University of Stirling
Department of Computing Science
PHP – Server Side Scripting

Practical 1
Experiments with PHP and its HTML Interface

You will find it helpful to have your lecture notes to hand.

1. Log-on in the usual way. The webserver to be used in the practicals is `shark.cs.stir.ac.uk`. All files you are using will have to be on `shark`. To access your file space on shark, you will need to map a network drive. In Windows Explorer, select the `Map Network Drive` option from the `Tools` menu. Select a drive and link this to the folder: `\\shark\www\`{your_username}. You will need your password to complete the link. Navigate to the newly created drive and create a directory `ITNP070`.
2. Copy the folder `Prac1` in `groups on wide` into your own file space. You also need to map this directory to a drive in Windows XP. As above, select the item 'Map Network drive' from the Tools menu. The path you need to select is: '`\\wide\groups`'. In the new drive you will find a folder called `ITNP070` and in there the folder `Prac1`. Copy `Prac1` into the `ITNP070` folder you created in the previous step.
3. Close the window on the `/groups/wide/ITNP070` folder, so you do not confuse its contents with your own copy.
4. Open a text editor (Wordpad, Notepad, Textpad). Open the file `hello.php` into the Textpad window. This file is the "Hello World" example from the lectures. Now try and view hello.php in a browser. You should direct the browser to: `shark.cs.stir.ac.uk/`{your_user_name}`/ITNP070/Prac1/hello.php`. Can you see the expected result? Check the page source for this page. Can you see any php code? – Why?
5. Change `hello.php` to print a second line, e.g. "Good Bye!". The newline character is '\n'. Does this produce the expected result? Why? Try and fix it if it does not.
6. Now make a deliberate mistake: for example, remove the semicolon between the first and second echo statements. This time, when you run it, you get an error message. However, the error message points to the wrong line of code. This shows that the error messages you will get are not always accurate. They usually point to the location where the error is noted, not the location where the error was made. You should keep this in mind for future experiments. Fix the error.

7. Make another deliberate mistake: change `echo` to `echoa` (say). Save and run again. Note the kind of error message you get when you use a word that PHP does not understand. It is usually not a good idea to take error messages literally.
8. Now create a new webpage, which contains some PHP statements (from scratch, in a fresh file called, (say) `me.php`). The PHP code should consist of three variables, `$givenName`, `$familyName`, and `$age`, and three assignment statements, one to each variable. Finally, include `echo` statements declaring to the world who you are and how old you are (or would like to be). Does the statement print what you expect? Now change your program in a way that you do not use three variables, but one array which holds the same information. Print the array using a `foreach` loop.
9. Change this program so as to make another deliberate mistake – this time, misspell one occurrence of one of your variables (maybe change the `N` in `$familyName` into lower-case). Run the program again: you will find that PHP does not complain, but produces incorrect output. This is not good. Note how PHP handles unknown variables.
10. The file `if.php` contains an example of the use of `if` and `elseif`, and `if-probs.php` contains illustration of some common pitfalls with `if` which were mentioned in the lectures. Try these out and change some values to experiment. Fix the problems in `if-probs.php`.
11. Open the file `simpleForm.php` into the Textpad window. This file is the form example from the lectures. View `simpleForm.php` in a browser (point to: `shark.cs.stir.ac.uk/{your_user_name}/ITNP070/Prac1/simpleForm.php`) Try and put in some string into the field and submit this data to the webserver.
12. Change `simpleForm.php` to use the GET request to transmit the data. Besides changing the method attribute in the form, what other changes do you need to do? Does the webpage work? What differences can you see compared to using the POST request? Can you also change the name of the text field? What other change do you need to carry out?
13. Add another text field to the form and change the PHP so you can access this field. Initially, just try and output it from within PHP to make sure the "infrastructure" is working.
14. Write a function which takes two parameters – the two values received from the form. The function should simply return a string which contains the concatenation of the two parameters. Print the string.
15. Add a test so the function is only called if both form values contain data. Otherwise, print an error message.
16. Now look at the file `BrokenMowers.php`. This is a webpage which contains some PHP that outputs a cheerful text (reproduced on the back of this handout), but unfortunately some variable names have been replaced by asterisks. (There were no asterisks in the original program text.) The "broken" text is listed below. Edit your copy of the file and get the PHP working again. You may find it useful to work out your changes in pencil on the listing, rather than starting straight away in the editor. You should also check the newline characters.

Make sure you understand this program and how it works.

17. Now change the one `for`-loop in the fixed mowers program to a `do-while` loop and the second `for`-loop to a `while`-loop. Do you need to insert any other additional statements? Could you swap the `do-while` and the `while` loops?

```
<html>
      <head><title>Campfire song, very jolly</title></head>
      <body>
<?php

# some bits have been replaced by ***

$ordinals = array("one", "two", "three", "four",
                  "five", "six", "seven", "eight",
                  "nine", "ten");

for (*** = 0 ; *** < 10 ; ***++) {
  echo "***[***] " ;
  if ($verse == 0) {
    echo "*** " ;
  }
  else {
    echo "*** " ;
  }
  echo "went to mow, went to mow a meadow\n" ;
  for (*** = *** ; $mower > 0; $mower--) {
    echo "*** men, " ;
  }
  if ($verse !=0) {
    echo "\n>" ;
  }
  echo "one man and his dog went to mow a meadow\n\n>" ;
}
?>
</body>
</html>
```

Checkpoint

one man went to mow, went to mow a meadow
one man and his dog went to mow a meadow

two men went to mow, went to mow a meadow
two men,
one man and his dog went to mow a meadow

three men went to mow, went to mow a meadow
three men, two men,
one man and his dog went to mow a meadow

four men went to mow, went to mow a meadow
four men, three men, two men,
one man and his dog went to mow a meadow

five men went to mow, went to mow a meadow
five men, four men, three men, two men,
one man and his dog went to mow a meadow

six men went to mow, went to mow a meadow
six men, five men, four men, three men, two men,
one man and his dog went to mow a meadow

seven men went to mow, went to mow a meadow
seven men, six men, five men, four men, three men, two men,
one man and his dog went to mow a meadow

eight men went to mow, went to mow a meadow
eight men, seven men, six men, five men, four men, three men, two men,
one man and his dog went to mow a meadow

nine men went to mow, went to mow a meadow
nine men, eight men, seven men, six men, five men, four men, three men, two men,
one man and his dog went to mow a meadow

ten men went to mow, went to mow a meadow
ten men, nine men, eight men, seven men, six men, five men, four men, three men, two men,
one man and his dog went to mow a meadow