

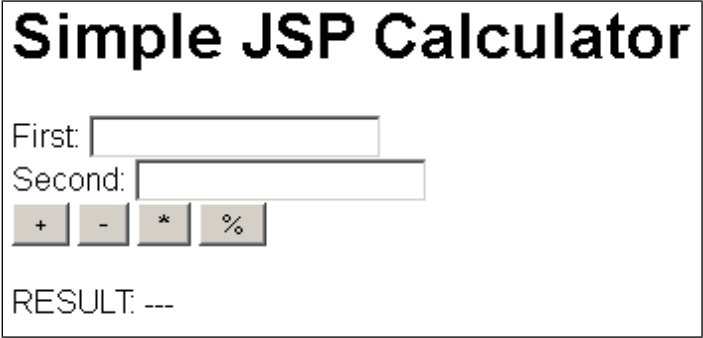
CPRG352 - Web Application Programming

Fall 2021

Topic: Simple JavaServer Pages (JSP)

Problem 1: A Simple JSP-based integer Calculator

Create a Java web application called **JSPCalculator**. Add a new file to the project called ***index.jsp*** and make it the default object for the application.¹ Using **only** the ***index.jsp*** file create a calculator application that looks as following when first run:



Simple JSP Calculator

First:

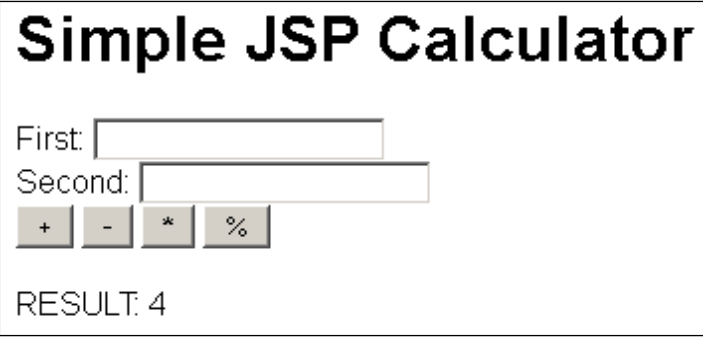
Second:

RESULT: ---

The user can enter two integer values in the text boxes provided, and then click on one of the buttons to tell the application which operation s/he wants to perform on the values.

If no calculation has been performed, or either or both of the values were not provided then the result should show as "---".

Example 1: adding 1 and 3



Simple JSP Calculator

First:

Second:

RESULT: 4

¹ We will not be using the default *index.html* page in this project because we need to execute code to generate the HTML output the application will send to the browser.

Example 2: Multiplying 3 and 7

Simple JSP Calculator

First:

Second:

RESULT: 21

Example 3: clicking on a button without provide either or both values

Simple JSP Calculator

First:

Second:

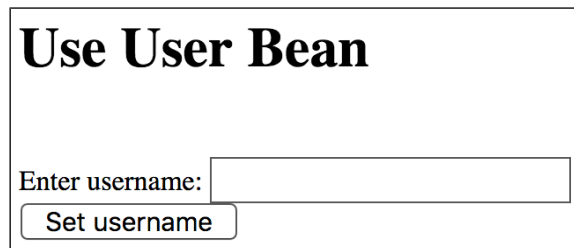
RESULT: ---

Problem 2: Using a JavaBean in a JSP

Create a new web application project called **JSPuseBean**. Add the following components to it:

- A JSP called **useBean.jsp**. This should be the default page shown when accessing the project, *not* **index.html**
- A JavaBean called **UserBean**, in a package called **ca.sait.itsd**. The only attribute needed in this class is a *String* called **username**. This class must follow the required conventions for being a Java bean

When accessed the application should display the following to the user in the **useBean.jsp** page:

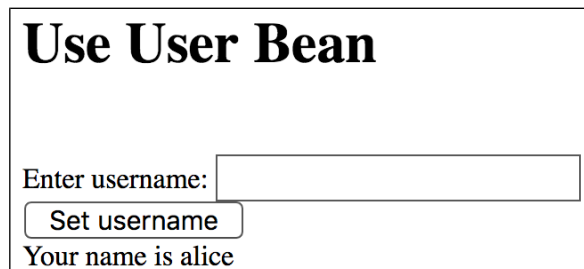


Use User Bean

Enter username:

If the user leaves the username textbox empty and clicks on the “**Set username**” button then the page simply refreshes (no error message needs to be shown).

