**Learning PHP by examples: a guided tour through the textbook**

Learning PHP, MySQL, JavaScript, CSS & HTML5

**4th Edition** By Robin Nixon (O'Reilly 2014, ISBN 978-1491918661)

------------------

# Part 1 - Basic setup

(Chapters 1-2)

* Read **chapter 1**
* Download a zip file containing all examples in the book, from <http://lpmj.net/>
* Read **chapter 2**
* Choose \***one**\* Apache/MySQL/PHP server solution for your personal computer / OS. Possible alternatives include: XAMPP (described in the textbook), EasyPHP, WAMPServer, MAMP (Pro), and many others.
* Download, install, configure, and test your WAMP/LAMP/MAMP server setup (follow the book instructions -- chapter 2)

------------------

# Part 2 - Basic PHP

(Chapters 3-7)

* Read **chapter 3**
* Run example 3-1 on your server setup; it should display a "Hello world" message.
* Modify example 3-3 to add an 'echo' for each variable and run it to check each variable's contents displayed on screen and run the modified version

<?php

$mycounter = 1; echo $mycounter; echo "<br />";

$mystring = "Hello"; echo $mystring;

echo "<br />";

$myarray = array("One", "Two", "Three");

echo $myarray; // not quite what you expected, right? echo "<br />";

foreach ($myarray as $item)

{

echo $item; echo "<br />";

}

?>

* Run example 3-4 and make sure you understand what it's doing (and how).
* Run example 3-5 and make sure you understand what it's doing (and how).
* Modify example 3-5 to add an 'echo' to display the entire array and run the modified version

<?php

$oxo = array(array('x', ' ', 'o'),

array('o', 'o', 'x'),

array('x', 'o', ' '));

echo"<pre>";

foreach ($oxo as $row)

{

foreach ($row as $symbol) echo "$symbol ";

echo "<br />";

}

echo"</pre>";

?>

* Run examples 3-6 through 3-9 and make sure you understand what they are doing (and how).
* Run example 3-10 and make sure you understand what it's doing (and how).
* Run example 3-11 and make sure you understand what it's doing (and how).
* Run examples 3-12 through 3-16 and make sure you understand what they are doing (and how).
* Run example 3-17 and make sure you understand what it's doing (and how).
* Modify example 3-17 to add an 'echo' to display the results of calling test() and run the modified version

<?php

$temp = "Calling function test()... "; echo $temp;

test();

echo "<br />";

$temp = "Calling function test() again... "; echo $temp;

test();

echo "<br />";

$temp = "... and again ... "; echo $temp;

test();

echo "<br />";

function test()

{

static $count = 0; echo $count;

$count++;

