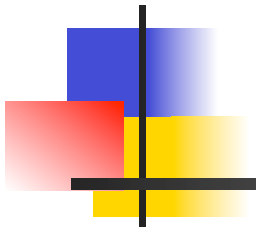


Java Server Pages

Java Server Pages (JSP)

- JavaServer Pages (JSP) is a J2EE component that enables to develop Web applications that work as if they had been created with Java servlets.

- The JSP specification defines JSP as "a technology for building the applications for generating dynamic Web content such as HTML, DHTML, SHTML, and XML

Why JSP ?

- Example Servlet: in the doGet() method

```
out.println("<HTML><BODY>");
out.println("Hello World ");
out.println("</BODY></HTML>");
```

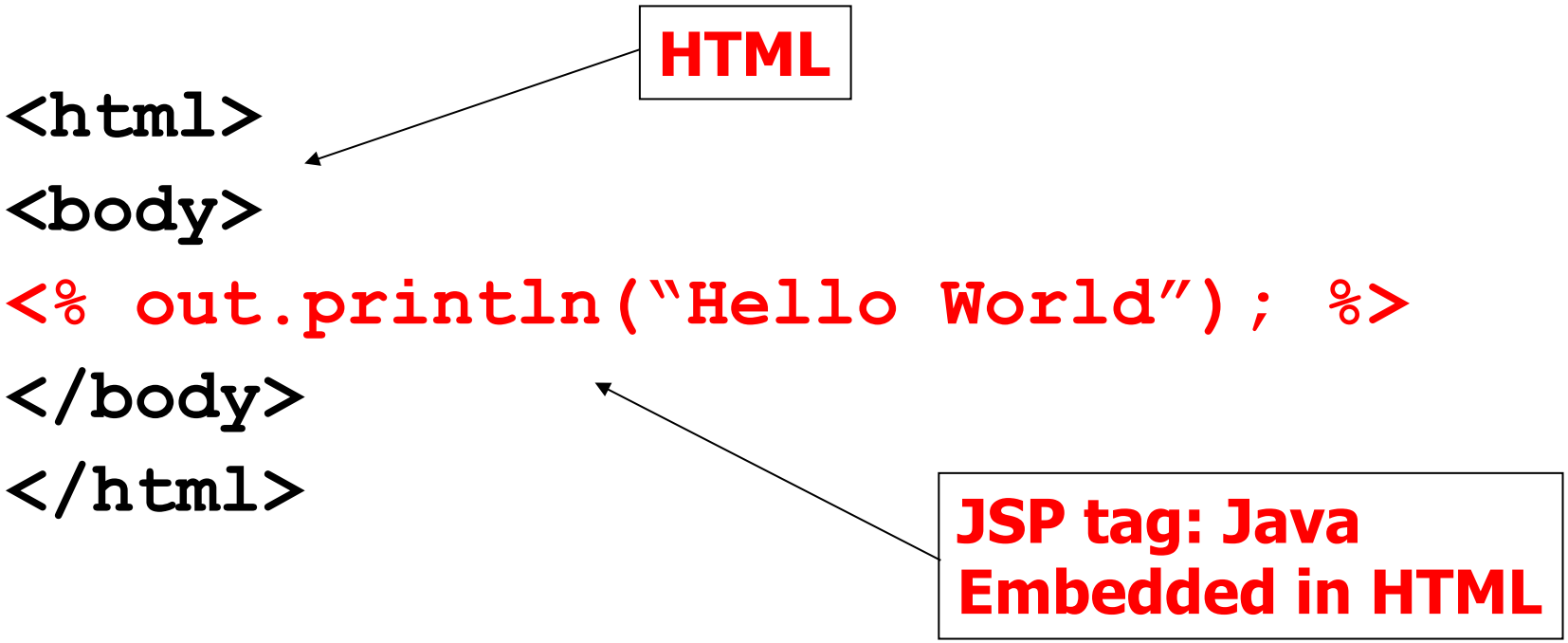
HTML embedded in Java

- What if you need to change the layout of this page —say you want to add a header and a footer?
- Because we have already learned about Java servlets, we can easily add several out.println()statements, and then recompile the servlet and deploy it again.
- A real Web application can generate dozens of Web pages and each of them may need to be modified.