If the user types in a name in the textbox and clicks on the button then the following message should be displayed:



Use User Bean

Enter username:

Your name is alice

Page Requirements:

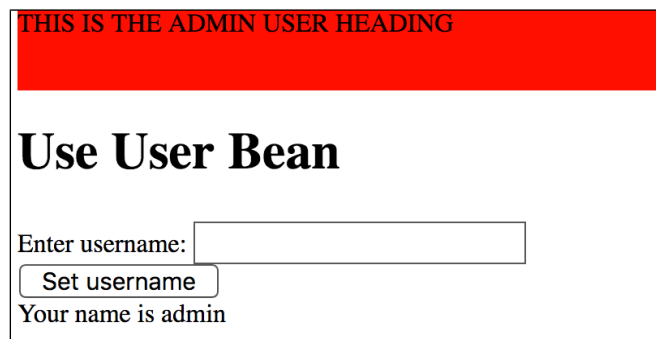
- The form in the JSP should submit data back to the same JSP page
- The page should instantiate an object of type **UserBean** using the **jsp:useBean** tag. The object should be stored in the **session** scope
- Using a Java scriptlet (<%%>) the page should check to see if there is a parameter called **username** being submitted to the page. If there is, then the page should set the **username** property in the **UserBean** instantiated in the page

- To display the “*Your name is...*” message Java code should again be used to check that the **username** property of the **UserBean** object is not **null** or the *empty string*, and display the message only if that is the case

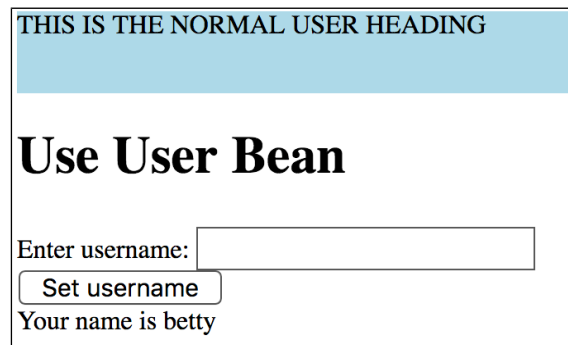
Problem 3: Includes in a JSP

Make a copy of the **JSPuseBean** project called **JSPincludeHeader** (you can do this in NetBeans by right-clicking on the **JSPuseBean** project and selecting “Copy...”).

This version of the project is the same as the original, except that a header will be displayed at the top of the JSP page. If the user’s name is “**admin**” then the following header should be shown:



Otherwise, for any other user name (or a blank user name) the header should look like this:



In both cases the header should be 50 pixels high, and 100% the screen width.

The header information should be stored as two *separate* files in a subdirectory of “**Web Pages**” called “**includes**”. Each header file only needs to contain the HTML for the header itself, not the HTML for a complete page (since the JSP page already contains all the required HTML for a valid HTML page).

The JSP page should examine the user name in the **UserBean** object using Java code and then use **jsp:include** to include the appropriate header file into the top of the JSP page.

Problem 4: Upper Case Word: Servlet as Controller

Create a project called **UpperCaseWord**. This application will get the user to enter a word (in lower case) and then display the upper case version of it.

This application will be different from the previous ones in that it will have a “**front servlet**” (a servlet which receives **all** requests to the application and then determines what to do and show next).

Another change will be that the JSP pages for this application will be stored in the “**WEB-INF**” directory under “**Web Pages**” and will therefore not be directly accessible to users.

The structure of the application should be as follows:

- **WEB-INF/wordForm.jsp**: the page that asks the user to enter a word in lower case
- **WEB-INF/upperCasePage.jsp**: the page that shows the upper case version of the word entered in **wordForm.jsp**
- **UCwordController**: a servlet stored in the package **ca.sait.itsd**. This is the front servlet for the application and must be set as the welcome object in the application via a **web.xml** file in the project

When initially accessed the application should run the **UCwordController** servlet, which should *forward* the user’s browser to the **wordForm.jsp** page:

Word Form

Enter a word in lower case:

If the user leaves the text box empty and clicks on the button then the form should be submitted to the **UCwordController** servlet, which should *forward* the browser back to the **wordForm.jsp** page and show the following message:

Word Form

Enter a word in lower case:

A word is required!

When the user enters a word and clicks on the “**Make Upper Case**” button the form should submit to the **UCwordController** servlet, which should convert the word to upper case and *forward* it on to the **upperCasePage.jsp** page for display, e.g. if the user enters the word “*orange*”:

Upper Case Page

The upper case version of the word you entered is ORANGE

[Back to word entry page](#)

The “**Back to word entry page**” link should send the user back to the **wordForm.jsp** page in the application so the user can enter another word if they want to.