}

?>

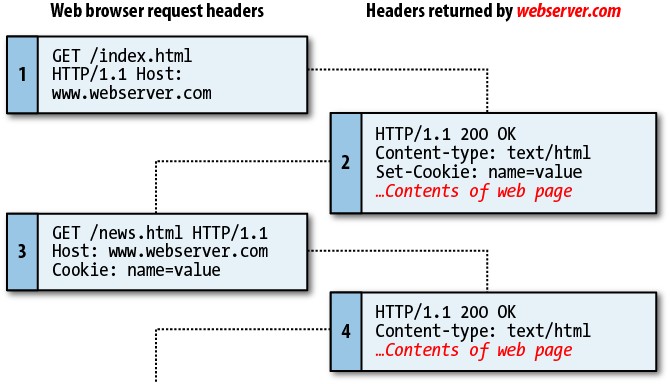
* Answer questions 1-20 at the end of the chapter and \*then\* check your answers against the official answers in App A.
* Read **chapter 4**
* Run examples 4-1 through 4-4 and make sure you understand what they are doing (and how).
* Run examples 4-12 through 4-16 and make sure you understand what they are doing (and how).
* Glance through examples 4-17 through 4-36 (they are similar to what you've seen in JavaScript and other languages) and make sure you understand what they are doing (and how).
* Run example 4-37 and make sure you understand what it's doing (and how).
* Answer questions 1-10 at the end of the chapter and \*then\* check your answers against the official answers in App A.
* Read **chapter 5**
* Run example 5-1 and make sure you understand what it's doing (and how).
* Run examples 5-2 through 5-5 and make sure you understand what they are doing (and how).
* Glance through examples 5-6 through 5-8 and make sure you understand the difference between *include, include\_once*, *require*, and *require\_once*.
* Run example 5-9 and make sure you understand what it's doing (and how).
* Run example 5-10 and make sure you understand what it's doing (and how).
* Run example 5-11 and make sure you understand what it's doing (and how).
* Run examples 5-12 and 5-13 and make sure you understand the difference between them.
* Glance through examples 5-14 through 5-20 and make sure you understand what they are doing (and how).
* Run example 5-21 and make sure you understand what it's doing (and how).
* Glance through examples 5-22 through 5-23 and make sure you understand what they are doing (and how).
* Run example 5-24 and make sure you understand what it's doing (and how).
* Run example 5-25 and make sure you understand what it's doing (and how).
* Run example 5-26 and make sure you understand what it's doing (and how).
* Run example 5-27 and make sure you understand what it's doing (and how).
* Answer questions 1-10 at the end of the chapter and \*then\* check your answers against the official answers in App A.
* Read **chapter 6**
* Run examples 6-1 through 6-3 and make sure you understand what they are doing (and how).
* Run example 6-4 and make sure you understand what it's doing (and how).
* Run example 6-5 and make sure you understand what it's doing (and how).
* Run examples 6-6 through 6-9 and make sure you understand what they are doing (and how).
* Run examples 6-10 through 6-11 and make sure you understand what it's doing (and how).
* Run examples 6-12 through 6-13 and make sure you understand what it's doing (and how).
* Run examples 6-14 through 6-15 and make sure you understand what it's doing (and how).
* Answer questions 1-7 at the end of the chapter and \*then\* check your answers against the official answers in App A.
* Read **chapter 7**
* Run examples 7-1 through 7-2 and make sure you understand what they are doing (and how).
* Run example 7-3 and make sure you understand what it's doing (and how).
* Modify example 7-3 to display "Date is valid".
* Run example 7-4 and make sure you understand what it's doing (and how).
* Locate the newly created file (testfile.txt) in your hard drive. Is it in the expected folder? Inspect its contents. Do they make sense?
* Run examples 7-5 through 7-6 and make sure you understand what they are doing (and how).
* Run example 7-8 and make sure you understand what it's doing (and how).
* Locate the newly created file (testfile2.txt) in your hard drive. Is it in the expected folder? Inspect its contents. Do they make sense?
* Run example 7-9 and make sure you understand what it's doing (and how).
* Locate the newly created file (testfile2.new) in your hard drive. Is it in the expected folder? Inspect its contents. Do they make sense?
* Run example 7-10 and make sure you understand what it's doing (and how). Did it delete testfile2.new successfully?
* Run example 7-11 and make sure you understand what it's doing (and how). Did it update the contents of testfile.txt successfully?
* Run example 7-12 and make sure you understand what it's doing (and how). Did it update the contents of testfile.txt successfully?
* Run example 7-13 and make sure you understand what it's doing (and how). Did it display the contents of testfile.txt successfully?
* Run example 7-14 and make sure you understand what it's doing (and how).
* Run example 7-15 (you must rename it to upload.php first) and make sure you understand what it's doing (and how).
* Run example 7-16 (you must rename it to upload2.php first) and make sure you understand what it's doing (and how).
* Run example 7-17 and make sure you understand what it's doing (and how).
* Answer questions 1-9 at the end of the chapter and \*then\* check your answers against the official answers in App A.

------------------

# Part 3 - Form processing, cookies, and authentication

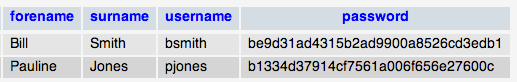
(Chapters 11 and 12)

* Read **chapter 11**
* Open example 11-1 in your favorite editor and save a copy as *formtest.php*.
* Run *formtest.php* and make sure you understand what it's doing (not a whole test, it turns out) (and how).
* Open example 11-2 in your favorite editor and save a copy as *formtest2.php*.
* Run *formtest2.php* and make sure you understand what it's doing (and how).
* Look at Example 11-9 and ensure that you understand the concept of “input sanitization” and how it’s implemented.
* Open example 11-10 in your favorite editor and save a copy as *convert.php*.
* Run *convert.php* and make sure you understand what it's doing (and how).
* Read **chapter 12**
* Study Fig 12-1 below carefully and ensure that you understand it.



