COMPSCI334S1T2008 Assignment 2

Lecturer: Ulrich Speidel, ulrich@cs.auckland.ac.nz

Due: April 23, 2008, no later than 23:59 pm via the Assignment Dropbox (https://adb.ec.auckland.ac.nz/adb/)

HTML forms

Comboforms (also called comboscripts) are a well-known technique in web programming. Traditionally, your HTML form and the script that processes the form's data sit in separate files. Your browser first requests the file with the form and displays it to the user. Once the user has completed the form, the browser then submits the form data to the script file that processes it.

A comboform combines these two files into the same URL, at least from the browser's perspective. When the browser requests a form from a comboform script, it issues an HTTP GET or POST request for the script. Presume for now that it's a POST request. The script then looks at the POST data (if any). If there is no POST data (or at least none from the form that goes with the script), the script returns the form to the browser. If there is POST data that looks as if it had come from the script's form, the script processes it.

There is a simple comboform example on the course web site (under Assignments) which demonstrates this principle. You can use it as a basis for your assignment if you wish.

If the script outputs the form to the browser, the form's action attribute points back to the script, i.e., the same script is also used to process the form data.

This has a number of advantages, the main one being that there are fewer files to maintain. In an advanced comboform (not for this assignment), the script could also use **`include()`** to include the form and the response page template from outside files to separate user interface and the business code in the script itself. It is then easier to give the form and response page in the external files to a designer and say "Make them look nicer" without running a big risk that the designer will damage your code. Anyway, I digress!

Another advantage is that we can react to erroneous user input by letting the script re-display the form with the "good" input fields pre-completed and an error message that says which field(s) weren't completed correctly. We can repeat this until the user provides proper input to our script.

In this assignment, I want you to write a reasonably complex comboform application in a file **`booking.php`**. The form must be able to be pre-completed by the script if the user input wasn't entirely correct.
If the user input is correct, a confirmation message must be output.

The Application

Consider the following (not entirely imaginary) problem:

From the beginning of the year 2008, car parking at Tamaki Campus will become even more difficult than now as the Department of Whiz Bang Science moves to Tamaki and immediately employs about 2,400 staff. Students must now book car parks at Tamaki. They can only book for five days in advance.

In this system, students and staff are shown a form that asks them for

- their AUID (seven-digit ID number),
- the car registration number of their car,
- which of the next five days they would like to park, via a number of checkboxes (**important**: it must be possible to make multiple selections here – you must find out yourself how this is done, it's not in the handout!). The labels on the checkboxes should show the day of the week (e.g., "Saturday" and the date and the month "17 March"). Any other date information (such as the year) is optional.

The form must not contain any other input elements. The POST name of the AUID field should be "AUID" and that of the registration number field "REGO". Including these two, there must not be more than three different input field names used in the form (i.e., all of the checkboxes must have the same name, hint, hint!). When the data is submitted, the script must check that:

- the AUID is valid, i.e., that it consists of seven digits
- the car registration number is valid. That is, it must show that it consists either:

  - of two uppercase letters followed by one to four digits, or
  - of three uppercase letters followed by one to three digits

  (we won't worry about personalized plates here)

- The script must also check whether the selected days are valid, i.e., whether the data submitted is consistent with one of the next five days. Example: If today is Saturday, 22nd March, you should only be able to book for Sunday, 23rd March to Thursday, 27th March. At least one day ***must*** be selected. For simplicity of checking, you may assume that the form download and form submission happen on the same day (i.e., that you don't have a form download before midnight followed by submission after midnight, turning tomorrow into today). Hint: Think carefully about the format in which you submit the data to the script.

Inputs that do not conform to these specifications must be rejected. If any of the inputs from the form are rejected, the script must return the form to the user. This time, the form must also show a message with a list of the input

problems that have occurred (e.g., “Your AUID must have seven digits”). Affected input fields must be cleared. All other input fields must show the previous input as default. This includes the checkboxes, which must show the correct days selected, if any.

