PHP (on Linux) Lab

Preliminaries

- You will be logging on to a Linux server. You will have a log-on account on that server, AND an account for the DB server (postgresql). Those accounts have the same name but they are DIFFERENT; however their passwords are synchronized. There will be one account per person. The account name is your e-mail handle
- It is assumed that you've done the PostgreSQL lab, previously assigned, and that you have now some familiarity with our linux server; in particular, that you can now log on to the server. Also, I'm assuming you've read the chapter on HTML and PHP.
- Remember that you will be working FROM your workstation, but the work will be done on the server. You need to keep straight what are you doing locally and what on the server.
- Login to the server (remember, start moba xterm, XliveCD or putty and xming, then type **`ssh -X yourusername@okaram.spsu.edu`** (remember to change yourusername for your username). Notice there is a space after the ssh and another after -X; the X is capitalized, and your username is your spsu email handle.)

PHP Preliminaries

- You will be writing HTML and PHP to communicate with your PostgreSQL database.
- You already have a directory called public_html inside your home directory. Anything that goes in that directory can be accessed from the web, starting with the URL http://okaram.spsu.edu/~yourself/. You also have a directory called php-examples inside it.
- php.net is your friend, especially their tutorial and section on psql

Accessing your directory on the web

On your web browser, go to **`http://okaram.spsu.edu/~yourname`** (substitute yourname for your username :). Browse the files.

Understand/Modify basic PHP

1. Remember you should be OUTSIDE psql !
2. type bluefish & to open bluefish.
3. once bluefish appears, use it to open the file test-db1.php, inside public_html/php-lab (CTRL-O opens a file, or use File menu)
4. Try to understand the code.
    - Notice php code is enclosed within <?php and ?> tags. Anything else is plain HTML
    - Variable references have a $ before them
    - pg_connect gets you a connection to the db, here we specify host (which happens to be on the local machine, and we need to use the loopback address, 127.0.0.1 NOT localhost), username and password. Notice the default database name is the same as your username and so it doesn't need to be specified. Assuming the username is okaram and the password is abc123, the connection string would be:

"host=127.0.0.1 user=okaram password=abc123"

- pg_query executes a query on a given connection.
- The for loop looks a lot like C/Java, except for the extra $
- pg_numrows returns the number of rows in the result set
- pg_fetchobject, fetches one row from the result set, and returns it as an 'object', which is more like a struct. Fields in the row become fields for the struct. All field names are converted to lower case.
- We can access the fields in the struct with the -> operator (as if it was a C/C++ pointer, equivalent to . in Java)
- The $ prompts substitution even inside a string ! (if the string is in double quotes, like this one)
- echo and print 'print' stuff, anything printed gets inserted into the HTML document
- The \n at the end is to make the code more readable; sometimes you need to see HTML code. Browsers will usually ignore it.

5. In your browser, go to the page (inside php-lab, page test-db1.php), open view/page source (just press CTRL-U in mozilla), check what is the actual 'output' of the script.
6. Modify the pg_connect so that it now connects to your database with your username (notice the name of your database is the same as your username; basically dbname and user should both be sent to your username). Test the page
7. In psql, add one more row. Test that your page reflects this change.
8. This program doesn't display the ID field. Modify it so it does.

HTML Tables

1. Open the file test-db2.php in your browser and in bluefish. This file displays an HTML table.
2. Notice it does NOT have a call to pg_connect, but instead has an include directive (looks like a function call). The file included is connect.php, and that file creates the connection. By including this file in all your pages you can keep the connection info in one place rather than replicate it all over.
3. modify pg_connect inside connect.php as in the previous example, make sure it displays your data now.
4. modify test-db2.php to display the id field

Forms

1. Open form1.html (inside php-lab)in your browser, you'll se a form, type a number, and it goes to form_test.php, which displays all rows with that age (there may be none, or one or even more, depending on your data and the number you type).
2. Check the code for the HTML file (form1.html), and the php script (form_test.php).
3. Modify the form, so that there are now 2 fields, with names age1 and age2
4. Modify the php script so that it now displays rows with ages between age1 and age2

More stuff to try

1. Create a new form, with 3 fields, one for id, one for name and another for age. Connect it to a new php page that adds a new row to the DB with that name and age.
2. Create a form and a php file for deleting a row based on its id
3. Create a php file that lists the number of students, with their average age