* Run Example 12-1 and make sure you understand what it's doing (and how).
* Run Example 12-2 and make sure you understand what it's doing (and how).
* Ensure that you have a valid *login.php* file (with your MySQL information and credentials) in the same directory as the examples in this chapter.
* Run Example 12-3 and make sure you understand what it's doing (and how). More specifically, look at the differences between the users’ actual passwords, their “salted” versions, and the values actually stored in the ‘users’ table in MySQL.

This is what I got:



* Close your browser and reopen it.
* Open Example 12-4 in your favorite editor and save a copy as *authenticate.php*.
* Run *authenticate.php* and make sure you understand what it's doing (and how).
* Close your browser and reopen it.
* Open Example 12-5 in your favorite editor and save a copy as *authenticate2.php*.
* Open Example 12-6 in your favorite editor and save a copy as *continue.php*.
* Run *authenticate2.php* and make sure you understand what it's doing (and how).
* Open Example 12-8 in your favorite editor and save a copy as *continue.php (overwriting the previous file with the same name)*.
* Close your browser and reopen it.
* Run *authenticate2.php* again.
* Press the browser’s reload button and see what happens. Can you understand why?

------------------ **Part 4 - AJAX** (Chapter 17)

* Read **chapter 17**
* Open example 17-1.js in your favorite editor and try to understand it. It is a (rather primitive) way of creating an ***XMLHttpRequest*** object that should work across multiple popular browsers. In practice, you’d probably use a library such as Prototype to take care of this.
* Read the explanation associated with Example 17-2 in detail. Then, run the *urlpost.html*

example.

* Run the *urlget.html* example and understand the differences between this example (using the HTTP GET method) and the previous one (using POST).
* Run the *xmlget.html* example and ensure that you understand what it does and how it does it.

------------------

# Part 5 - MySQL

(Chapters 8-9)

* Access your MySQL account on lamp.cse.fau.edu following the instructions at <http://tsg.eng.fau.edu/servers/lamp-cse-fau-edu/accessing-mysql/> and the email received from TSG at the time your account was created (you will need it for the MySQL password).
* Read **chapter 8**
* Go to **https://lamp.cse.fau.edu/phpMyAdmin/**, select your default database (it has your username), and enter the commands from Example 8.3 using the 'SQL' tab.
* Notice that the newly created table (classics) will appear on the left sidebar.
* Double-check it by typing DESCRIBE classics;
* Make sure you understand the names, types, and meaning of each field in the table.
* Add a new column called id to the table classics with autoincrementing, following the syntax in example 8.5.
* Delete the table classics with the command DROP TABLE classics;
* Re-create the table classics using the syntax in example 8.6.
* Populate the table classics using the syntax in example 8.8.
* Execute a query to see all contents of the newly populated table: SELECT \* FROM classics;
* Rename the table: ALTER TABLE classics RENAME pre1900;
* Rename it again (back to the original name): ALTER TABLE pre1900 RENAME classics;
* Change the data type of a column: ALTER TABLE classics MODIFY year

SMALLINT;

* Add a new column: ALTER TABLE classics ADD pages SMALLINT UNSIGNED;
* Inspect the results of the last two steps: DESCRIBE classics;
* Rename a column: ALTER TABLE classics CHANGE type category VARCHAR(16);
* Remove a column: ALTER TABLE classics DROP pages;
* Create an index following the syntax in example 8.10.
* Delete the table classics with the command DROP TABLE classics;
* Re-create the table classics using the syntax in example 8.12.
* Populate the table classics using the syntax in example 8.8 (*modified* to replace 'type' with 'category').
* Try to create a new column using the syntax: ALTER TABLE classics ADD isbn CHAR(13) PRIMARY KEY;
* You should get an error message (see textbook for explanation)
* Create and populate a new 'isbn' column with data and using a primary key following the syntax in Example 8.13.
* Add a FULLTEXT index to the table 'classics' using: ALTER TABLE classics ADD

FULLTEXT(author,title);

