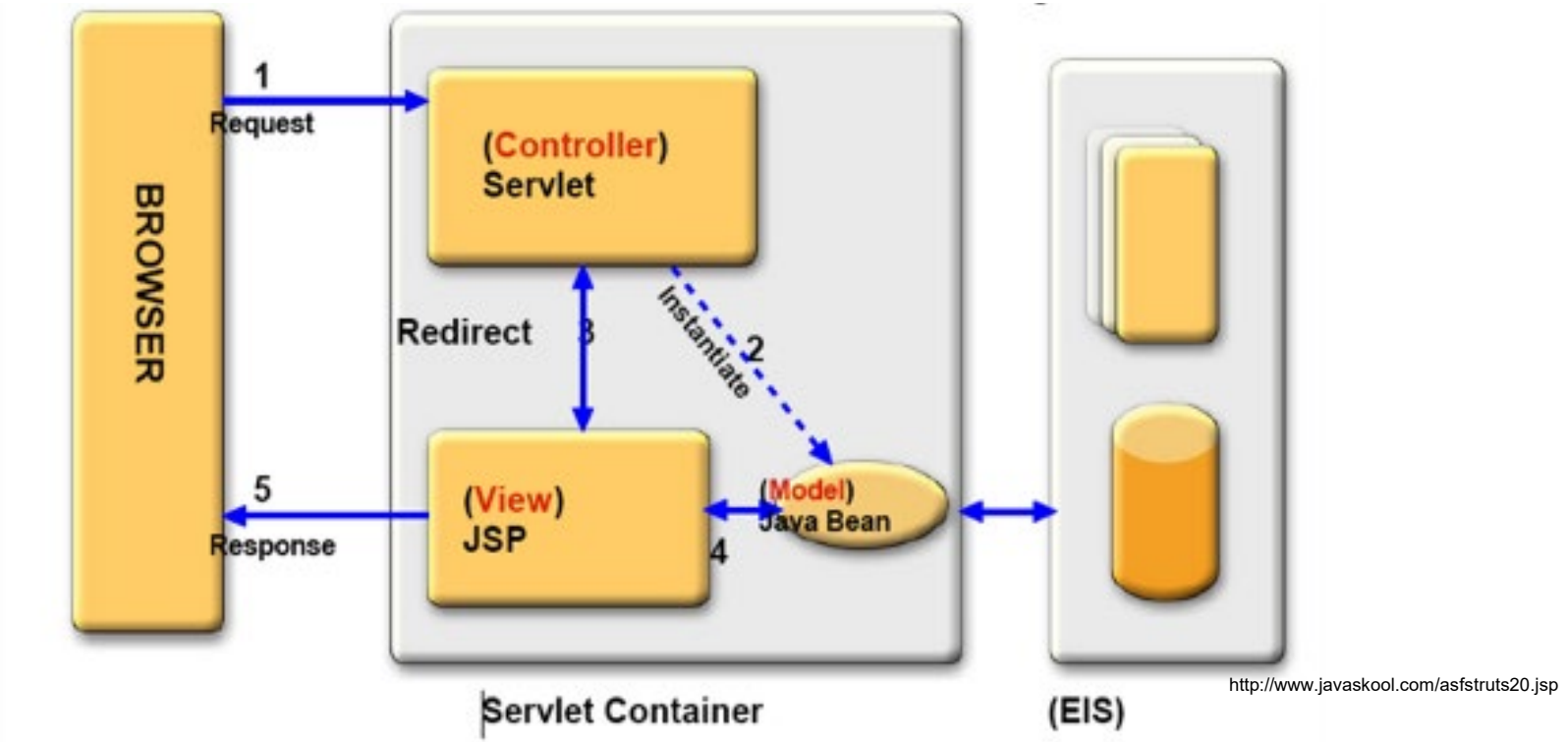
CT5106

JSP (Java Server Pages)

MVC Architecture

- There are a few different interpretations / implementations of this model, but we will be using the following model:
  - Model:
    - Represents the application data. It includes the logic to manage access to and modification of the data (e.g. basic CRUD operations).
  - View:
    - The view presents the model to the user, and gets user inputs / requests to change the data(model). It gets data from the model and specifies how that data should be presented. A view also forwards user input to a controller.
  - Controller:
    - The controller is the glue between the model and the view. It is responsible for controlling the flow of the program as well as processing updates from the model to the view and visa versa. A controller selects the next view to display based on the user interactions and the outcome of the model operations.

