Java Web

Li Jing
lijingjing@bupt.edu.cn

Tips

- Tools
  - JDK(Latest Release, old edition is ok)
  - MyEclipse (6.5 edition and above，IDE，integrated development environment)
  - MySQL DataBase (5.0 edition and above)
  - MySQL-Front
  - Optional
    - Tomcat (5.o edition and above)

Outline

- Java Web Basics
- SERVLET
  - B/S & C/S
- JSP
  - Java Web Application Server
    - Install and Configure Tomcat

B/S & C/S

- B/S
  - B means Browser and S means Server
  - People can use B/S web application without any additional software installed except an IE browser.
- C/S
  - C means Client and S means Server
  - People must install the application's special client software.
  - It usually need upgrade and patching from time to time.
- They are architectures of web application.
- Each one has its own suitable situation.

Java Web Application Server

- Some famous web application servers:
  - Tomcat (Apache, open source)
  - Jboss (open source)
  - WebLogic (BEA)
  - WebSphere (IBM)
- We choose Tomcat because it is enough to serve normal web applications.

Install and Configue Tomcat

- Download Tomcat from :
  - http://www.apache.org
- During the installation, you must pay more attention to :
  - Tomcat server's port（服务器端口）. The default port is 8080.
  - The install path of JDK. There must be a full installation of JDK.
- Tomcat also can be integrated in MyEclipse IDE.

Install and Configue Tomcat - cont

- After installation, you can start it .
- How to start Tomcat?
  - Find ‘Apache Tomcat x.x’ in the start menu
  - Open ‘monitor tomcat’
  - You can see a button  in the taskbar
  - Right click it, and select ‘start service’
  - If you see  , it means tomcat works.
  - Run an ie browser, and open http://localhost:8080, you will see a cute cat. 

Install and Configue Tomcat - cont

- In the install directory of Tomcat, you will see

Directory for command files

Directory for configuration files

Directory for all deployed web applications

Directory for web server’s lib files

Directory for log files

Directory for temp work web applications

Outline

- Java Web Basics
- SERVLET
  - Source of Servlet
  - What is a servlet
  - Servlet’s advantages
  - Servlet Basic
  - How servlet handles HTTP Request
  - Example
- JSP

Source of servlet

- HTTP : HyperText Transfer Protocol (超文本传输协议)
- In the beginning, HTTP was employed to transfer static contents.
  - a plain text file or
    - http://www.bupt.edu.cn/jwxt/file.txt
  - an image file or
    - http://www.bupt.edu.cn/jwxt/image.jpg
  - an html web page
    - http://www.bupt.edu.cn/jwxt/newpage.html