* Execute the two SELECT statements (one at a time) from Example 8.16.
* Count the number of rows using: SELECT COUNT(\*) FROM classics;
* Insert a new record following the syntax in Example 8.18.
* Execute the two SELECT statements (one at a time) from Example 8.19 and make sure that you understand the differences between them.
* Remove the last entry: DELETE FROM classics WHERE title='Little Dorrit';
* Execute the two SELECT statements (one at a time) from Example 8.21.
* Execute the three SELECT statements (one at a time) from Example 8.22.
* Execute the three SELECT statements (one at a time) from Example 8.23.
* Execute the three SELECT statements (one at a time) from Example 8.24.
* Execute the two SELECT statements (one at a time) from Example 8.25.
* Execute the two UPDATE statements (one at a time) from Example 8.26.
* Execute the two SELECT statements (one at a time) from Example 8.27.
* Run the query: SELECT category, COUNT(author) FROM classics GROUP BY category; and try to understand what it does.
* Create a customers table following the syntax of example 8.28.
* Execute the SELECT statement from Example 8.29 and notice how it joins information from both tables in a meaningful way.
* Repeat the previous step using the syntax SELECT name,author,title FROM customers NATURAL JOIN classics;
* Repeat it again, this time using the syntax:

SELECT name,author,title FROM customers

JOIN classics ON customers.isbn=classics.isbn;

* Do it again, this time using:

SELECT name,author,title from

customers AS cust, classics AS class WHERE cust.isbn=class.isbn;

* Execute the three SELECT statements (one at a time) from Example 8.30.
* Read **chapter 9**
* Look at Table 9.1 (below) and make sure you understand why its design is (highly) inefficient.

*Table 9-1. A highly inefficient design for a database table*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Author 1** | **Author 2** | **Title** | **ISBN** | **Price (USD)** | **Customer name** | **Customer address** | **Purch. date** |
| *David Sklar* | *Adam Trachtenberg* | *PHP*  *Cookbook* | *0596101015* | *44.99* | Emma Brown | 1565 Rainbow Road, Los Angeles, CA 90014 | Mar 03  2009 |
| Danny Goodman |  | Dynamic HTML | 0596527403 | 59.99 | **Darren Ryder** | **4758 Emily Drive, Richmond, VA 23219** | **Dec19**  **2008** |
| Hugh E Williams | David Lane | PHP and MySQL | 0596005436 | 44.95 | Earl B. Thurston | 862 Gregory Lane, Frankfort, KY 40601 | Jun 22  2009 |
| *David Sklar* | *Adam Trachtenberg* | *PHP*  *Cookbook* | *0596101015* | *44.99* | **Darren Ryder** | **4758 Emily Drive, Richmond, VA 23219** | **Dec19**  **2008** |
| Rasmus Lerdorf | Kevin Tatroe & Peter MacIntyre | Programming PHP | 0596006815 | 39.99 | David Miller | 3647 Cedar Lane, Waltham, MA 02154 | Jan 16  2009 |

- Look at Tables 9.2 and 9.3 (below) and make sure you understand why this design is better than the one in Table 9.1.

*Table 9-2. The result of stripping the author columns from Table 9-1*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Title** | **ISBN** | **Price (USD)** | **Customer name** | **Customer address** | **Purchase date** |
| *PHP Cookbook* | *0596101015* | *44.99* | Emma Brown | 1565 Rainbow Road, Los Angeles, CA 90014 | Mar 03 2009 |
| Dynamic HTML | 0596527403 | 59.99 | **Darren Ryder** | **4758 Emily Drive, Richmond, VA 23219** | **Dec 19 2008** |
| PHP and MySQL | 0596005436 | 44.95 | Earl B. Thurston | 862 Gregory Lane, Frankfort, KY 40601 | Jun 22 2009 |
| *PHP Cookbook* | *0596101015* | *44.99* | **Darren Ryder** | **4758 Emily Drive, Richmond, VA 23219** | **Dec 19 2008** |
| Programming PHP | 0596006815 | 39.99 | David Miller | 3647 Cedar Lane, Waltham, MA 02154 | Jan 16 2009 |