MVC = Model View Controller



MVC- an example

A simple JSP page

today.jsp

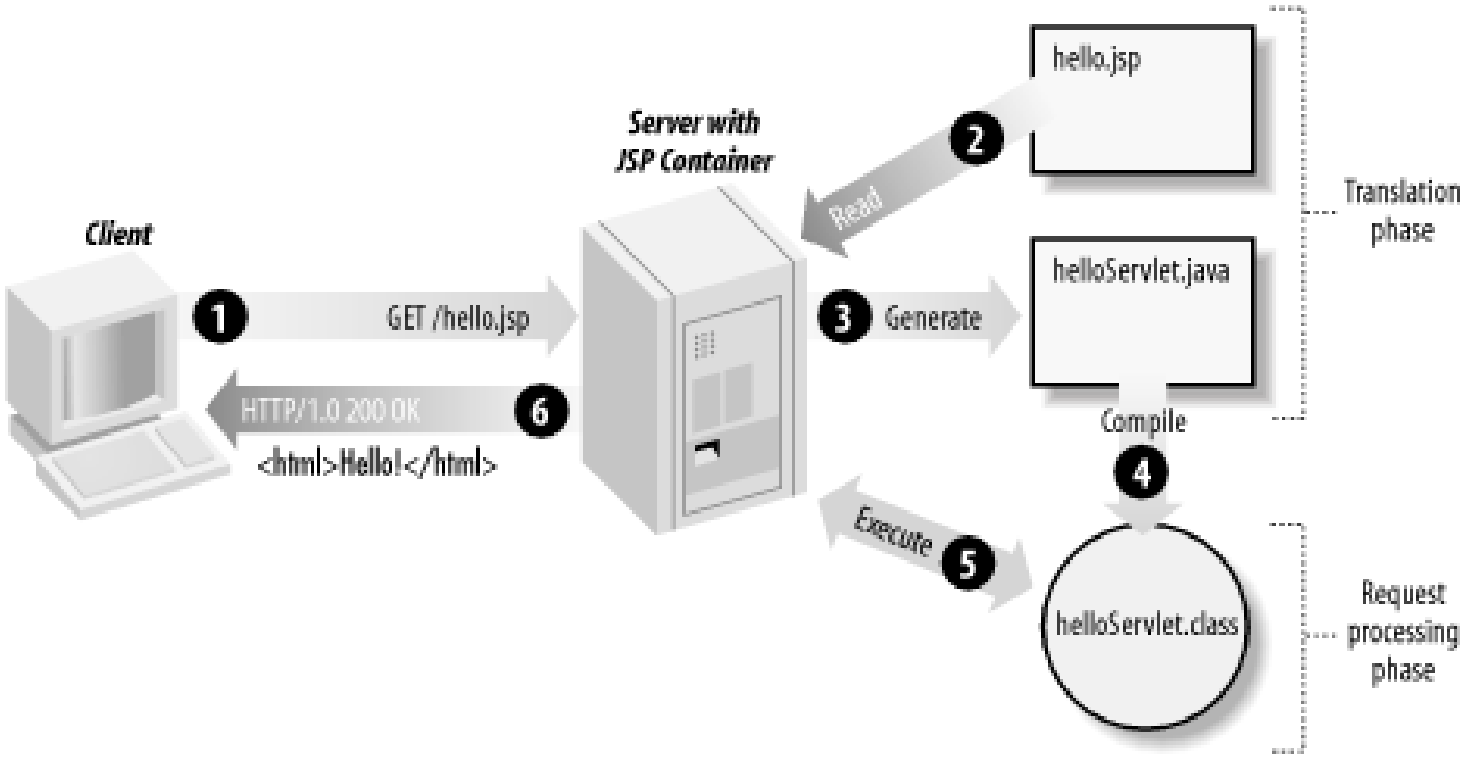
```
<%@page import="java.io.PrintWriter"%>
<%@page import="java.util.Date"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <% Date today = new Date();%>

        <h1>Hello, today is <%=today%></h1>

        <h2>
            <%
                PrintWriter writer = new PrintWriter(out);
                writer.printf("Just another %tA in the month of %tB in the year %tY%n", today, today, today);
            %>
        </h2>

    </body>
</html>
```

Some explanations

- <%@page import="java.util.Date"%>
  - Like an import in regular Java – here, we want to use the Date class
- <% Date today = new Date(); %>
  - This is a JSP *scriptlet (Java in the JSP page)*. It is executed every time the page is requested
  - So we have a longer example also where the Java is inside the <% … %> tags
- PrintWriter writer = new PrintWriter(out);
  - Here we get a PrintWrite object using the 'out' variable – one of a number of useful variables pre-defined in JSP pages
- <%=date %>
  - This is a JSP expression that is evaluated every time the page is requested

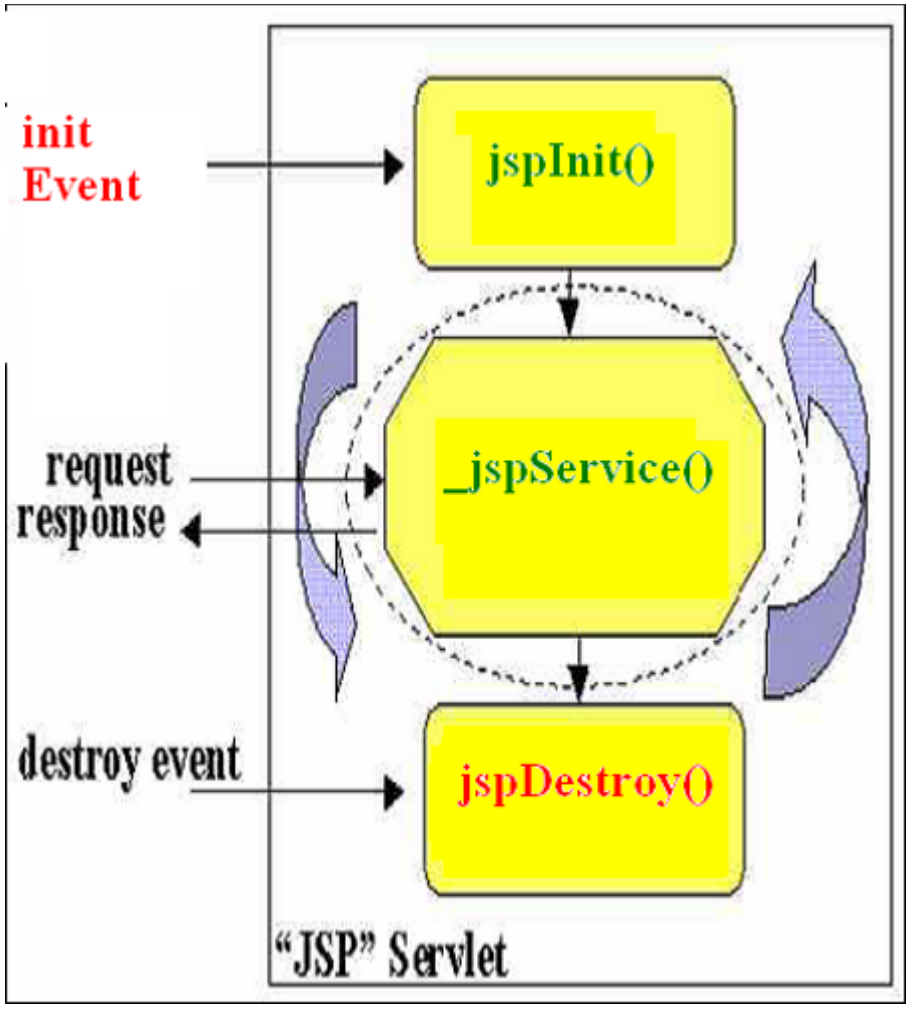
How are JSP's executed?

