WKW SCHOOL

2020

CI6206 Internet Programming

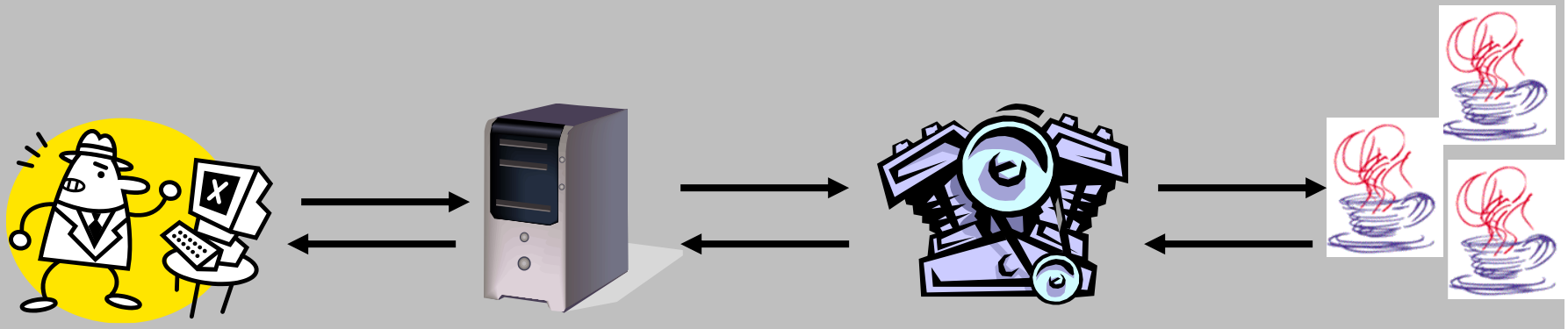
SERVLET & JSP

Wong Twee Wee

Ver1.1

PREVIOUS LECTURE …

- Java-based Web component that generates dynamic content
- Managed by a **container/servlet** engine
  - Part of a Web server or application server
  - Processes HTTP requests and responses



PREVIOUS LECTURE …

- Web applications – resources that make up a complete application on a Web server
  - Servlets, JSP pages, Java classes
  - Static documents (HTML, images, sounds, etc.)
  - Descriptive meta information (deployment descriptor)
- Represented by a hierarchy rooted at context path
  - http://host.com/context-path/…

Context-path

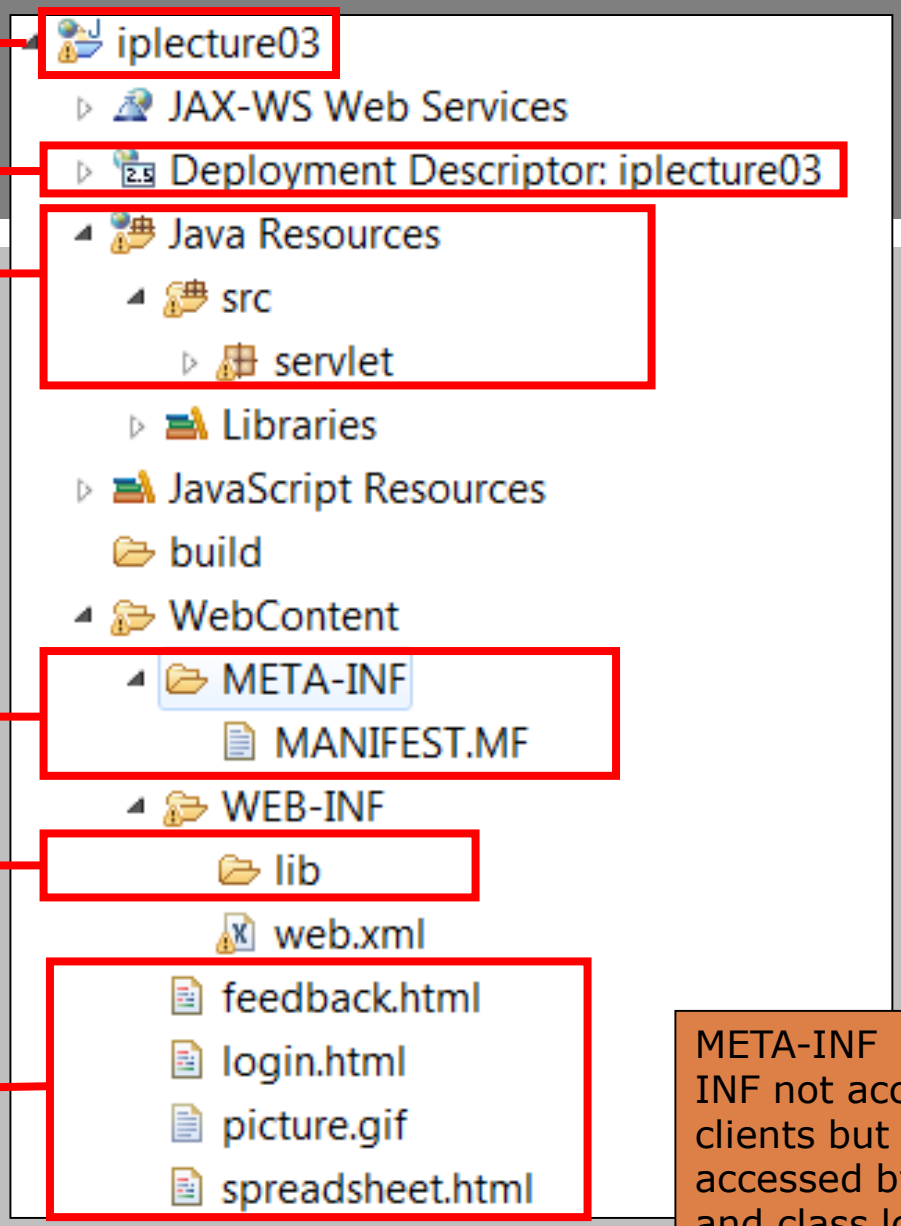
Deployment descriptor

Servlets and Java utility classes can be in subdirectories

Meta information for WAR files

Jar files

JSPs and regular Web content (HTML, CSS, images, etc).



META-INF and WEB-INF not accessible to clients but can be accessed by servlets and class loaders

TOPIC

- The need for JavaServer Pages
- The JSP Approach
- JSP Syntax
  - directives
  - scripting elements
  - action elements
- RequestDispatcher

- http://www.jsptut.com/

THE NEED FOR JSP

- With servlets, it is easy to
  - Read form data
  - Read HTTP request headers
  - Set HTTP status codes and response headers
  - Perform session tracking
  - Share data among servlets

But it is difficult to…

- *Generate HTML using* `println` *methods*
- *Maintain HTML code*

```
import java.io.*;
import javax.servlet.*;

public HelloWorldServlet extends HttpServlet {
   public void doGet( HttpServletRequest  req,
       HttpServletResponse res) throws
                    ServletException, IOException {
       res.setContentType("text/html");
       PrintWriter out = res.getWriter();
       out.println("<html>");
       out.println("<head>");
       out.println("<title>FirstServlet</title>");
       out.println("</head>");
       out.println("<body>");
       out.prinltn("<h1>Hello World!!</h1>");
       out.println("</body>");
       out.println("</html>");
       out.close();
   }
}
```

