

ECHANGE D'INFORMATIONS PHP

<https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql>

<http://php.net/manual/fr/langref.php>

Open Class Room

SOMMAIRE

Sommaire.....	1
Echanges d'informations en PHP	3
Introduction : passer des variables dans l'URL.....	3
Recherche dans google de : « php variable \$_GET »	3
Exercice.....	3
Installation des fichiers de test.....	4
Méthode Get - URL - Le tableau \$_GET	4
Problématique	4
Exemple-1 : et tableau \$_GET.....	4
Toujours vérifier les informations fournies par l'utilisateur – 1.....	7
Présentation	7
Exemple-2 : href-GET-sans-verifications.....	7
Exemple-3 : href-GET-avec-verifications.....	7
Fonction de gestion des variables	8
Méthode POST - Formulaires - Le tableau \$_POST	9
Rappels sur les formulaires.....	9
Exemple 4 : formulaire-POST.....	10
Exemple 5 : formulaire-GET.....	11
Toujours vérifier les informations fournies par l'utilisateur – 2 : faille XSS.....	12
Never trust user input !	12
Exemples 4 et 5.....	12
Solution : exemple 6.....	12
Fonctions de traitement de chaînes de caractères	13
Se protéger contre les failles : aide en ligne.....	13
TP 1 : Exercice password.....	14
Première version	14
Remarque : crypter les mots de passe.....	14
Deuxième version : monofichier.....	15
Exemple 7 : saisie d'un utilisateur.....	16
Structure du code	16
Conclusion	20
\$_SESSION.....	21
Présentation	21
Utilisation du tableau associatif \$_SESSION	21
Terminer une session	21
TP 2 : Synthèse : site users.....	23
Objectif :	23
Etape 1 : affichage des utilisateurs avec un Header et un Nav.....	23
Etape 2 : ajout du formulaire admin-mot de passe.....	25
Etape 3 : page d'administration.....	26
Etape 4 : vérifier le login.....	28

Etape 5 : enregistrer le nouveau user.....	29
Etape 6 : création d'un fichier index.php – fonction header(URL)	30

Les variables superglobales 31

Présentation	31
Ce sont des tableaux associatifs	31
\$_GET - \$_POST - \$_SESSION	31
\$_COOKIE et \$_REQUEST	31
\$_FILE.....	31
\$_SERVER et \$_ENV	31
Afficher les superglobales : print_r() ou var_dump()	31
\$_FILE	32
Objectif : upload	32
Technique : un formulaire spécial	32
Exemple 8 : \$_FILE	32
Les fichiers	34
Présentation	34
Autoriser l'écriture de fichier sur le serveur : CHMOD.....	34
Syntaxe générale de la manipulation des fichiers	34
Algorithmique des fichiers.....	35
Exemple 9 – tableau-users-Etape-4 : codage avec fichier	36
\$_COOKIES : exemple 10	38
Présentation	38
Principes de codage.....	38
Modifier et supprimer un cookie	39
Précisions.....	39

Bilan..... 40

Construction d'un site et variables de page	40
Echanges entre les pages.....	40
Appel d'une page	41
< a href= « url » >.....	41
< form action= « url » method= « » >	41
header('location : url').....	41

.HTACCESS et .HTPASSWD 42

Présentation	42
.htaccess	42
.htpasswd	42

Edition : mars 2019

ECHANGES D'INFORMATIONS EN PHP

Manuel de Référence du PHP : <http://php.net/manual/fr/langref.php>

Introduction : passer des variables dans l'URL

Recherche dans google de : « php variable \$_GET »

Résultats

L'URL du résultat est la suivante :

https://www.google.fr/#safe=active&q=php+variable+%24_get

Derrière <https://www.google.fr/> on trouve plusieurs éléments séparés par des « & ».
On va trouver :

« **q=php+variable+%24_GET** ».

La variable q contient les informations « php » + « variable » + « %24_get »
Le %24 vient remplacer le \$.