- The first time a JSP is requested, it is converted into a servlet (java), then compiled (turned into a .class file) and executed



JSP Life Cycle

- When the JSP is converted into a class and loaded into the servlet container it is ready to be called
- When it is called first, it is initialised *(jspInit)*
- Every request after that is handled by the *jspService* method.



Generated source

- Should be under the Payara server folder in the domain which you create when installing Payara from NetBeans, e.g.:

"C:\Users\o_molloy\Payara_Server\glassfish\domains\domain1\generated\jsp\week3-1.0-SNAPSHOT\org\apache\jsp\hellojsp_jsp.java"

- By default in later versions this is switched off, to switch it on..

Need to turn keepgenerated back on

- Edited default-web.xml in the comain config folder:
  - C:\Users\0063190s\Payara_Server\glassfish\domains\domain1\config
- Added this parameter in the <servlet> section
  - <init-param>
  - <param-name>keepgenerated</param-name>
  - <param-value>true</param-value>
  - </init-param>

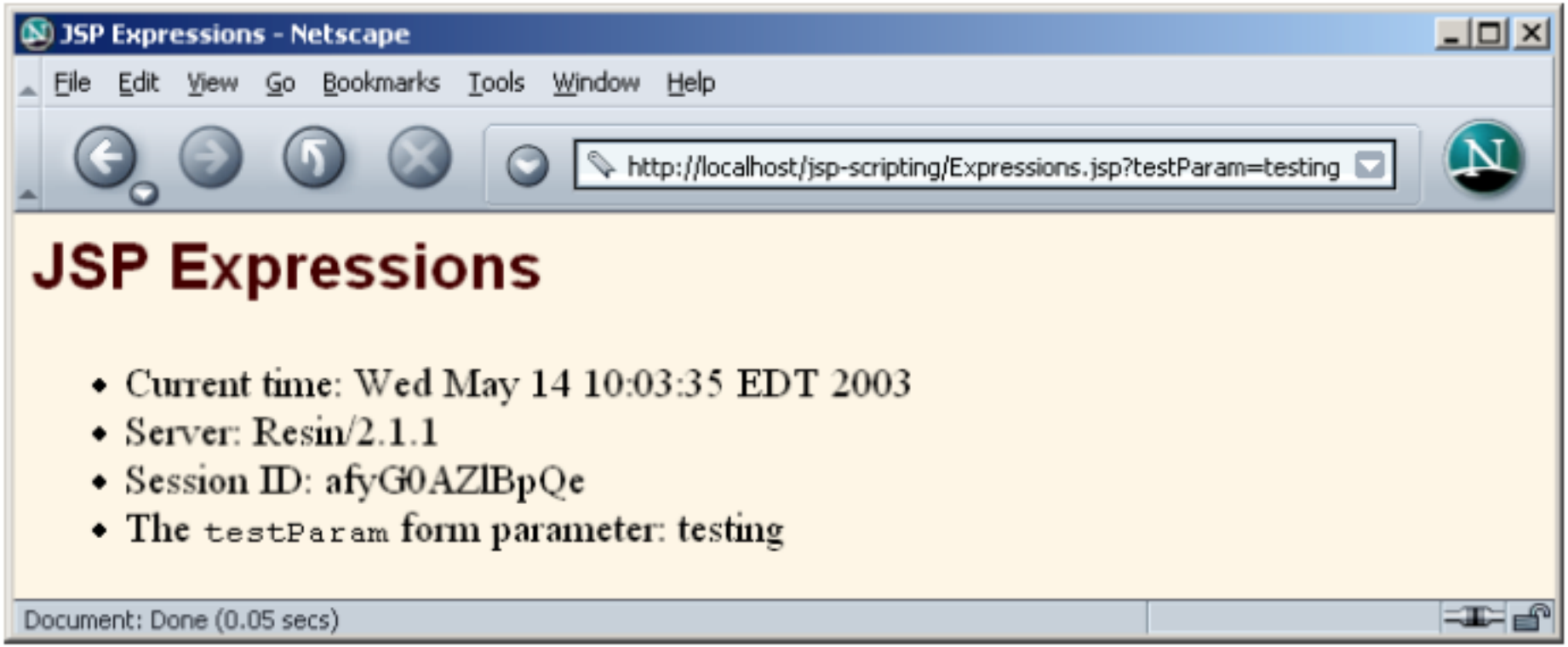
JSP Pre-defined variables

- We will only be using a few of these (we've already see 'out'), for example:
  - request
  - response
  - session
  - application

```
public void _jspService(HttpServletRequest request, HttpServletResponse response)
  throws java.io.IOException, ServletException {
    JspFactory _jspxFactory = JspFactory.getDefaultFactory();
    PageContext pageContext = _jspxFactory.getPageContext(...);
    HttpSession session = pageContext.getSession();
    ServletContext application = pageContext.getServletContext();
    ServletConfig config = pageContext.getServletConfig();
    JspWriter out = pageContext.getOut();
    Object page = this;
    ...
```

Listing 11.3 Expressions.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>JSP Expressions</TITLE>
<META NAME="keywords"
      CONTENT="JSP,expressions,JavaServer Pages,servlets">
<META NAME="description"
      CONTENT="A quick example of JSP expressions.">

<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H2>JSP Expressions</H2>
<UL>
  <LI>Current time: <%= new java.util.Date() %>
  <LI>Server: <%= application.getServerInfo() %>
  <LI>Session ID: <%= session.getId() %>
  <LI>The <CODE>testParam</CODE> form parameter:
      <%= request.getParameter("testParam") %>
</UL>
</BODY></HTML>
```


JSP Expressions - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost/jsp-scripting/Expressions.jsp?testParam=testing
JSP Expressions
Current time: Wed May 14 10:03:35 EDT 2003
Server: Resin/2.1.1
Session ID: afyG0AZlBpQe
The testParam form parameter: testing
Document: Done (0.05 secs)

JSP Comment

- **Description:**
- Developer comment that is not sent to the client
- **Example:**

  <%-- Blah --%>

JSP Expression

- **Description:**
- Expression that is *evaluated* and sent to the client each time the page is requested
- **Example:**

<%= *Java Value* %>

