Databases and the Web – Exercise 9

This week we'll enhance the data entry pages built in exercises 5-8 by deleting & editing some data.

Outcomes:

When you've finished this exercise you should:

- Understand how to use PHP and created GET requests to pass information.
- Know how to delete data and appreciate the difference between POST and GET.

Debugging: Because PHP is *server-side* you **must** check your code in the browser through the web server "http://..." *not* by loading the page as "file://"! It is a good idea to:

i. Open the PHP page in the browser (http://localhost/… or http://studentnet.kingston.ac.uk/~k*xxxxxxx*/...), leaving the page open.

ii. Switch to your editor, make small changes to code.

iii. Save the file (upload to StudentNet if necessary).

iv. Switch to the browser and refresh the page.

v. Switch back to the editor to fix any reported syntax errors.

vi. Test the changes in the browser again.

Repeat *ad nauseum*. (Use Alt-Tab/Windows-Tab to cycle through windows ☺)

Task 1: Delete a city!

Deleting data is technically trivial – it's a simple query after all… just make sure you have a backup of your database as accidentally executing "`DELETE FROM city`" is bad ;-)

1. As usual, create a PHP page that has a `title` and all the usual stuff. Call it, for example, `deletecity1.php`.

There are two methods for sending data from the browser to the server (POST and GET) and there are two ways of preparing a GET request … we'll use the obvious one first:

2. Create a `form` that submits to itself (using `PHP_SELF` sanitised with `htmlentities` or `basename`) with the GET method, inside of which we'll create HTML form elements to let the user choose which data to delete.
3. The user needs to be able to select the city to delete so list them!
   - i There are quite a few … for the sake of sanity & the exercise it's worth limiting them, *e.g.* to the smallest group of cities with the same first letter …
     - ➢ Quick aside: Can you construct a query to work this out?
   - ii To delete one data item at a time it's sufficient to have one button per item.
   - iii For the quickest turnaround, make sure the button is a *submit* button.
     - ➢ There's no "there can be only one" rule ;-)
     - ➢ Can you see a potential drawback to this? Perhaps you could use JavaScript to say "Are you sure?"

iv Since the button will be the only entry in the `$_GET` array it'll be easy to use it, so you can label the button with the name of the city. Give it the appropriate attribute so that PHP can do a "DELETE FROM City WHERE …" query that is guaranteed to be unique.

- What's the database field we need?
- How do we pass it to the server?

4. Now is a convenient time to test the page … as we're using GET you can see what data were passed to the server in the URL so try it! Does the query string look like you expect it to?
5. The "submission" branch within this page (*i.e.* whether the form is displayed or a city is deleted) can be based on `count($_GET)==1`
6. Inside `$_GET` we should have a "key => value pair" with the city ID in the *key* (if it were in the value it would have been the button label text…)

   i PHP has "foreach" and list as well as array walking functions based on a "current array position" which starts out at the start of the array … so as we only have one element `current($_GET)` returns the current value (city name) and `key($_GET)` returns the key name (city ID.)

   ii Assign these to a couple of variables for use later.

7. There's not much validation that can be done <sigh of relief?>

   i This is a deletion page … access should be restricted in some way (outside scope.)

   ii We've only two fields: the key and value of the submit button that was pressed, but we should (for completeness) check their contents for sense:

   - The key is presumably numeric?
   - The value is the city name.
   - Provided these are satisfied it's probably safe to proceed…

8. Form a query string, submit the query to the database and display a result … something like this example page.

Task 2: DIY query strings

If the city "Xi´an" is selected in my example page above then the following URL appears in the browser (all one line):

`http://staffnet.kingston.ac.uk/~ku13043/WebDB/ex/week07/deletecity.php?1901=Xi%B4an`

It's the query string on the end that contains the information passed to PHP in $_GET and `PHP_SELF` contains (almost) the rest so if we can arrange to fake the query string in a link URL then we can get rid of the form … this is very convenient at times so let's try it!

1. Take a copy of "Task 2" (*e.g.* copy it to `deletecity2.php`).
2. Remove the form from the HTML.

3. In place of each input element we must write an anchor tag `<a…>…</a>`
   i Its `href` attribute value must consist of
      - the contents of `PHP_SELF` followed by
      - a `?` denoting the start of the query string and then
      - the city ID, an = and finally
      - the city name (so that it matches exactly example page 1)
   ii The text between the opening and closing anchor tags describes the destination of the link … *e.g.* "Delete city "Aba"?"
4. The processing code is unchanged as the GET request ends-up being identical so hopefully you end up with something like this example.

Task 4: Delete lots of cities

Back to forms … Modify your code from "Task 2" above (*e.g.* copy it to `deletecity3.php`) so that the user can select multiple cities to delete, something like the example discussed in lecture 7.

- What's the obvious form element to use?
- How do we ensure PHP sees a collection of these elements all together?
- It should be easy enough to loop over these entries, submitting a new query for each…
- Now's the time to change from using GET to POST.