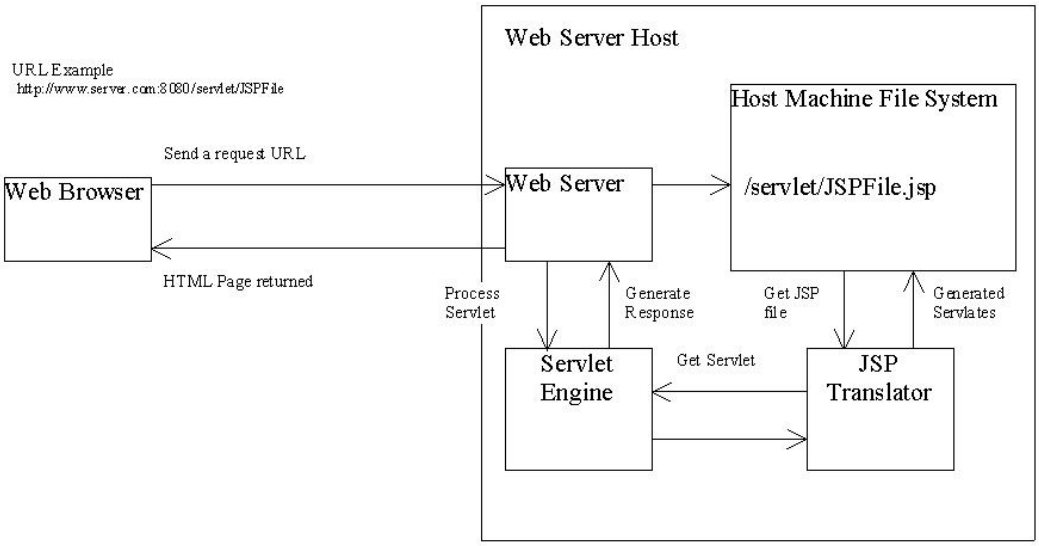
COMP4321 Lab 4 - Java Server Pages

1 Introduction

A web search engine usually provides a friendly user interface for users to input their queries and for displaying the search results to the user. In this lab, we focus on the JavaServer Pages(JSP) as the server side language to generate some simple pages for the search engine. For a web search engine, you should design the pages containing functions which include retrieving the query from user input, performing stemming and stopword removal on the input query, connecting to the database to retrieve the indexed pages, calculating the scores of the pages for the query and outputting the results to the user. Since JSP puts Java inside HTML pages and as a result we can use our implemented components in previous labs directly into our project, JSP is a good candidate for the server side language of our project.

2 Working Mechanism of JSP



3 Selected Topics on JSP

JSP Constructs

- expression `<%= Java-expression %>`
  The expression is evaluated, converted into a string, and sent to the output stream of the servlet.
- scriptlet `<% Java-statement %>`
  A JSP scriptlet enables you to insert a Java statement into the servlet's jspService method, which is invoked by the service method.
- declaration `<%! Java method or field declaration %>`
  A JSP declaration is for declaring methods or fields into the servlet. It is not a good idea to declare variable here since these are declarations, and will only be evaluated once when the page is loaded.

Comments

- HTML comments
  The comment will appear in the resultant HTML file.
- JSP comments `<%-- JSP Comment --%>`
  The comment will appear in the generated servlet but not the resultant HTML.

JSP Predefined variables

- request - represents the client's request, which is an instance of HttpServletRequest. You can use it to access request parameters, HTTP headers such as cookies, hostname, etc.
- response - represents the servlet's response, which is an instance of HttpServletResponse. You can use it to set response type and send output to the client.
- out - represents the character output stream, which is an instance of PrintWriter obtained from response.getWriter(). You can use it to send character content to the client.
- session - represents the HttpSession object associated with the request, obtained from request.getSession().
- application - represents the ServletContext object for storing persistent data for all clients. The difference between session and application is that session is tied to one client, but application is for all clients to share persistent data.
- config - represents the ServletConfig object for the page.
- pagecontext - represents the PageContext object. PageContext is a new class introduced in JSP to give a central point of access to many page attributes.
- page - is an alternative to this.

