Tasks to Solve During Seminar 2

Internet Applications, ID1354

Write a document with your answers to the following tasks, and upload it to Canvas at the end of the seminar. The format of the document is not important. You solve the tasks together in the group, but each group member must write and upload their own document.

Mandatory Task

All members of the group shall, in turn, explain and motivate their program to the other group members. Write a brief summary of differences, and comment on advantages and disadvantages of the different solutions. Do not write about minor layout differences.

Evaluate that your web sites meet the requirements specified in the lab tasks, and that all PHP code is well-structured and easy to understand. A typical mistake when writing HTML forms is to set the `action` attribute to a PHP function, which is wrong. The value of the `action` attribute must be a URL. Check that nobody made this mistake. Also evaluate that your reports meet the requirements in the report template and in the lab tasks.

Optional Tasks

Solve as many of these tasks as time allows, and write your solutions in the same document as the mandatory task. If the solution is a piece of code, just paste the code in the document, without bothering about figures or formatting. The tasks may be solved in any order, you do not have to start from task one.

Task 1, PHP Session Handling

The exact functioning of the PHPSESSID cookie is quite often misunderstood. Call a PHP script with different combinations of `session_start`, `session_destroy` and `session_id`. Use for example firebug to carefully examine when the PHPSESSID cookie is set and what it contains. Try to manually delete the cookie (with firebug or another tool) and see what happens to session data. Also try to change the value of the cookie.

Task 2, Anonymous Functions and Closures

a) The code in listing 1 uses an anonymous function and a closure. Try to understand how it works, and what it prints on the last line. Run it and check your answer.

```php
<?php

function getAdder($operand1) {
    return function ($operand2) use ($operand1) {
        return $operand1 + $operand2;
    };
}

$adder = getAdder(2);
$result = $adder(3);
echo $result;
```

Listing 1: PHP code with a closure

b) Try to write a string concatenation function that works as follows, using a closure.

- The function takes one string parameter. The first time it is called it returns a new function, which also takes one string parameter. Each time it is called, it appends the parameter to previously passed strings and returns a new function. When the parameter is an empty string (length zero), the function returns the string resulting from all previous concatenations.
- Here is a sample execution.
  1. **`$concat = concatenation("Hello ")`** returns a new function, which is stored in the variable **`$concat`**. Internally, the function stores the specified string, **`"Hello "`**.
  2. **`$concat = $concat("World")`** also returns a new function, which is stored in the variable **`$concat`**. Internally, the string **`"World"`** is appended to the previous string, **`"Hello "`**.
  3. **`$concat = $concat("!")`** works exactly as the call in bullet two above.
  4. **`$concat = $concat("")`** returns the resulting string, since the parameter has length zero. The returned string is **`"Hello World!"`**.

Task 3, Your Own Choice

Experiment with a PHP feature you are interested in. It could be for example a variable-length argument list (see slide 49 in the lecture notes from lecture four), PHP arrays (slides 37-44 in the lecture notes from lecture four) or parameter passing (slides 20-21 in the lecture notes from lecture five).