(https://profile.intra.42.fr/searches) jtian

(https://profile.intra.42.fr)

SCALE FOR PROJECT PISCINE PHP (/PROJECTS/PISCINE-PHP) / DAY 04 (/PROJECTS/42-PISCINE-C-FORMATION-PISCINE-PHP-DAY-04)

You should evaluate 1 student in this team

★

Git repository

```
git@vogsphere-v2.hive.fi:vogsphere/intra-uuid-ac94cda2-ed03-4e12-834
```

Guidelines

You already know how peer2peer evaluation work by now and during this PHP Piscine, it works as any other Piscine projects you have evaluated so far. Take the time to discuss the assignments and to make sure that the evaluated has understood the notions of the day.

Attachments

User sessions (EN subs)

User sessions (EN subs) (https://cdn.intra.42.fr/video/video/1215/piscine-PHP-J04-0-EN-SUBS.mp4)

Subject (https://cdn.intra.42.fr/pdf/pdf/10079/d04.en.pdf)

Preliminaries

There are a lot of exercises, you will have to manage your time accordingly. Execute the simplest, fastest and most efficient tests. Be careful, this evaluation form won't necessarily tell you what to test, you have to come up with your own! As you most likely have created a bunch of them for your own assignments, feel free to re-use them. Once an exercise is wrong, the evaluation must stop but you can still discuss the rest of the assignments.

The basics

- The whole team must be present.
- Only grade the work that is in the student or group's GiT repository.
- You must use Chrome for this defense.
- If there is nothing in the repository, the evaluation stops here. Discuss what went wrong and how to avoid facing the same situation tomorrow.

✓ Yes ✕ No

Ex00

Session

Execute the examples provided in the subject and then make
your own tests.

Check that `index.php` contains a form which uses the GET method, the username field is called `login`, the password field `passwd`, the validation button is called `submit` and its value is `OK`.
If that part is correct, add **3 points**.

Example of output:

```
$> curl -v -c cook.txt 'http://localhost:8080/d04/ex00/index.php'
...
> GET /ex00/index.php HTTP/1.1
[...]
> Host: localhost:8080
> Accept: */*
>
< HTTP/1.1 200 OK
[...]
* Added cookie PHPSESSID="ji22ln1iua24au9eo97ubbego0" for domain localhost, pa
< Set-Cookie: PHPSESSID=ji22ln1iua24au9eo97ubbego0; path=/
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
[...]
<html><body>
<form method="get" action="index.php">
   Username: <input type="text" name="login" value="" />
   <br />
   Password: <input type="password" name="passwd" value="" />
  <input type="submit" name="submit" value="OK" />
</form>
</body></html>
$>
```

Check that the submitted `login` and `passwd` values are saved in the session, then used as default values in the previous form.
If that part is correct, add **2 points**.

Example of output:

```
$> curl -v -b cook.txt 'http://localhost:8080/d04/ex00/index.php?login=sb&passv
[...]
>
GET /ex00/index.php?login=sb&passwd=beeone&submit=OK HTTP/1.1
[...]
>
Host: localhost:8080
> Accept: */*
> Cookie: PHPSESSID=ji22ln1iua24au9eo97ubbego0
>
< HTTP/1.1 200 OK
[...]
<

<
<html><body>
<form method="get" action="index.php">
   Username: <input type="text" name="login" value="sb" />
   <br />
   Password: <input type="password" name="passwd" value="beeone" />
  <input type="submit" name="submit" value="OK" />
</form>
</body></html>
$>
```

Rate it from 0 (failed) through 5 (excellent)

Ex01

Create account

Execute the examples provided in the subject and then make
your own tests.
- Check that a simple account creation works, try to create several new accounts. The created accounts must be stored in `/private/passwd`
If that part is correct, add **2 points**.

```
$> curl -d login=truc -d passwd=titi1 -d submit=OK 'http://localhost:8080/d04/
OK
$> more /path/to/mamp/apache2/htdocs/d04/private/passwd
a:1:{i:0;a:2:{s:5:"login";s:4:"truc";s:6:"passwd";s:128:"2bdd45b3c828273786937
```

Check the error management: it should not be possible to use the same username twice or to save a blank password.
If that part is correct, add **2 points**.

```
$> curl -d login=truc -d passwd=titi1 -d submit=OK 'http://localhost:8080/d04/
ERROR
$> curl -d login=machin -d passwd= -d submit=OK 'http://localhost:8080/d04/ex0
ERROR
$>
```

- If the password is stored in plain text, the exercise is KO and the evaluation stops here.
- Check that the hash algorithm used for the password is not md5 or sha1. If that part is correct, add **1 point**.

Rate it from 0 (failed) through 5 (excellent)

Ex02

Modif account

Execute the examples provided in the subject and then make your own tests.
- Remove the `/private/passwd` file to prevent any problem.
- Create an account using the `create.php` file that must be present in the folder of the ex01.

```
$> rm /path/to/mamp/apache2/htdocs/d04/private/passwd
$> curl -d login=z -d passwd=21 -d submit=OK 'http://localhost:8080/d04/ex01/c
OK
```

- Change the password and check that it has been updated in `/private/passwd` .
If that part is correct, add 2 **points**.

```
$> curl -d login=z -d oldpw=21 -d newpw=42 -d submit=OK 'http://localhost:8080
OK
$> more /path/to/mamp/apache2/htdocs/d04/private/passwd
a:1:{i:0;a:2:{s:5:"login";s:1:"z";s:6:"passwd";s:128:"76936d1331bb87561dead903
```

- Check that an error is returned when providing a wrong password. Also check that it should not be possible to put an empty password.
If that part is correct, add **2 points**.

```
$> curl -d login=x -d oldpw=12 -d newpw=21 -d submit=OK 'http://localhost:8080
ERROR
$> curl -d login=x -d oldpw=42 -d newpw= -d submit=OK 'http://localhost:8080/d
ERROR
```

- If the password is stored in plain text, the exercise is KO and the evaluation stops here.
- Check that the hash algorithm used for the password is not md5 or sha1. If that part is correct, add **1 point**.