Modification de l'URL

On peut modifier directement l'URL.
On met par exemple POST à la place de GET.
On obtient la page de recherche de « php variable \$_POST »

Simplification

On peut écrire directement dans l'URL :
https://www.google.fr/search?q=php+variable+%24_POST

Conclusion

C'est à l'adresse de l'URL qui donne la page à afficher, avec des paramètres en plus.

Exercice

Google

Chercher « java » directement dans l'URL de google.

Page du cours

Regardez l'URL de la page du cours : [ici](#).
Quelle est la question posée dans l'URL ?
Modifiez la valeur de l'article, par exemple 216. Que constatez-vous ?

Installation des fichiers de test

Les exemples du cours sont dans un fichier zip fournis avec l'article du cours : 01-exemples_GET_POST.zip

Chargez ce fichier et mettez-le dans le dossier Partie_2 du répertoire web « www » du serveur WAMP.

Les exemples sont présentés dans un chapitre en vert.

Les exercices à faire sont présentés dans un chapitre en jaune.

REMARQUE :

Tous les fichiers d'exemples commencent souvent par ces trois lignes :

```
echo '<h1>CODE PHP</h1>';  
highlight_file('fichier.php');  
echo '<h1>RESULTATS</h1>';
```

Ce code affiche deux balises h1 avec CODE PHP puis RESULTATS

La fonction « highlight_file » permet d'afficher le contenu du fichier proposé. Quand on teste le code, on commence par afficher le code. Ça permet de voir le code en même temps que les résultats.

Pour généraliser le code, on écrit : highlight_file(basename(__FILE__));

basename(__FILE__) permet de récupérer le nom du fichier en cours de traitement.

Méthode Get - URL - Le tableau \$_GET

Problématique

Comment faire passer des informations d'une page web à une autre ?

Passer d'une page à une autre peut se faire avec des .

On vient de voir qu'on peut passer des informations sur l'URL.

On va donc pouvoir passer des informations d'une page à une autre de cette manière.

Exemple-1 : et tableau \$_GET

Objectif

L'objectif est, à partir d'une première page (la page d'appel) de faire appel à une deuxième page (la page appelée) en passant des informations qui circuleront à travers l'URL.

Principes : un passage de paramètres

- On utilise un link text
- Dans l'URL, on va mettre les paramètres.
- Dans la page appelée, les paramètres se retrouvent dans un tableau associatif de paramètres : \$_GET

Page appelante : a href

Fichier : index_appel_bonjour.php

C'est une page HTML ou PHP (il faudra l'exécuter via le serveur web) avec un <a href> :

```
<DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>$_GET-Appelant</title>
  </head>

  <body>
    <h1>Test de $_GET</h1>
    <p>Rendez-vous sur
      <a href="01-1-bonjour.php?prenom=Bertrand&nom=Liaudet">
        un bonjour personnalisé avec $_GET
      </a>
    <!-- syntaxe : ?nom de variable=valeur&nom de variable=etc.
-->
    </p>
  </body>
</html>
```

href="bonjour.php?prenom=Bertrand&nom=Liaudet">

- Le nom du fichier : bonjour.php
- un ?
- nom de variable = information (prenom=Bertrand)

Les informations envoyées vont remplir le tableau associatif \$_GET

- **Les variables sont séparées par des « & ». On écrit « & »** car le « & » tout seul crée une confusion en HTML.
- On a une deuxième variable avec sa valeur : nom=Liaudet

Page appelée : \$ _GET

Fichier : bonjour.php

```
<?php
echo '<h1>CODE PHP : </h1>';
highlight_file(basename(__FILE__));
echo '<h1>RESULTATS : </h1>';

echo '<h3>TABLEAU $_GET : </h3>';
echo '<pre>' ; print_r($_GET); echo '</pre>';
?>

<DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>$_GET-Appelé</title>
  </head>

  <body>
    <h1>Page appelée par un &lt;a href="URL"&gt; </h1>
```

```
<h2>Des paramètres sont passées l'URL dans le tableau $_GET :
</h2>
<p> Bonjour
  <?php
      echo $_GET['prenom'] . ' ' . $_GET['nom'];
  ?>
</p>
</body>
</html>
```

Explications

Dans la page appelée, on a un tableau associatif \$_GET qui contient les clés et les valeurs fournies dans la page appelante :

```
array (
    [prenom] => Bertrand
    [nom] => Liaudet
)
```

On peut donc exploiter ce tableau associatif comme on veut dans la page appelée.