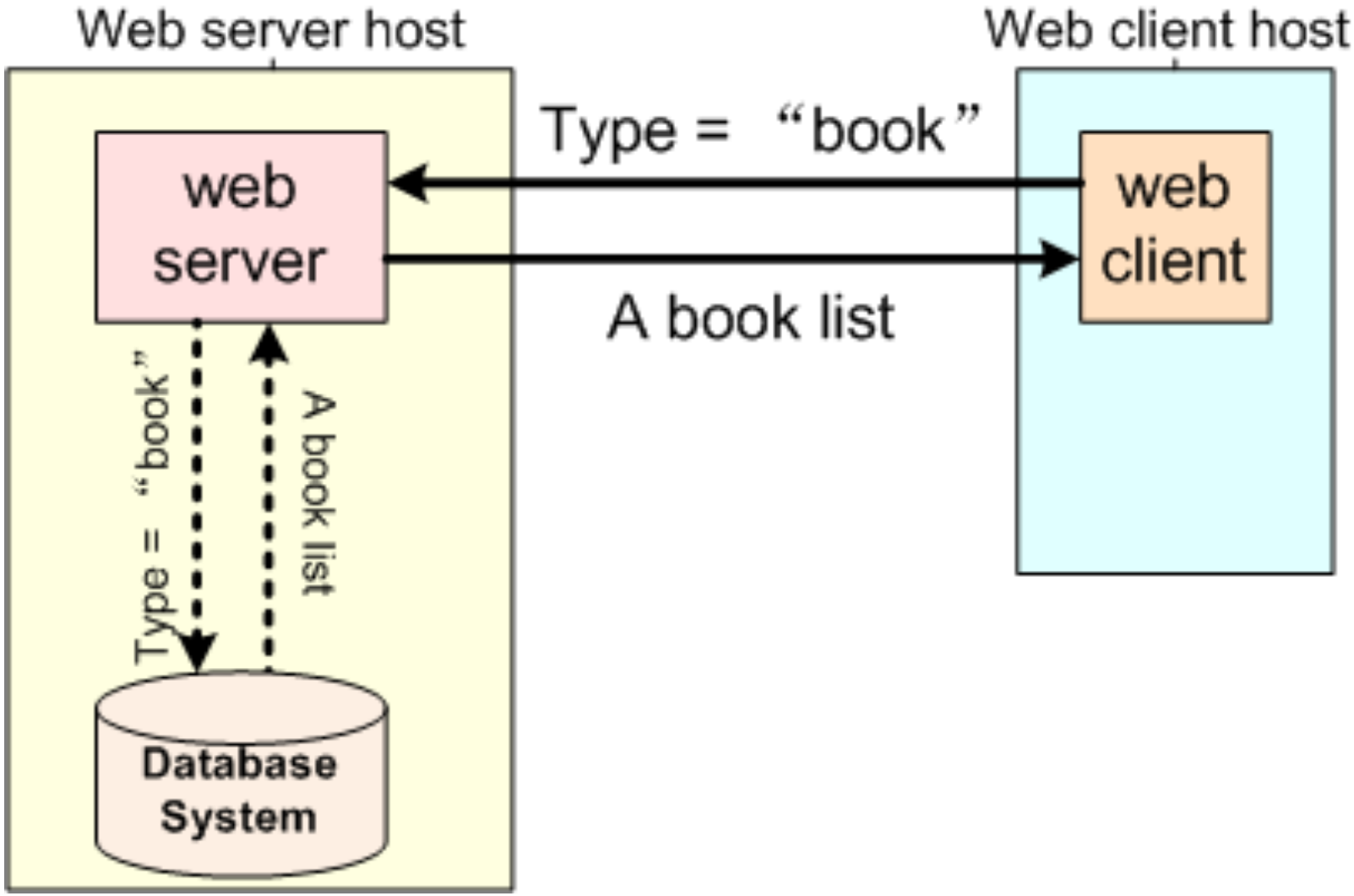
Source of servlet - cont

- As the web evolved, web applications began to allow a browser user to retrieve data based on dynamic information entered during an http session.

Source of servlet - cont

- E.g. an online shop
- requires fetching remote data based on data entered by a client at runtime.
  - allows a user to key in data, which is then used to formulate a query to retrieve data from a database, and the outcome is displayed to the user.

Source of servlet - cont



Source of servlet - cont

- Applied to the web information system, it is desirable to
  - allow a client to submit data during a web session
  - retrieve data from the web server host
  - be displayed by the web browser

Source of servlet - cont

- A generic HTTP server does not possess the application logic for fetching the data from the data source.
- Instead, an external process that has the application logic will serve as an intermediary.
- The external process runs on the server host, accepts input data from the web server, exercises its application logic to obtain data from the data source, returns the outcome to the web server, which transmits the outcome to the client.

Source of servlet - cont

- Java Servlet is adopted to augment HTTP in supporting run-time generated web contents.

What Is a Servlet?

- Servlets
  - are modules of Java code that run in a server application to answer client requests.
  - are written in the highly portable Java language and follow a standard framework.
  - provide a means to create sophisticated server extensions in a server and operating system independent way.

What Is a Servlet? - cont

- Typical things that HTTP Servlets can do:
  - Processing and/or storing data submitted by an HTML form.
  - Providing dynamic content
    - e.g. returning the results of a database query to the client.
  - Managing state information on top of HTTP.
  - ……

Servlet's Advantages

- Have significantly *less overhead*
  - A Servlet does not run in a separate process. This removes the overhead of creating a new process for each request.
- Have long lifetime
  - A Servlet stays in memory between requests. There is only a single instance which answers all requests concurrently. This saves memory and allows a Servlet to easily manage persistent data.
- Easily to write
  - Servlets API provides high abstract, programmers do not need to pay more attention to decode and extract the query string.

