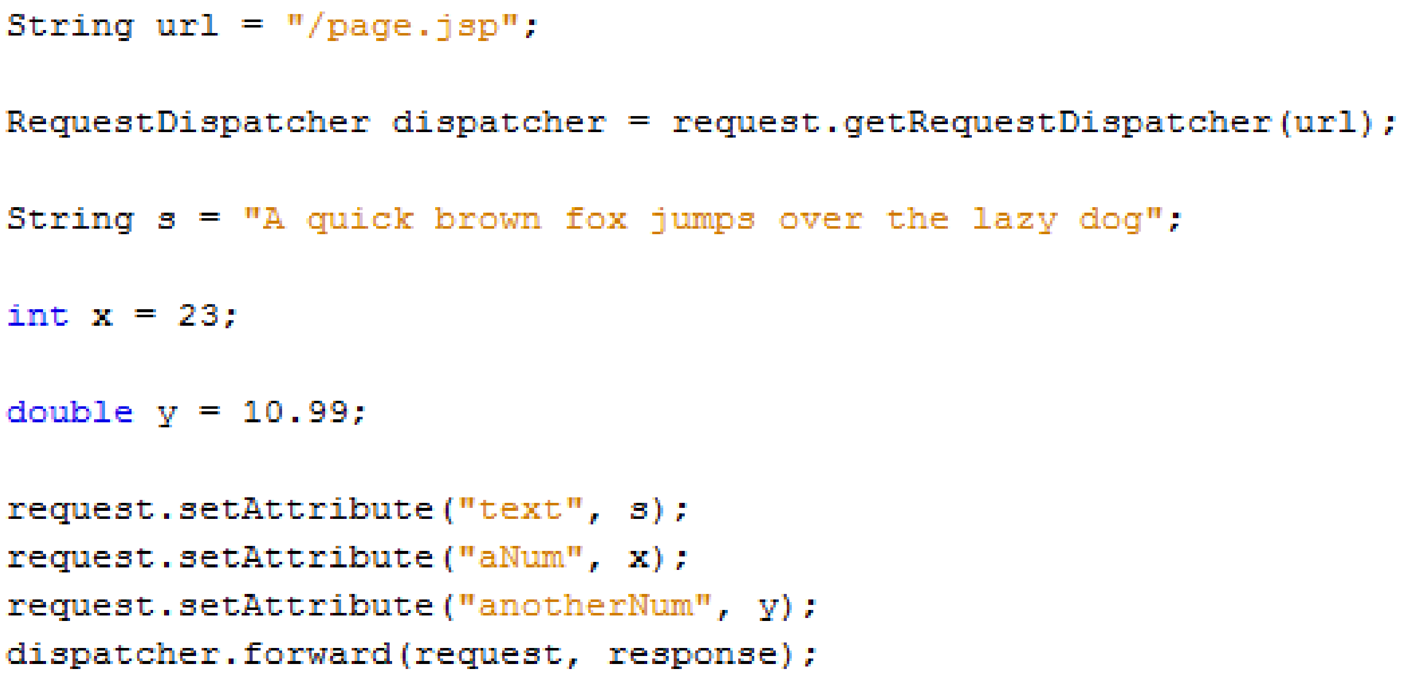
Tom
Costello
tc@lit.ie

Brendan
Watson
bw@lit.ie

Seamus
Doyle
sd@lit.ie

Forwarding Requests and Redirecting Responses: Example 3: Forward primitive types

