WAD LABORATORY 1

NOTE: In this, and all other Laboratories, there is more material than can be covered in a 2 hours Lab. To fully cover the learning objectives of the Unit, you are required to complete all Lab exercises. You will need to complete part of the exercises in the lab and complete other exercises by working at home or on campus, following the lab. You will find that this provides great help towards completing the assignments for the Unit.

1 Objectives covered in this laboratory

- ❑ Get familiar with the lab software used for the unit
- ❑ Create simple PHP pages
- ❑ Know how to run an Ajax program
- ❑ Get a general idea of how Ajax works

2 Exercises

NOTE: Questions (a) – (e) are required exercises for all students, while Question (f) is an additional exercise.

(a) to (e) are two points each

a) Get familiar with the lab software used for the unit.
   1) Activate your web account
   2) Log onto the web server mercury
   3) Create a simple html test file with very little functionality (call it **test.htm**) into your mercury account and run the test page in your browser by going to http://mercury.swin.edu.au/<unitcode>/<username>/<filename>
      (You can find a step-by-step example of using Mercury web server "A step-by-step example of using Mercury.pdf" in the Lab 1 folder downloaded from Canvas)

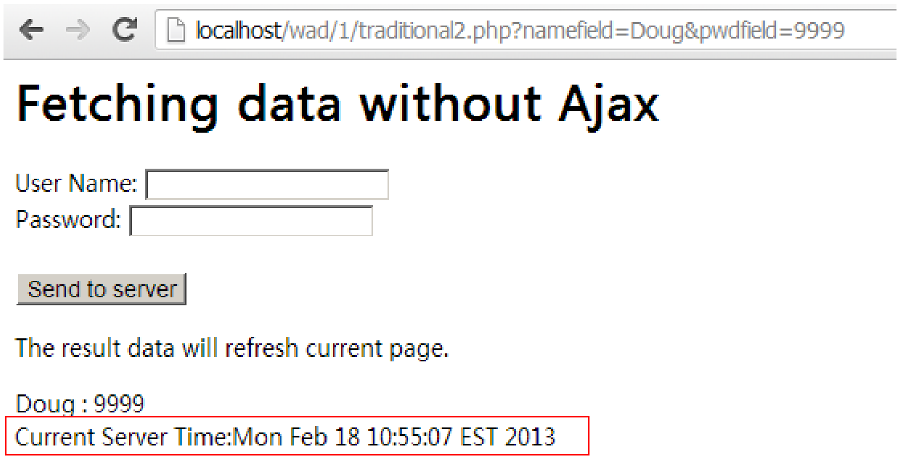
b) Copy the code from "get data with PHP" example from Canvas into your mercury account. Load them into your preferred editor, and just read the files carefully, so that you get a good feeling for the structure of the example, and appreciate how the html and PHP technologies work together. In both IE and Firefox, go to the URL of the PHP file (simplephp.php) and run the PHP example. The PHP program has a "sleep" statement in it, so that the server takes time to respond.

I suggest that you keep copies of all the files for each variant of the system. Thus for this part, copy all the files into another directory, suitably named, and then make the changes required. That way you have a complete record of all files for all examples, suitably stored in directories with suitable names. e.g. you can copy the code of “get data with PHP” to lab01 that you created.

c) Copy the code of “get data with Ajax” example from Canvas, into your mercury account. Load them into your preferred editor, and just read the files carefully, so that you get a general picture for the structure of the example, and appreciate how the html, JavaScript and PHP technologies work together with Ajax. In both IE and Firefox, go to the URL of the HTML files and run the Ajax example (simpeajax.htm). The PHP program has a “sleep” statement in it, so that the server takes time to respond. Experiment with various client interactions in the waiting period – for example, try entering different data whilst waiting for the server to respond, and see what the eventual response is.

d) We now add a new feature based on b). Modify the “simplephp.php” page so that it also displays current server time on the page (see Figure 1) after the user click ‘Send to server’ button. Use PHP built-in functions to finish this task.

Hint: you may need to use “date” and “date_default_timezone_set” function. You could find documents of PHP built-in functions at *http://php.net/quickref.php*.


localhost/wad/1/traditional2.php?namefield=Doug&pwdfield=9999

Fetching data without Ajax

User Name:
Password:

The result data will refresh current page.

Doug : 9999
Current Server Time:Mon Feb 18 10:55:07 EST 2013

Figure 1 Display server time on the page

e) Modify the "simpeajax.js" file in c). Change the third parameter of *'XMLHttpRequest.open'* function from "true" to "false", i.e., change *'xhr.open("GET", url, true)'* to *'xhr.open("GET", url, false)'*. Then run both examples in c) and e) and try to find the difference between them. (Hint: The difference between these two examples is due to asynchronous/synchronous mode of Ajax).

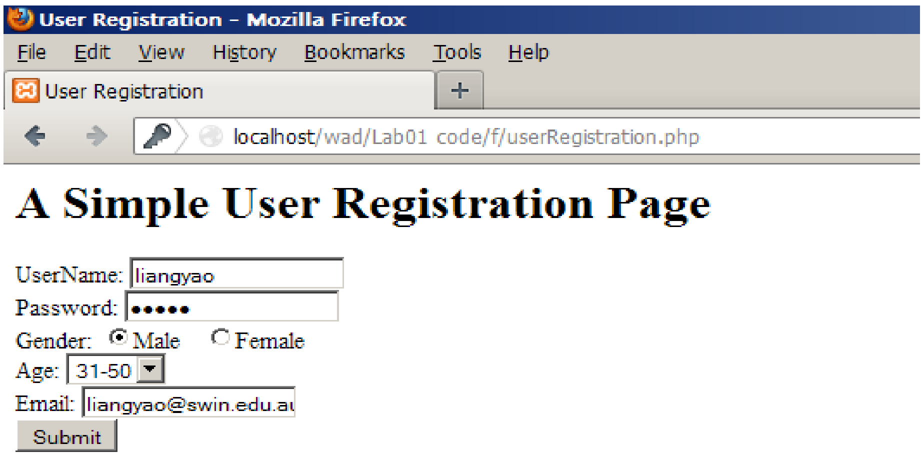
Additional exercise

f) Modify the "simplephp.php" file in d). We now try to extend this page to a user registration page.

1) Change the file name to 'userRegistration.php'.

2) Use HTML form inputs to get user information (see Figure 2). We ask users to input their Username/Password/Gender/Age Range/Email. Try to use different form input types such as text/password/

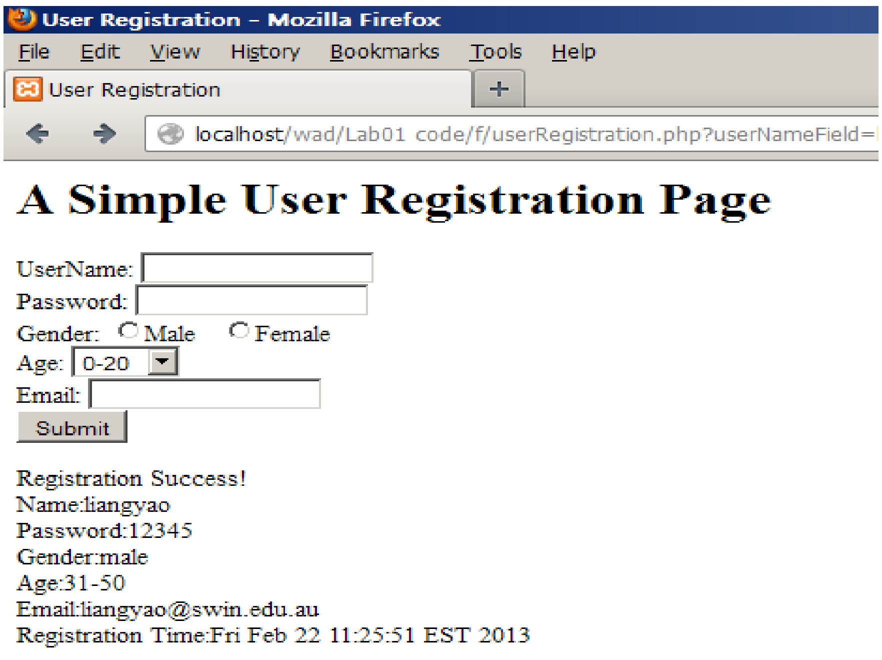
3) When the user click 'submit' button, the PHP page simply check whether user input all required information. If all the information exists, the PHP page displays all input data with the registration time to users. (see Figure 3).


User Registration - Mozilla Firefox
File Edit View History Bookmarks Tools Help
User Registration
localhost/wad/Lab01 code/f/userRegistration.php

A Simple User Registration Page

UserName: liangyao
Password: ●●●●●
Gender: [✓ Male] Female
Age: 31-50
Email: liangyao@swin.edu.au
Submit

Figure 2 Input information


User Registration - Mozilla Firefox
File Edit View History Bookmarks Tools Help
User Registration
localhost/wad/Lab01 code/f/userRegistration.php?userNameField=
Submit

A Simple User Registration Page

UserName:
Password:
Gender: ○ Male ○ Female
Age: 0-20
Email:

```
Registration Success!
Name:liangyao
Password:12345
Gender:male
Age:31-50
Email:liangyao@swin.edu.au
Registration Time:Fri Feb 22 11:25:51 EST 2013
```

Figure 3 Display registered information