The JSP approach

- JSP enables to separate GUI design from business logic
  - a Web designer can take care of the HTML part
  - while the Java developer concentrates on the programming of the business functions required by the application.

Simple.jsp



- A JavaServer Page is nothing but a servlet that is automatically generated by a JSP container from a file containing valid HTML and JSP tags.

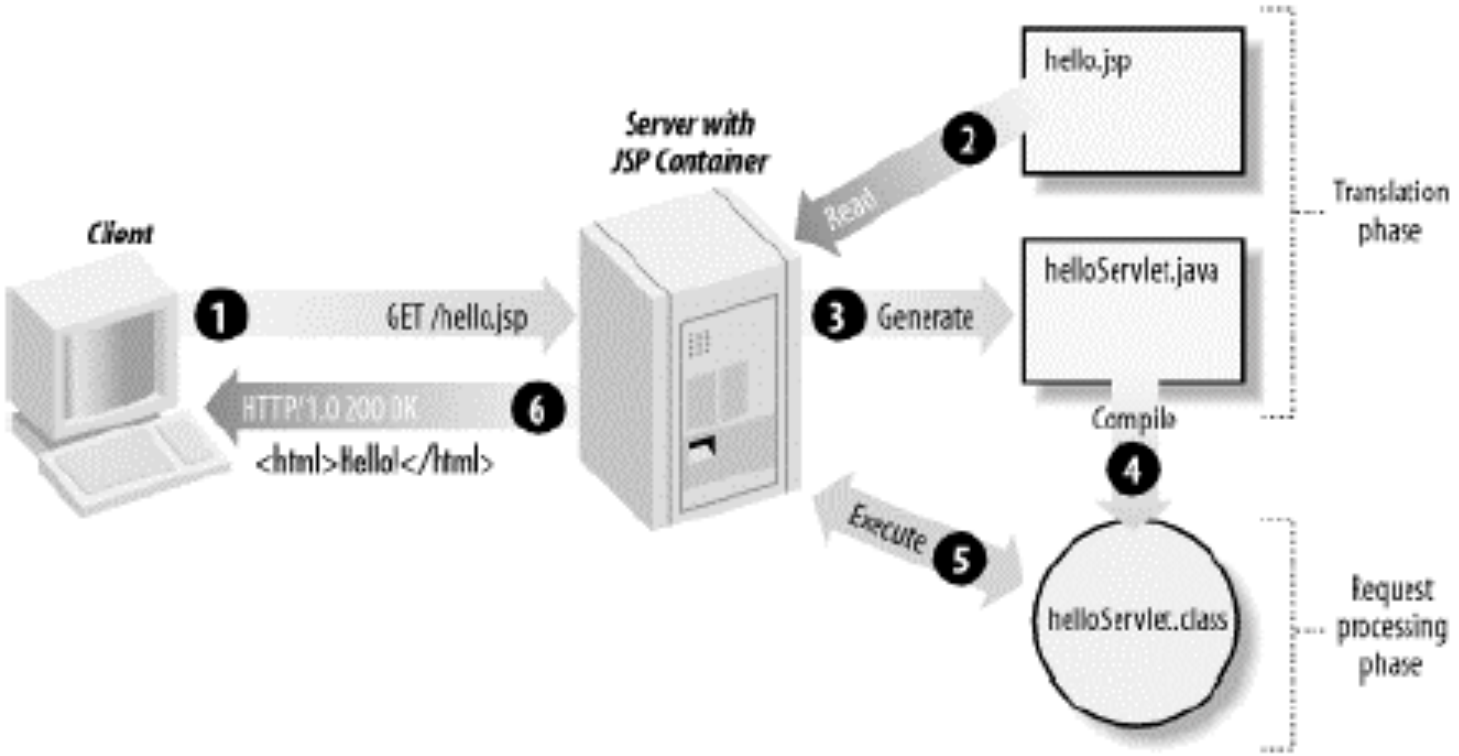
A more interesting example ☺

```
<HTML>
<BODY>
You may not know that 2 + 2 is <%= 2 + 2%>
<p>The code within the JSP tag was
created by a Java developer,
while the rest of the page has been done
by the HTML person.
</BODY>
</HTML>
```

