



Piscine PHP

Jour 04

Staff 42 piscine@42.fr

Résumé:

Ce document est le sujet du jour 04 de la piscine PHP de 42.

Table des matières

I	Consignes	2
II	Préambule	3
III	Exercice ex00 : session	4
IV	Exercice ex01 : create_account	7
V	Exercice ex02 : modif_account	9
VI	Exercice ex03 : auth	11
VII	Exercice ex04 : 42chat	13

Chapitre I

Consignes

- Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Seul le travail rendu sur votre dépôt sera pris en compte par la correction.
- Comme lors de la piscine C, vos exercices seront corrigés par vos camarades de piscine ET/OU par une moulinette.
- La Moulinette est très stricte dans sa notation. Elle est totalement automatisée. Il est impossible de discuter de sa note avec elle. Soyez d'une rigueur irréprochable pour éviter les surprises.
- L'utilisation d'une fonction interdite est un cas de triche. Toute triche est sanctionnée par la note de -42.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- Vous ne devez laisser dans votre répertoire aucun autre fichier que ceux explicitement spécifiés par les énoncés des exercices.
- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Votre manuel de référence s'appelle **Google**.
- Pensez à discuter sur le forum. La solution à votre problème s'y trouve probablement déjà. Sinon, vous en serez l'instigateur.
- Lisez attentivement les exemples. Ils pourraient bien requérir des choses qui ne sont pas autrement précisées dans le sujet ...
- Réfléchissez ! Nom d'une pipe !

Chapitre II

Préambule


Vous devez avoir fait les jours de piscine précédent et visionné la vidéo du jour 04, en particulier le passage sur les questions de sécurité des algorithmes de hash.



L'outil que vous utiliserez pour votre serveur à compter d'aujourd'hui est PAMP, développé par le bocal de 42. Celui-ci étant en beta, aidez nous à l'améliorer en remontant les bugs que vous croisez par ticket ou le [forum](#).

Chapitre III

Exercice ex00 : session

	Exercice : 00
session	
Dossier de rendu : <i>ex00/</i>	
Fichiers à rendre : index.php	
Fonctions Autorisées : echo, session_start(), \$_GET, \$_SESSION	
Remarques : n/a	

Créez une page `index.php` qui contient un formulaire permettant de créer/modifier son identifiant et son mot de passe.

- le formulaire devra appeler “`index.php`” et utiliser la méthode “GET”
- le champ identifiant devra s'appeler “login”
- le champ mot de passe devra s'appeler “passwd”
- le bouton submit devra s'appeler “submit” et avoir comme valeur “OK” (si vous ne recevez pas cette valeur dans le champ submit, ne modifiez pas les valeurs stockées dans la session)
- tous les tags du formulaire devront être chacun sur une et une seule ligne (voir l'exemple)
- les champs déjà renseignés précédemment devront être pré-remplis.

Exemple d'utilisation, on commence par tester qu'on reçoit bien le cookie de session au premier accès (il manque volontairement des lignes dans les headers pour la clareté et des bouts d'HTML dans le formulaire pour vous laisser travailler en paix) :

```
$> curl -v -c cook.txt 'http://eXrXpX.42.fr:808x/j04/ex00/index.php'
> GET /ex00/index.php HTTP/1.1
>
< HTTP/1.1 200 OK
* Added cookie PHPSESSID="mfpmngdi4qjbfli03nfgrevu17" for domain j04.local.42.fr, path /, expire 0
< Set-Cookie: PHPSESSID=mfpmngdi4qjbfli03nfgrevu17; path=/
<
<html><body>
<form ...>
  Identifiant: <input ... name="login" value="" />
  <br />
  Mot de passe: <... />
  <input type="submit" ... />
</form>
</body></html>
$>
```