Servlet Basic - Architectural Support

- Requires the existence of a module known as a servlet engine (servlet引擎) or servlet container (servlet容器).
- The most commonly used implementation that provides the servlet architecture is Apache Tomcat.
- Servlet support is also available on commercial application servers:
  - Weblogic
  - iPlanet
  - WebSphere

Servlet Basic - HttpServlet

```
java.lang.Object
  |
  +--javax.servlet.Servlet
   |
  +--javax.servlet.GenericServlet
        |
        +--javax.servlet.http.HttpServlet
```

All Implemented Interfaces: java.io.Serializable, Servlet, ServletConfig

Servlet Basic - HttpServlet

- An HTTP servlet
  - Abstract details of request into *HTTPServletRequest* object.
  - Abstract details of response into *HTTPServletResponse* object.
  - overrides the *doPost() (POST METHOD)* and/or *doGet() (GET METHOD)* method.

Servlet Basic - HttpServlet

- *HttpRequest* objects capture(encapsulate) request details from requests submitted via Web page forms, including data availability, protocol types, security levels, and so forth.
- *HttpResponse* objects capture response details.
- *HttpSession* objects specific to each user handle user session information in the server. The servlet developer can add and remove information about the user during execution of the servlet.

Servlet Basic - HttpServlet

- The HttpServlet class provides methods, such as *service()*/*doGet()*/*doPost()*, for handling HTTP-specific services.

Servlet Basic -Key HTTP Servlet Mtehods

Method	Description
protected void *doGet*(*HttpServletRequest* req, *HttpServletResponse* resp)	Called by the server to allow a servlet to handle a GET request.
protected void *doPost*(*HttpServletRequest* req, *HttpServletResponse* resp)	Called by the server to allow a servlet to handle a POST request.

Servlet Basic - HttpServletRequest Object

- The ***HttpServletRequest*** object allows you to obtain the arguments that the client sent as part of the request.
- To access data:
  - ***getHead*** – returns the value of the specified request header.
  - ***getMethod*** – returns the name of the HTTP method with which the request was made.
  - ***getQueryString*** – returns the query string sent with the request.
  - ***getParameter*** – returns the value of a request parameter.
  - ***getParameterNames*** – returns the names of the parameters contained in this request.
  - ***getParameterValue***s – returns all the values the given request parameter has.

Servlet Basic - HttpServletResponse Objects

- An ***HttpServletResponse*** object provides ways of returning data to the user:
  - ***setContentType*** – sets the content type of the response being sent to the clien.
  - ***addHeader*** - adds a response header line with the given name and value.
  - ***sendError*** – sends an error response to the client using the specified status code and description.
  - ***setHeader*** – sets a response header with the given name and value.
  - ***getOutputStream*** – returns a ServletOutputStream object suitable for writing binary data in the response.
  - ***getWriter*** – returns a PrintWriter object that can send character text to the client.
  - ***sendRedirect*** - invokes the execution of the servlet at the URL specified in the location string.

How Servlets Handles HTTP Requests

- After receiving an HTTP request, Application Server determines which servlet should be executed to fulfill the request.
- The servlet selection is typically determined by the Web's *uniform resource locator* (URL) that specifies the location of the requested resource, its type, and its protocol.

Example

- Form.html
- formServlet.java
- Web.xml

Mozilla Firefox

文件(F) 编辑(E) 查看(V) 转到(G) 书签(B) 工具(T) 帮助(H)

http://s-c92scq1aqh1wr:7001/formServlet//form.html

转到

My Form!

Username:

Your age:

Submit

完成

开始 收 2 M. L. 3 22:08

Mozilla Firefox

文件(F) 编辑(E) 查看(V) 转到(G) 书签(B) 工具(T) 帮助(H)

http://s-c92scq1aqh1wr:7001/formServlet//form.html 转到

My Form!

Username: steve

Your age: 30

Submit

Input information here and click submit

完成

开始 收 2 M. L. 2 22:09

Form Completed - Mozilla Firefox

文件(F) 编辑(E) 查看(V) 转到(G) 书签(B) 工具(T) 帮助(H)

http://s-c92scq1aqh1wr:7001/formServlet//formServlet

转到

The information you have submitted is as follows:

age: 30

submit: Submit

username: steve

完成

开始 22:09

```html
<html>
<body bgcolor=#FFFFFF>
<hl>
My Form!
</hl>

<p>
<font face="Helvetica">
<form method="post" action="formServlet">
  <table border="0" bgcolor=#eeeeee align=center cellspacing=10>
    <tr>
      <td>Username:</td>
      <td>
        <input type="TEXT" name="username">
      </td>
    </tr>
    <tr>
      <td>Your age:</td>
      <td>
        <input type="TEXT" name="age">
      </td>
    </tr>
  </table>
  <p>
  <center>
    <input type="SUBMIT" name="submit" value="Submit">
```

```java
1  package com.learnweblogic.examples.ch3;
2
3  import java.io.*;
4  import java.util.*;
5  import javax.servlet.*;
6  import javax.servlet.http.*;
7
8  public class formServlet extends HttpServlet{
9
10   /*  The doGet method handles the initial invocation of
11       the servlet.  The default service() method recognizes
12       that it has received a GET and calls this method
13       appropriately. It responds with a form, which will
14       use the POST method to submit data.
15   */
16   public void doGet(HttpServletRequest req, HttpServletResponse res)
17     throws IOException, ServletException
18   {
19     res.setContentType("text/html");
20     res.setHeader("Pragma", "no-cache");
21
22     PrintWriter out = res.getWriter();
23
24     out.println("<html>");
25     out.println("<body bgcolor=#FFFFFF>");
26     out.println("<h1>");
27     out.println("My Form!");
28     out.println("</h1>");
```

```java
public void doGet(HttpServletRequest req, HttpServletResponse res)
  throws IOException, ServletException
{
  res.setContentType("text/html");
  res.setHeader("Pragma", "no-cache");

  PrintWriter out = res.getWriter();

  out.println("<html>");
  out.println("<body bgcolor=#FFFFFF>");
  out.println("<h1>");
  out.println("My Form!");
  out.println("</h1>");
  out.println("<font face=Helvetica>");
  out.println("<form method=post action=FormServlet>");
  out.println("<table border=0 bgcolor=#eeeeee cellspacing=10>");
  out.println("<tr>");
  out.println("<td>Username:</td>");
  out.println("<td>");
  out.println("<input type=TEXT name=username>");
  out.println("</td>");
  out.println("</tr>");
  out.println("<tr>");
  out.println("<td>Your age:</td>");
  out.println("<td>");
  out.println("<input type=TEXT name=age>");
  out.println("</td>");
  out.println("</tr>");
```

```
    out.println("</tr>");
    out.println("</table>");
    out.println("<p>");
    out.println("<center>");
    out.println("<input type=SUBMIT name=submit value=Submit>");
    out.println("</center>");
    out.println("</form>");
    out.println("</font>");
    out.println("</body>");
    out.println("</html>");
  }

  /* Finally, include a separate doPost() r
     when the user responds by clicking on

  /*
    Responds to the "POST" query from the
    original form supplied by the doGet() method.
  */
  public void doPost HttpServletRequest req, HttpServletResponse res)
    throws IOException, ServletException
  {

    // Set the content type of the response.
    res.setContentType("text/html");
    res.setHeader("Pragma", "no-cache");

    PrintWriter pw = res.getWriter();
```

Here doPost will be the right method to handle client's request.

```
62  public void doPost(HttpServletRequest req, HttpServletResponse res)
63    throws IOException, ServletException
64  {
65
66    // Set the content type of the response.
67    res.setContentType("text/html");
68    res.setHeader("Pragma", "no-cache");
69
70    PrintWriter pw = res.getWriter();
71
72    pw.println("<HTML><HEAD><TITLE>Form Completed</TITLE></HEAD>");
73    pw.println("<BODY>The information you have" +
74      " submitted is as follows:");
75    pw.println("<P>");
76
77    // Loop through all of the name/value pairs.
78    Enumeration ParamNames = req.getParameterNames();
79
80    // Loop through all of the name/value pairs.
81    while(ParamNames.hasMoreElements()) {
82
83      // Get the next name.
84      String ParamString = (String)ParamNames.nextElement();
85
86      // Print out the current name's value:
87      pw.println("<b>" + ParamString + ":</b> " +
88        req.getParameterValues(ParamString)[0]);
89      pw.println("<P>");
```

Write HTML page

Get parameters' names, username and age

Get parameters' values, steve and 30.

<servelt> </servelt> and <servlet-mapping> </servlet-mapping>should appear at the same time.

Root element of web.xml

The package and classname of the servlet.