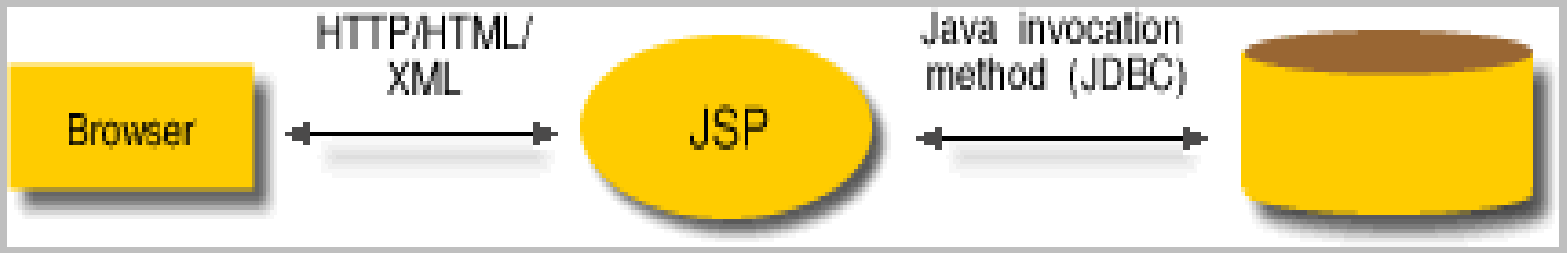
Use PrintWriter to generate dynamic HTML stream for the client,

THE NEED FOR JSP

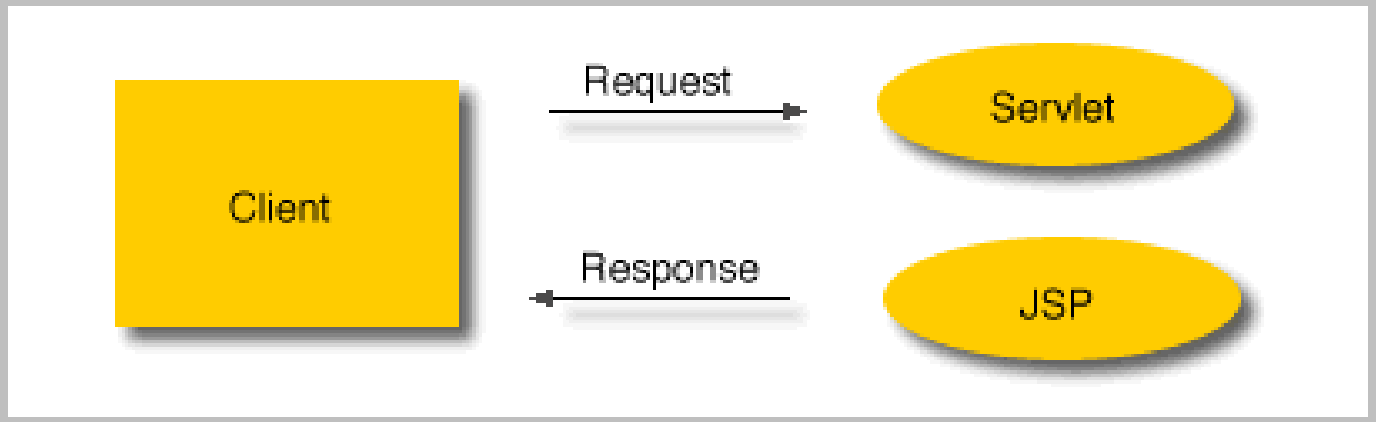
- JSP makes it easier to
  - Read, write and maintain HTML – Easier to maintain than servlet.
- JSP makes it possible to
  - use standard HTML tools
  - have different members of a team do HTML layout and Java programming
- JSP encourages separation of dynamic and static content
  - *The dynamic contents are generated via programming logic and inserted into the static template*
- Emphasizing reusable components (tag libraries)
- Built on Java Technology so it is platform independent.

APPLICATION MODEL

- Simple Application



- Flexible Application with Java Servlets



THE JSP APPROACH

- "HTML with embedded Java code"
  - Use regular HTML for most of page
  - Mark servlet code with special tags
  - Compilation
    - JSP page complied into a servlet (once)
  - Execution
    - Servlet actually invoked during HTTP requests
  - JSP and servlets are equivalent

```
<HTML><HEAD><TITLE>Order Confirmation</TITLE>
<LINK REL=STYLESHEET HREF="JSP-styles.css" TYPE="text/css"></HEAD>
<BODY><H1>Order Confirmation</H1>
Thanks for ordering <I><%= request.getParameter("title") %></I>!
</BODY></HTML>
```

ADV OF JSP OVER SERVLETS

- Extension to Servlet
  - JSP technology is the extension to Servlet technology.
  - Able to use all the features of the Servlet in JSP.
- Easy to maintain
  - JSP: Separation of business logic with presentation logic.
  - Servlet: mix business logic with the presentation logic.
- Fast Development: No need to recompile and redeploy
  - JSP : IF modified, not required to recompile and redeploy the project.
  - Servlet: If modified needs to be updated and recompiled.
- Less code than Servlet
  - The use many tags reduces the need to write more codes.

A SIMPLE JSP PAGE

```
<%@ page language=“java”%>
<%@ page contentType="text/html; charset=ISO-8859-1"%>
<html>
<head>
<title>A simple case</title>
</head>
<body bgcolor="#FFFFFF" text="#000000">
<% String temp="the time now is:";%>
<%=temp%>
<%=(new java.util.Date()).toString()%>
</body>
</html>
```