Ensuite on soumet le formulaire et on observe le résultat

```
$> curl -v -b cook.txt 'http://eXrXpX.42.fr:808x/j04/ex00/index.php?login=sb&passwd=beeone&submit=OK'
> GET /ex00/index.php?login=sb&passwd=beeone&submit=OK HTTP/1.1
> Cookie: PHPSESSID=mfpmngdi4qjbfli03nfgrevu17
>
< HTTP/1.1 200 OK
<
<html><body>
<form ...>
  Identifiant: <input ... name="login" value="sb" />
  <br />
  Mot de passe: <... value="beeone" />
  <input type="submit" ... />
</form>
</body></html>
$>
```

Puis on recharge la page sans passer les valeurs dans l'url pour vérifier qu'elles sont toujours présentes


```
$> curl -v -b cook.txt 'http://eXrXpX.42.fr:808x/j04/ex00/index.php'
> GET /ex00/index.php HTTP/1.1
> Cookie: PHPSESSID=mfpmngdi4qjbfli03nfgrevu17
>
< HTTP/1.1 200 OK
<
<html><body>
<form ...>
  Identifiant: <input ... name="login" value="sb" />
  <br />
  Mot de passe: <... value="beeone" />
  <input type="submit" ... />
</form>
</body></html>
$>
```

Dernière étape on enlève le cookie de la requête, et on constate que le formulaire est de nouveau vierge et que PHP nous envoie un nouveau cookie de session

```
$> curl -v 'http://eXrXpX.42.fr:808x/j04/ex00/index.php'
> GET /ex00/index.php?login=sb&passwd=beeone&submit=OK HTTP/1.1
>
< HTTP/1.1 200 OK
< Set-Cookie: PHPSESSID=r7u3bnggoe91negv7aqnjk0q6; path=/
<
<html><body>
<form ...>
  Identifiant: <input ... name="login" value="" />
  <br />
  Mot de passe: <... value="" />
  <input type="submit" ... />
</form>
</body></html>
$>
```

Chapitre IV

Exercice ex01 : create_account

	Exercice : 01
create_account	
Dossier de rendu : <i>ex01/</i>	
Fichiers à rendre : <code>index.html</code> , <code>create.php</code>	
Fonctions Autorisées : <code>echo</code> , <code>hash()</code> , <code>file_get_contents()</code> , <code>file_put_contents()</code> , <code>serialize()</code> , <code>unserialize()</code> , <code>\$_POST</code> , <code>file_exists()</code> , <code>mkdir()</code>	
Remarques : n/a	

Créez une page `index.html` qui contient un formulaire permettant de créer son compte avec un identifiant et son mot de passe. Un compte valide est constitué d'un identifiant et mot de passe non nuls (pas de chaîne vide), sinon renvoyer "ERROR\n". En cas de succès renvoyer "OK\n".

- le formulaire devra appeler "create.php" et utiliser la méthode "POST"
- le champ identifiant devra s'appeler "login"
- le champ mot de passe devra s'appeler "passwd"
- le bouton submit devra s'appeler "submit" et avoir comme valeur "OK" (si vous ne recevez pas cette valeur dans le champ submit, ne créez pas le compte et répondez "ERROR\n")
- tous les tags du formulaire devront être chacun sur une et une seule ligne (voir l'exemple)
- vous devrez stocker les comptes ainsi créés dans `/private/passwd`. Ce fichier doit être un tableau serialisé, voir l'exemple.
- Chaque compte doit être un élément du tableau, et être un tableau lui-même avec un champ "login" contenant l'identifiant et "passwd" contenant le mot de passe hashé.
- vous ne devez pas stocker les mots de passe en "clair" mais "hashés" (voir la vidéo

du jour)


- choisissez bien votre algorithme de hashage (voir la vidéo du jour)
- si l'identifiant soumis existe déjà dans le tableau vous devrez renvoyer "ERROR\n"