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>

  <servlet>
    <servlet-name>formServlet</servlet-name>
    <servlet-class>com.learnweblogic.examples.ch3.formServlet</servlet-class>
  </servlet>//**** the location of the compiled .class file**//

  <welcome-file-list>
    <welcome-file>/form.html</welcome-file>
  </welcome-file-list>

  <servlet-mapping>
    <servlet-name>formServlet</servlet-name>
    <url-pattern>/formServlet</url-pattern>//***the location of the formServlet.j
  </servlet-mapping>

</web-app>
```

Nickname of the following servlet

It makes

http://localhost:8080/formServlet/form.html = http://localhost:8080/formServlet

Nickname of the above servlet

The path '/formServlet' should be handled by a servlet named 'formServlet'.

JSP

- JavaServer Pages
- An extension of servlet technology, JSPs offer a simplified way to develop servlets.
- Like servlets, they generate dynamic output that is sent back to the client's web browser, thus bridging the client and middle tier.

Outline

- Java Web Basics
- SERVLET
- JSP
  - What is JSPs ?
  - Why JSPs?
  - The Lifecycle of a JSP
  - When to use Servlets or JSPs?

What is JSPs?

- JSP means Java Server Pages
- It is developed by SUN
- JSPs provide essentially the same services as the servlet API
- but they have a higher level, more user-friendly, HTML-tag-like development interface.
- JSPs, also a part of the J2EE standards, greatly accelerate the process of creating Web application that use dynamic and personalized Web content.

What Is JSPs ? - cont

- A JSP is translated into Java servlet before being run, and it processes HTTP requests and generates responses like any servlet.

Why JSPs?

two distinct groups work together to create a Web application:

- the programming team
- the Web design team

```
16    public void doGet(HttpServletRequest req, HttpServletResponse res)
17      throws IOException, ServletException
18    {
19
20    }
21
22    /* Finally, include a separate doPost() method to be called
23       when the user responds by clicking on the submit button: */
24
25    /*
26      Responds to the "POST" query from the
27      original form supplied by the doGet() method.
28    */
29    public void doPost(HttpServletRequest req, HttpServletResponse res)
30      throws IOException, ServletException
31    {
32      HttpSession session = req.getSession();
33
34      session.setAttribute("user", req.getParameter("username"));
35      session.setAttribute("age", req.getParameter("age"));
36      session.setAttribute("method", req.getMethod());
37
38      res.sendRedirect("./welcome.jsp");
39
40    }
41  }
42
```

```
1   <%@ page language="java" contentType="text/html; charset=ISO-
2       pageEncoding="ISO-8859-1"%>
3   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN
4   <HTML>
5   <HEAD>
6   <TITLE>Form Completed</TITLE>
7   </HEAD>
8
9   <BODY>
10  The information you have submitted is as follows:
11  <P>
12  <B>username:<%= (String)session.getAttribute("user")%></B><P>
13  <B>age:<%= session.getAttribute("age")%></B><P>
14  <B>method:<%= session.getAttribute("method")%></B><P>
15  </html>
```