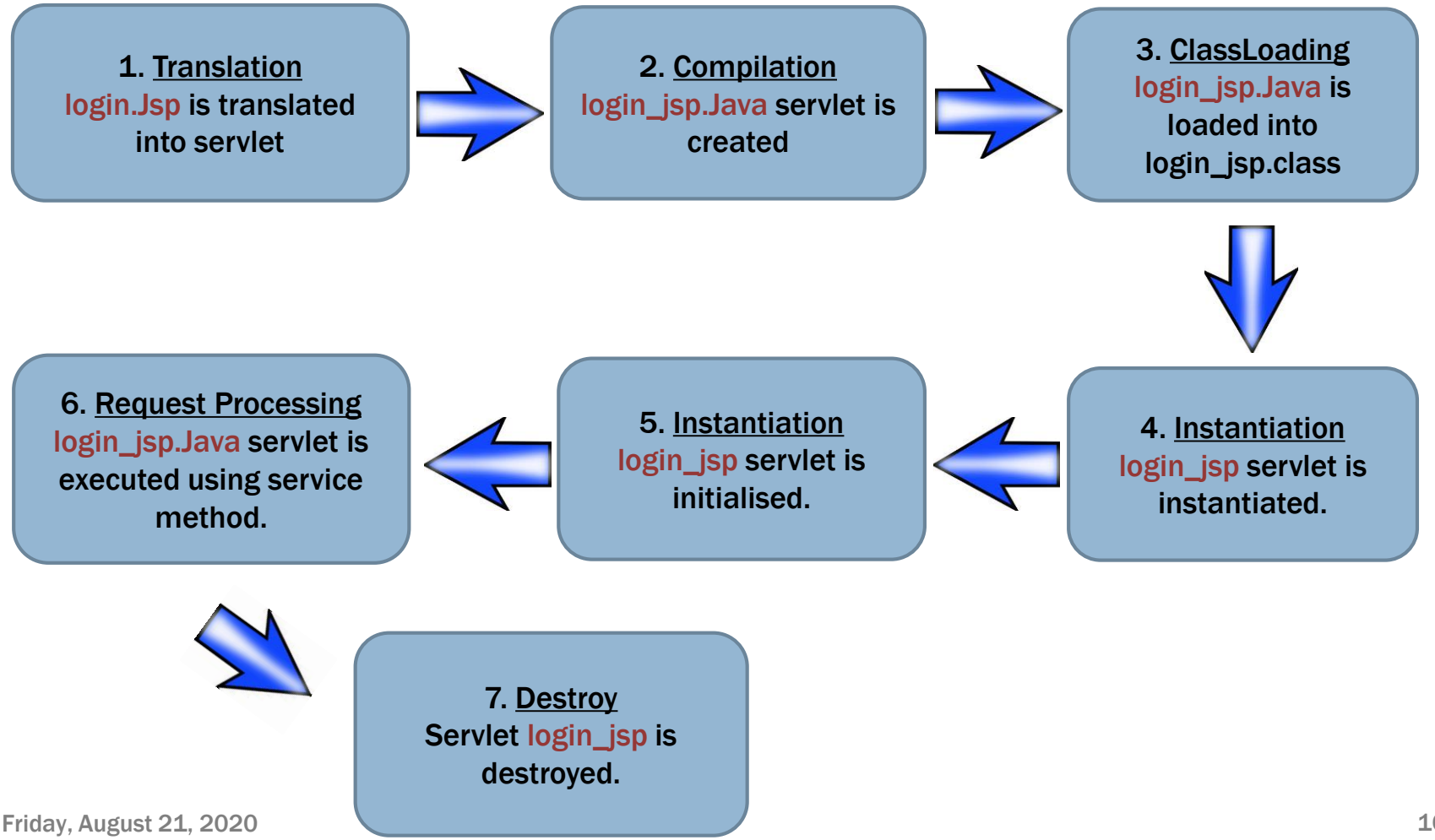
JSP LIFECYCLE

- JSP has two phases
  - Translation phase
  - Request processing phase

- **Translation phase**
  - The process of converting jsp into an equivalent Servlet and then generating class file of the Servlet.
  - Only happens :
    - When first request is a given to the jsp.
    - When a jsp is modified.

- **Request processing phase**
  - The process of executing service() of jsp and generating response data on to the browser.

JSP LIFECYCLE

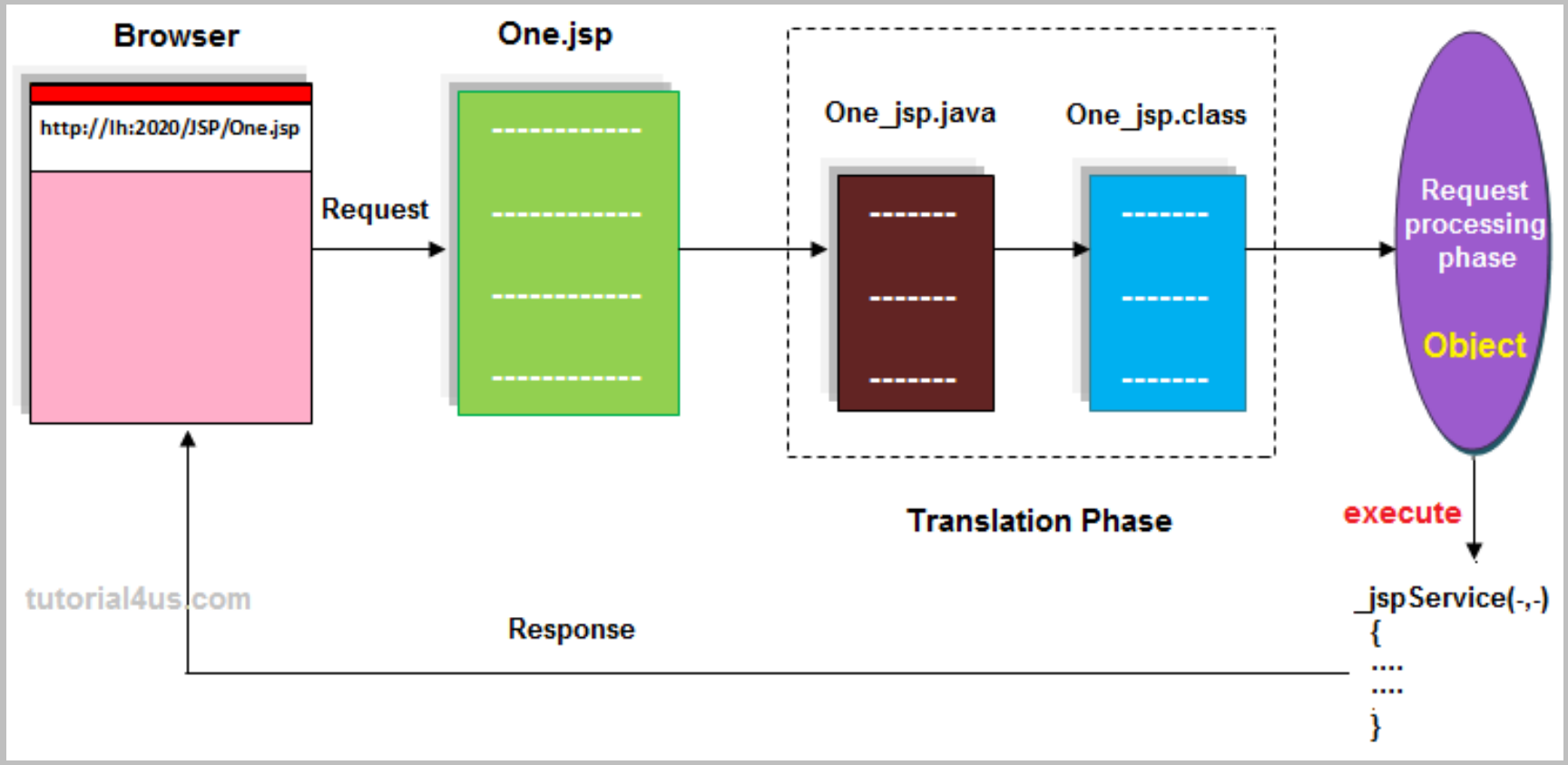


JSP LIFECYCLE

- Translation of JSP page
- Compilation of JSP page(Compilation of JSP page into _jsp.java)
- Classloading (_jsp.java is converted to class file _jsp.class)
- Instantiation(Object of generated servlet is created)
- Initialisation(_jspinit() method is invoked by container)
- Request Processing(_jspservice() method is invoked by the container)
- Destroy (_jspDestroy() method invoked by the container

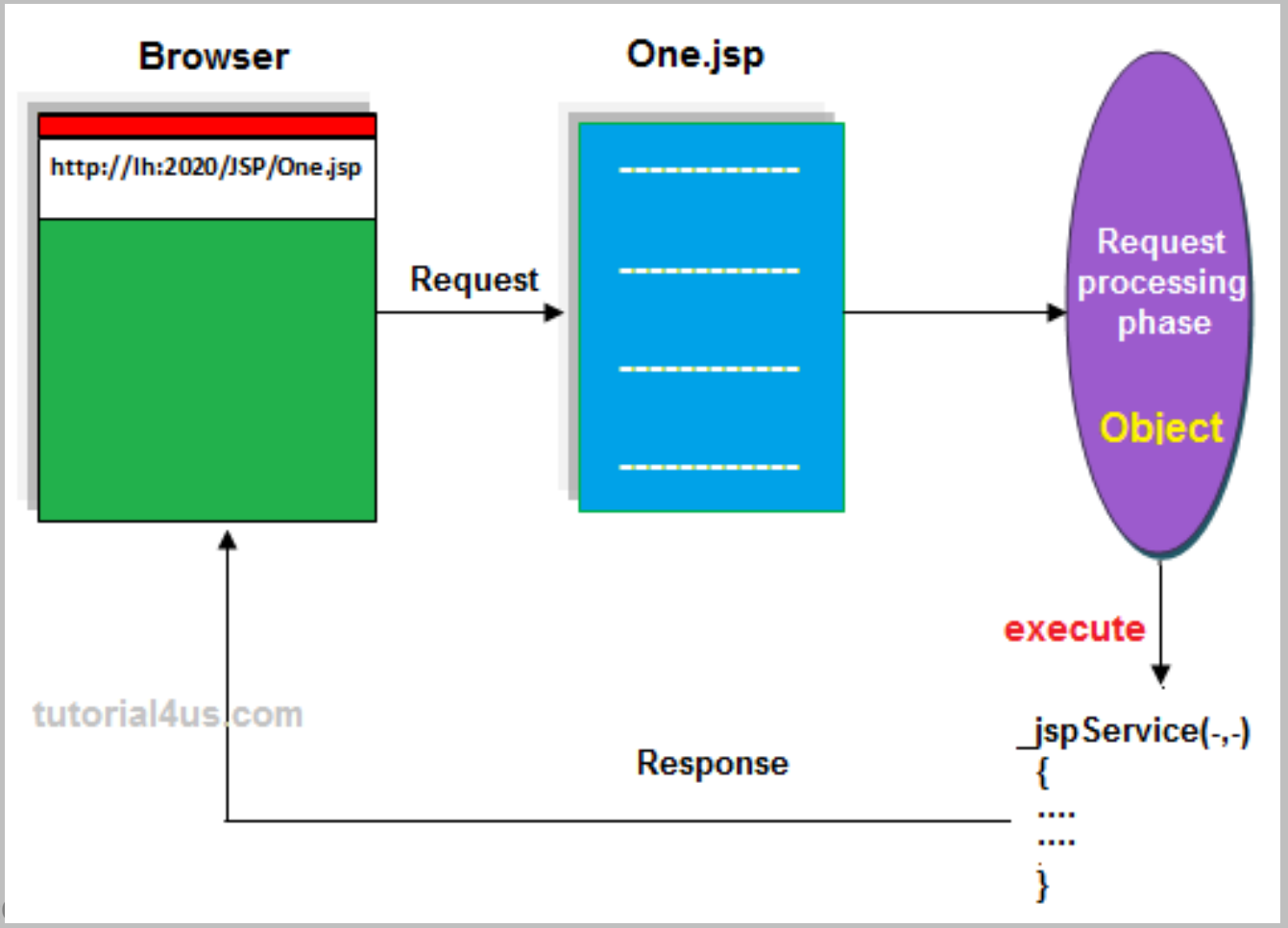
JSP PROCESS

First Request or Modified JSP or deleted class.java files



JSP PROCESS

Next request



TRANSLATION OF JSP

Demo.jsp

```
<html>
<head>
<title>Demo JSP</title>
</head>
<%
int demvar=0;%>
<body>
Count is:
<% Out.println(demovar++); %>
<body>
</html>
```

```
1  Public class demp_jsp extends HttpServlet{
2     Public void _jspservice(HttpServletRequest request, HttpServletResponse response)
3           Throws IOException, ServletException
4       {
5  PrintWriter out = response.getWriter();
6  response.setContentType("text/html");
7  out.write("<html><body>");
8  int demovar=0;
9  out.write("Count is:");
10 out.print(demovar++);
11 out.write("</body></html>");
12 }
13 }
14
```

JSP SYNTAX

- In general, JSP syntax can be divided into three forms.
  - Directives
    - <%@ … %>
  - Scripting Elements
    - Expressions <%= … %>
    - Scriplets <% … %>
    - Declarations <%! … %>
  - Action Elements (JSP tags)
    - <jsp:include>, <jsp:forward>
    - <jsp:useBean>
    - <jsp:setProperty>, <jsp:getProperty>

http://beginnersbook.com/2013/11/jsp-include-action-tag/

JSP DIRECTIVES

- JSP directives are the messages to JSP container.
  - provides global information about an entire JSP page.
  - Provides special instruction to a container for translation of JSP to servlet code.
- Format
  - `<%@ directive attribute="value" %>`
- Types
  - `page` – controls structure of servlet
  - `include` – inserts file into a JSP
  - `taglib` – defines custom tags

```
<%@ directive_name
  attribute1="value1"
  attribute2="value2"
  ......
  attributeN="valueN" %>
```

THE JSP PAGE DIRECTIVE

- `import` **attribute**
  - `<%@ page import="package.class1, ..., package.classN"%>`
  - `<%@ page import="java.util.*" %>`
  - `<%@ page import="java.util.*,java.text.*" %>`
- `contentType` **attribute**
  - `<%@ page contentType= "application/vnd.ms-excel" %>`
  - `<%@ page contentType="text/html" %>`
  - **Use** `response.setContentType()`
- `errorPage` **attribute**
  - `<%@ page errorPage = "error.html" %>`
  - **If error occurs, redirect the client to the error.html page.**

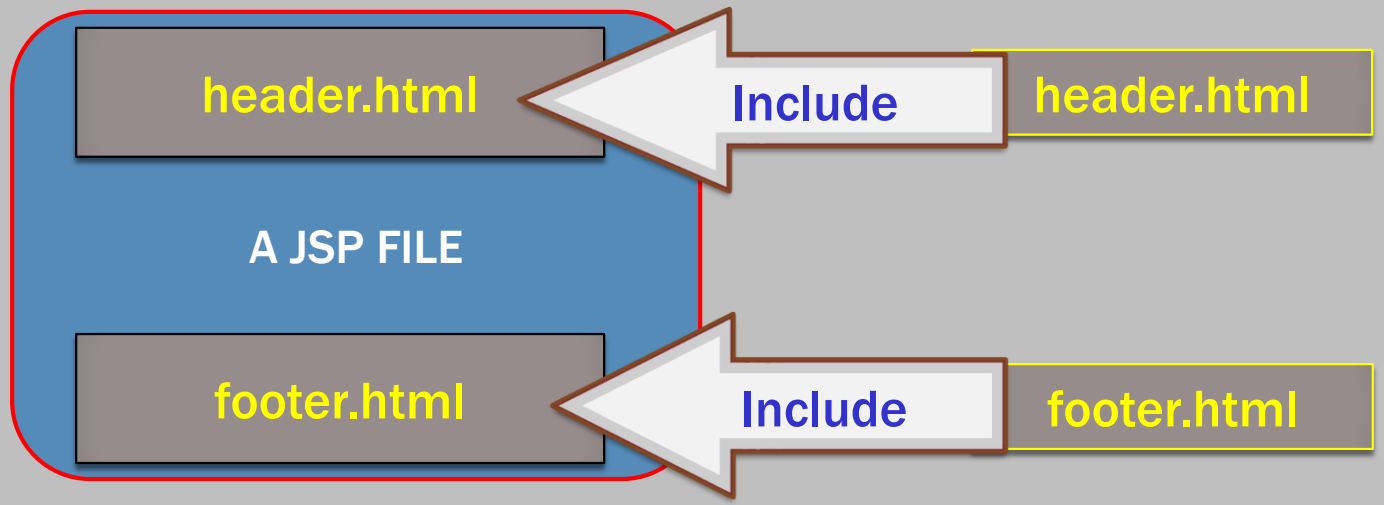
THE JSP INCLUDE DIRECTIVE

- include **directive**
  - <%@ include file="*Relative URL*" %>
  - **Preprocessor - Invoked at page translation time**
  - **Inserts contents of file into main JSP page**
  - **Composite page is then translated into servlet**

```
<%@ include file="header.html" %>
......
<%@ include file="footer.html" %>
```

EXAMPLE:

```
<html>
  <body>
       <%@ include file="header.html" %>
       This is my jsp file
       <%@ include file="footer.html" %>
  </body>
</html>
```



EXAMPLE

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Home Page</title>
</head>
<body>

    <%@include file="header.html" %>

    <hr/>
    <h2>This is main content</h2>
    <hr/>

    <%@include file="footer.html" %>

</body>
</html>
```

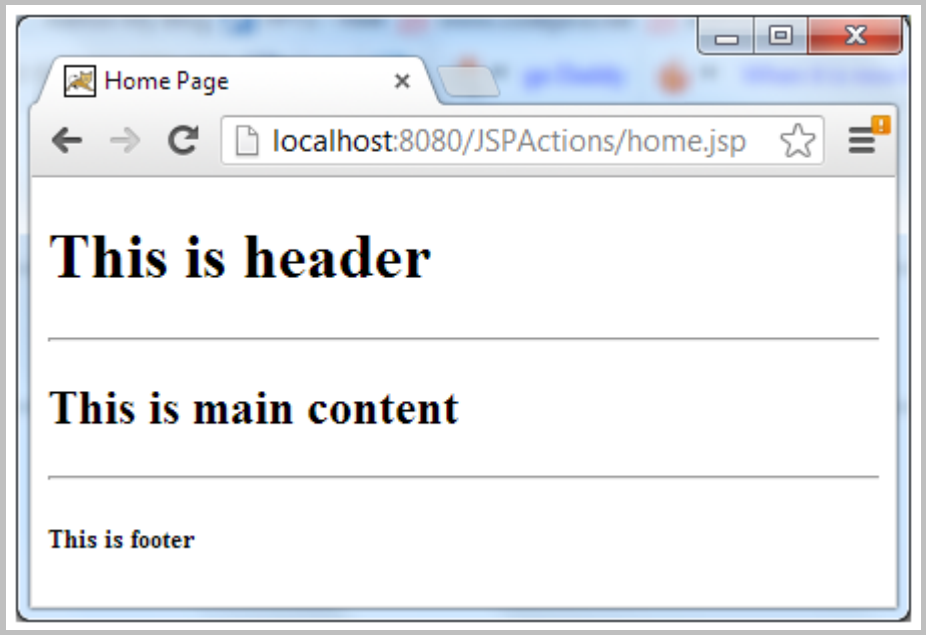
EXAMPLE

Header.html

```
<h1>This is header</h1>
```

Footer.html

```
<h5>This is footer</h5>
```



Try them from your IDE Eclipse!

<JSP:INCLUDE>

```
<html>
<head>
<title>JSP Include example</title>
</head>
<body>
<b>index.jsp Page</b><br>
<jsp:include page="Page2.jsp" />
</body>
</html>
```

JSP SCRIPTING ELEMENTS

- There are three kinds of scripting elements
  - Expressions
  - Declarations
  - Scriptlets

JSP EXPRESSION

- To insert a single Java expression directly into the response message.
- ***<%= expression %>***

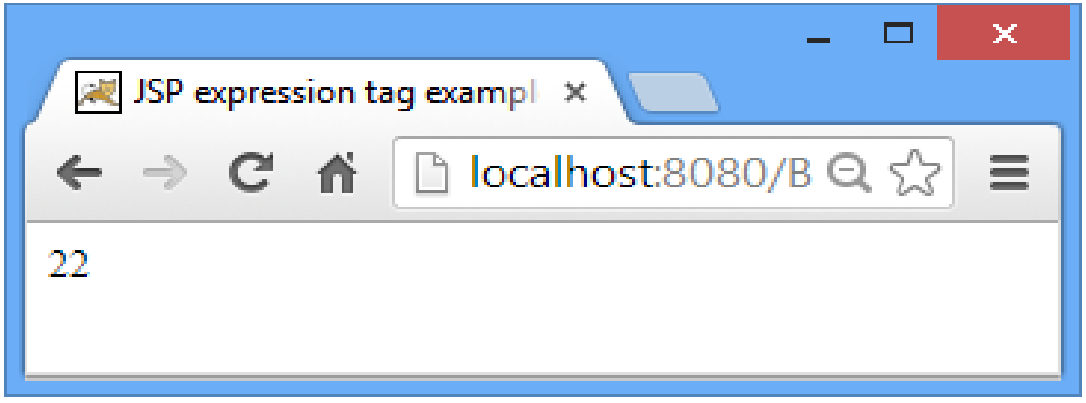
  <jsp:expression>expression </jsp:expression>

- Expression evaluated, converted to *String*, and placed into HTML page at same location as in JSP page

```
<ul>
<li>Current time: <%= new java.util.Date() %>
<li>Your hostname: <%= request.getRemoteHost() %>
</ul>
```

JSP EXPRESSION

```
<html>
 <head>
   <title>JSP expression tag example1</title>
 </head>
 <body>
   <%= 2+4*5 %>
 </body>
 </html>
```



JSP SCRIPTLETS

- To write JAVA related codes within a JSP page
- **<% *Java code* %>**
- <jsp:scriptlet>*Java code*</jsp:scriptlet>
  - Allows to write blocks of Java code inside HTML.
  - Each block is known as a “Scriptlet”
  - Java code inserted into compiled servlet
  - Scriptlet has access to all implicit objects

SCRIPTLET EXAMPLE

```
<HTML>
<BODY>
<%
// This is a scriptlet.

System.out.println( "Evaluating date now" );
java.util.Date date = new java.util.Date();

%>

The time is now <%= date %>
</BODY>
</HTML>
```

JAVA CODES IN HTML

```
<HTML>
<BODY>
<%
   // This scriptlet declares and initializes "date"
   System.out.println( "Evaluating date now" );
   java.util.Date date = new java.util.Date();
%>
The time now is :
<%
   // This scriptlet generates HTML output
   out.println( String.valueOf( date ));
%>
</BODY>
</HTML>
```

MORE EXAMPLES

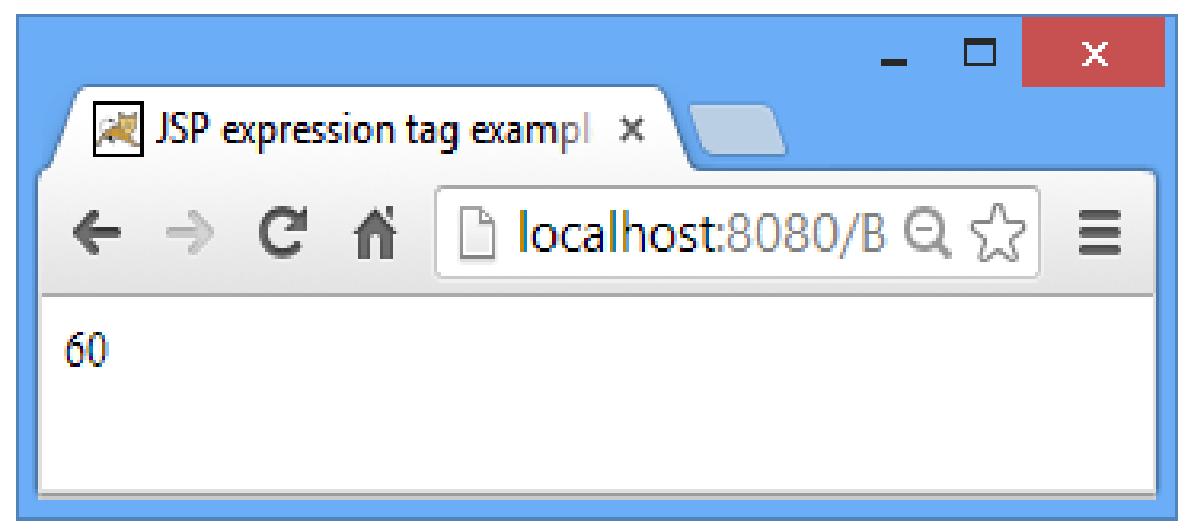
```
<HTML> <BODY>
<%
    // This scriptlet declares and initializes "date"
    System.out.println( "Evaluating date now" );
    java.util.Date date = new java.util.Date();
%>
The time now
<%
    out.println( date );
    out.println( "<BR>Your machine's address is " );
    out.println( request.getRemoteHost()); %>
</BODY>
</HTML>
```

MORE EXAMPLES

```
<HTML> <BODY>
<TABLE BORDER=2>
<%     for ( int i = 0; i < 10; i++ ) {    %>
<TR>
  <TD> Number       </TD>
  <TD> <%= i+1 %> </TD>
</TR>

<%     } %>

</TABLE>
</BODY> </HTML>
```

EXPRESSION & SCRIPTLETS

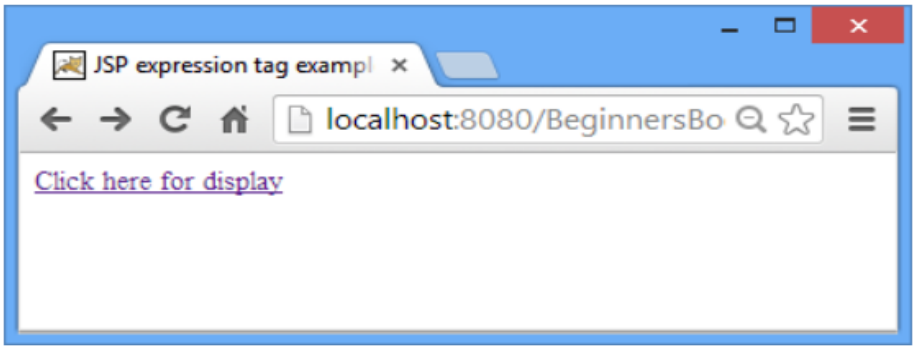
```
<html>
<head>
   <title>JSP expression tag example2</title>
</head>
<body>
  <%
  int a=10;
  int b=20;
  int c=30;
  %>
  <%= a+b+c %>
</body>
</html>
```



EXPRESSION & SCRIPTLETS

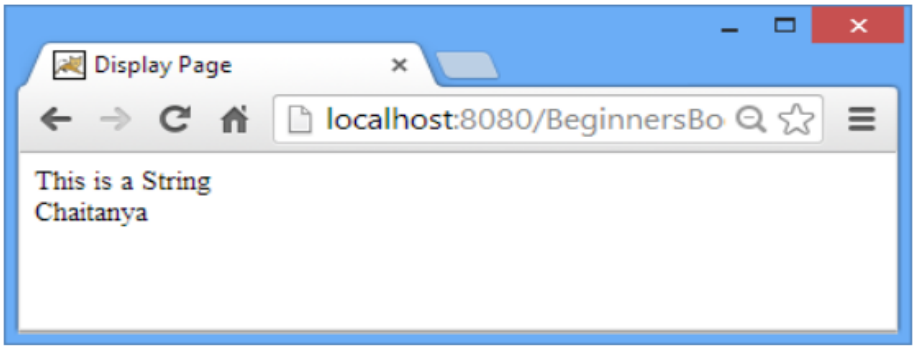
Index.jsp

```
<html>
<head>
<title> JSP expression tag example3 </title>
</head>
<body>
  <% application.setAttribute("MyName", "Chaitanya"); %>
  <a href="display.jsp">Click here for display</a>
</body>
</html>
```



Display.jsp

```
<html>
<head>
<title>Display Page</title>
</head>
<body>
 <%="This is a String" %><br>
 <%= application.getAttribute("MyName") %>
</body>
</html>
```



JSP DECLARATION

- To define methods & instance variables
  - Does not produce any output that is sent to client
  - Embedded in <%! and %> delimiters
  - <jsp:declaration>*Field/method definition*</jsp:declaration>
  - **Code is inserted verbatim into servlet's class definition outside of any existing methods**
    - **Useful for fields but use separate class for methods**

```
<%! private int someField = 0; %>
<%!
  private String someMethod() {return "Math.random()";}
%>
... <%= someMethod() %> ...
```

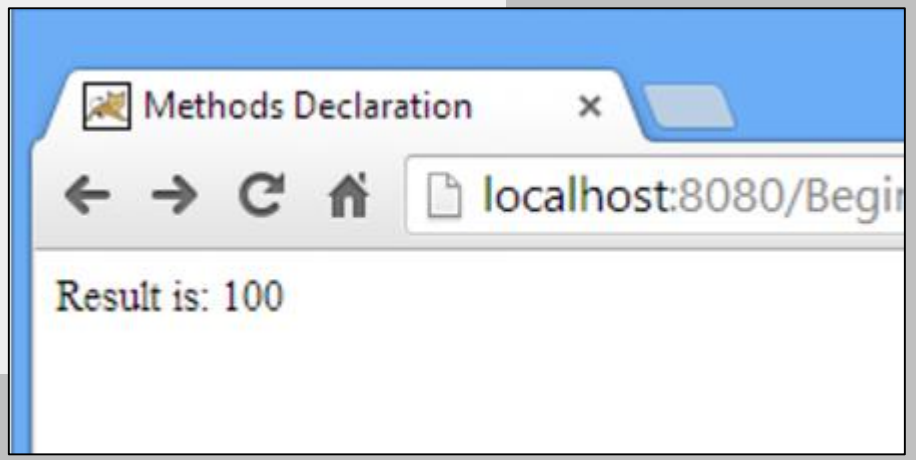
EXAMPLES

```
<HTML>
 <HEAD>
  <TITLE>Creating a Method in jsp</TITLE>
 </HEAD>

 <BODY>
  <H1>Creating a Method in jsp</H1>
  <%!
    int Double(int number){
       return 2*number;
    }
  %>
  <%
   out.println("The Double of 3 is = " + Double(3));
  %>
 </BODY>
</HTML>
```

EXAMPLE

```
<html>
<head>
 <title>Methods Declaration</title>
</head>
<body>
  <%!
  int sum(int num1, int num2, int num3){
  return num1+num2+num3;
  }
  %>

  <%= "Result is: " + sum(10,40,50) %>
</body>
</html>
```

Use of **declaration tag** and displaying results on client using **expression tag**



JSP IMPLICIT OBJECTS

- provide access to server side objects – e.g request, page, session, or application
- can directly use them within Scriptlet without initializing and declaring them.

Implicit Object	In Servlet
request	HTTPServletRequest
response	HTTPServletResponse
out	PrintWriter - out
session	HttpSession
application	ServletContext
config	ServletConfig

JSP IMPLICIT OBJECTS

Implicit	Servlet
out	javax.servlet.jsp.JspWriter
request	javax.servlet.http.HttpServletRequest
response	javax.servlet.http.HttpServletResponse
session	javax.servlet.http.HttpSession
application	javax.servlet.ServletContext
exception	javax.servlet.jsp.JspException
page	java.lang.Object
pageContext	javax.servlet.jsp.PageContext
config	javax.servlet.ServletConfig

IMPLICIT OBJECTS

List of Implicit Objects

- request: Reference to the current request
- response: Response to the request
- session: session associated woth current request
- application: Servlet context to which a page belongs
- pageContext: Object to access request, response, session and application associated with a page
- config: Servlet configuration for the page
- out: Object that writes to the response output stream
- page: instance of the page implementation class (this)
- exception: Available with JSP pages which are error pages

EXAMPLES

```
<html>
  <head>
    <title>Implicit Objects</title>
  </head>
  <body style="font-family:verdana;font-size:10pt">
    <p>
      Using Request parameters...<br>
      <b>Name:</b> <%=
          request.getParameter("name") %>
    </p>
    <p>
      <% out.println("This is printed using the out
          implicit variable"); %>
    </p>
    <p>
      Storing a string to the session...<br>
      <% session.setAttribute("name", "Meeraj");
          %>
      Retrieving the string from session...<br>
      <b>Name:</b> <%=
          session.getAttribute("name") %>
    </p>
    <p>
      Storing a string to the application...<br>
      <% application.setAttribute("name", "Meeraj"); %>
      Retrieving the string from application...<br>
      <b>Name:</b>
      <%= application.getAttribute("name") %>
    </p>
    <p>
      Storing a string to the page context...<br>
      <% pageContext.setAttribute("name", "Meeraj"); %>
      Retrieving the string from page context...</br>
      <b>Name:</b>
      <%= pageContext.getAttribute("name") %>
    </p>
  </body>
</html>
```

FORMS AND JSP

Index.html

```
<html>
<head>
<title>Enter UserName and Password</title>
</head>
<body>
<form action="userinfo.jsp">
Enter User Name: <input type="text" name="uname" /> <br><br>
Enter Password: <input type="text" name="pass" /> <br><br>
<input type="submit" value="Submit Details"/>
</form>
</body>
</html>
```

USERINFO.JSP

```
<%@ page import = " java.util.* " %>
<html>
<body>
<%
String username=request.getParameter("uname");
String password=request.getParameter("pass");
out.print("Name: "+username+" Password: "+password);
%>
</body>
</html>
```

EXCEPTION HANDLING

2 methods

- Exception handling using exception **implicit object**
- Exception handling using try catch blocks within scriptlets

EXCEPTION WITH IMPLICIT OBJECT

```jsp
<%@ page errorPage="errorpage.jsp" %>
<html>
<head>
  <title>JSP exception handling example</title>
</head>
<body>
<%
     //Declared and initialized two integers
     int num1 = 122;
     int num2 = 0;

     //It should throw Arithmetic Exception
     int div = num1/num2;
%>
</body>
</html>
```

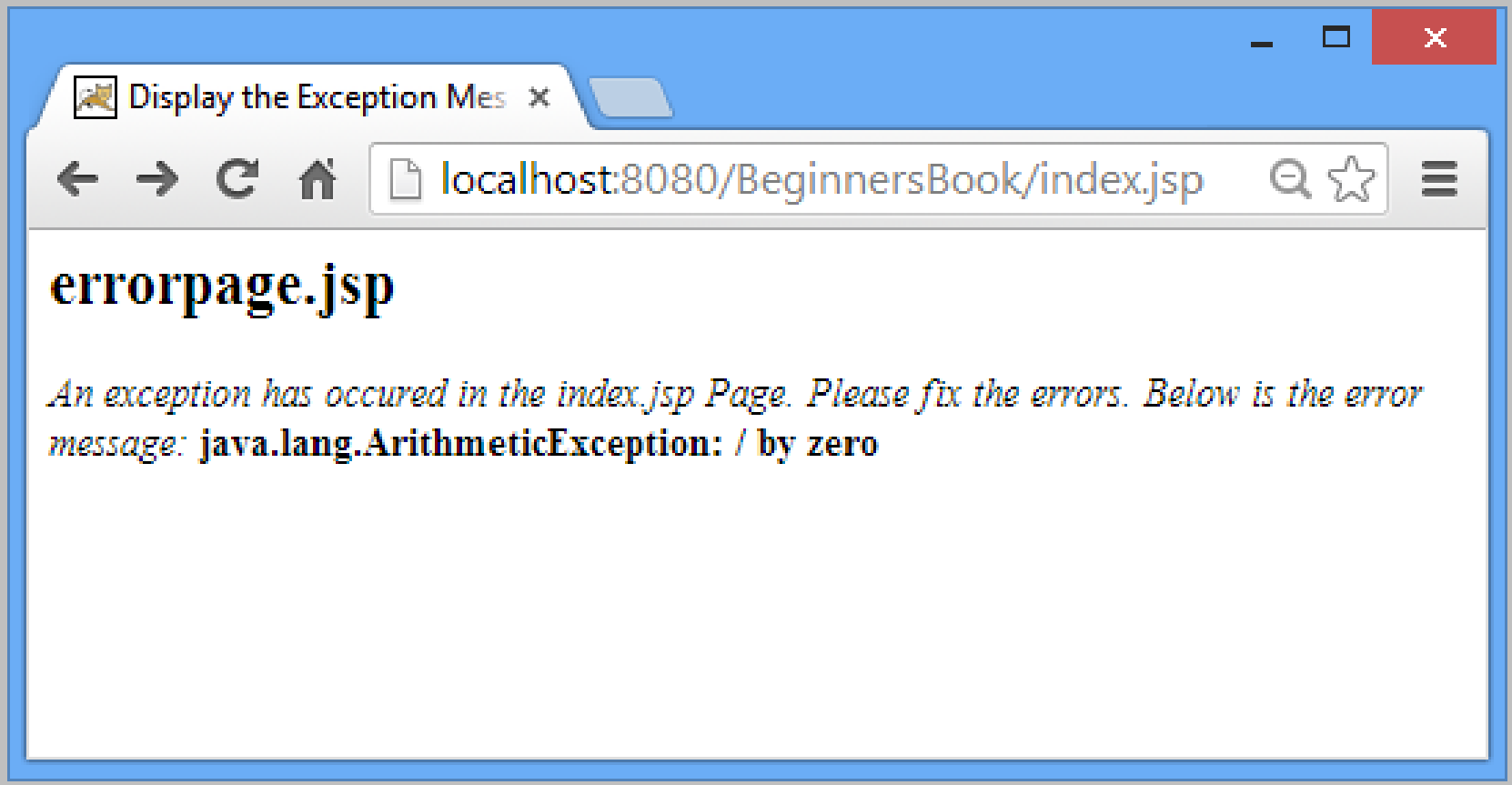
Index.jsp

ERRORPAGE.JSP

The handler page should have **isErrorPage set to true** in order to use **exception implicit object**.

```
<%@ page isErrorPage="true" %>
<html>
<head>
  <title>Display the Exception Message here</title>
</head>
<body>
   <h2>errorpage.jsp</h2>
   <i>An exception has occurred in the index.jsp Page.
   Please fix the errors. Below is the error message:</i>
   <b><%= exception %></b>
</body>
</html>
```

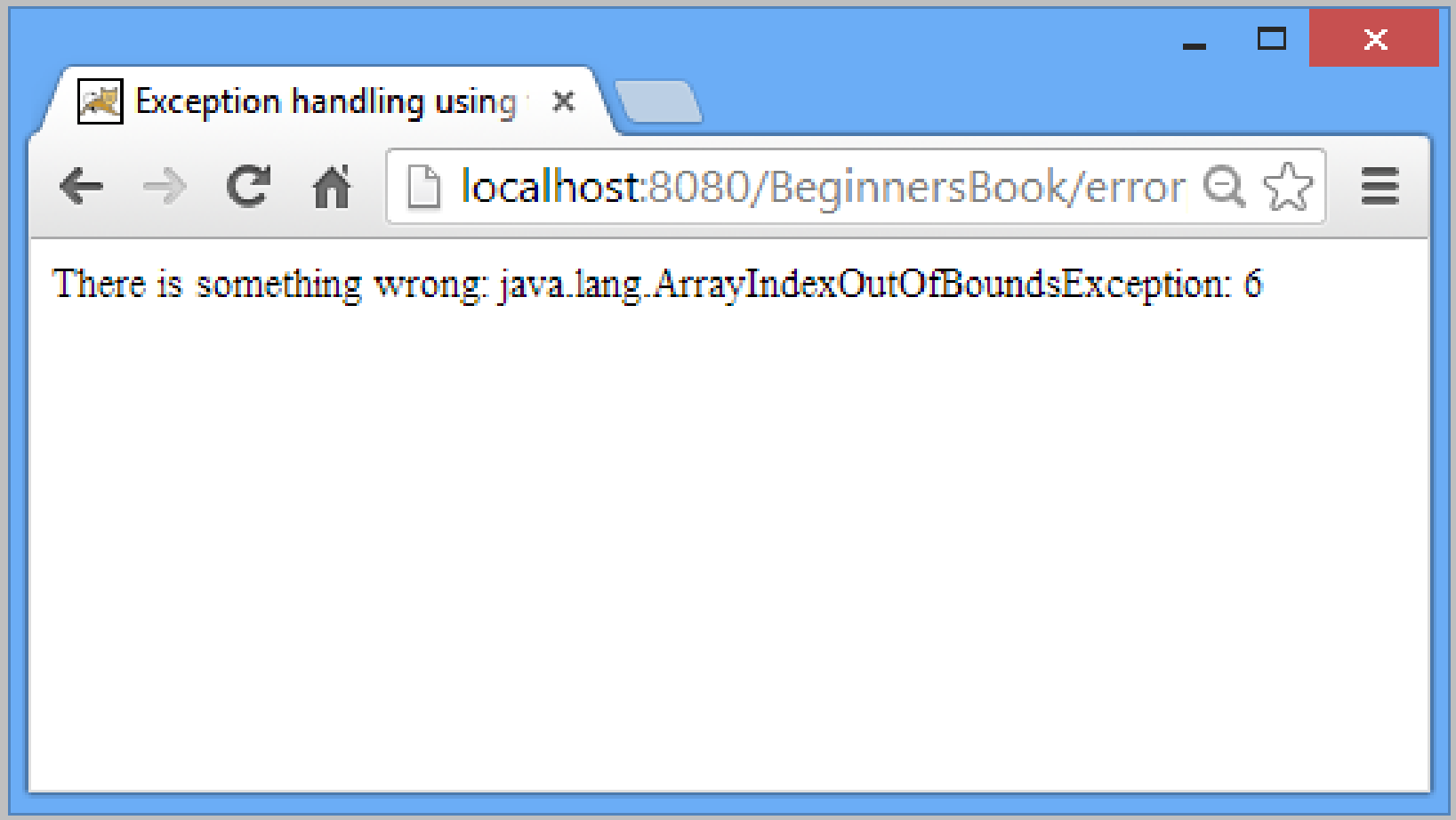
OUTPUT


Display the Exception Mes
localhost:8080/BeginnersBook/index.jsp
errorpage.jsp
An exception has occured in the index.jsp Page. Please fix the errors. Below is the error message: java.lang.ArithmeticException: / by zero

EXCEPTION HANDLING USING TRY CATCH BLOCKS SCRIPTLETS

```
<html>
 <head>
 <title>Exception handling using try catch blocks</title>
 </head>
 <body>
 <%
 try{
     //I have defined an array of length 5
     int arr[]={1,2,3,4,5};
     //I'm assinging 7th element to int num
     //which doesn't exist
     int num=arr[6];
     out.println("7th element of arr"+num);
 }
 catch (Exception exp){
     out.println("There is something wrong: " + exp);
 }
 %>
 </body>
 </html>
```

OUTPUT


Exception handling using
localhost:8080/BeginnersBook/error
There is something wrong: java.lang.ArrayIndexOutOfBoundsException: 6

JSP ACTION ELEMENTS

Action elements are tags used embedded in JSP

- **Some commonly used tags**
  - `<jsp:include page="……"  />`
  - `<jsp:forward page="……" />`
  - `<jsp:param……/>`
  - `<jsp:useBean……/>`
  - `<jsp:setProperty……/>`
  - `<jsp:getProperty……/>`
  - `<jsp:session …/>`

JSP ACTION ELEMENTS

- `<jsp:include>`
  - Includes static or dynamic resources into current JSP page
  - Processed when a JSP page is executed
  - More flexible than `<%@ include….%>`
    - It can pass parameters to other dynamic contents (JSP, servlet)
    - The JSP or servlets to be included can access the parameters using `getParameter()` method of `request` implicit object

JSP ACTION ELEMENTS

- **Example** `<jsp:include>`

```
<jsp:include page="newJSP.jsp" />
```

(Append parameters to request object)

```
<jsp:include page="newJSP.jsp">
   <jsp:param name="Name" value="ME">
   <jsp:param name="Address" value="Toa Payoh">
</jsp:include>
```

include-action.jsp; inc-header.jsp

<JSP:SESSION>

Index.html

```
<html>
<head>
<title>Welcome Page: Enter your name</title>
</head>
<body>
<form action="session.jsp">
<input type="text" name="inputname">
<input type="submit" value="click here!!"><br/>
</form>
</body>
</html>
```

<JSP:SESSION>

The **session.jsp** page displays the name which user has entered in the index page and it stores the the same variable in the **session object** so that it can be fetched on any page until the session becomes inactive.

Session.jsp

```
<html>
<head>
<title>Passing the input value to a session variable</title>
</head>
<body>
<%
String uname=request.getParameter("inputname");
out.print("Welcome "+ uname);
session.setAttribute("sessname",uname);
%>
<a href="output.jsp">Check Output Page Here </a>
</body>
</html>
```

<JSP:SESSION>

In this page we are fetching the variable's value from session object and displaying it.

output.jsp

```
<html>
<head>
<title>Output page: Fetching the value from session</title>
</head>
<body>
<%
String name=(String)session.getAttribute("sessname");
out.print("Hello User: You have entered the name: "+name);
%>
</body>
</html>
```

<JSP:FORWARD>

- Transfer control to another web component from a JSP page
- Also have ability to pass parameters
  - Similarly, parameters accessible forwarded JSP's request object
- Similar to servlet's request dispatcher forward
  - If any data has been returned to client before forwarding, an exception will be thrown: `IllegalStageException`.

JSP ACTION ELEMENTS

- **Example** `<jsp:forward>`

```
<jsp:forward page="newJSP.jsp" />
```

Append parameters to request object)

```
<jsp:forward page="newJSP.jsp">
   <jsp:param name="Name" value="ME">
   <jsp:param name="Address" value="Toa Payoh">
</jsp:forward>
```

DRAWBACKS

- Embedding Java scriptlets in HTML causes readability problems
  - Difficult to trace through the codes therefore relatively difficult to debug.
  - Web page designers must know Java programming in order to create JSP pages.
- Database connection is not very easy. Many codes are needed.
- No clear separation between logic and presentation.
  - Everything is in one file (.jsp)