*Table 9-3. The new Authors table*

**ISBN Author**

0596101015 David Sklar

0596101015 Adam Trachtenberg

0596527403 Danny Goodman 0596005436 Hugh E Williams 0596005436 David Lane

0596006815 Rasmus Lerdorf

0596006815 Kevin Tatroe

0596006815 Peter MacIntyre

* Look at Tables 9.4 and 9.5 (below) and make sure you understand why this design is better than the one in Table 9.2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Table 9-4. The new Titles table* | | | | |
| **ISBN** | **Title** | **Price** |  |  |
| 0596101015 | PHP Cookbook | 44.99 |  |  |
| 0596527403 | Dynamic HTML | 59.99 |  |  |
| 0596005436 | PHP and MySQL | 44.95 |  |  |
| 0596006815 | Programming PHP | 39.99 |  |  |
| *Table 9-5. The Customer details from Table 9-2* | | |  |  |
| **ISBN** | **Customer name** | **Customer address** | | **Purchase date** |
| 0596101015 | Emma Brown | 1565 Rainbow Road, Los Angeles, CA 90014 | | Mar 03 2009 |
| 0596527403 | Darren Ryder | 4758 Emily Drive, Richmond, VA 23219 | | Dec 19 2008 |
| 0596005436 | Earl B. Thurston | 862 Gregory Lane, Frankfort, KY 40601 | | Jun 22 2009 |
| 0596101015 | Darren Ryder | 4758 Emily Drive, Richmond, VA 23219 | | Dec 19 2008 |
| 0596006815 | David Miller | 3647 Cedar Lane, Waltham, MA 02154 | | Jan 16 2009 |

* Look at Tables 9.6 and 9.7 (below) and make sure you understand why this design is better than the one in Table 9.5.

*Table 9-6. The new Customers table*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CustNo** | **Name** | **Address** | **City** | **State** | **Zip** |
| 1 | Emma Brown | 1565 Rainbow Road | Los Angeles | CA | 90014 |
| 2 | Darren Ryder | 4758 Emily Drive | Richmond | VA | 23219 |
| 3 | Earl B. Thurston | 862 Gregory Lane | Frankfort | KY | 40601 |
| 4 | David Miller | 3647 Cedar Lane | Waltham | MA | 02154 |

*Table 9-7. The new Purchases table*

|  |  |  |
| --- | --- | --- |
| **CustNo** | **ISBN** | **Date** |
| 1 | 0596101015 | Mar 03 2009 |
| 2 | 0596527403 | Dec 19 2008 |
| 2 | 0596101015 | Dec 19 2008 |
| **CustNo** | **ISBN** | **Date** |
| 3 | 0596005436 | Jun 22 2009 |
| 4 | 0596006815 | Jan 16 2009 |

- Look at Tables 9.8 through 9.11 (below) and make sure you understand why this design is better than the one in Table 9.6.

*Table 9-8. Third Normal Form Customers table*

|  |  |  |  |
| --- | --- | --- | --- |
| **CustNo** | **Name** | **Address** | **Zip** |
| 1 | Emma Brown | 1565 Rainbow Road | 90014 |
| 2 | Darren Ryder | 4758 Emily Drive | 23219 |
| 3 | Earl B. Thurston | 862 Gregory Lane | 40601 |
| 4 | David Miller | 3647 Cedar Lane | 02154 |

*Table 9-9. Third Normal Form Zip codes table*

|  |  |
| --- | --- |
| **Zip** | **CityID** |
| 90014 | 1234 |
| 23219 | 5678 |
| 40601 | 4321 |
| 02154 | 8765 |

*Table 9-10. Third Normal Form Cities table*

|  |  |  |
| --- | --- | --- |
| **CityID** | **Name** | **StateID** |
| 1234 | Los Angeles | 5 |
| 5678 | Richmond | 46 |
| 4321 | Frankfort | 17 |
| 8765 | Waltham | 21 |

*Table 9-11. Third Normal Form States table*

|  |  |  |
| --- | --- | --- |
| **StateID** | **Name** | **Abbreviation** |
| 5 | California | CA |
| 46 | Virginia | VA |
| 17 | Kentucky | KY |
| 21 | Massachusetts | MA |

- (OPTIONAL) create those tables in MySQL (via phpMyAdmin) and "play" with them.