< ;a permet d'avoir un <

> ; permet d'avoir un >

code spéciaux HTML : [ici](#).

Précisions

➤ ***URL : on voit les paramètres***

...bonjour.php?prenom=Bertrand&nom=Liaudet
(dans l'adresse les espaces sont traduits par %20)

➤ ***Sécurité – Confidentialité***

L'information qui circule est visible sur le lien URL. Il faut donc faire attention à la limiter à une information non confidentielle.

➤ ***Limite***

En général, les navigateurs n'acceptent pas des url de plus de 256 caractères.
Il faut donc éviter de passer trop d'informations par ce moyen.

Toujours vérifier les informations fournies par l'utilisateur – 1

Présentation

On a vu qu'on pouvait voir dans l'URL les informations fournies par l'utilisateur.
On peut aussi modifier ces informations ou les supprimer directement dans l'URL.
Ca peut être gênant pour le fonctionnement du site.
Il faut donc faire des vérifications.

Exemple-2 : href-GET-sans-verifications

Code sans vérifications

On peut tester les résultats si on donne de mauvaises valeurs en changeant l'URL.
On peut ralentir le serveur ou générer une erreur.

Exemple-3 : href-GET-avec-verifications

Code avec vérifications

On vérifie tous les cas à problème possibles :

```
<?php
echo '<h2>on attend un nombre du programme appelant</h2>' ;

if( !isset($_GET['nombre']) ){
    echo '<p> Pas de nombre reçu !</p>';
}
else{
    // cast: si $_GET n'est pas un entier, on obtient 0
    $nombre=(int)$_GET['nombre'];
    if( $_GET['nombre'] >10000 OR $_GET['nombre'] <=0){
        echo '<p> Le nombre reçu n'est pas valide !</p>';
    }
    else{
        echo '<p> Nous allons compter jusqu'à ' . $nombre. '</p>';
        for($i=1; $i<=$_GET['nombre']; $i++){
            echo('<p>' . $i. '</p>');
        }
    }
}
?>
```

Vérifier que les informations attendues existent : isset(\$var)

isset() est vrai s'il y a quelque chose dans la variable, faux sinon. On pourrait aussi utiliser la fonction empty()

Transtypage (cast) : forcer le type d'une variable : (int)

\$_GET['nombre'] est forcément une chaîne de caractères.

On pourrait tester si elle contient des chiffres avec la fonction ctype_digit().

On peut aussi « caster » directement \$_GET['nombre'] avec un (int). Si \$_GET['nombre'] n'est pas un entier, le cast renvoie 0.

Fonction de gestion des variables

Il existe de nombreuses fonctions qui permettent de faire ces vérifications.

Ce sont particulièrement les fonctions de gestion des variables :

<http://php.net/manual/fr/ref.var.php>

Méthode POST - Formulaires - Le tableau \$_POST

Rappels sur les formulaires

Principes

Les formulaires sont l'outil de base pour échanger des informations avec les visiteurs.

http://www.w3schools.com/html/html_forms.asp

http://www.w3schools.com/tags/att_form_action.asp

http://bliaudet.free.fr/IMG/pdf/HTML_4_Fonctionnalites_avancees.pdf

chapitre 3 : les formulaires, pages 9 à 12 : <form> et <input>

Syntaxe de base

- Exemple :

```
<form action="action.php" method="POST" >
  <p>
    <label for="nom">nom</label>
    <input type="text" name="nom" id="nom" maxlength="10"
placeholder="votre nom">
  </p>
  <p><input type="submit" value="Valider"></p>
</form>
```

- La balise form contient **2 attributs principaux : method et action**.
- La balise form contient des **balises incluses** qui permettent de :
 - caractériser le type de formulaire (champ de saisie, bouton, menu déroulant, etc.)
 - fournir des variables à passer à la page appelée via les tableaux associatifs \$_GET ou \$_POST.

