



# PHP Piscine

## Day 03

Staff 42 [piscine@42.fr](mailto:piscine@42.fr)

*Summary:*

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

# Contents

<b>I</b>	<b>Foreword</b>	<b>2</b>
<b>II</b>	<b>General Instructions</b>	<b>3</b>
<b>III</b>	<b>Exercise 00: Dat vhost !</b>	<b>4</b>
<b>IV</b>	<b>Exercise 01: phpinfo</b>	<b>5</b>
<b>V</b>	<b>Exercise 02: print_get</b>	<b>6</b>
<b>VI</b>	<b>Exercise 03: cookie_crisp</b>	<b>7</b>
<b>VII</b>	<b>Exercise 04: raw_text</b>	<b>8</b>
<b>VIII</b>	<b>Exercise 05: read_img</b>	<b>9</b>
<b>IX</b>	<b>Exercise 06: members_only</b>	<b>10</b>

# Chapter I

## 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.



The tool you will use for your server from now on is PAMP, developed by 42. That one is still in beta, please help us make it better by bringing up the bugs you will encounter by ticket or on the [forum](#).


# Chapter II

## 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 III

## Exercise 00: Dat vhost !

	Exercise 00
	Dat vhost !
	Turn-in directory : <i>ex00/</i>
	Files to turn in : <i>D03.php</i>
	Allowed functions :
	Notes : <i>n/a</i>

This first exercise will make you to install and configure your web server embedded in the PAMP tool. You must accomplish this exercise only if PAMP is at least version 3.0.0, otherwise you can skip it. Therefore, you should know how to configure a vHost, this would be useful and really interesting.

Before using PHP for the web, you must configure your first `VirtualHost` such as :

- PAMP should load your site's elements from `$HOME/http/MyWebSite/d03`
- it must listen to the port 8100

Create the `D03.conf` configuration file that fulfill all the requirements written above. This file would be turned-in in your repository.

If you want to test your configuration file, you can add the following line at the end of the `$HOME/http/conf/my.conf` file :


```
Include <votre-dossier-de-rendu>/ex00/D03.conf
```



Now, exercises and rushes will use PAMP and its associated tools.  
For more information, you can follow [this link](#)

# Chapter IV

## Exercise 01: phpinfo


	Exercise 01
phpinfo	
Turn-in directory : <i>ex01/</i>	
Files to turn in : <b>phpinfo.php</b>	
Allowed functions : <b>phpinfo()</b>	
Notes : <b>n/a</b>	

Create a page named `phpinfo.php` that will execute and show the result on `phpinfo()`;

```
$> curl 'http://eXrXpX.42.fr:8xxx/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 V

## Exercise 02: print\_get

	Exercise 02
print_get	
Turn-in directory : <i>ex02/</i>	
Files to turn in : <b>print_get.php</b>	
Allowed functions : <b>echo</b>	
Notes : <b>n/a</b>	


Create a page named `print_get.php` that will display all the variables passed in the url.

Example:

```
$> curl 'http://eXrXpX.42.fr:8xxx/ex02/print_get.php?login=mmontinet'
login: mmontinet
$> curl 'http://eXrXpX.42.fr:8xxx/ex02/print_get.php?gdb=pied2biche&barry=barreamine'
gdb: pied2biche
barry: barreamine
$>
```

# Chapter VI

## Exercise 03: cookie\_crisp

	Exercise 03
print_crisp	
Turn-in directory : <i>ex03/</i>	
Files to turn in : <i>cookie_crisp.php</i>	
Allowed functions : <i>echo, setcookie() et time()</i>	
Notes : <i>n/a</i>	

Create a page `cookie_crisp.php` that will allow to create, read and erase a cookie.


Example:

```
$> curl -c cook.txt 'http://eXrXpX.42.fr:8xxx/ex03/cookie_crisp.php?action=set&name=plat&value=choucroute'
$> curl -b cook.txt 'http://eXrXpX.42.fr:8xxx/ex03/cookie_crisp.php?action=get&name=plat'
choucroute
$> curl -c cook.txt 'http://eXrXpX.42.fr:8xxx/ex03/cookie_crisp.php?action=del&name=plat'
$> curl -b cook.txt 'http://eXrXpX.42.fr:8xxx/ex03/cookie_crisp.php?action=get&name=plat'
$>
```



# Chapter VII

## Exercise 04: raw\_text

	Exercise 04
	raw_text
	Turn-in directory : <i>ex04/</i>
	Files to turn in : <b>raw_text.php</b>
	Allowed functions : <b>header()</b>
	Notes : <b>n/a</b>

Create a page named `raw_text` that will show the same thing on the screen if you look at its source code with `curl` or its html rendered in Chrome.

```
$> curl 'http://eXrXpX.42.fr:8xxx/ex04/raw_text.php'
<html><body>Hello</body></html>
$>
```


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

```
$> lynx -dump 'http://eXrXpX.42.fr:8xxx/ex04/raw_text.php'
<html><body>Hello</body></html>

$> lynx -source 'http://eXrXpX.42.fr:8xxx/ex04/raw_text.php'
<html><body>Hello</body></html>
$>
```

# Chapter VIII

## Exercise 05: read\_img

	Exercise 05
read_img	
Turn-in directory : <i>ex05/</i>	
Files to turn in : <code>read_img.php</code>	
Allowed functions : <code>header()</code> , <code>readfile()</code>	
Notes : <i>n/a</i>	


Create a page 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 in the following folder “`/img/42.png`” so that we can use it again in other exercises.



```
$> curl --head http://eXrXpX.42.fr:8xxx/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 IX

## Exercise 06: members\_only

	Exercise 06
members_only	
Turn-in directory : <i>ex06/</i>	
Files to turn in : <i>members_only.php</i>	
Allowed functions : <i>header()</i> , <i>echo</i> , <i>\$_SERVER</i> , <i>file_get_contents</i> , <i>base64_encode</i>	
Notes : <i>n/a</i>	

Create a page named *members\_only.php* that will require a login/password at the http protocol level. If the login is “zaz” and the password “jaimelespetitsponeys” the answer must be an html page that contains an *img* tag whose source is directly the image “/img/42.png” but not its url (careful! We will probably change the content of 42.png for the correction, so no solid content value)

You need to reproduce the following example:

```
$> curl --user zaz:jaimelespetitsponeys http://eXrXpX.42.fr:8xxx/ex06/members_only.php
<html><body>
Hello Zaz<br />
<img src='data:image/png;base64,iVBORwOKGgoAAAA...
...
...6MIHnr2t+ee04Fr+v/H80AmcVvzqAfAAAAAE1FTkSuQmCC'>
</body></html>
$>
```

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

```
$> curl -v --user root:root http://eXrXpX.42.fr:8xxx/ex06/members_only.php
* About to connect() to eXrXpX.42.fr port 8xxx (#0)
*   Trying xxx.xxx.xxx.xxx...
* connected
* Connected to eXrXpX.42.fr (xxx.xxx.xxx.xxx) port 8xxx (#0)
* Server auth using Basic with user 'root'
> GET /ex06/members_only.php HTTP/xxx
> Authorization: Basic xxxxxxxxxxxxxxxx
> User-Agent: curl/xxxxxx (x86_64-apple-darwin12.0) libcurl/xxxxxx OpenSSL/xxxxxx zlib/xxxxxx
> Host: eXrXpX.42.fr:8xxx
> 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/xxxxxx
< 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
$>
```