------------------

# Part 6 – MySQL, MySQLi, and PHP

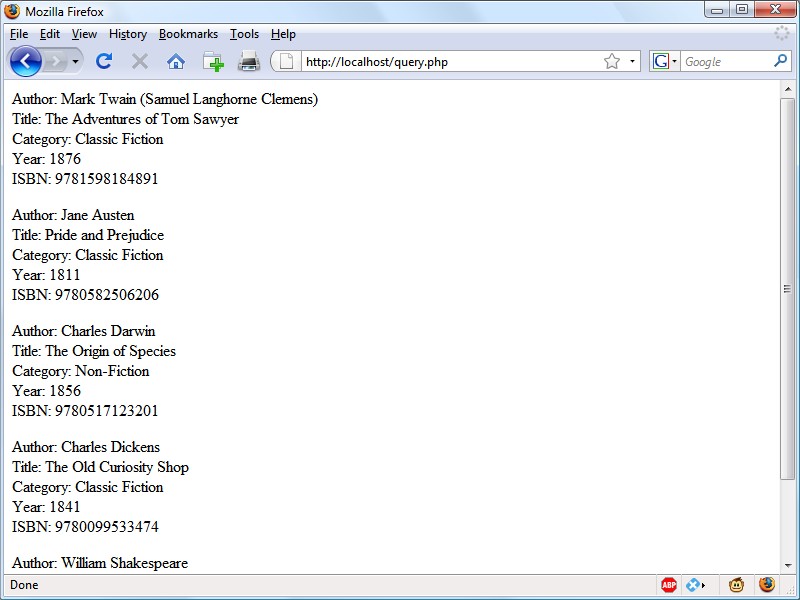
(Chapter 10)

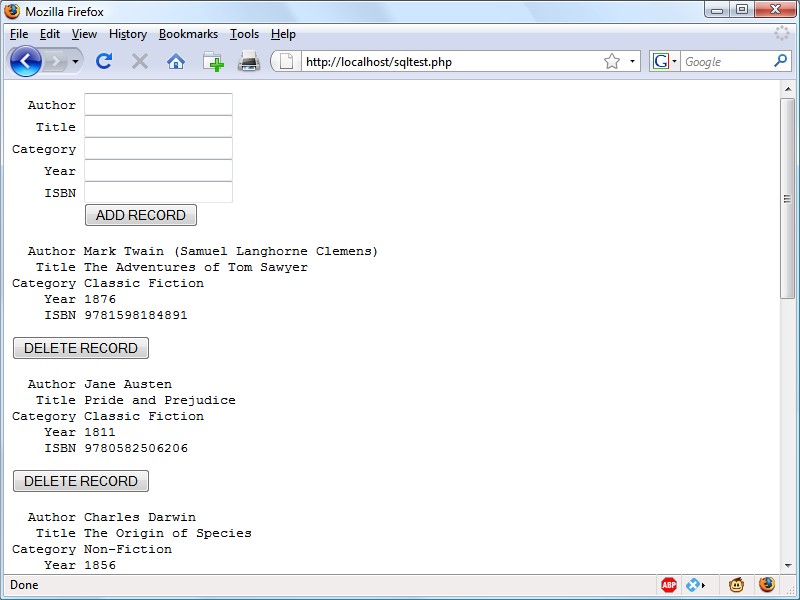
* Read **chapter 10**
* You will build a simple web-based application, using PHP to enable access to some of the previously created tables.
* You have two options: running them on your local (MAMP/LAMP/WAMP) server (recommended) **or** uploading them and running them on lamp.cse.fau.edu.

**The Process**

The process of using MySQL with PHP is:

* 1. Connect to MySQL.
  2. Select the database to use.
  3. Build a query string.
  4. Perform the query.
  5. Retrieve the results and output them to a web page.
  6. Repeat Steps 3 through 5 until all desired data has been retrieved.
  7. Disconnect from MySQL.
* Modify Example 10.1 to refer to your database name and user credentials and save it to a separate directory with the name *login.php*.
* Populate the ‘classics’ table according to the examples in Chapter 8.
* Copy Example 10.5 to your test directory and rename it to *query.php*.
* Test *query.php*. You should see results similar to Fig. 10.1 in the book.

*Figure 10-1. The output from the query.php program in Example 10-5*

* Copy Example 10.8 to your test directory and rename it to *sqltest.php*.
* Run *sqltest.php*. You should see results similar to Fig. 10.2 in the book.

