Java Server Pages (JSP)

Static Web Pages

- Web pages with content that cannot change without a developer editing its source code
- Display the exact same information whenever anyone visits it. Static Web pages do not have to be simple plain text. They can feature detailed multimedia design and even videos.

Dynamic Web Pages

- Web pages which display different content from the same source code
- Capable of producing different content for different visitors from the same source code file.
- Determining parameters may be based on what operating system or browser the visitor is using, whether she is using a PC or a mobile device, or even the source that referred the visitor

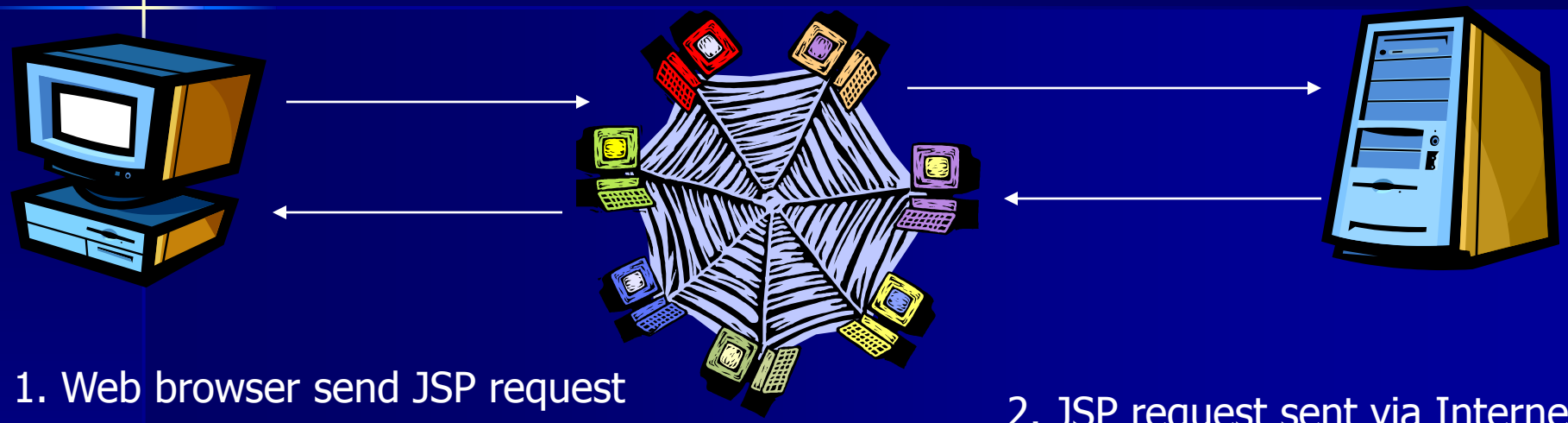
Apache Tomcat

- It is an open-source Java Servlet Container that implements several Java EE specifications including Java Servlet, Java Server Pages (JSP), Java EL, and WebSocket, and provides a HTTP web server environment in which Java code can run.

JSP - Introduction

JSP technology is used to create web application. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, jstl etc.

How JSP works?



1. Web browser send JSP request

2. JSP request sent via Internet to the web server
3. The web server sends the JSP file (template pages) to JSP servlet engine
4. Parse JSP file
5. Generate servlet source code
6. Compile servlet to class
7. Instantiate servlet

8. HTML/JSP sent back to the browser

Template Pages

Server Page Template

```
<html>
<title>
A simple example
</title>

<body color="#FFFFFF">
The time now is
<%= new java.util.Date() %>
</body>
</html>
```

Resulting HTML

```
<html>
<title>
A simple example
</title>

<body color="#FFFFFF">
The time now is
Tue Nov 5 16:15:11 PST 2002
</body>
</html>
```

Predefined Variables (Implicit Objects)

- They are created automatically when a web server processes a JSP page.
- request: The HttpServletRequest
- response: The HttpServletResponse
- session
  - The HttpSession associated with the request (unless disabled with the session attribute of the page directive)
- out
  - The stream (of type JspWriter) used to send output to the client
- application
  - The ServletContext (for sharing data) as obtained via getServletConfig().getContext().
- page, pageContext, config, exception

JSP Components

- There are three main types of JSP constructs that you embed in a page.
  - Scripting elements
    - You can specify Java code
    - Expressions, Scriptlets, Declarations
  - Directives
    - Let you control the overall structure of the servlet
    - Page, include, Tag library
  - Actions
    - Transfer control between pages

JSP Expressions

- Format
  - <%= Java Expression %>
- Result
  - Expression evaluated, converted to String, and placed into HTML page at the place it occurred in JSP page
  - That is, expression placed in _jspService inside out.println
- Examples
  - Current time: <%= new java.util.Date() %>
  - Mathematical Expression: <%= 10*4 %>

JSP/Servlet Correspondence

- Original JSP

```
<%= Math.random() %>
```

- Possible resulting servlet code

```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response)
  throws ServletException, IOException {
 request.setContentType("text/html");
 HttpSession session = request.getSession(true);
 JspWriter out = response.getWriter();

  out.println(Math.random());
 ...
 }
```

JSP/Servlet Correspondence

- Original JSP

```
<%= foo() %>
```

- Possible resulting servlet code

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

JSP Scriptlets

- Format: <% Java Code %>
- Result
  - Code is inserted verbatim into servlet's _jspService
- Example
  - <%
    String queryData = request.getQueryString();
    out.println("Attached GET data: " + queryData);
    %>

JSP/Servlet Correspondence

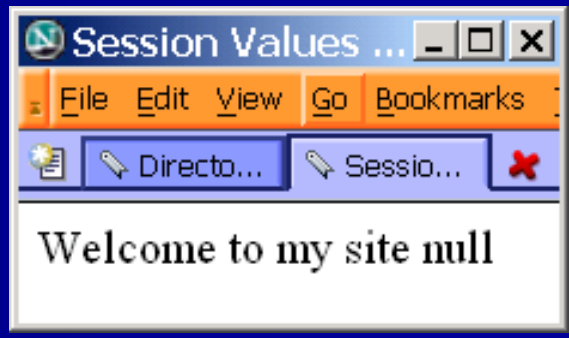
- Possible resulting servlet code

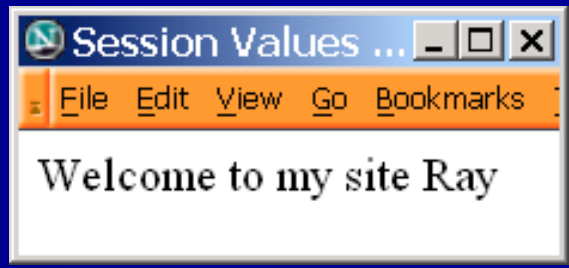
```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response)
    throws ServletException, IOException {
  request.setContentType("text/html");
  HttpSession session = request.getSession(true);
  JspWriter out = response.getWriter();

  String queryData = request.getQueryString();
  out.println("Attached GET data: " + queryData);
  ...
}
```

Working with Session object

- The session object has many useful methods that can alter or obtain information about the current session.
  - setMaxInactiveInterval(second)

```
<html><head>
<title>Session Values</title>
</head><body>
<%
session.setMaxInactiveInterval(10);
String name = (String)
    session.getAttribute("username");
out.print("Welcome to my site " + name + "<br>");
%>
</body></html>
```





JSP Declarations

- Format
  - <%! Java Code %>
- Result
  - Code is inserted verbatim into servlet's class definition, outside of any existing methods
- Examples
  - <%! private int someField = 5; %>
  - <%! private void someMethod(...) {...} %>

Declarations vs. Scriptlets

```
<%! int count=100; %>
<%= ++count %>

public final class
   _scopeExpermnt1_xjsp
{
  int count=100;

  public void _jspService
    (HttpServletRequest request,
     HttpServletResponse response)
     throws java.io.IOException
  {
    JspWriter out =
   pageContext.getOut();



   out.print( String.valueOf(  ++co
   unt  ) );

  }
}
```

```
<% int count=100; %>
<%= ++count %>

public final class
   _scopeExpermnt2_xjsp
{
  public void _jspService
    (HttpServletRequest request,
     HttpServletResponse response)
     throws java.io.IOException
  {
     JspWriter out =
   pageContext.getOut();

     int count=100;

out.print( String.valueOf(  ++count
     ) );


  }
}
```

JSP Directives

- JSP directives provide directions and instructions to the container, telling it how to handle certain aspects of JSP processing.
- Affect the overall structure of the servlet
- Two possible forms for directives
  - <%@ directive attribute="value" %>
  - <%@ directive attribute1="value1"
    attribute2="value2"…
    attributeN="valueN" %>
- There are three types of directives
  - page, include, and taglib
  - We will only go through page

The import Attribute

- Format
  - <%@ page import="package.class" %>
  - <%@ page import="package.class1,...,package.classN" %>
- Purpose
  - Generate import statements at top of servlet

Example of import Attribute

```
<BODY><H2>The import Attribute</H2>
<%-- JSP page directive --%>
<%@ page import="java.util.*" %>

<%-- JSP Declaration --%>
<%!
private String randomID() {
  int num = (int)(Math.random()*10000000.0);
  return("id" + num);
}
private final String NO_VALUE = "<I>No Value</I>";
%>
</BODY></HTML>
```

Example of buffer Attribute

```
<html>
    <body>
        <%@ page buffer="16kb" %>
        Today is:
        <%= new java.util.Date() %>
     </body>
</html>
```

Including Pages

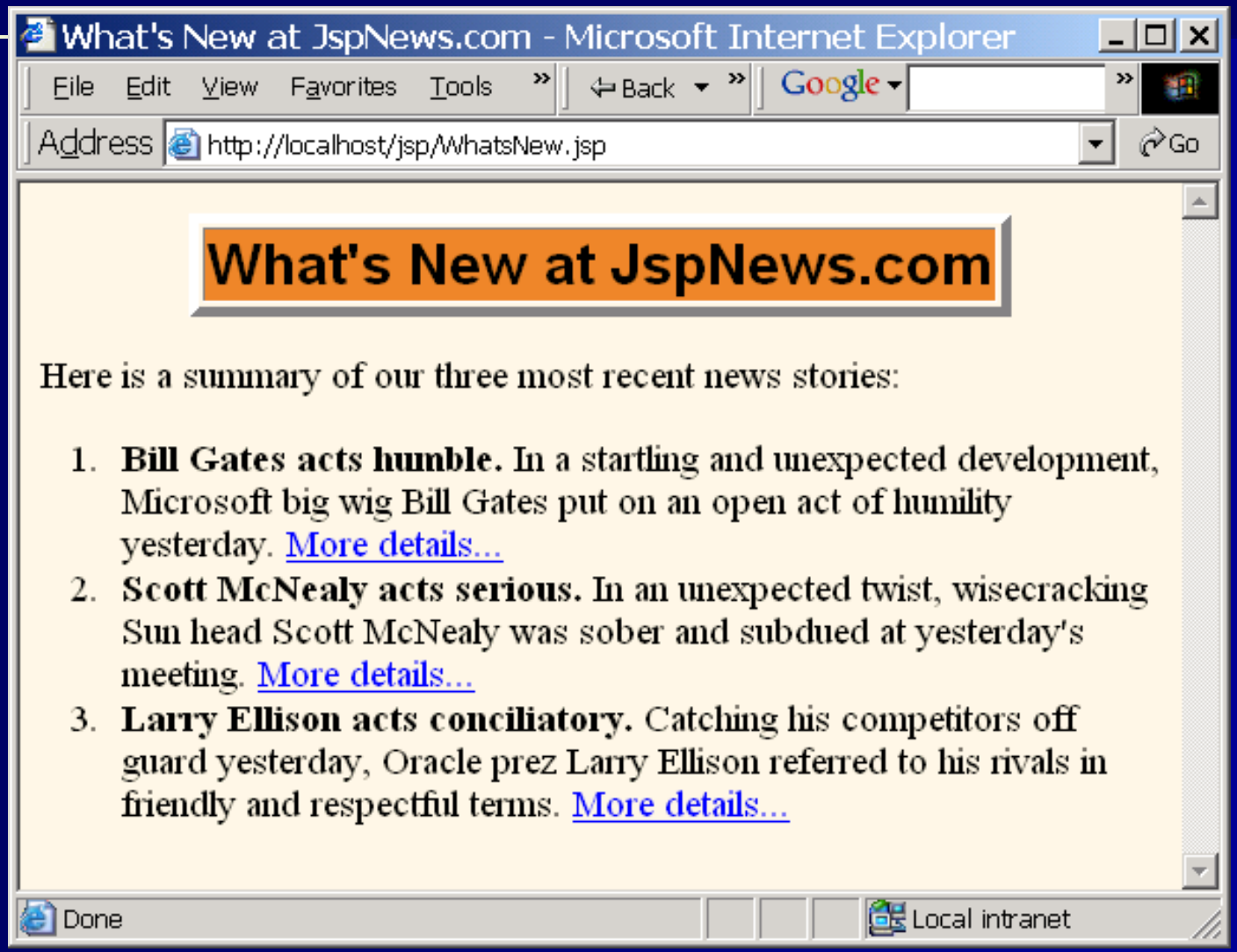
- Format
- <%@ include file="relative url" >

- Purpose
  - To reuse JSP, HTML, or plain text content
  - JSP content cannot affect main page: only output of included JSP page is used
  - To permit updates to the included content without changing the main JSP page(s)

Including Pages: Example Code

```
...
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
      What's New at JspNews.com</TABLE>
<P>
Here is a summary of our three most recent news stories:
<OL>
  <LI><%@ include file=" news/Item1.html"  >
  <LI <%@ include file=" news/Item2.html" >
  <LI <%@ include file=" news/Item3.html" >
</OL>
</BODY></HTML>
```

Including Pages: Result



JSP with JDBC