```
<h1>A random number is: <%= Math.random() %></h1>
```

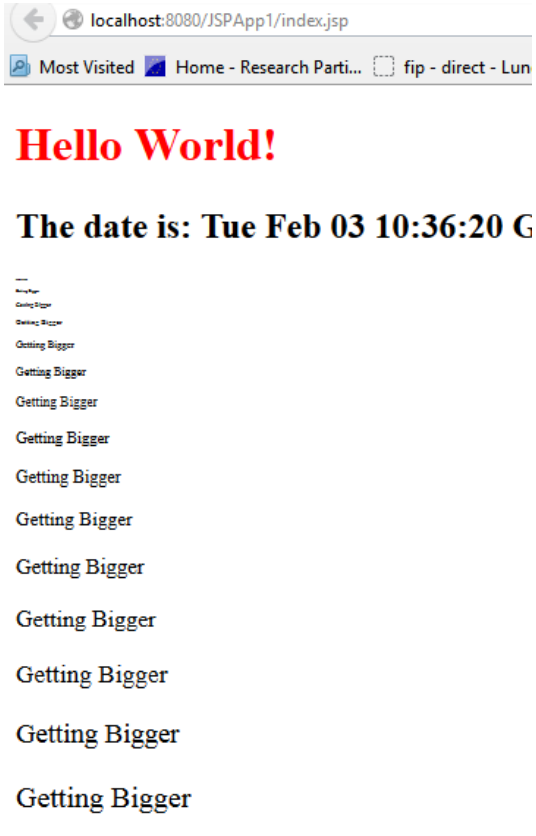
JSP Scriptlet

```
<% Date date = new Date(); %>

<h1>Hello World!</h1>
<h2>The date is: <%=date %></h2>
```

Mixing html and jsp script

fontLoop.jsp

```
<% for (int i=1;i<=15;i++)
        {
        %>
        <p style="font-size:<%=i%>pt"> Getting Bigger </p>
        <%
        }
        %>
```



Declaring variables

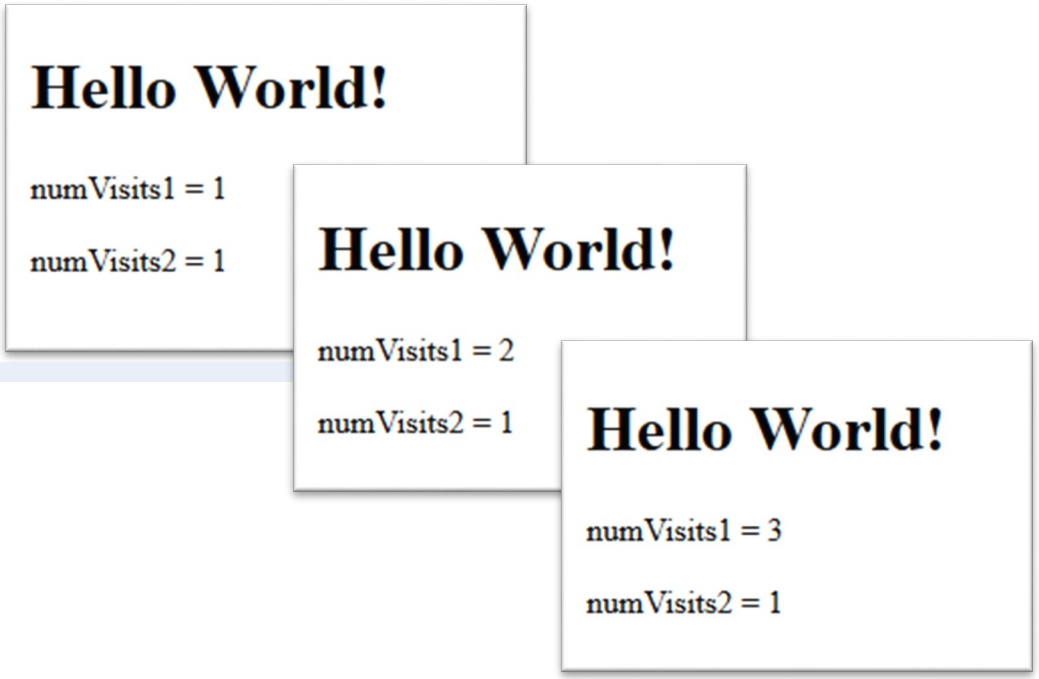
- Difference between the following?
  1. <%! int numVisits1 = 0; %>
  2. <% int numVisits2 = 0; %>

  1. Using <%! int x=0; %> is like declaring a variable once at the class level (called once at *init*)
  2. using Using <% int x=0 %> declares a variable locally (so it is re-run every time we call the service method)

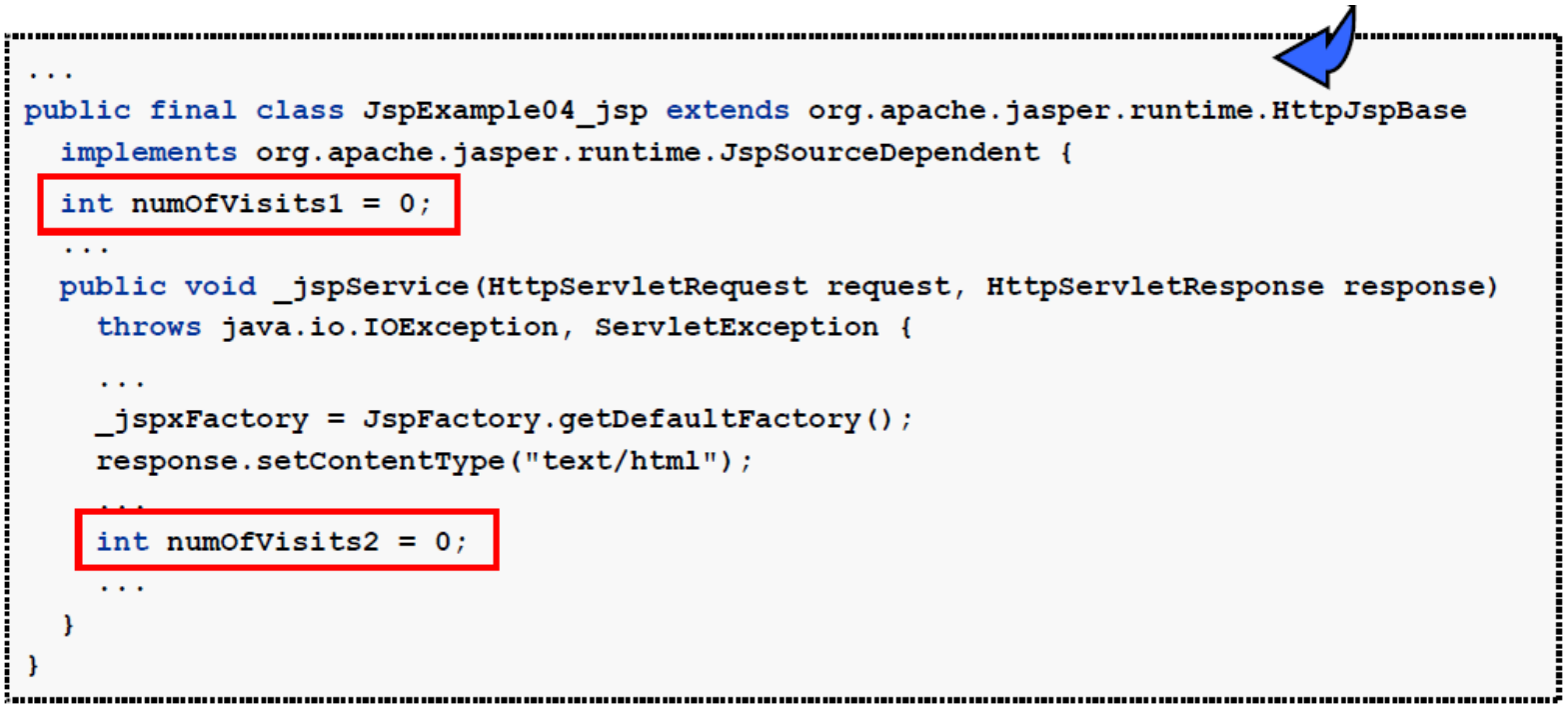
Example

visit.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello World!</h1>
        <%! int numVisits1 = 0; %>
        <% int numVisits2 = 0; %>

        <p>numVisits1 = <%=++numVisits1%></p>
        <p>numVisits2 = <%=++numVisits2%></p>

    </body>
</html>
```



