T5 - Web Development Seminar

T-WEB-500

Day 09

PHP

3.2

This day is tested by the autograder!

Now that you now the basics of programming in PHP, let's use it to interact with HTML and forms.

Setup a web server

Contrary to HTML, CSS and Javascript which are interpreted client-side by your web browser, PHP is a back-end language. It needs a server which will interpret the language (as does the `php` command you've used yesterday) when an URL is called from the browser.
We won't go into details here but it'll be usefull for you to install and configure a simple PHP web server for testing purpose.
Here's one tutorial: How to Install LAMP, but you can find many more on the internet.

Apache or Nginx are great webserver capable of processing PHP.

Task 01

Turn in: `./task01.php`

Create a `display_menu` function which takes no parameters and returns a string containing HTML capable of rendering a menu, like this:

```
<ul>
    <li><a href="home.php">Home</a></li>
    <li><a href="product.php">Products</a></li>
    <li><a href="about.php">About Us</a></li>
    <li><a href="contact.php">Contact</a></li>
</ul>
```

If you've setup a web server, you can test the function and see your browser render the menu by creating a page like `index01.php` containing:

```
<!doctype html>
<html lang="en">
```

```
<head>
  <meta charset="utf-8">
  <title>Task 01</title>
</head>
<body>
    <?php
        require("task01.php");
        $menu = display_menu();
        echo $menu;
    ?>
</body>
</html>
```

Now, if you go to `http://127.0.0.1/index01.php` you should see your page with the menu.

TASK 02

Turn in: `./task02.php`

PHP can be used to render dynamic content without having to re-write all the page structure for each now page of your website.
Remembers day01 of the pool? If you keep only the content of the `<body>` of each page in separate htmls files like `home.html`, `php.html` and `sql.html`, we'll be able to `include` it in our structure.
Create a `render_body` function which takes a string as parameter. If the string is `home`, `php` or `sql`, returns the content of the corresponding html file (it'll be present in the same directory as `task02.php`).
If the parameter is unknown just return: "
Unknown page
".
If you've setup a web server, you can test the function and see your browser render the body by creating a page like `index02.php` containing:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Task 02</title>
</head>
<body>
    <?php
        require("task02.php");
        $body = render_body("home");
        echo $body;
    ?>
</body>
</html>
```

Now, if you go to `http://127.0.0.1/index02.php` you should see your page with the content of `home.html` inside the `<body>` tag. If you change the value of the parameter, the content of the page changes.

TASK 03

Turn in: `./task03.php`

The previous task was good, but it's not very dynamic as you have to edit your index file to change the value of the parameter.
We'll now become truly dynamic by using *URL parameters* (also called *GET parameters*).
Create a `dynamic_body` function which takes no paramter. It'll check the content of the GET parameter called `page`. Depending of the content of this `page` parameter, do the same thing as the previous task.
If there is no `page` parameter or if the content of it is unknown just return: "
Unknown page
".

There's a special variable in PHP called `$_GET`.

If you've setup a web server, you can test the function and see your browser render the body by creating a page like `index03.php` containing:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Task 03</title>
</head>
<body>
    <?php
        require("task03.php");
        $body = dynamic_body();
        echo $body;
    ?>
</body>
</html>
```

Now, if you go to `http://127.0.0.1/index03.php?page=home` you should see your page with the content of `home.html` inside the `<body>` tag. Now you can change the value of `page` from the URL to load another body content. That's what I call dynamic, you no longer have to edit your file. All your web pages now shares the same structure using a single file.

TASK 04

Turn in: `./task04.php`

Now that you know how to handle URL parameters with `$_GET` we'll see another way of sending data: POST. GET and POST are different in terms of limitations and usage. POST is most commonly used to handle forms data.

More information on GET vs POST and example here: `https://www.w3schools.com/php/php_forms.asp`.

Create a function `whoami` which takes no parameters and prints "Hi, my name is <name> and I'm <age> years old.". The `name` and `age` parameters will come as **POST** data.
If there's no name print: "Hi, I have no name and I'm <age> years old."
If there's no age or it's not a valid age print: "Hi, my name is <name>."
If there's neither a name nor a valid age, guess what you should print.
If you've setup a web server, you can test the function using the `curl` command.
Examples:

```
Terminal
~/T-WEB-500> cat index04.php

~/T-WEB-500> curl -d "name=Jane&age=21" -H "Content-Type: application/x-www-form-ur
-X POST http://127.0.0.1/index04.php
Hi, my name is Jane and I'm 21 years old.
~/T-WEB-500> curl -d "nom=John&age=48" -H "Content-Type: application/x-www-form-url
-X POST http://127.0.0.1/index04.php
Hi, I have no name and I'm 48 years old.
```

TASK 05

Turn in: `./task05.php`

We'll now connect our php script to a real HTML form.
You have to make the following page work (call it `index5.php`, you don't have to turn it in!).

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Task 05</title>
</head>
```

```
<body>
  <?php
   require("task05.php");
   if (form_is_submitted()) {
   ?>
  <p><?php whoami(); ?></p>
  <?php } else { ?>
  <form method="post">
    <div>
      <label for="name">Name</label>
      <input type="text" id="name" name="name" />
    </div>
    <div>
      <label for="age">Age</label>
      <input type="number" id="age" name="age" min="0" />
    </div>
    <div>
      <label for="curriculum">Curriculum</label>
      <select name="curriculum" id="curriculum">
        <option value="">--Please choose an option--</option>
        <option value="pge">PGE (Programme Grande Ecole)</option>
        <option value="msc">MSc Pro</option>
        <option value="coding">Coding Academy</option>
        <option value="wac">Web@cademie</option>
      </select>
    </div>
    <div>
      <input type="submit" name="submit" value="Send" />
    </div>
  </form>
  <?php } ?>
</body>
</html>
```

Thus, you have to create two functions:

- `form_is_submitted`: return a boolean value indicating if the form has already been submitted or not.

Check the `isset` php function.

- `whoami`: works as in the previous tasks with an added features. It must add "I'm a student of <curriculum>." if there's a curriculum specified after the first sentence. Example: "Hi, I have no name and I'm 48 years old. I'm a student of MSc Pro."