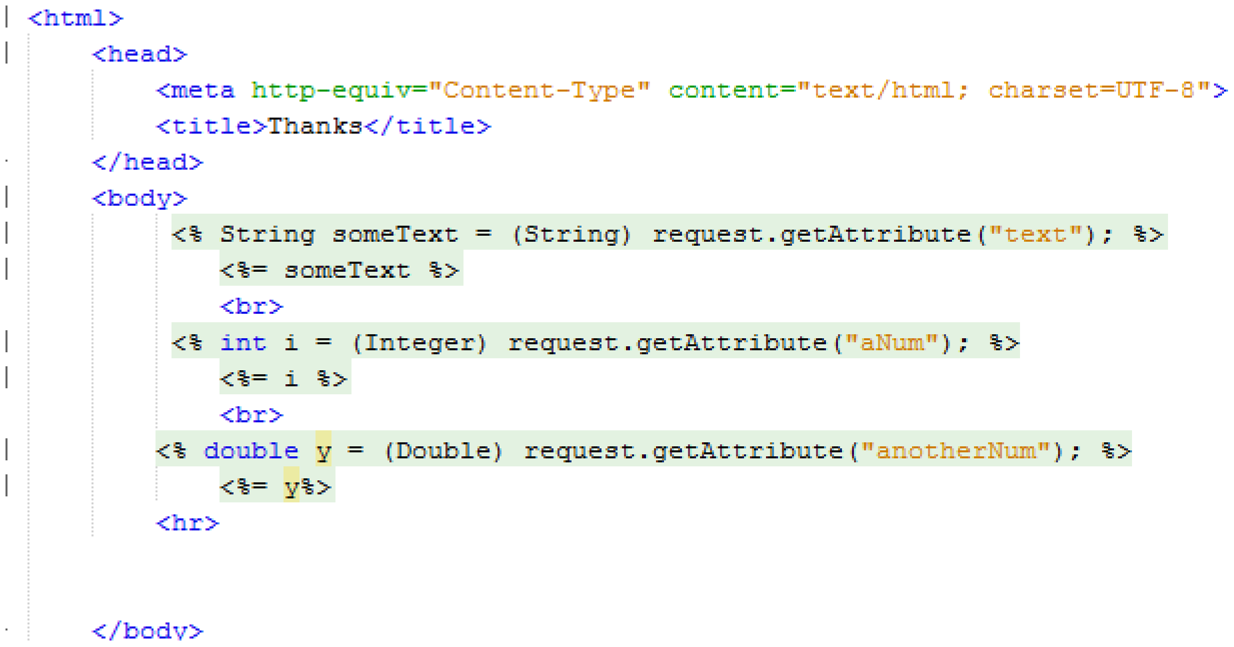
In the Servlet define and initialise variables.



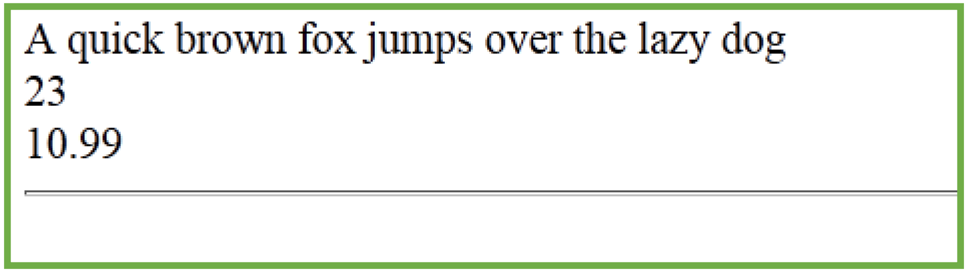
```
String url = "/page.jsp";

RequestDispatcher dispatcher = request.getRequestDispatcher(url);

String s = "A quick brown fox jumps over the lazy dog";

int x = 23;

double y = 10.99;

request.setAttribute("text", s);
request.setAttribute("aNum", x);
request.setAttribute("anotherNum", y);
dispatcher.forward(request, response);
```

Forwarding Requests and Redirecting Responses: Example 3: Forward primitive types

In the JSP (page.jsp) access the variables through the request.



```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Thanks</title>
    </head>
    <body>
         <% String someText = (String) request.getAttribute("text"); %>
            <%= someText %>
            <br>
         <% int i = (Integer) request.getAttribute("aNum"); %>
            <%= i %>
            <br>
        <% double y = (Double) request.getAttribute("anotherNum"); %>
            <%= y%>
        <hr>


    </body>
```