Principes

C'est un passage de paramètres (comme dans un appel de fonction) de page à page.

Les paramètres se retrouvent dans un tableau associatif de paramètres : \$_GET ou \$_POST

Le code HTML de la page appelante remplit le \$_GET ou le \$_POST

Dans la page appelée, on accède aux éléments du tableau associatif \$_GET ou \$_POST

Les attributs de la méthode form

- L'attribut method** permet de choisir la technique de transfert d'informations. Il y en a 2 : GET ou POST.

La méthode GET est celle déjà vue qui fait transiter l'information dans l'URL. On limite l'information à 256 caractères.

La méthode POST permet d'envoyer des gros contenus d'informations et de cacher le transit aux utilisateurs.

On utilise préférentiellement la méthode POST dans les formulaires.

- L'attribut action** dit quelle autre page sera appelée avec cette balise form, quand on valide sur un bouton par exemple, et donc à quelle page on enverra des informations. Ce sera une page php (ou n'importe quel autre page d'un langage serveur) qui sera capable de traiter les informations transmises. On peut aussi faire appel à la même page.

Les balises incluses dans la balise form

- **La balise input** : c'est la principale balise de saisie. Elle fournit en général un champ de saisie. Mais aussi des saisie « typées » : téléphone, mail, couleurs, etc. Elle fournit aussi les boutons de validation.
- **La balise textarea** : fournit une zone de texte à saisir
- **La balise select** : fournit un menu déroulant
- **La balise fieldset** : permet de regrouper dans un cadre plusieurs éléments de saisie. La balise legend joue le rôle d'un label pour ce cadre.
- **La balise label** : elle permet de mettre un texte associé à la zone de saisie.

http://www.w3schools.com/html/html_form_input_types.asp

2 attributs fondamentaux des balises incluses : name et value, pour remplir \$ _GET ou \$ _POST

Pour chaque élément d'un formulaire (un champ de saisie, un bouton, un menu déroulant, etc.), on peut définir un « name » qui sera le nom de la variable qu'on retrouvera dans \$_GET ou \$_POST et qui sera accessible dans la page appelée. Ce « name » est donc une variable passée en paramètre pour la page appelée.

On peut définir aussi une « value » qui sera la valeur de la variable définie par le « name ».

La « value » pourra être fournie à travers une variable avec le php ou n'importe quel langage serveur. A noter que la « value » peut aussi être une valeur qui s'affiche dans le formulaire (par exemple le texte du bouton de validation).

Exemple 4 : formulaire-POST

Page appelante : < form action="actionPOST.php" method="POST" >

Code du formulaire

```
<form action="actionPOST.php" method="POST" >
  <p><label for="prenom">prenom</label>
  <input type="text" name="prenom" id="prenom" placeholder="votre
prenom"></p>
  <p><label for="nom">nom</label>
  <input type="text" name="nom" id="nom" placeholder="votre
nom"></p>
  <p><input type="submit" value="Valider"></p>
</form>
```

Dans l'input :

- Le type définit le type de saisie : ici du texte
- Le name est une clé du tableau associatif \$_POST ou \$_GET, ici \$_POST
- L'id sert pour le CSS
- Si le for du <label> est égal à l'id, un clic sur le label conduit dans l'input pour éviter les confusions.
- placeholder : une info pour la saisie, en grisée

Page appelée : \$ _POST

```
<p> Bonjour
  <?php echo $_POST['prenom'] .' ' . $_POST['nom'] ; ?>
</p>
```

Le tableau \$_POST fonctionne comme le tableau \$_GET : c'est un tableau associatif.
Les name sont des clés (key) du tableau \$_POST : ce sera le cas pour tout les autres formulaires.
Quand on teste, on constate qu'il n'y a pas d'information dans l'URL.