Exemple :

```
$> rm ~/http/Piscines/j04/htdocs/private/passwd
$> curl -d login=toto1 -d passwd=titi1 -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex01/create.php'
OK
$> more ~/http/Piscines/j04/htdocs/private/passwd
a:1:{i:0;a:2:{s:5:"login";s:5:"toto1";s:6:"passwd";s:128:"2bdd45b3c828273786937ac1b4ca7908a431019
e8b93c9fd337317f92fac80dace29802bedc33d9259c8b55d1572cb8a6c1df8579cdaa02256099ed52a905d38";}}
$> curl -d login=toto1 -d passwd=titi1 -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex01/create.php'
ERROR
$> curl -d login=toto2 -d passwd= -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex01/create.php'
ERROR
$>
```

Chapitre V

Exercice ex02 : `modif_account`

	Exercice : 02
	<code>modif_account</code>
	Dossier de rendu : <code>ex02/</code>
	Fichiers à rendre : <code>index.html</code> , <code>modif.php</code>
	Fonctions Autorisées : <code>echo</code> , <code>hash()</code> , <code>file_get_contents()</code> , <code>file_put_contents()</code> , <code>serialize()</code> , <code>unserialize()</code> , <code>\$_POST</code>
	Remarques : n/a

Créez une page `index.html` qui contient un formulaire permettant de modifier le mot de passe associé à son compte. L'utilisateur devra fournir son identifiant, son mot de passe actuel (que l'on nommera "ancien" pour éviter toute ambiguïté), et son nouveau mot de passe. On ne modifiera le mot de passe bien évidemment que si le login correspond à un compte déjà existant et que l'ancien mot de passe est bien celui stocké dans nos fichier au moment de la modification. Si l'identifiant n'existe pas, ou que l'ancien mot de passe est erroné, ou que le nouveau mot de passe n'est pas valide, `modif.php` devra renvoyer "ERROR\n" sans autre forme de détail, sinon "OK\n"

- le formulaire devra appeler "`modif.php`" et utiliser la méthode "POST"
- le champ "Identifiant" devra s'appeler "login"
- le champ "Ancien mot de passe" devra s'appeler "oldpw"
- le champ "Nouveau mot de passe" devra s'appeler "newpw"
- le bouton submit devra s'appeler "submit" et avoir comme valeur "OK" (si vous ne recevez pas cette valeur dans le champ submit, ne modifiez pas le compte et répondez "ERROR\n")
- tous les tags du formulaire devront être chacun sur une et une seule ligne (voir l'exemple)
- vous devrez utiliser et modifier le fichier `/private/passwd` qui contient les comptes

créés dans l'ex01


- vous ne devez pas stocker les mots de passe en “clair” mais “hashés” (voir la vidéo du jour)
- choisissez bien votre algorithme de hashage (voir la vidéo du jour)

Exemple :

```
$> rm ~/http/Piscines/j04/htdocs/private/passwd
$> curl -d login=x -d passwd=21 -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex01/create.php'
OK
$> curl -d login=x -d oldpw=21 -d newpw=42 -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex02/modif.php'
OK
$> more ~/http/Piscines/j04/htdocs/private/passwd
a:1:{i:0;a:2:{s:5:"login";s:1:"x";s:6:"passwd";s:128:"fee32be7c00e73eab97a39549d79af73aec87b6fa22a0b
56867a4975fe82344cd9776c6d6dff419e0f2e415c492340bb8329bbfac0c872934df66466c2e0e5d3";}}
$> curl -d login=x -d oldpw=21 -d newpw=42 -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex02/modif.php'
ERROR
$> curl -d login=x -d oldpw=42 -d newpw= -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex02/modif.php'
ERROR
$>
```

Chapitre VI

Exercice ex03 : auth

	Exercice : 03
auth	
Dossier de rendu : <i>ex03/</i>	
Fichiers à rendre : <code>auth.php</code> , <code>login.php</code> , <code>logout.php</code> , <code>whoami.php</code>	
Fonctions Autorisées : <code>echo</code> , <code>hash()</code> , <code>file_get_contents()</code> , <code>file_put_contents()</code> , <code>serialize()</code> , <code>unserialize()</code> , <code>session_start()</code> , <code>\$_GET</code> , <code>\$_SESSION</code>	
Remarques : n/a	

Créez un fichier `auth.php` contenant la fonction suivante :

```
function auth($login, $passwd);
```

Cette fonction devra renvoyer `TRUE` si le couple `login/passwd` correspond bien à un compte dans `/private/passwd` et `FALSE` dans le cas contraire. La variable `$passwd` contient le mot de passe en clair, donc AVANT hashage.