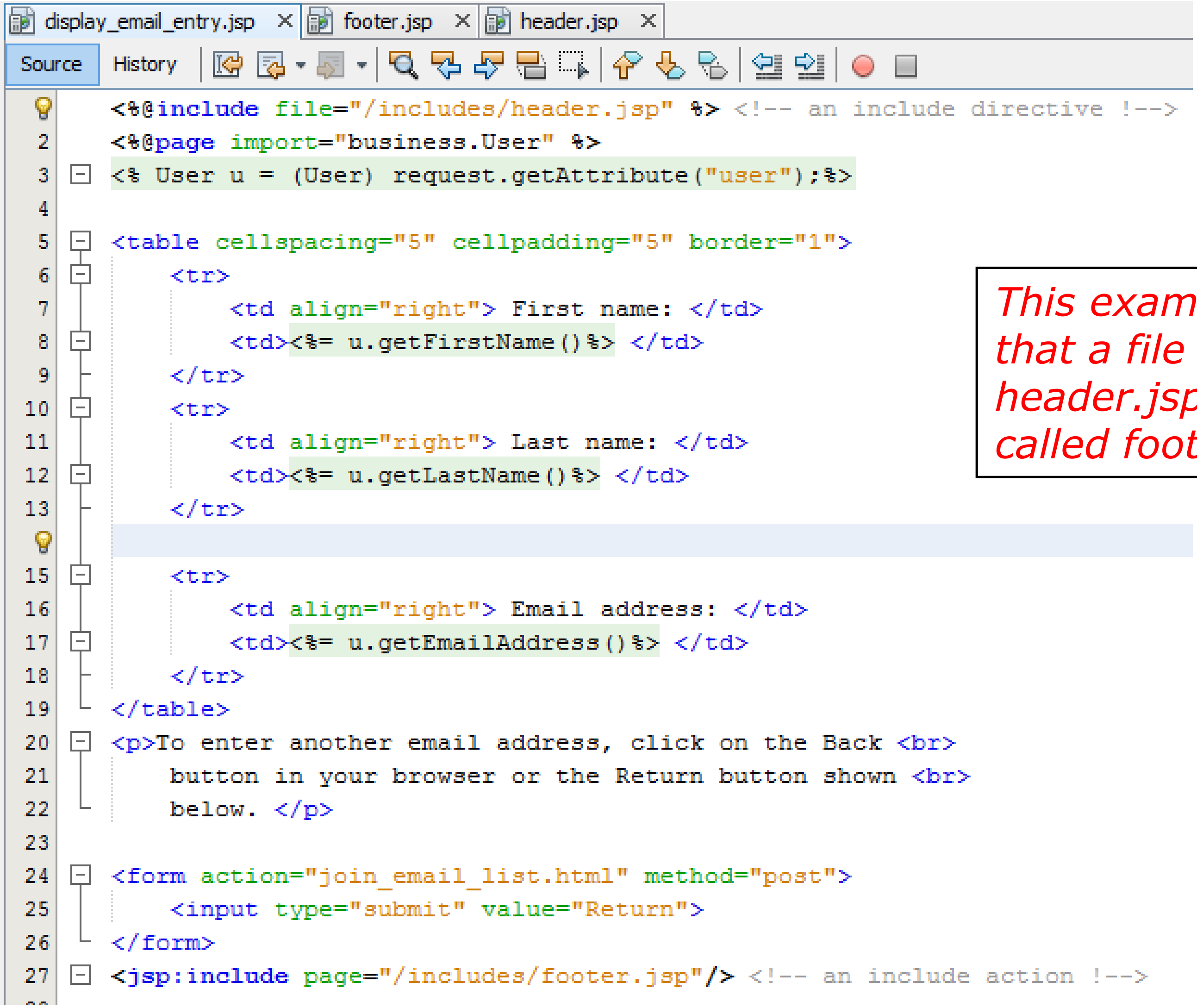


```
A quick brown fox jumps over the lazy dog
23
10.99
```

How to include a file in a JSP

- When you’re coding a web app, you may want to include the same block of code in several JSP’s.
  - Headers and footers.
  - Reuse menus etc.
- If so, you can store this code in a separate file, then you can include the code in that file in a JSP.
- These files are often called includes.
  - Includes can reduce redundant code and improve maintenance.
- An example appears on the next slide.

How to include a file in a JSP



```
<%@include file="/includes/header.jsp" %> <!-- an include directive !-->
<%@page import="business.User" %>
<% User u = (User) request.getAttribute("user");%>

<table cellspacing="5" cellpadding="5" border="1">
    <tr>
        <td align="right"> First name: </td>
        <td><%= u.getFirstName()%> </td>
    </tr>
    <tr>
        <td align="right"> Last name: </td>
        <td><%= u.getLastName()%> </td>
    </tr>

    <tr>
        <td align="right"> Email address: </td>
        <td><%= u.getEmailAddress()%> </td>
    </tr>
</table>
<p>To enter another email address, click on the Back <br>
    button in your browser or the Return button shown <br>
    below. </p>

<form action="join_email_list.html" method="post">
    <input type="submit" value="Return">
</form>
<jsp:include page="/includes/footer.jsp"/> <!-- an include action !-->
```

This example assumes that a file called header.jsp and another called footer.jsp exist.

Two techniques for Including files

- When you include a file at compile-time, the code within the file becomes part of the generated servlet.
- When you include a file a runtime, the included file never becomes part of the generated servlet so the serlvet makes a runtime call to get the included file each time the page is requested.

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/