

## Chapter 35 JavaServer Pages

1. A JSP page must first be processed by a Web server before it can be displayed in a Web browser. The Web server must support JSP, and the JSP page must be stored in a file with a .jsp extension. The Web server translates the .jsp file into a Java servlet, compiles the servlet, and executes it. The result of the execution is sent to the browser for display. Figure 35.2 shows how a JSP page is processed by a Web server.
2. For a JSP file to run from Tomcat, it must be placed in c:\jakarta-tomcat-4.1.27\webapps\examples\jsp (or you can specify another location in the Tomcat configuration file.)
3. You cannot display a JSP file by simply typing the file name, because JSP must be compiled into servlets and executed by a JSP-enabled Web server.
4. A JSP *expression* is used to insert a Java expression directly into the output. It has the following form:

```
<%= Java-expression %>
```

The expression is evaluated, converted into a string, and sent to the output stream of the servlet.

A JSP *scriptlet* enables you to insert a Java statement into the servlet's `jspService` method, which is invoked by the `service` method. A JSP scriptlet has the following form:

```
<% Java statement %>
```

A JSP *declaration* is for declaring methods or fields into the servlet. It has the following form:

5.

Three errors:

- (1) `<%! int k %>` must have a semicolon for Java statement to end
- (2) should have braces to enclose the statement inside the loop body even though there is only one statement.
- (3) `<%= j; %> <br>` should have no semicolon for JSP expression.

6.

```
<%! int k; %>    k is an instance variable because it is defined in the  
declaration  
<%! int i; %>    i is an instance variable because it is defined in the  
declaration
```

```
<% for (int j = 1; j <= 9; j++) k += 1;%> k is a local variable because it is defined in the statement tag.
```

```
<%= k><br> <%= i><br> <%= getTime()><br>  
<% private long getTime() {  
    long time = System.currentTimeMillis(); time is a local variable  
    return time; } %>
```

7. You can use variables in JSP. For convenience, JSP provides eight predefined variables from the servlet environment that can be used with JSP expressions and scriptlets. These variables are also known as *JSP implicit objects*. The predefined variables are request, response, out, session, application, config, and page.

8. No. Because they are local variables in the service method. The predefined variables (e.g., request, response, out) correspond to local variables defined in the servlet methods doGet and doPost. They must appear in JSP scriptlet, not in JSP declaration.

9. Same reason as in 8.

10. A JSP directive is a statement that gives the JSP engine information about the JSP page. The page directive lets you provide information for the page, such as importing classes and setting up content type. The page directive can appear anywhere in the JSP file.

11. No. JDK 1.2 or higher requires a class to be in a package for it to be imported.

12. If you use a custom class from a JSP, it should be placed in c:\jakarta-tomcat-4.1.27\wepapps\examples\WEB-INF\classes\packageNameOfTheClass (or you may reconfigure Tomcat to specify a new location for class files).

13. The difference lies in the scope. JSP allows you to share the object of a class among different pages. However, the scope of object created using the new operator is limited to the page only.

14. The scope attribute specifies the scope of the object. Four scopes are page, application, session, and request.

15. When <jsp:useBean id="objectName" scope="scopeAttribute" class="ClassName" /> is processed, the JSP engine first searches for the object of the class with the same id and scope. If found, the preexisting bean is used; otherwise, a new bean is created.

16. Use `<jsp:setProperty name="beanId" property="*" />` for example.

17. Web applications developed using JSP generally consist of many pages linked together. JSP provides a forwarding tag in the following syntax that can be used to forward a page to another page.

```
<jsp:forward page="destination" />
```