Generated Java shows what happened

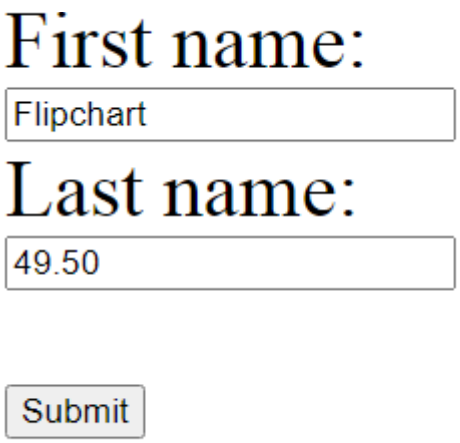


```
...
public final class JspExample04_jsp extends org.apache.jasper.runtime.HttpJspBase
  implements org.apache.jasper.runtime.JspSourceDependent {
  int numOfVisits1 = 0;
  ...
  public void _jspService(HttpServletRequest request, HttpServletResponse response)
    throws java.io.IOException, ServletException {
    ...
    _jspxFactory = JspFactory.getDefaultFactory();
    response.setContentType("text/html");
    ...
    int numOfVisits2 = 0;
    ...
  }
}
```

Form can call jsp directly

form.html

```
<body>
    <form action="form.jsp" method="post">
        First name:<br>
        <input type="text" name="product" value="Flipchart"><br>
        Last name:<br>
        <input type="text" name="price" value="49.50"><br><br>
        <input type="submit" value="Submit">
    </form>
</body>
```



Retrieve in JSP using request

form.jsp

```
<body>
    <h1>Request Parameters</h1>
    <h2><%= request.getParameter("product")%> </h2>
    <h2><%= request.getParameter("price")%> </h2>
</body>
```

Request Parameters

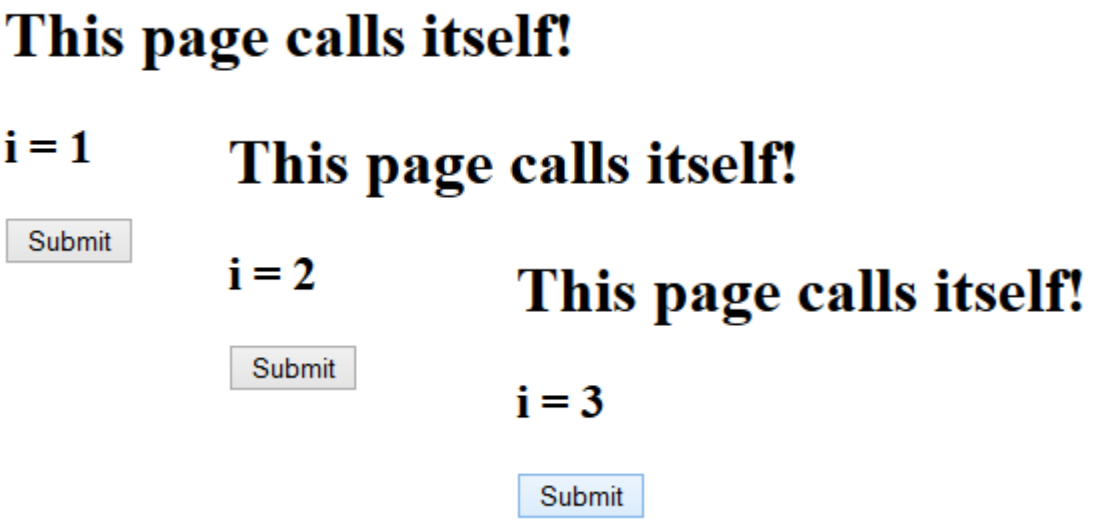
Flipchart

49.50

JSP page calls itself using <form>

callMyself.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>This page calls itself!</h1>
        <%! int i = 0; // declares a class variable %>