If the input test is successful, the script must return a conformation page to the user, showing the input data, including a list of the days selected.

Your Implementation

Your implementation must come in one file called **`booking.php`** (case-sensitive!). DO NOT USE DIFFERENT FILENAMES – OR YOUR ASSIGNMENT WILL NOT BE MARKED.

Please use the assignment dropbox to submit:

`https://adb.ec.auckland.ac.nz/adb/`

Editing the script and HTML

You can edit the PHP and HTML code in your script with any text editor, such as Notepad, BBEdit, Jext, or whatever your preferred editor is (MS Word is not a good text editor, and I suggest that you stay away from HTML editors such as FrontPage, DreamWeaver, and PageMill until you know what you’re doing).

Note that PHP does not run in your browser, it only runs on webservers that are configured to run PHP. If you don’t have your own (see class website for advice on how to set one up), you can use m3r (see below).

Putting things onto the university web server (web334)

Note that the web334 web server is not visible outside the university network. This note only applies if you are working from one of the labs or on the university wireless network. You can use you own machine as well (see the course web site for setup instructions). **NOTE: For marking purposes, your assignment must work on web334.**

If you want to use web334, you need to open an ssh connection to:

web334. cs.auckland.ac.nz

Use your university login and password. To open an ssh connection, you can use tools such as Putty or SSH Secure Shell, or the ssh2 command (sftp2 for file transfers) under MS Windows. The following instructions assume that you use ssh2 and sftp2, and that you are a little familiar with Unix.

On the Windows command line, cd to the directory of your assignment file. Then issue the command:

`ssh2 <`*`your_upi`*`>@web334.cs.auckland.ac.nz`

Accept the key when prompted, and log in with your university password. You should see a login message ending in this type of prompt:

<*your_upi*>@web334:~$

Assume your UPI is **jblo123**. After login, issue the **ls** command and you will see a directory listing with the **public_html** directory:

```
jblo123@web334:~$ ls
public_html
```

Change into the **public_html** directory and create a directory for your assignment:

```
jblo123@web334:~$ cd public_html/
jblo123@web334:~/public_html$ mkdir ass2
jblo123@web334:~/public_html$ ls ass2
ass2
```

Now, for testing, create an index file:

```
jblo123@web334:~/public_html$ echo Hello! > index.html
```

You should now be able to see this (mini) webpage online. Point your browser at

http://web334.cs.auckland.ac.nz/~jblo123/

(or whatever your UPI is) and you should see the page with “Hello!” on the screen.

Assume you want to upload **booking.php** into the **ass2** directory. Using **sftp2** this is just as easy:

```
sftp2 jblo123@web334.cs.auckland.ac.nz
```

Log in with your password and the following prompt will appear:

```
sftp>
```

Change to the **public_html/ass2** directory

```
sftp> cd public_html/ass2
/var/home/jblo123/public_html/ass2
sftp> put booking.php
booking.php                |          3 kB |   0.1 kB/s |
TOC: 00:00:01 | 100%
```

Do's and Don'ts

Copying

You must not copy other people's code. The above assignment contains enough scope to exclude the possibility of similarities between individual solutions. Do not ask other students for their code. Do not offer them yours. This includes not letting other students see the contents of your folders. Offenders will be dealt with according to the departmental policy on cheating:

http://www.cs.auckland.ac.nz/CheatingPolicy.php

HTML Code

Any HTML that your script outputs to the browser must be clean, i.e., there must be no open or overlapping tags, etc.

Dangerous Code

If we find code in your assignment that could compromise server security (such as unchecked user input), we will deduct some marks. Be smart, think like a hacker and don't leave any back doors open.

Remarking

Note the assignment marking policy on

http://www.cs.auckland.ac.nz/compsci334s1t/assignments/

I do not publish marking guides in advance or make sample solutions available – assignments are for you to experiment and find out.