Rate it from 0 (failed) through 5 (excellent)

Ex03

Auth

Execute the examples provided in the subject and then make
your own tests.

- Create a new account and test the `login.php` : a wrong login/password combination should fail, a good one should succeed and set the `loggued_on_user` session variable. Check the result with the file `whoami.php` .
If that part is correct, add **3 points**.

```
$> rm /path/to/mamp/apache2/htdocs/d04/private/passwd
$> curl -d login=toto -d passwd=titi -d submit=OK 'http://localhost:8080/d04/e
OK
$> curl -c cook.txt 'http://localhost:8080/d04/ex03/login.php?login=toto&passw
OK
$> curl -b cook.txt 'http://localhost:8080/d04/ex03/whoami.php'
toto
$>
```

- The `login.php` file must include `auth.php` and use the auth function. If that part is correct, add **1 point**.
- The `logout.php` must properly destroy the session cookie. If that part is correct, add **1 point**.

```
$> curl -b cook.txt 'http://localhost:8080/d04/ex03/logout.php'
$> curl -b cook.txt 'http://localhost:8080/d04/ex03/whoami.php'
ERROR
$>
```

Rate it from 0 (failed) through 5 (excellent)

Ex04

42chat - Part 1/2

Execute the examples provided in the subject and then make
your own tests.

- Check that the `index.html` page has a sign-in form, there is a link to create a new account, a link to modify an existing account, if so, add **1 point**.
- Check that once registered, it's possible to sign in and after doing so, the page displays two iframes, if so, add **1 point**.

```
$> curl -d login=user1 -d passwd=pass1 -d submit=OK 'http://localhost:8080/d04
OK
$> curl -d login=user2 -d passwd=pass2 -d submit=OK 'http://localhost:8080/d04
OK
$> curl -c user1.txt -d login=user1 -d passwd=pass1 -d submit=OK 'http://local
[...]
  <iframe name="chat" src="chat.php" width="100%" height="550px"></iframe>
  <iframe name="speak" src="speak.php" width="100%" height="50px"></iframe>
[...]
```

- Check that the bottom iframe contains a form that uses `speak.php` to allow a use to add a new message in the chat. If that part is correct, add **1 point**.

```
$> curl -b user1.txt -d submit=OK -d msg=Bonjours 'http://localhost:8080/d04/e
[...]
$> curl -c user2.txt -d login=user2 -d passwd=pass2 -d submit=OK 'http://local
[...]
$> curl -b user2.txt -d submit=OK -d msg=Hello 'http://localhost:8080/d04/ex04
[...]
```

- Check that the `/private/chat` file is filled with every new message, if so, add **1 point**.

```
$> more /path/to/mamp/apache2/htdocs/d04/private/chat
a:2:{i:0;a:3:{s:5:"login";s:5:"user1";s:4:"time";i:1364287362;s:3:"msg";s:8:"B
```

- Check that the `chat.php` file reads and displays the content of the `/private/chat` with the proper format. If that part is correct, add **1 point**.

```
$> curl -b user2.txt 'http://localhost:8080/d04/ex04/chat.php'
[09:42] <b>user1</b>: Bonjours<br />
[09:43] <b>user2</b>: Hello<br />
$>
```

Rate it from 0 (failed) through 5 (excellent)

42chat - Part 2/2

- Check that after submission of the `speak.php` form, the other iframe (chat.php) reloads, if so, add **1 point**.
- Check that it's possible to log out, after which it's not possible to post any new messages with `speak.php` . If that part is correct, add **1 point**.

```
$> curl -b user1.txt -c user1.txt 'http://localhost:8080/d04/ex04/logout.php'
$> curl -b user1.txt -d submit=OK -d msg=Bonjours 'http://localhost:8080/d04/e
ERROR
$>
```

- Check that **flock** was used in the `speak.php` file to prevent two different users from simultaneously writing a message in the `/private/chat` file. If that part is correct, add **2 points**.
- Check that **flock** was used in the `chat.php` file to prevent reading the `/private/chat` file when it's being modified. If that part is correct, add **1 point**.

Rate it from 0 (failed) through 5 (excellent)

Ratings

Don't forget to check the flag corresponding to the defense

✔ Ok

Empty work Cheat Crash Forbidden function

Conclusion

Leave a comment on this evaluation

Finish evaluation

Terms of use for video surveillance (https://signin.intra.42.fr/legal/terms/1)
Rules of procedure (https://signin.intra.42.fr/legal/terms/4)
Declaration on the use of cookies (https://signin.intra.42.fr/legal/terms/2)
General term of use of the site (https://signin.intra.42.fr/legal/terms/6)
Legal notices (https://signin.intra.42.fr/legal/terms/3)
Privacy policy (https://signin.intra.42.fr/legal/terms/5)