        <% i++;  // executed every time the jspService method is called %>

        <h2> i = <%= i%> </h2>
        <form action="index.jsp" method="post">
            <input type="submit" value="Submit">
        </form>

    </body>
</html>

```



Including other pages in JSP

Listing 13.1 WhatsNew.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>What's New at JspNews.com</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
      What's New at JspNews.com</TABLE>
```

Listing 13.1 WhatsNew.jsp *(continued)*

```
<P>
Here is a summary of our three most recent news stories:
<OL>
  <LI><jsp:include page="/WEB-INF/Item1.html" />
  <LI><jsp:include page="/WEB-INF/Item2.html" />
  <LI><jsp:include page="/WEB-INF/Item3.html" />
</OL>
</BODY></HTML>
```

Listing 13.2 /WEB-INF/Item1.html

```
<B>Bill Gates acts humble.</B> In a startling and unexpected
development, Microsoft big wig Bill Gates put on an open act of
humility yesterday.
<A HREF="http://www.microsoft.com/Never.html">More details...</A>
```

Listing 13.3 /WEB-INF/Item2.html

```
<B>Scott McNealy acts serious.</B> In an unexpected twist,
wisecracking Sun head Scott McNealy was sober and subdued at
yesterday's meeting.
<A HREF="http://www.sun.com/Imposter.html">More details...</A>
```

Listing 13.4 /WEB-INF/Item3.html

```
<B>Larry Ellison acts conciliatory.</B> Catching his competitors
off guard yesterday, Oracle prez Larry Ellison referred to his
rivals in friendly and respectful terms.
<A HREF="http://www.oracle.com/Mistake.html">More details...</A>
```

Figure 13–1 Including files at request time lets you update the individual files without changing the main page.

An MVC example

```
<body>
    <h2><a href="hello.jsp">Hello </a></h2>
    <h2><a href="maths.jsp"> Maths</a></h2>
    <h2><a href="expression.jsp">Expression </a></h2>
    <h2><a href="preDefinedVariables.jsp">Pre-defined Variables </a></h2>
    <h2><a href="fontLoop.jsp">For Loop</a></h2>
    <h2><a href="table.jsp">Table </a></h2>
    <h2><a href="tableForLoop.jsp">Table with For Loop </a></h2>
    <h2><a href="definitionList.jsp">Definition List </a></h2>
    <h2><a href="callMyself.jsp">JSP Calls Itself </a></h2>
    <h2><a href="requestParams.jsp?param1=Web&amp;param2=Application">JSP Request Parameters </a></h2>
    <h2><a href="form.html"> Simple Form</a></h2>
    <h2><a href="createProduct">Call <i>createProduct</i> servlet </a></h2>
</body>
```

Simple Servlet

createProduct.java

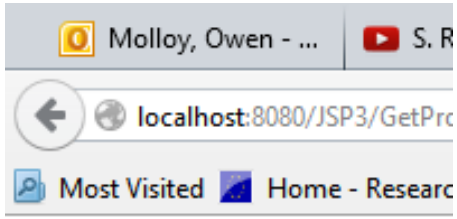
- Just puts an array of strings into the request object:

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    String products[] = {"Flipchart","Projector","Whiteboard","Chair"};

    request.setAttribute("products", products);

    RequestDispatcher dispatcher = request.getRequestDispatcher("/catalog.jsp");

    dispatcher.forward(request, response);  // forwards request to catalog.jsp
    }
```

catalog.jsp displays the string array in a table

catalog.jsp

```
<body>
      <h1>Products</h1>
      <table>
         <thead> <td> <b> Products </b></td></thead>
         <%
            String products[] = (String[]) request.getAttribute("products");
            for (int i = 0; i < products.length; i++)
            {
         %>
         <tr> <td>
           <% out.println(products[i]); %> </td> </tr>
         <%
            }
         %>
   </table>
</body>
```



Products

Products

Flipchart

Projector

Whiteboard

Chair

So far ……

- So far with JSP, we have see how to use implicit objects to access their attributes:
  - request.getParameter(“username”);
- Write Scriptlets (just ordinary Java code):
  - <% Date date = new Date(); %>
- Write expressions that can be evaluated, e.g.:
  - <%=date %>

We can write as much java as we like……

```
<html>
<body>
<%
String name=request.getParameter("uname");
out.print("welcome "+name);
%>
</form>
</body>
</html>
```

Other examples

- To run through quickly in lecture:
  - definitionList.jsp
  - expression.jsp
  - table.jsp
  - tableForLoop.jsp

To get the most out of JSP

- We need to use:
  - JSP Standard Tag Library (JSTL)
  - Expression Language (EL)

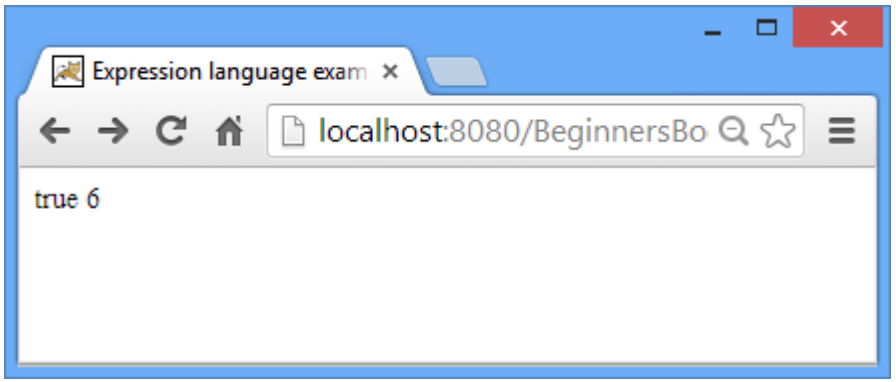
JSTL

- Like html tags, JSTL tags are used to simplify programming JSP.
- We will be using just some of the Core JSTL tags, which we will introduce as we use them, e.g.
  - <c:out>
    - Like <%=…%>
  - <c:if>
    - Used like an if in normal code
  - <c:forEach>
    - Used to loop over a collection / array
  - <c:url>
    - used to specify a url, which we can use like a href
- ***NB*** Include the following line in the header of your jsp page:
  - <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

Expression Language (EL)

- Mainly used to access properties of data classes (beans) which are attached as attributes of the request, session, etc..
- The syntax is:
  - ${ *expression* }
- For example:

```
<body>
${1<2}
${1+2+3}
</body>
```



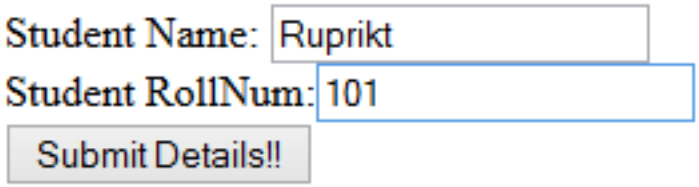
Simple EL usage

- If the servlet creates and instance (e.g. called **myUser**) of User bean class, and adds it to the request, like:
  - request.setAttribute("user", myUser);
- If the servlet forwards the request to a JSP page, it can access the beans properties like so:
  - ${user.userName}
- You don't have to specify the context – it will search the page, request, session and application, in that order, for an attribute of that name

Putting params on the request

student.jsp

```
<!DOCTYPE html>
<html>
    <head>
        <title>Expression language example2</title>
    </head>
    <body>
        <form action="display.jsp">
            Student Name: <input type="text" name="stuname" /><br>
            Student RollNum:<input type="text" name="rollno" /><br>
            <input type="submit" value="Submit Details!!"/>
        </form>
    </body>
</html>
```



Have to use *${param.xxx}*

display.jsp

```
<html>
    <head>
        <title>Display Page</title>
    </head>
    <body>
        Student name is ${ param.stuname } <br>
        Student Roll No is ${ param.rollno }
    </body>
</html>
```

Student name is Ruprikt
Student Roll No is 101

Example of Bean class

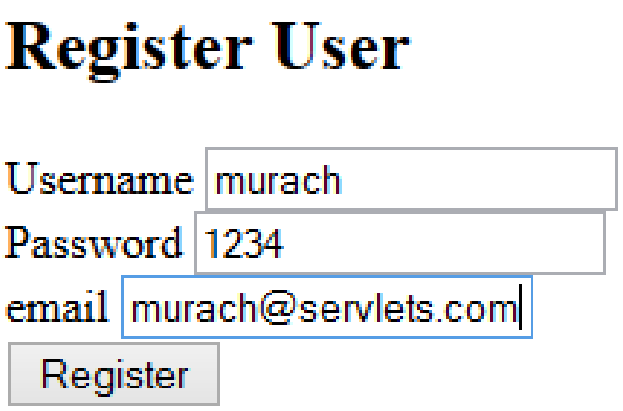
User.java

```
import java.io.Serializable;

public final class User {

    private String userName;
    private String password;
    private String email;

    public User() {

    }

    public User (String name, String password, String email)
    {
        setUserName(name);
        setPassword(password);
        setEmail(email);
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }
```

```
    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

register.html


```
<h2>Register User</h2>
<form action="registerUser" method="POST">
    Username <input type="text" name="username" value="" /><br>
    Password <input type="text" name="password" value="" /><br>
    email <input type="text" name="email" value="" /><br>
    <input type="submit" value="Register" />
</form>
```



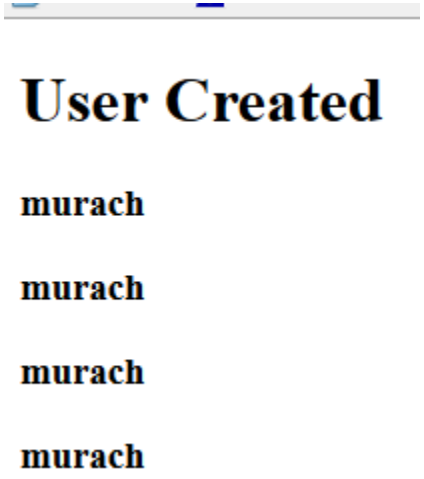
Servlet – registerUser.java

RegisterUser.java

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    String username = request.getParameter("username");
    String password = request.getParameter("password");
    String email = request.getParameter("email");

    User u1 = new User (username, password, email);
    request.setAttribute("user", u1);

    RequestDispatcher dispatcher = request.getRequestDispatcher("userCreated.jsp");
    dispatcher.forward(request, response);
}
```

userCreated.jsp


```
<body>
    <h1>User Created</h1>

    These lines all do the same thing!
    <h3>${user.userName}</h3>
    <h3><c:out value="${user.userName}" /></h3>
    <h3>${requestScope.user.userName}</h3>
    <h3><c:out value="${requestScope.user.userName}" /></h3>

</body>
```



<c:out

Just displays the result of the expression specified in *value*

Product bean

Product.java

```
*/
public final class Product  {

    private String name;
    private String description;
    private double price;
    private int ID;
    private Supplier supplier;

    public Product() {

    }

    public Product(String name, String description, double price, int ID) {
        setName(name);
        setDescription(description);
        setPrice(price);
        setID(ID);
    }

    public Supplier getSupplier()
    {
        return supplier;
    }
```

Supplier bean