Dynamic content with JSP elements

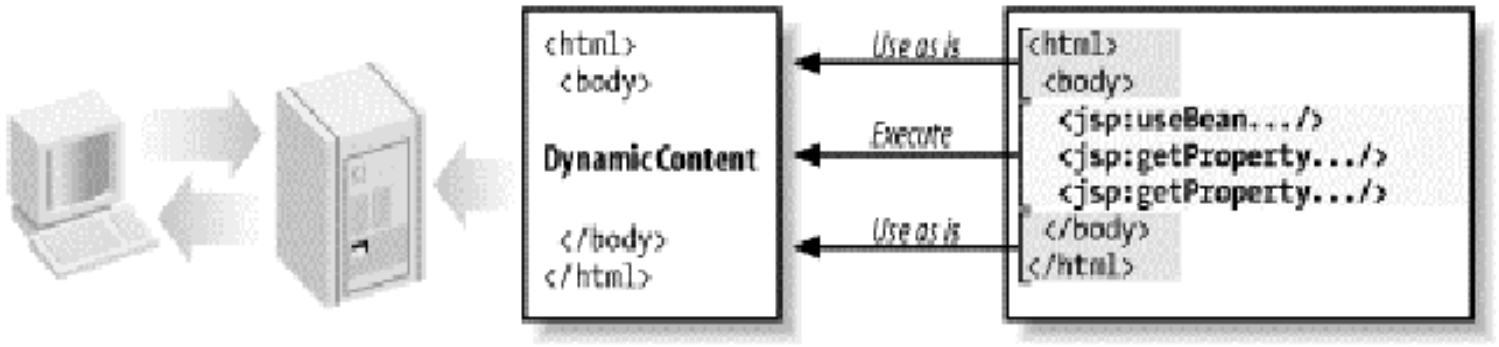
- A JSP page contains standard markup language elements, such as HTML tags, just like a regular web page.
- However, a JSP page also contains special JSP elements (**tags**) that allow the server to insert dynamic content in the page.
- When a user asks for a JSP page, the server
  - executes the JSP elements,
  - merges the results with the static parts of the page in a servlet, and
  - sends the dynamically composed page back to the browser

JSP lifecycle



Dynamic content with JSP elements

- the JSP schema



Other Solutions: ASP

Active Server Pages (ASP)

- Microsoft's Active Server Pages (ASP) is a popular technology for developing dynamic web sites. Just like JSP, ASP lets a page author include logic, such as VBScript and JScript code, in regular web pages to generate the dynamic parts.
  - For complex code, COM (ActiveX) components written in a programming language such as C++ can be invoked by the scripting code.
  - Due to its reliance on native COM code as its component model, it's primarily a solution for the Windows platform. Limited support for other platforms, such as the Apache web server on Unix, is available through third-party products such as Sun Chili!Soft ASP (Sun Microsystems, Inc.) and InstantASP (Halcyon Software).

- **ASP.NET**, the latest version of web application framework from Microsoft, adds a number of new features. As an alternative to scripting, dynamic content can be generated by HTML/XML-like elements similar to JSP action elements.
  - ASP.NET is a part of the complete .NET platform, with the potential for better support on non-Windows platforms.

- You can read more about ASP and ASP.NET on Microsoft's web site, http://www.microsoft.com.

Other Solutions: PHP

PHP

- PHP 1 is an open source web scripting language. Like JSP and ASP, PHP allows a page author to include scripting code in regular web pages to generate dynamic content.
- PHP has a C-like syntax with some features borrowed from Perl, C++, and Java. Complex code can be encapsulated in both functions and classes.
- PHP 4, the current version, compiles a page when it's requested, executes it and merges the result of executing the scripts with the static text in the page, before it's returned to the browser.
- PHP is supported on a wide range of platforms, including all major web servers, on operating systems like Windows, Mac, and most Unix flavors, and with interfaces to a large number of database engines.
- More information about PHP is available at http://www.php.net.

JSP nuts and bolts

- Syntactic elements:

<%-- comments --%>
<%! declarations %>
<%@ directives %>
<%= expressions %>
<% scriptlets %>

<jsp: actions />

- Implicit Objects:
  - request
  - response
  - pageContext
  - session
  - application
  - out
  - config
  - page

Comments

- JSP comments and are not included in the output Web page:

- Syntax:

<%-- Some comments --%>

- If you need to include comments in the source of the output page, use standard HTML comments, which start with <!-- and end with -->.

