Java Server Pages
JSP

Agenda

- JSP Pages
- Script elements
  - Expressions <%= ....%>
  - Declarations <%! ....%>
  - Scriplets <% ....%>

- Directive elements
- Actions (not covered in the scope of this class)

Anatomy of a JSP page

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="wroxtags" tagdir="/WEB-INF/tags" %>
<jsp:useBean id="info" class="com.wrox.begjsp.ch2.Info" scope="request"/>
<html>
        <head>
                <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
                <title>JSP Page</title>
        </head>
        <body>
                <h1>Book Information</h1>
                <hr>
                <p> the name of the book is <wroxtags:bookTitle /> </p>
                <p> it is published by <wroxtags:publisher /> </p>
                <% if (request.getParameter("outside").equals("true")){
                info.setMessage(request.getParameter("texteMessage"));%>
                <p> <%=info.getMessage()%> </p>
                <jsp:include page="copyright.html" />
                <%}%>
        </body>
</html>
```

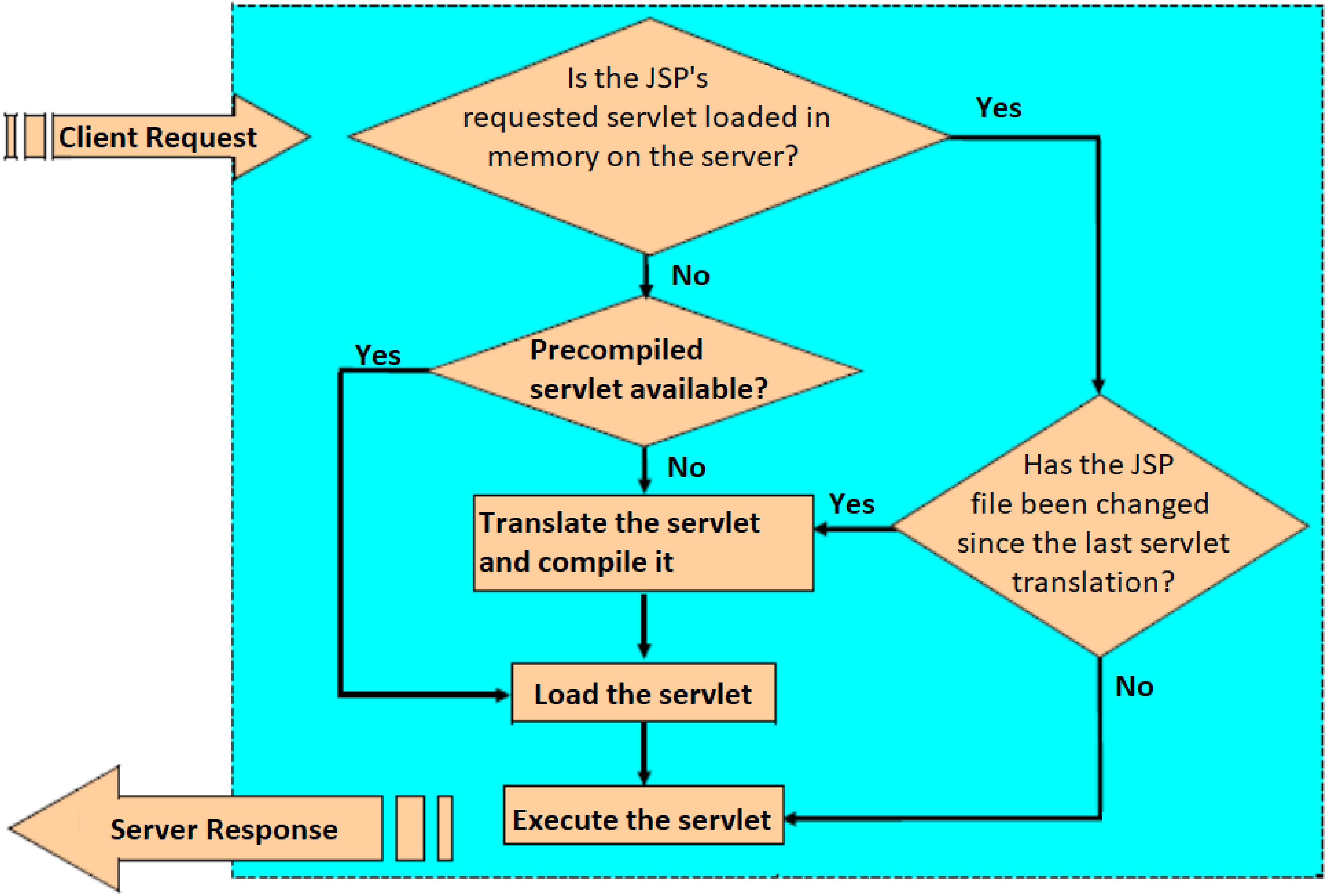
Directives

Action

Static
contain
er

Script
elements

JSP's Execution Algorithm


Client Request
Is the JSP's requested servlet loaded in memory on the server?
Yes
No
Precompiled servlet available?
Yes
No
Translate the servlet and compile it
Has the JSP file been changed since the last servlet translation?
Yes
No
Load the servlet
Execute the servlet
Server Response

Script Elements: Expressions

- Format
  - <%= Java Expression %>
- Result
  - The expression is evaluated, its result is converted to a String and place in the HTML page where the JSP page specifies.
  - No semicolon after expression.
  - In other words, the expression is placed in out.print inside jspService
- Examples
  - Current time: <% = new java.util.Date ()%>
  - Your host name: <% = request.getRemoteHost ()%>

Expressions: JSP / Servlet Correspondence

- JSP Original

```
<H1>A Random Number</H1>
<%= Math.random() %>
```

Possible result code in the servlet

```
public void _jspService (HttpServletRequest request,
                         HttpServletResponse response)
      throws ServletException, IOException {
  response.setContentType ("text / html");
  HttpSession session = request.getSession (true);
  JspWriter out = response.getWriter ();
  out.println ("<H1> A Random Number </ H1>");
  out.println (Math.random ());
  ...
}
```

Using predefined variables that match their servlet equivalent

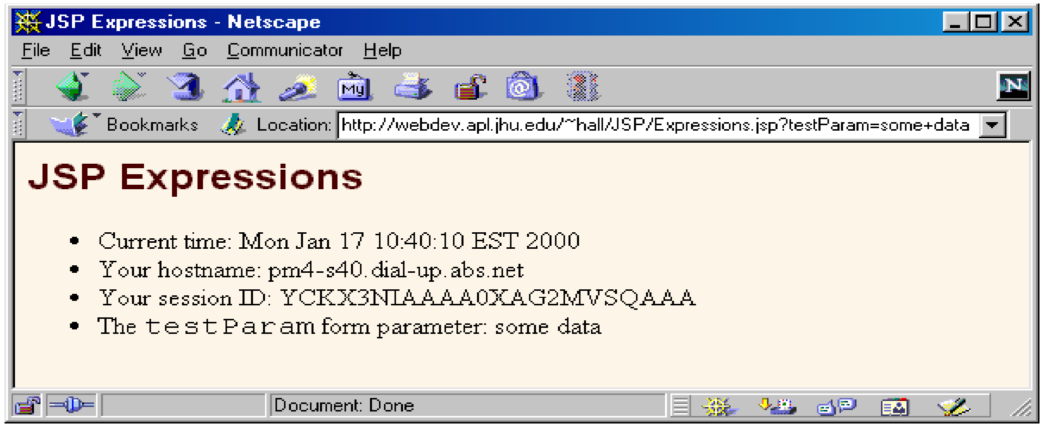
Variables	**Signification**
request	the object of type HttpServletRequest
response	the object of type HttpServletResponse
out	a PrintWriter to write in the body of the answer
session	the object of type HttpSession is obtained as follows: response.getSession(true);
application	the ServletContext object is obtained as follows: getServletConfig().getServletContext();
config	the ServletConfig object obtained using: getServletConfig ()
pageContext	the object defining a set of features specific to jsp (detailed later)
page	synonymous with this, but must be *cast* before being used

Using JSP Expressions

```
<BODY>
<H2> JSP Expressions </ H2>
<UL>
    <LI> Current time: <% = new java.util.Date ()%>
    <LI> Your hostname: <% = request.getRemoteHost ()%>
    <LI> Your session ID: <% = session.getId ()%>
    <LI> The <CODE> testParam </ CODE> form parameter:
        <% = request.getParameter ("testParam")%>
</ UL>
</ BODY>
```

Notice that we have access to objects :

- request
- Session
- out


JSP Expressions - Netscape
File Edit View Go Communicator Help
Bookmarks
Location:
Document: Done

http://webdev.apl.jhu.edu/~hall/JSP/Expressions.jsp?testParam=some+data

JSP Expressions

- Current time: Mon Jan 17 10:40:10 EST 2000
- Your hostname: pm4-s40.dial-up.abs.net
- Your session ID: YCKX3NIAAAA0XAG2MVSQAAA
- The `testParam` form parameter: some data

Script Elements: Scriptlets

- Format
  - <%   Java Code   %>
- Result
  - The code is inserted where the servlet's _jspService method
  - specifies. Declared variables are local to the service's method.
  - Method declaration is impossible here because as the code is inserted directly into the _jspService method of the servlet, this would declaring a method in another which is forbidden in Java.
- Example

  -
    ```
    <%
    String queryData = request.getQueryString();
    out.println("Attached GET data: " + queryData);
    %>
    ```
  - `<% response.setContentType("text/plain"); %>`

JSP Scriptlet Examples

```
<HTML>
....
<%
String bgColor = request.getParameter("bgColor");
boolean hasExplicitColor;
if (bgColor != null) {
  hasExplicitColor = true; }
  else {
    hasExplicitColor = false;
    bgColor = "WHITE"; }
  %>
  <BODY BGCOLOR="<%= bgColor %>">
  <H2 ALIGN="CENTER">Color Testing</H2>
  <%
  if (hasExplicitColor)
  {
  %>
  <h1> You specified the color </h1> <% =bgColor %>
  <%} else { %>
  <h1> You did not specify a color </h1>
  <%
  }
  %>
```

Script Elements: declarations

- Format
  - <%! Java Code %>
- Result
  - The code is embedded in the servlet class definition (outside of any methods). They are generated automatically by the servlet.

  A declaration:
    - of a variable results in an instance variable in the servlet.
    - of a method results in a method in the servlet.
- Examples
  - <%! private int someField = 5; %>
  - <%! private void someMethod(...) {...} %>
- Conceptual point of view:
  - Method fields can be useful. It is usually best to define the method in a separate class, rather than inline.
  - Do not put java instructions other than declarations of variables and methods here.

JSP/Servlet Correspondence

🗣 JSP original

```
<H1>Some Heading</H1>
<%!
   private String
    randomHeading() {
      return("<H2>" +
    Math.random() + "</H2>");
     }
  %>
  <%= randomHeading() %>
```

- **(Alternative: make randomHeading a static method in a separate java class)**

🗣 Code en Servlet

```
public class xxxx implements HttpJspPage {
  private String randomHeading() {
   return("<H2>" + Math.random() +
     "</H2>");
 }

public void _jspService(HttpServletRequest
     request,
HttpServletResponse response)
      throws ServletException, IOException {
     response.setContentType("text/html");
    HttpSession session =
     request.getSession(true);
    JspWriter out = response.getWriter();
     out.println("<H1>Some
    Heading</H1>");
    out.println(randomHeading());
    ...
 } ...
}
```

Example using JSP declarations

```
<!DOCTYPE HTML PUBLIC
            "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD><TITLE>JSP Declarations</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>

<BODY>
<H1>JSP Declarations</H1>

<%! private int accessCount = 0; %>
<H2>Accesses to page since ?????…….:
<%= ++accessCount %></H2>

</BODY>
</HTML>
```

Design Strategy: Limit Java Code in JSP Pages

Two options :
- Put 25 lines of Java code directly in the JSP page
- Put these 25 lines in a separate Java class and put a line in the JSP page to call the java code

Why is the second option better?

1. Debugging:

If you have any syntax errors, you will see them right away at compile time. In case of bug the compiler will show the error in the servlet created by the server from the .jsp file (and not in your .jsp file).

2. Testing is simpler.

3. The code is reusable and can be called on any number of pages.

Directive Elements

- Used to control certain characteristics of the page
- They do not generate a display
- We use the following syntax:
  - <% @ .... directive ....%>
- There are 3 types of directives allowed in JSP :
  - The directive page
  - The taglib directive
  - The include directive
- We will see these directives in more detail later in the course.

```
<%@ page import="Java.util.*, Java.math.*" %>
<%@ page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib prefix="wroxtags" tagdir="/WEB-INF/tags" %>
<%@ include file="url relatif" %>
```

Page Directive

import attribute describes the packages whose classes are used in the jsp page. <% @ page import = "Java.util. *, Java.math." %>

contentType attribute <%@ page contentType = "text / html"%>
isTreadSafe attribute <%@ page isThreadSafe = "false"%>
errorPage attribute indicates which page should be called in case the current page generates an exception that no catch intercepts.
<%@ page errorPage = "relative url"%>
isErrorPage attribute indicates whether the current page can be a page called in response to a raised but uncaught exception in a page.
<%@ page isErrorPage = "true"%>
other attributes

JSP also offers the following page-type directives: session, buffer, autoflush, extends, info and language.

The Include Directive

- The directive <% @ include file = "relative url"%> in a document will be completely replaced by the contents of the file whose relative URL is specified in the directive.
- The file will be included when the server generates the servlet from the jsp page. If the file to be included is changed later, the servlet will remain unchanged.
- If you want the file to be included every time the user accesses the jsp page, use the following directive:

```
<jsp:include page="url relatif" flush="true" />
```