```
1   package com.mycompany.week3;
2
3   public class Supplier
4   {
5
6
7       private String name;
8       private String address;
9       private String telephone;
10      private String email;
11
12      public Supplier()
13      {
14      }
15
16      public Supplier(String name, String address, String telephone, String email)
17      {
18          this.name = name;
19          this.address = address;
20          this.telephone = telephone;
21          this.email = email;
22      }
23
24      public String getName()
25      {
26          return name;
27      }
```

getProducts servlet

```
23      * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
24      * methods.
25      *
26      * @param request servlet request
27      * @param response servlet response
28      * @throws ServletException if a servlet-specific error occurs
29      * @throws IOException if an I/O error occurs
30      */
31     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
32             throws ServletException, IOException {
33
34         Product p1 = new Product("Whiteboard", "Just a white board", 50.00, 101);
35         Product p2 = new Product("Stapler", "Just a staple stapler", 10.00, 102);
36         Product p3 = new Product("Chair", "Standard office chair", 40.00, 103);
37         Product p4 = new Product("Lamp", "Anglepoise!", 25.00, 104);
38
39         Supplier s1 = new Supplier ("ACME Products", "Coyote Avenue", "1800-34800", "sales@acme.com");
40         Supplier s2 = new Supplier ("Staples Office Supplies", "Beijing", "00-86-6513-0828", "info@staples.cn");
41
42         p1.setSupplier(s2);
43         p2.setSupplier(s2);
44         p3.setSupplier(s1);
45         p4.setSupplier(s1);
46
47         List<Product> products = new ArrayList<>();
48         products.add(p1);
49         products.add(p2);
50         products.add(p3);
51         products.add(p4);
52
53         HttpSession session = request.getSession();
54         session.setAttribute("catalogue", products);
55
56         RequestDispatcher dispatcher = request.getRequestDispatcher("displayProducts.jsp");
57         dispatcher.forward(request, response);
58     }
```

displayProducts.jsp

Source History

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Products</title>
    </head>
    <body>
        <h1>Product Catalogue</h1>
        <table>
            <thead>
                <tr>
            <b>
                <td>Code</td><td>Name</td><td>Description</td><td>Price</td><td>Supplier</td>
            </b>
        </tr>
    </thead>
    <c:forEach var="prod" items="${catalogue}">
        <tr>
            <td>${prod.ID}</td><td>${prod.name}</td><td>${prod.description}</td><td>${prod.price}</td>
            <td>
                <form action="viewSupplier.jsp" method="post">
              <input type="hidden" name="id" value="${prod.ID}">
                    <input type="submit" value="View">
                </form>
            </td>
        </tr>
    </c:forEach>
</table>
</body>
</html>
```

viewSupplier.jsp

```jsp
<a href="viewSupplier.jsp"></a>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
        <link href="css/styles.css" rel="stylesheet">
        <title>Supplier Details</title>
    </head>
    <body>
        <h1>Supplier Details</h1>
        <table>
            <c:forEach var="prod" items="${catalogue}">
                <c:if test="${prod.ID == param.id}">
                    <tr>
                        <td>Name</td><td>${prod.supplier.name}</td>
                    </tr>
                    <tr>
                        <td>Address</td><td>${prod.supplier.address}</td>
                    </tr>
                    <tr>
                        <td>Telephone</td><td>${prod.supplier.telephone}</td>
                    </tr>
                    <tr>
                        <td>Email</td><td>${prod.supplier.email}</td>
                    </tr>
                </c:if>
            </c:forEach>
        </table>
    </body>
</html>
```

output

Product Catalogue

Code	Name	Description	Price	Supplier
101	Whiteboard	Just a white board	50.0	View
102	Stapler	Just a staple stapler	10.0	View
103	Chair	Standard office chair	40.0	View
104	Lamp	Anglepoise!	25.0	View

Supplier Details

Name	ACME Products
Address	Coyote Avenue
Telephone	1800-34800
Email	sales@acme.com

Explanation

- <c:forEach
  - Like a for loop – you specify the collection you want to iterate over (*items*), and what the individual item variable is (*var*)
  - So, in this case, each product object in the collection stored in the attribute *catalogue*, will be stored in turn in the variable "prod"
  - So, then ${prod.name} for example, just evaluates to the name attribute of a Product object stored in the collection
  - What really happens behind the scenes is the *getter* is called for that attribute
- Important not to have any other attributes on the request, session etc., called "prod" in this case, as it could cause an error

<c:set>

- Used to set a property of a Java Bean object. The ${…} part is an *expression*, so it is evaluated

```
<c:set var="salary" scope="session" value="${2000*2}"/>
<c:out value="${salary}"/>
```

  - Will output 4000

```
<c:set var="num" scope="page" value="${125*3.2}"/>
<p> ${num} </p
```

  - Will output 400

- ***Generally, we try to avoid setting attributes in EL, as the JSP is supposed to be the View, not the Controller!***

<c:if>

```
<c:if test="${num < 500}">
    <p> smaller than 500! </p>
</c:if>
```

- Fairly self-explanatory !

<c:choose>
<c:when>
<c:otherwise>

- Bit like a *switch* statement

```
<c:set var="salary" scope="session" value="${2000*2}"/>
<p>Your salary is : <c:out value="${salary}"/></p>

<c:choose>
    <c:when test="${salary <= 0}">
        Salary is very low to survive.
    </c:when>
    <c:when test="${salary > 1000}">
        Salary is very good.
    </c:when>
    <c:otherwise>
        No comment sir...
    </c:otherwise>
</c:choose>
```

Your salary is : 4000

Salary is very good.

Java

```
int day = 3;
String dayString;
switch (day) {
    case 1:
        dayString = "Monday";
        break;
    case 2:
        dayString = "Tuesday";
        break;
    case 3:
        dayString = "Wednesday";
        break;
    case 4:
        dayString = "Thursday";
        break;
    case 5:
        dayString = "Friday";
        break;
    default:
        dayString = "Weekend!";
        break;
}
```

<c:import>

- Used to fetch content from a url and put it into a variable (or import into the page if not variable specified)
- E.g.

```
<c:import var="data" url="http://www.it.nuigalway.ie"/>
<c:out value="${data}"/>
```

- Gives: