PHP Piscine

Day 03

Staff 42 piscine@42.fr

Summary:

This document is the day03's subject for the PHP Piscine.

Contents

Chapter 1

Foreword

Here is some of what Wikipedia has to say about Apaches:

The Apache are culturally related Native American tribes from the Southwestern United States and Northern Mexico. These indigenous peoples of North America speak Southern Athabaskan languages, which are related linguistically to Athabaskan languages in Alaska and western Canada.

Apache people traditionally have lived in Eastern Arizona, Northern Mexico (Sonora and Chihuahua), New Mexico, West Texas, and Southern Colorado. Apacheria, their collective homelands, consists of high mountains, sheltered and watered valleys, deep canyons, deserts, and the southern Great Plains. The Apache tribes fought the invading Spanish and Mexican peoples for centuries. The first Apache raids on Sonora appear to have taken place during the late 17th century. In 19th-century confrontations during the American-Indian wars, the U.S. Army found the Apache to be fierce warriors and skillful strategists.

Apache groups are politically autonomous. The major groups speak several different languages and developed distinct and competitive cultures. The current post-colonial division of Apache groups includes Western Apache, Chiricahua, Mescalero, Jicarilla, Lipan, and Plains Apache (also known as the Kiowa-Apache). Apache groups live in Oklahoma and Texas and on reservations in Arizona and New Mexico. Apache people have moved throughout the United States and elsewhere, including urban centers.

Chapter 2

General Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Only the work submitted on the repository will be accounted for during peer-2-peer correction.
- As when you did C Piscine, your exercises will be corrected by your peers AND/OR by Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Using a forbidden function is considered cheating. Cheaters get **-42**, and this grade is non-negotiable.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We **will not** take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- You cannot leave any additional file in your repository than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called **Google / the Internet / http://www.php.net / ....**
- Think of discussing on the Forum. The solution to your problem is probably there already. Otherwise you will start the conversation.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject …
- By Odin, by Thor ! Use your brain !!!

Chapter 3

Exercise 00 : Install

Turn-in directory : ex00/
Files to turn in:
Allowed functions:

For the first exercise, your only task will be to install **bitnami MAMP**. Watch carefully the video for the installation. We strongly encourage you to explore the configuration possibilities for MAMP and, most importantly, to read the documentation as you will need it for all the exercises of today.

For the examples in the following exercises, we will consider that you have a directory **day03** in the **htdocs** and that you are using Apache (in the **apache2/** directory).

You can find the manager for bitnami MAMP here `/path/to/mamp/manager-osx.app` **which you can use to open the software.**

Chapter 4

Exercise 01 : phpinfo

Turn-in directory : `ex01/`
Files to turn in: `phpinfo.php`
Allowed functions: `phpinfo()`

Create a file named **phpinfo.php** that will execute and show the result of **phpinfo()**.

```
$> curl 'http://localhost:8080/d03/ex01/phpinfo.php'
<!DOCTYPE html PUBLIC ``-//W3C//DTD XHTML 1.0 Transitional//EN'' ``DTD/xhtml1-
transitional.dtd''>
<html xmlns=''http://www.w3.org/1999/xhtml''><head>
<style type=''text/css''>
...
<title>phpinfo()</title><meta name=''ROBOTS'' content=''NOINDEX,NOFOLLOW,
NOARCHIVE'' /></head>
...
$>
```

Chapter 5

Exercise 02 : print_get

Turn-in directory : `ex02/`
Files to turn in: `print_get.php`
Allowed functions: `echo`

Create a file named **print_get.php** that will display all the variables passed in the url.

Example:

```
$> curl 'http://localhost:8080/d03/ex02/print_get.php?login=mmontinet'
login: mmontinet
$> curl 'http://localhost:8080/d03/ex02/print_get.php?gdb=pied2biche&barry=
barreamine'
gdb: pied2biche
barry: barreamine
$>
```

Chapter 6

Exercise 03 : cookie_crisp

Turn-in directory : `ex03/`
Files to turn in: `cookie_crisp.php`
Allowed functions: `echo, setcookie() and time()`

Create a file **cookie_crisp.php** that will allow you to create, read and erase a cookie.

Example:

```
$> curl -c cook.txt 'http://localhost:8080/d03/ex03/cookie_crisp.php?action=set
&name=food&value=choucroute'
$> curl -b cook.txt 'http://localhost:8080/d03/ex03/cookie_crisp.php?action=get
&name=food'
choucroute
$> curl -c cook.txt 'http://localhost:8080/d03/ex03/cookie_crisp.php?action=del
&name=food'
$> curl -b cook.txt 'http://localhost:8080/d03/ex03/cookie_crisp.php?action=get
&name=food'
$>
```

Chapter 7

Exercise 04 : raw_text

Turn-in directory : `ex04/`
Files to turn in: `raw_text.php`
Allowed functions: `header()`

Create a file named **raw_text.php** that will display the exact same thing whether you look at your browser or if you look at its source code with **curl**.

```
$> curl 'http://localhost:8080/d03/ex04/raw_text.php'
<html><body>Hello</body></html>
$>
```

If you have lynx, you can test it like that (right down to the newline).

```
$> lynx -dump 'http://localhost:8080/d03/ex04/raw_text.php'
<html><body>Hello</body></html>

$> lynx -source 'http://localhost:8080/d03/ex04/raw_text.php'
<html><body>Hello</body></html>
$>
```

Chapter 8

Exercise 05 : read_img

Turn-in directory : `ex05/`
Files to turn in: `read_img.php`
Allowed functions: `header()`, `readfile()`

Create a file named **read_img.php** that will return to the browser the file **42.png** with the right Content-Type. You will find this file in the attachment section on the intranet.
You must submit it in your repository inside a folder named **img** so that we can use it again in other exercises.

```
$> ls ../img/
42.png

$> curl --head http://localhost:8080/d03/ex05/read_img.php
HTTP/1.1 200 OK
Date: Tue, 26 Mar 2013 09:42:42 GMT
Server: Apache
X-Powered-By: PHP/5.4.26
Content-Type: image/png

$>
```

Chapter 9

Exercise 06 : members_only

Turn-in directory : ex06/
Files to turn in: members_only.php
Allowed functions: header(), echo. $_SERVER, file_get_contents, base64_encode

---

Create a file **members_only.php** that will require a login/password at the HTTP protocol level. If the login is "**zaz**" and the password is "**Ilovemylittleponey**", the response must be an HTML page that contains an img tag which source is directly the image "**/img/42.png**". You must not provide the image URL (we will probably change the content of the image during the evaluation).

You need to reproduce the following example:

```
$> curl --user zaz:Ilovemylittleponey http://localhost:8080/d03/ex06/
members_only.php
<html><body>
Hello Zaz<br />
<img src='...
...
...
...6MIHnr2t+eeO4Fr+v/H80AmcVvzqAfAAAAAElFTkSuQmCC'>
</body></html>
$>
```

If the login/password doesn't match **"zaz"/"Ilovemylittleponey"**, return an error message exactly like in the following example:

```
$> curl -v --user root:root http://localhost:8080/d03/ex06/members_only.php
* About to connect() to j03.local.42.fr port 8080 (#0)
*   Trying 127.0.0.1...
* connected
* Connected to j03.local.42.fr (127.0.0.1) port 8080 (#0)
* Server auth using Basic with user 'root'
> GET /ex05/members_only.php HTTP/1.1
> Authorization: Basic cm9vdDpyb290
> User-Agent: curl/7.24.0 (x86_64-apple-darwin12.0) libcurl/7.24.0 OpenSSL
/0.9.8y zlib/1.2.5
> Host: j03.local.42.fr:8080
> Accept: */*
>
* HTTP 1.0, assume close after body
< HTTP/1.0 401 Unauthorized
< Date: Tue, 26 Mar 2013 09:42:42 GMT
< Server: Apache
< X-Powered-By: PHP/5.4.26
< WWW-Authenticate: Basic realm=''Member area''
< Content-Length: 72
< Connection: close
< Content-Type: text/html
<
<html><body>That area is accessible for members only</body></html>
* Closing connection #0
$>
```