```html
<html>
<body bgcolor=#FFFFFF>
<h1>
My Form!
</h1>

<p>
<font face="Helvetica">
<form method="post" action="./formProcess.jsp">
  <table border="0" bgcolor=#eeeeee align=center cellspacing=10>
    <tr>
      <td>Username:</td>
      <td><input type="TEXT" name="username"></td>
    </tr>
    <tr>
      <td>Your age:</td>
      <td><input type="TEXT" name="age"></td>
    </tr>
  </table>
  <p>
  <center>
    <input type="SUBMIT" name="submit" value="Submit">
  </center>
</form>
</font>
</body>
</html>
```

This form will be sent to a jsp page

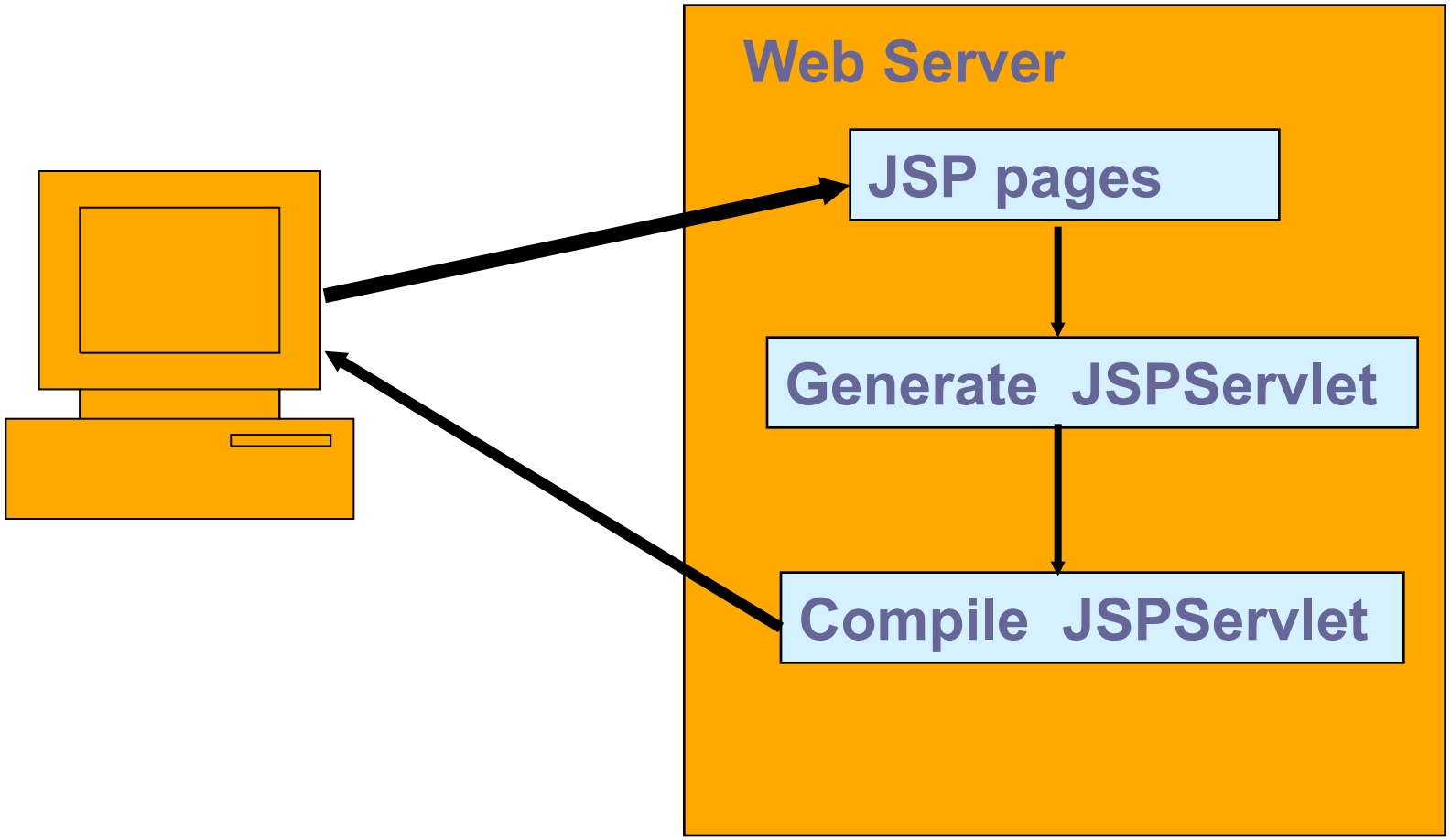
```
1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2      pageEncoding="ISO-8859-1"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http:
4  <HTML>
5  <HEAD>
6  <TITLE>Form Process</TITLE>
7  </HEAD>
8
9  <BODY>
10 <% session.setAttribute("user", request.getParameter("username"));
11     session.setAttribute("age", request.getParameter("age"));
12     session.setAttribute("method", request.getMethod());
13
14      response.sendRedirect("./welcome.jsp");%>
15 </BODY>
16 </html>
```

Process the request and send response

The Lifecycle of a JSP

- Because JSPs are compiled into classes very similar to servlets the first time they are requested, their behavior and controls are identical to servlets after the first request.
  - Initialization
  - Loading and Instantiation
  - Request Handling
  - End of Service

The first time a JSP is requested



When to use Servlets or JSPs?

Use JSPs if:

- You are building HTML pages that are not trivial (having **more than a few lines** and with advanced features such as tables and so forth).
- You have one development group doing the interface design (Web pages) while another is building the Java code.
- Your HTML code changes much more frequently than the presentation logic specified by Java code.

When to use Servlets or JSPs?

Use servlets if:

- You are planning to **service** clients other than Web browsers, such as application clients.
- You have a complex user interaction model, which includes complicated Web pages that

  are highly customized.(like the struts framework)