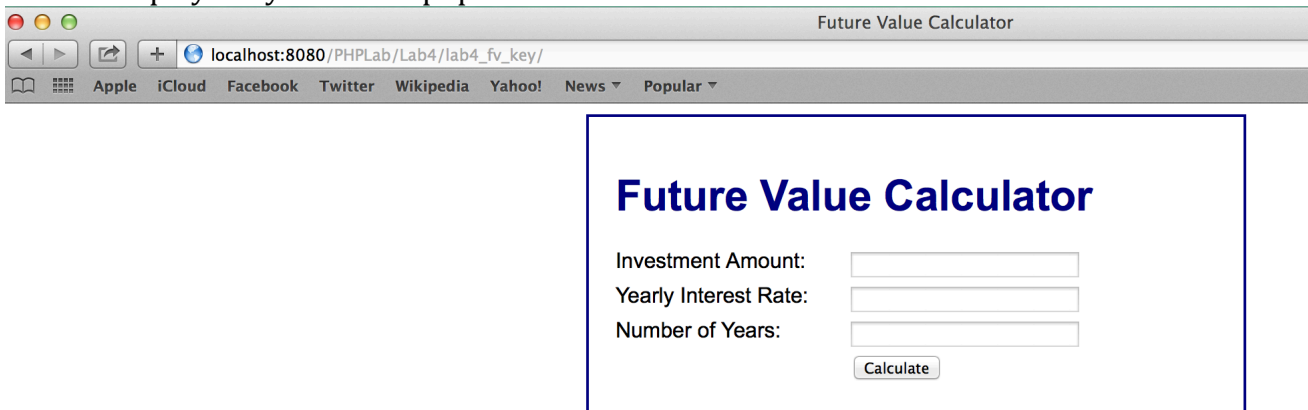


Lab #4 Assignment:

Create a Future Value Calculator using the folder lab4_fv_skeleton and template that I have provided. You will use index.php to display the form. The first time it is displayed, the form will be empty. This is shown in Figure 1 below. Figure 2 shows the form correctly filled out.

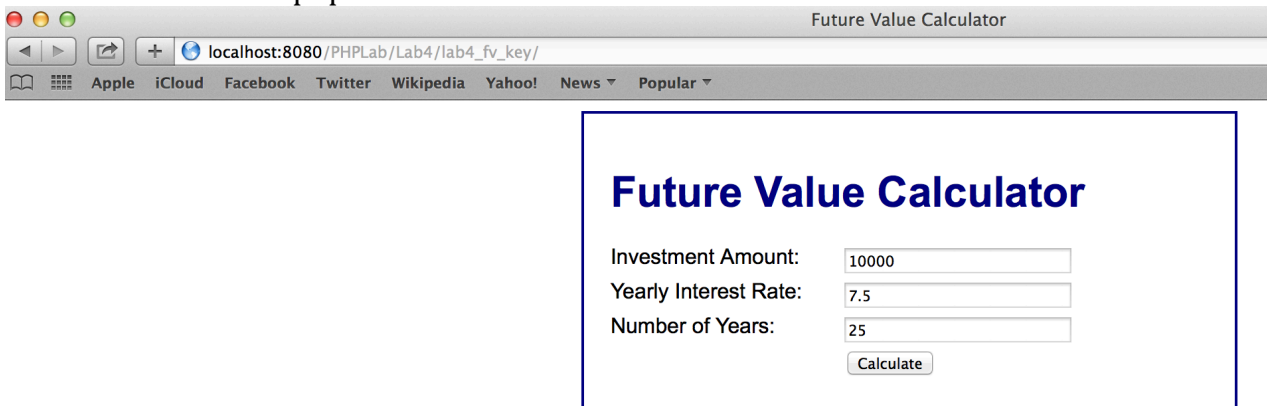
If the form has to be displayed again (if the user entered some values incorrectly)

Figure 1: The First Page: This is how the form is displayed if nothing has been entered yet. This form is displayed by the index.php file.



The screenshot shows a web browser window titled "Future Value Calculator". The address bar displays "localhost:8080/PHPLab/Lab4/lab4_fv_key/". The browser's menu bar includes "Apple", "iCloud", "Facebook", "Twitter", "Wikipedia", "Yahoo!", "News", and "Popular". The main content area features a blue heading "Future Value Calculator". Below the heading are three input fields labeled "Investment Amount:", "Yearly Interest Rate:", and "Number of Years:". Each field is currently empty. A "Calculate" button is positioned below the "Number of Years" field.

Figure 2: The correctly completed form: Here, the user has correctly entered the numeric values. This is still the index.php file.



The screenshot shows the same web browser window as Figure 1, but the form is now filled with values. The "Investment Amount" field contains "10000", the "Yearly Interest Rate" field contains "7.5", and the "Number of Years" field contains "25". The "Calculate" button remains visible below the "Number of Years" field.

Figure 3: When the user clicks on Calculate, you are transferred to the display_results.php file, where the following is displayed. This is assuming that the user entered the data correctly. BTW, I included the future value calculation in the skeleton display_results.php file.