*Figure 10-2. The output from Example 10-8, sqltest.php*

* Play with *sqltest.php* and test its add/delete capabilities.
* Run Example 10.9 and ensure that it created a table (cats) as expected.
* Run Example 10.10 and ensure that it works as expected.
* Run Example 10.11 and ensure that it drops the table (cats) as expected.
* Run Example 10.9 again and re-create the table.
* Run Example 10.12 to populate the table with contents.
* Add more records to the ‘cats’ table (either by editing Example 10.12 or manually via phpMyAdmin).
* Run Example 10.13 and ensure that it works as expected.
* Run Example 10.14 and ensure that it works as expected.
* Run Example 10.15 and ensure that it works as expected.
* Run Example 10.16 and ensure that it works as expected.
* Ensure that you (still) have tables ‘customers’ and ‘classics’ from Chapter 8. If not, re- create them.
* Run Example 10.17 and ensure that it works as expected.
* Read the section on “Preventing Hacking Attempts” and ensure that you understand the risks involved and the recommended best practices for PHP (and JavaScript) developers.

------------------

# Part 7 - Bringing it all together

(Chapter 21)

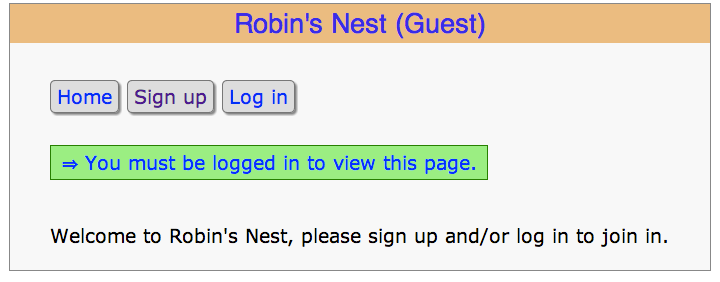
* Read **chapter 21**
* Explore the Robin’s Nest example following the procedure described below.

***Procedure (follow these instructions carefully)***

1. Download the “Robin’s Nest” example from: <http://lpmj.net/>and save a copy in your working directory.
2. Tweak the parameters at the top of the *functions.php* file to reflect your WAMP/MAMP/LAMP setup.
3. Use phpMyAdmin to create a new database and call it ‘rndata’ (to be consistent with line 3 of

*functions.php*).

1. Run the setup code (*setup.php*) to create the necessary tables.
2. (OPTIONAL) Double-­‐check (via phpMyAdmin) that the four tables were, indeed, created (under the ‘rndata’ database).
3. Run the sign up code (*signup.php*). You should see an opening screen like this:



1. Sign up, creating a username and password. (Notice how it checks the username availability “on-­‐the-­‐ fly”, i.e., using AJAX.)
2. (OPTIONAL) Double-­‐check (via phpMyAdmin) that your user information has been stored in table ‘members’.
3. Log in using the newly created username/password. You should see a screen like this (in this case, my username is ogemarques):
4. Edit your profile and upload a picture of yourself. Ask yourself: where is the image stored? How does the PHP script that displays your profile + picture know where to find it?
5. Create a few (2-­‐5) additional members and establish “friendship” links among some of them (if you’re running this on your localhost, you cannot easily invite others to join your network; you have to create your network yourself). In my case I created users ‘alice’, ‘bob’, carol’, and ‘joe’ and “friended” each one of them. This has to be done in several steps: first you follow all of them, then you log out as yourself, log in as each of them separately and “follow back”, i.e. “reciprocate”.
6. Log in as yourself and leave a public “Test” message and a private “Hello” message to one of the people that follow you.
7. Log out, log in as one of your “friends” and check to see if they can see your public and private messages.
8. Play with the app to understand its strengths, weaknesses, and features from a user standpoint. Try to be critical, but positive. **Keep a running list of “things you’d do differently”.**
9. Now it’s time to look at the PHP code behind the scenes. Examine the key PHP scripts, listed below, one at a time, in as much detail as needed to understand *what* they do and *how* they do it.
   * functions.php
   * header.php
   * setup.php
   * index.php
   * signup.php
   * checkuser.php
   * login.php
   * profile.php
   * members.php
   * friends.php
   * messages.php
   * logout.php
10. At this point you know how the sample code works, what it does (and what it doesn’t), and how it does it. If you decide to tweak the code and improve its usefulness and functionality, I suggest taking a look at the list you produced at step 14.