Declarations

- A declaration is a block of Java code used to:

  define class-wide variables (global variables) and methods in the generated servlet

- They are initialized when the JSP page is initialized and remain available in the current page.

Declarations

- Syntax

```
<%! DECLARATION %>
```

- Examples:

```
<%! String nome="pippo"; %>
<%! public String getName()
    {return nome;} %>
```

Directives

- A directive is used as a message mechanism to:

  pass information from the JSP code to the Web container **at compilation time**
- Directives do not generate screen output
- Main directives:
  - page
  - include (for including other STATIC resources at compilation time)
  - taglib (for including custom tag libraries)

Directives: page

- Syntax:

```
<%@ DIRECTIVE {attributo=valore} %>
```

- main attributes:

```
<%@ page language=java session=true %>
<%@ page import=java.awt.*,java.util.* %>
<%@ page isThreadSafe=false %>
<%@ page errorPage=URL %>
<%@ page isErrorPage=true %>
```

Directives: include

header.jsp

```
<html>
     <head><title><%= title %></title></head>
<body>
```

footer.jsp

```
</body>
</html>
```

```
<%! String title = "Addition"; %>
<%@ include file="header.jsp" %>
     The sum of <%= request.getParameter("x") %>
     and <%= request.getParameter("y") %> is
     <%= Integer.parseInt(request.getParameter("x")) +
     Integer.parseInt(request.getParameter("y")) %>
<%@ include file="footer.jsp" %>
```

N.B. the include directive adds the content of the included page at the time of *compilation*

Other Directives

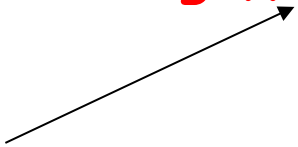
- contentType="*type*"
- info="*description*"
- errorPage="*path*"
- isErrorPage="*boolean*"
- import="*package*"

Expressions

- An expression is a shorthand notation that sends the evaluated Java expression back to the client (in the form of a String).

- Examples:

```
<%@ page import=java.util.* %>
Sono le <%= new Date().toString() %>
```



Please note that there is no semicolon (;) at the end of the expression !!

Expressions: example

```
<html><body>
<%! String nome="pippo" %>
<%! public String getName()
    {return nome;} %>

<H1>
Buongiorno
<%= getName() %>
</H1>
</body></html>
```

Scriptlet

- A scriptlet is a block of Java code executed during the request-processing time.
  - In Tomcat all the scriptlets gets put into the service() method of the servlet.
  - They are therefore processed for every request that the servlet receives.

Scriptlet

- Examples:

Servlet's tyle

(1)
```
<% z=z+1; %>
```

(2)
```
<% // Get the Employee's Name from the request
   out.println("<b>Employee: </b>" +
   request.getParameter("employee"));
   // Get the Employee's Title from the request
   out.println("<br><b>Title: </b>" +
   request.getParameter("title"));
   %>
```

Note on Scriptlet

- Even though the JSP syntax enables insertion of Java code fragments, variables, and method declarations, you should try to minimize the amount of Java code in the JSP body
- Remember that the whole point of using JSP is to separate business processing from the presentation.
- That's why the best practice is either to move Java code into JavaBeans or to use custom tag libraries (JSTL: JSP Standard Tag Library).

Actions

- JSP actions provide runtime instructions to the JSP containers.
- For example, a JSP action can include a file, forward a request to another page, or create an instance of a JavaBean.
- Examples:

```
<jsp:include page "header.jsp" />
<jsp:forward page="someOther.jsp" />
<jsp:useBean id="User"
  class="com.connexiaair.AirUser" />
```

We will present JavaBeans separately
In the following slides we will discuss the rest of the standard JSP action tags in this section.

Action→ <jsp: include

- Include a file in the current JSP page

```
<jsp:include page "header.jsp" />
```

- Note the difference:
  - the include directive adds the content of the included page at the time of *compilation*
  - the jsp:include action does it at *runtime*.
    - this adds flexibility to JSP, because the decision about what page to include can be made based on the user's actions or other events that may take place when the Web application is already running.

Action→ <jsp: forward

- the jsp:forward action enables to redirect the program flow to a different HTML file, JSP, or servlet while maintaining the same request and response objects.

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

- This directive works in the same way as the forward() method of the RequestDispatcher class within the Servlet model

Example: Forward (1)

```
//File: index.jsp
<html>


<%
   double freeMem = Runtime.getRuntime().freeMemory();
   double totlMem = Runtime.getRuntime().totalMemory();
   double percent = freeMem/totlMem;
   if (percent < 0.5) {
%>

<jsp:forward page="one.jsp"/>

<% } else { %>

<jsp:forward page="two.html"/>

<% } %>
```

Example: Forward (2)

```
//File: one.jsp
<html>

<body bgcolor="white">
<font color="red">

VM Memory usage < 50%.
</html>
```

```
//File: two.html

<html>

<body bgcolor="white">
<font color="red">

VM Memory usage > 50%.
</html>
```

JSP nuts and bolts

- Syntactic elements:

<%-- comments --%>
<%! declarations %>
<%@ directives %>
<%= expressions %>
<% scriptlets %>

<jsp: actions/>

- Implicit Objects:
  - request
  - response
  - pageContext
  - session
  - application
  - out
  - config
  - page

Implicit JSP objects

- Within the Servlet framework we have learned how to use such objects as **`HttpServletRequest`**, **`HttpServletResponse`**, **`HttpSession`**, and **`ServletContext`**
- Since JSPs live by the same rules as servlets, they also need to be able to get access to these objects.
  - JSPs provide a number of predefined variables that give you access to these vital objects.

request & response

request

- This variable points at HttpServletRequest . The following example gets the flight destination entered by the user on the HTML page:

```
<%String dest=
    request.getParameter("destination ");%>
```

response

Use this variable to access the HttpServletResponse object. For example:

```
<%response.setContentType("text/html ");%>
```

out , session, exception

out

This variable represents the JspWriter class, which has the same functionality as the PrintWriter class in servlets. Here's an example:

```
<%out.println("Enter flight destination ");%>
```

session

This variable represents the instance of the HTTPSession object.

exception

This variable represents an instance of the uncaught Throwable object and contains error information.

This variable is only available from the JSP error page that contains the directive i**sErrorPage=true**

page & pageContext

page

This variable represents the instance of the JSP's Java class processing the current request.

pageContext

This variable represents the javax.servlet.jsp.PageContext class, which contains methods for dealing with scoped data.

application & config

application

This variable gives you access to the ServletContext object without your having to call getServletConfig().getContext().

config

This variable provides access to the ServletConfig object.

A Tiny Example

```
<% response.addDateHeader("Expires", 0); %>
<%! int hits = 0; %>
….
<html>
  <head><title>JSP</title></head>
  <body>
    <h1>Hello World!</h1>

    You are visitor number
    <% synchronized(this) { out.println(++hits); } %>
    since the last time the service was restarted.
    <p>
    This page was last updated:
    <%= new java.util.Date().toLocaleString() %>
  </body>
</html>
```

Be Careful About Declarations

```
<% response.addDateHeader("Expires", 0); %>
<html>
  <head><title>JSP</title></head>
  <body>
    <h1>Hello World!</h1>
    <% int hits = 0; %>
    You are visitor number
    <% synchronized(this) { out.println(++hits); } %>
    since the last time the service was restarted.
    <p>
    This page was last updated:
    <%= new java.util.Date().toLocaleString() %>
  </body>
</html>
```

This page counter is always 1...

Esercizio

- Costruire una JSP che legga due numeri da un form e fornisca in output come risultato una pagina html che mostri
  - la loro somma
  - la loro differenza
  - il loro prodotto
  - il loro quoziente

Gestire gli errori direttamente dal codice

Using Error Pages

```
<%@ page errorPage="error.jsp" %>
<html>
  <head><title>Division</title></head>
  <body>
    <% int n = Integer.parseInt(request.getParameter("n")); %>
    <% int m = Integer.parseInt(request.getParameter("m")); %>
    <%= n %>/<%= m %> equals <%= n/m %>
  </body>
</html>
```

Error.jsp ⟶

```
<%@ page isErrorPage="true" %>
<html>
  <head><title>Error</title></head>
  <body>
     Something bad happened:
     <%= exception.getMessage() %>
  </body>
</html>
```