JSP Directives
A JSP directive is a statement that gives the JSP engine information about the JSP page. For example, if your JSP page uses a Java class from a package other than the java.lang package, you have to use a directive to import this package. The general syntax for a JSP directive is as follows:

- page - 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.
  - Attributes for page Directives:
    - import - specifies one or more packages to be imported for this page. For example, the directive imports java.util.* and java.text.*.
    - contentType - specifies the MIME type for the resultant JSP page. By default, the content type is text/html for JSP. The default content type for servlets is text/plain.
    - session - specifies a boolean value to indicate whether the page is part of the session. By default, session is true.
    - buffer - specifies the output stream buffer size. By default, it is 8KB. For example, the directive specifies that the output buffer size is 10KB. The directive specifies that a buffer is not used.
    - autoFlush - specifies a Boolean value to indicate whether the output buffer should be automatically flushed when it is full or whether an exception should be raised when the buffer overflows. By default, this attribute is true. In this case, the buffer attribute cannot be null.
    - isThreadSafe - specifies a Boolean value to indicate whether the page can be accessed simultaneously without data corruption. By default, it is true. If it is set to false, the JSP page will be translated to a servlet that implements the SingleThreadModel interface.
    - errorPage - specifies a JSP page that is processed when an exception occurs in the current page. For example, the directive specifies that HandleError.jsp is processed when an exception occurs.
    - isErrorPage - specifies a boolean value to indicate whether the page can be used as an error page. By default, this attribute is false.
- include - lets you insert a file to the servlet when the page is translated to a servlet. The include directive must be placed where you want the file to be inserted.
- tablib - lets you define custom tags.

4 Example

- Example 1: example1.jsp
  It is a simple example demonstrating JSP constructs, JSP comments, and JSP directives.
- Example2: form.html, example2.jsp
  It is an example demonstrating Form Processing in JSP
- Example3: example3.jsp
  It is an example demonstrating the usage of Cookies in JSP
- Example4: example4.jsp
  It is an example demonstrating the usage of Sessions in JSP
- To set up and publish the jsp in your local machine, follow these steps:
  1. Download Tomcat
  2. Unzip and put under some path
  3. Add environment var:
     CATALINA_HOME = "your tomcat path (e.g. `C:\apache-tomcat-9.0.70`)"
     JAVA_HOME = "your jdk path (e.g. `C:\Program Files\Java\jdk-19`)"
  4. To start up, run: `%CATALINA_HOME%\bin\startup.bat`
  5. To shut down, run: `%CATALINA_HOME%\bin\shutdown.bat`
  6. Go to `%CATALINA_HOME%\webapps` and create folder "lab4"
  7. Write and put your files under `%CATALINA_HOME%\webapps\lab4`
  8. Open in your browser: http://localhost:8080/lab4/lab4.html to test your program

5 Exercise

You should write a HTML file called lab4.html. It contains one text input box for entering a list of words. The user is expected to separate each word by space. For example, the user may type "information retrieval computer science".

You should write a JSP file named lab4.jsp. It gets the string in the input box from the web page lab4.html. The program should divide it into a list of words, and output it line by line on the browser.

The following web page screen shots show the expected result of your program.

Figure 1: The outlook of the web page lab4.html

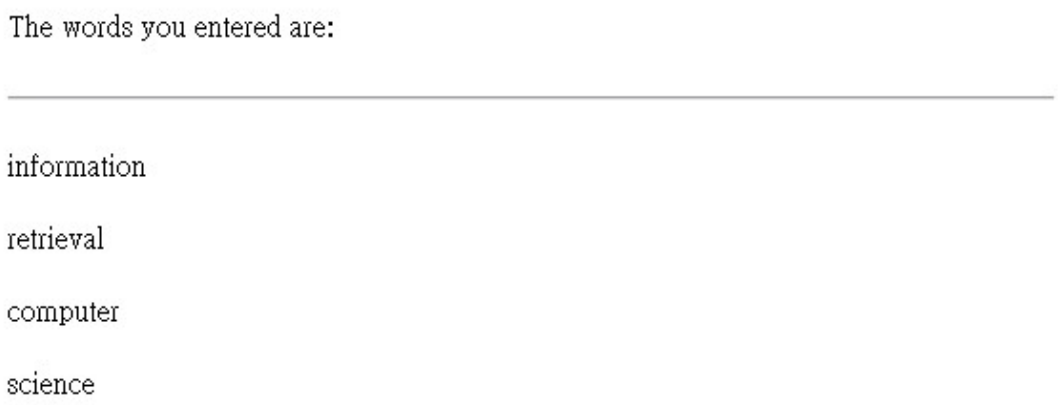


Figure 2: Expected output after user clicking the button "Enter"

Submission
no need to submit

Useful Links

1. JSP tutorial: http://www.jsptut.com/
2. JSP official page: http://java.sun.com/products/jsp/