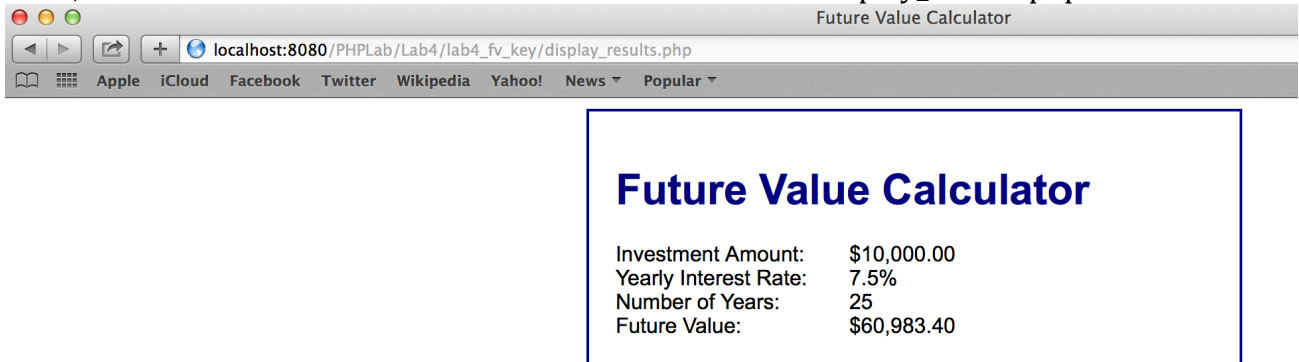


Figure 4: What if the user doesn't enter something correctly? Below, in index.php, the user entered \$10000, which uses the non-numeric character "\$".

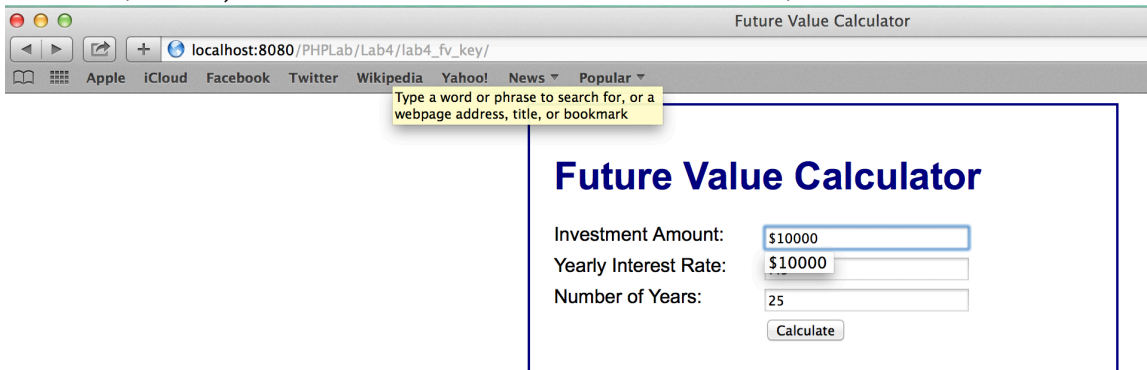
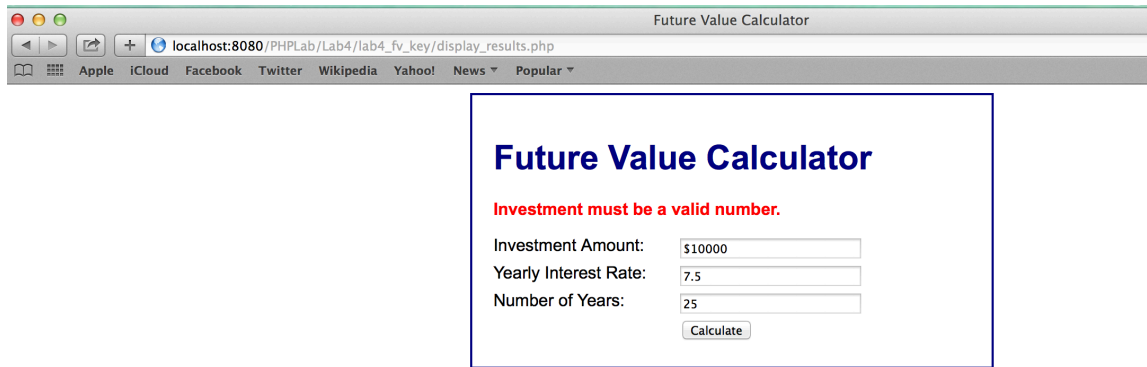


Figure 5: In display_results.php, before actually displaying the results, the program tests for valid user input. If the input is all valid, it displays as in Figure 3, above. If there are errors, then you should "include" the index.php file, which will display the form again, but this time with an error message, and with the current input values filled in.



There are two different ways that you can achieve this effect:

Method #1 (the way I'm showing it in this example)

When `index.php` is displayed, it doesn't "know" whether this is the first time being displayed, or whether it is being displayed because there was an error. Therefore, `index.php` has to test a variable (which you might want to call `$error_message`), to see if it's empty. If it's empty, then this is the first time `index.php` is being displayed, so just display the form. If there's an error message, print it. The css file (which I have included) will color it red. For this method, since you are showing `index.php` (which displays the form) as your first page, you need to set the action to `display_results.php`. This is because the first time that you show the form, when the user clicks Calculate, you have to send the form to the `display_results.php` page. After that point, if there are any errors entered, the `display_results.php` page will catch the error and "include" the `index.php` page once again. At that point, all of the code from `index.php` is included in `display_results.php`, and you could technically use a postback (`action=""`). But since you don't want to create two different files, just leave the `action=display_results.php`.

Method #2 (the way I displayed it for the Welcome example in class)

In class, we used a form that had `action=""` --it was a postback to the same page. You can do that here if you want to, but then you would have the `display_results.php` page be the first page shown, and test to see if it's the first time that you are attempting to display. If it is the first time, then include the form, and have `action=""`. It works very similarly.

I just wanted to show you different ways that you can program the same "look".