Créez également un fichier login.php prenant en paramètre dans l'url une variable login et une variable passwd. Cette page démarre une session, vérifie la validité du couple login/passwd, et stock dans la session une variable "logged_on_user" contenant :

- soit le login de l'utilisateur qui a soumis un couple login/passwd correct
- soit une chaîne vide dans le cas contraire

Si le couple login/passwd est correct la page doit afficher "OK\n", sinon "ERROR\n". Ce fichier devra se servir de auth.php via un include.

```
$> rm ~/http/Piscines/j04/private/passwd
$> curl -d login=toto -d passwd=titi -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex01/create.php'
OK
$> curl 'http://eXrXpX.42.fr:808x/j04/ex03/login.php?login=toto&passwd=titi'
OK
$>
```


Créez également un fichier logout.php qui ne prend aucun paramètre mais se sert du cookie de session pour supprimer cette dernière. Cette page n'affiche rien.

Créez également un fichier whoami.php qui ne prend aucun paramètre mais se sert du cookie de session pour afficher le login contenu dans variable de session "logged_on_user" suivi d'un "\n". Si cette variable n'existe pas ou contient une chaîne vide, afficher seulement "ERROR\n"

```
$> rm ~/http/Piscines/j04/htdocs/private/passwd
$> curl -d login=toto -d passwd=titi -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex01/create.php'
OK
$> curl -c cook.txt 'http://eXrXpX.42.fr:808x/j04/ex03/login.php?login=toto&passwd=titi'
OK
$> curl -b cook.txt 'http://eXrXpX.42.fr:808x/j04/ex03/whoami.php'
toto
$> curl -b cook.txt 'http://eXrXpX.42.fr:808x/j04/ex03/logout.php'
$> curl -b cook.txt 'http://eXrXpX.42.fr:808x/j04/ex03/whoami.php'
ERROR
$>
```

Chapitre VII

Exercice ex04 : 42chat

	Exercice : 04
42chat	
Dossier de rendu : <i>ex04/</i>	
Fichiers à rendre : <code>index.html</code> , <code>create.html</code> , <code>modif.html</code> , <code>auth.php</code> , <code>create.php</code> , <code>modif.php</code> , <code>login.php</code> , <code>logout.php</code> , <code>speak.php</code> , <code>chat.php</code>	
Fonctions Autorisées : <code>echo</code> , <code>session_start()</code> , <code>header()</code> , <code>hash()</code> , <code>file_get_contents()</code> , <code>file_put_contents()</code> , <code>serialize()</code> , <code>unserialize()</code> , <code>\$_GET</code> , <code>\$_POST</code> , <code>\$_REQUEST</code> , <code>fopen()</code> , <code>flock()</code> , <code>fclose()</code> , <code>time()</code> , <code>date()</code> , <code>date_default_timezone_set()</code> , <code>file_exists()</code> , <code>mkdir()</code>	
Remarques : n/a	

Le but de cet exercice est de créer un chat multi-utilisateur. Vous devrez commencer par reprendre vos fichiers des exercices précédents pour faire l'appli suivante :

- la page d'accueil `index.html` propose un formulaire pour se connecter au chat (le formulaire a pour action `login.php`, en méthode POST, modifiez `login.php` en conséquence), mais aussi un lien pour créer son compte (`create.html`) et un pour modifier son mot de passe (`modif.html`).
- `create.html` est le formulaire de l'ex01. Il appelle `create.php` selon les mêmes règles.
- `modif.html` est le formulaire de l'ex02. Il appelle `modif.php` selon les mêmes règles.
- `create.php` et `modif.php` doivent afficher "OK" en cas de succès ET rediriger l'utilisateur vers la page d'accueil au moyen d'un header "Location :". L'utilisateur devra ensuite s'identifier.
- `login.php` en cas de succès doit afficher une iframe, splittée horizontalement, la partie du haut doit mesurer 550px de la page et contenir "chat.php", la partie du bas doit mesurer 50px et contenir "speak.php"
- `speak.php` permet d'ajouter un message au chat. Il doit vérifier que l'utilisateur est correctement logué. C'est un formulaire qui s'appelle lui-même avec simplement

le champ msg passé en POST. L'identité de l'utilisateur est tiré de la session. speak.php doit remplir un tableau serialisé dans un fichier /private/chat. Chaque élément du tableau est un tableau contenant : "login", "time", "msg". Le champ "login" est bien sûr le login de l'utilisateur qui a posté le message. Le champ time est la date/heure à laquelle le message a été envoyé, au format timestamp. Le champ msg est évidemment le message.

- Attention ! Vous devez locker le fichier avec flock durant toute son utilisation (lecture / écriture) pour ne pas le corrompre en cas d'écriture simultanée par 2 utilisateurs, ni de lecture partielle pendant l'écriture par un autre processus.
- chat.php permet d'afficher le contenu du fichier /private/chat en le formatant. Voir l'exemple (attention à la timezone, <http://www.php.net/manual/en/timezones.php>)

Exemple :

```
$> curl -d login=user1 -d passwd=pass1 -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex04/create.php'
OK
$> curl -d login=user2 -d passwd=pass2 -d submit=OK 'http://eXrXpX.42.fr:808x/j04/ex04/create.php'
OK
$> curl -c user1.txt -d login=user1 -d passwd=pass1 'http://eXrXpX.42.fr:808x/j04/ex04/login.php'
...
<iframe name="chat" src="chat.php" width="100%" height="550px"></iframe>
<iframe name="speak" src="speak.php" width="100%" height="50px"></iframe>
...
$> curl -b user1.txt -d submit=OK -d msg=Bonjours 'http://eXrXpX.42.fr:808x/j04/ex04/speak.php'
...
$> curl -b user1.txt -c user1.txt 'http://eXrXpX.42.fr:808x/j04/ex04/logout.php'
$> curl -b user1.txt -d submit=OK -d msg=Bonjours 'http://eXrXpX.42.fr:808x/j04/ex04/speak.php'
ERROR
$> curl -c user2.txt -d login=user2 -d passwd=pass2 'http://eXrXpX.42.fr:808x/j04/ex04/login.php'
...
$> curl -b user2.txt -d submit=OK -d msg=Hello 'http://eXrXpX.42.fr:808x/j04/ex04/speak.php'
...
$> more ~/http/Piscines/j04/htdocs/private/chat
a:2:{i:0;a:3:{s:5:"login";s:5:"user1";s:4:"time";i:1364287362;s:3:"msg";s:8:"Bonjours";}i:1;a:3:{s:5:
"login";s:5:"user2";s:4:"time";i:1364287423;s:3:"msg";s:5:"Hello";}}
$> curl -b user2.txt 'http://eXrXpX.42.fr:808x/j04/ex04/chat.php'
[09:42] <b>user1</b>: Bonjours<br />
[09:43] <b>user2</b>: Hello<br />
$>
```

Bonus gratuit :

Une fois que vous avez fait tout ça, vous remarquerez que l'iframe chat.php ne se recharge pas toute seule quand vous utilisez speak.php. Voici un bout de javascript à ajouter dans le head de votre speak.php :

```
<script language="javascript">top.frames['chat'].location = 'chat.php';</script>
```

Attention : mettez bien à jour le fichier /private/chat AVANT de renvoyer le code html de votre page pour éviter que le navigateur recharge chat.php trop tôt. Bon courage ;-)