JSP Exercises

Exercise 1 : Dynamic Content and Expressions

Q1.1. Data Manipulation

Create a jsp date.jsp page

Display the date the page is loaded. To do this, create the following date.jsp file:

```
<HTML>
<BODY> Today is <%= new java.util.Date() %> </BODY>
</HTML>
```

Q1.2. Access this jsp page via a browser. What happens?

Exercise 2 : Scriplets

Programming cannot be reduced to simply adding expressions to html pages. It is possible to write bloacks of Java code in a JSP. A Java block (scriptlet) is exectued each time the JSP is invoked.

Take the jsp code from 1.1 and manipulate the date using scruptlets.

Q2.1. Modify the jsp of the question 1.1 to be able to display the date adding java code

- Create a variable 'date' of type java.util.Date
- Instantiate the variable 'date' using java.util.Date();
- Write the date (value of the variable 'date')

Q2.2. Generate an html table. For example, the table contains the numbers 1 to n (n =5). For this example, you must write a scriptlet that instantiates a variable n. Then display all integers from 1 to n in the form of a table (like below).

Nombre	1
Nombre	2
Nombre	3
Nombre	4
Nombre	5

Q2.3. Generate a page according to the value of a Boolean variable. To do this write a scriptlet which initializes a boolean variable 'statuser', Then generates an HTML page which displays 'welcome' if 'statuser' is true and 'goodbye' in the opposite case.

(Use Math.random () to determine whether the variable is true or false)

Exercise 3 : Directives

A JSP directive has the following syntax: **<%@ string %>**

Use the data from Q2.1 and add to the first line the dirstive '@ page':

<%@ page import="java.util.*,java.text.*" %>

Q.3.1. Modify the JSP accordingly (consider import).

We now want to import the 'hello' class of the 'hello' package defined by the following program 'hello.java':

```
package bonjour;

public class hello{
  public String affiche(){
    return "Bonjour";
  }
}
```

Q.3.2. Write the JSP page that allows you to import the 'hello' class and execute the 'display' method.

We will now consider the contentType attribute of the page directive that specifies the MIME type (**MIME** stands for **Internet media type**), and the character encoding used by the JSP response.
The default Mime type is
'Text / html' and the character encoding is 'ISO-8859-1'.

Consider two JSPs that display different results after assigning the value of the contentType attribute of the page directive. In the first case, the page-contenttype-html.jsp page displays a table in html format (contentType is 'text / html') while the second JSP, page-contenttype-xml.jsp, displays a result in xml format (contentType is 'text / xml').

Q3.3. Write these pages and view them in a browser

Q3.4. Write a JSP and include the JSP file 'hello.jsp'

using the '@ include' directive:

<% @ include file = "hello.jsp"%>

Exercise 4 : Declarations

JSPs are compiled into classes and all JSP scriptlets are placed in a single method of the class. We can then declare variables and methods and use them in scriptlets and expressions.

Q4.1.

Resume the JSP of Exercise 1 dedicated to the manipulation of the date. We will modify it with declarations to perform the same task (display the date):

- Declare a variable theDate and initialize it with the current date
- Write the getDate() function that returns the date (value of theDate variable)
- Display the date by calling the getDate() function and using an expression that calls new Date()

Access the modified JSP with a browser. What do you notice?

Q4.2. Modify the previous JSP to correct its limitations.s.

Exercise 5 : Tags

Q5.1. Repeat the exercises dealing with the include and forward directives and rewrite the JSP pages using the jsp: include and jsp: forward tags. (do not hesitate to use Google for further information)

Exercise 6 : Sessions & Cookies

Q6.1. Write an HTML page representing a form that allows you to enter the name of a person.

Q6.2. Write a JSP that saves the previously entered name in a session by using the 'theName' attribute. This JSP also has a link to the next page called savename.jsp.

Q6.3. Write the page nextpage.jsp that displays the name entered at the beginning.

Q6.4. Repeat this exercise but consider this time the person's last name, first name, address, age and relationship situation (write a bean and use the import directive)

Cookies are data sent by the web server to the client browser. Cookies are stored on the client's hard drive as small text files. They help the web server to identify users.
We will handle cookies using JSP pages. For this, we will first add a cookie using a JSP page, then display the values of the same cookie using another JSP page.

Q6.6. Start by writing a page (cookie-form.html) that asks the user to enter their name.

Q6.7. Once the user name is entered, the setcookie.jsp page then adds the cookie. It will be necessary to use the getParameter function of the 'request' object to retrieve the name entered by the user and store it in the 'username' variable

- Then create the cookie: Cookie cookie = new Cookie ("username", username);
- Set the maximum age of the cookie (times before it expires) by using the setMaxAge function of the cookie object.
- Add the cookie using the addCookie function of the response object.
- This JSP page also displays a link to the JSP, whose role is to display the values of the cookie.

Q6.8. Finally write the JSP that displays the values of the cookie.