Exemple 5 : formulaire-GET

Page appelante : < form action="actionPOST.php" method="GET" >

Code du formulaire

```
<form action="actionPOST.php" method="GET" >
```

Page appelée : \$_GET

```
<p> Bonjour  
    <?php echo $_GET['prenom'] .' '. $_GET['nom'] ; ?>  
</p>
```

Dans ce cas, les variables transiteront via l'URL. On peut les voir dans l'URL.

Toujours vérifier les informations fournies par l'utilisateur – 2 : faille XSS

Never trust user input !

Principe : envoyer du code

Avec la méthode POST, l'information envoyée est cachée. Mais on peut envoyer n'importe quoi et particulièrement du code HTML ou JavaScript qui nuire à l'utilisation normale du site, en affichant par exemple des contenus inadaptés.

L'envoi de code à la place d'un texte, c'est ce qu'on appelle la faille XSS

Exemples 4 et 5

Les exemples précédents ne sont pas protégés.

On peut saisir une simple balise HTML, ou du code JavaScript, ou des balises HTML complexes (un formulaire !)

➤ *Saisie de balise*

Dans le test précédent, à la place de saisir un simple nom : « Bertrand », on peut saisir

« `<h1>Bertrand</h1>` »

Dans ce cas, la balise `<h1>` sera interprétée et le texte apparaîtra en très gros.

➤ *Saisie de code javascript*

On peut aussi ajouter un code JavaScript : `<script type="text/javascript">alert('Un virus a été détecté ! Veuillez vous rendre sur b্লাউডেট.ফ্রী.ফ্র)</script>` !

Les 2 exemples précédents ne sont pas très grave, mais avec du code JavaScript, un pirate peut récupérer les informations privées d'un utilisateur.

➤ *Rôle des navigateurs*

Safari (apple) protège automatiquement contre les intrusions XSS (et pas firefox).

Firefox 50.0.2, laisse passer le Javascript

Solution : exemple 6

Principes

PHP fournit deux fonctions qui permettent de traiter le code interprétable comme une simple chaîne de caractères.

Fonction htmlspecialchars

La fonction htmlspecialchars permet que le code inséré soit traité comme du texte normal.

`htmlspecialchars($_POST['nom'])`

Ainsi si on saisit : `<h1>Bertrand</h1>`, le site affichera : bonjour `<h1>Bertrand</h1>` : la balise h1 n'aura pas été prise en compte.

<http://php.net/manual/fr/function htmlspecialchars.php>

Transformation du caractère spécial « < » en caractère normal « < »

Quand on regarde le code source d'une page avec un htmlspecialchars, on voit que le « < » qui est interprété par le navigateur comme le début d'une balise est remplacé par « < » qui permet au navigateur d'afficher un <

C'est le cas dans Safari, mais pas dans Firefox ni Chrome.

Fonction htmlentities

La fonction htmlentities est équivalente à la fonction htmlspecialchars en encore plus restrictive : tous les caractères qui ont des équivalents HTML sont traduits.

<http://php.net/manual/fr/function.htmlentities.php>

Fonctions de traitement de chaînes de caractères

htmlspecialchars et htmlentities sont des fonctions de traitement de chaîne de caractères.

Toutes les fonctions : <http://php.net/manual/fr/ref.strings.php>

On y trouve par exemple :

ltrim (suppression des espaces en trop),
ucfirst (upper case pour la première lettre), etc.

Mais aussi :

printf et sprintf

La fonction printf permet d'écrire une chaîne avec des variables « formatées (comme un « echo » mais formaté). La fonction sprintf permet de retourner le résultat dans une chaîne.

Exemple

```
printf(« Le prix est %.2f euros », $prix) ;
```

Le résultat affiché pour le prix le sera avec 2 chiffres après la virgule.

%s : string

%f : float avec plusieurs chiffres après la virgule.

%d : entier

%c : caractère

etc.

Toute la documentation : <http://php.net/manual/fr/function.sprintf.php>

Tester du code : <http://phptester.net>

Se protéger contre les failles : aide en ligne

<https://openclassrooms.com/courses/protegez-vous-efficacement-contre-les-failles-web>

TP 1 : Exercice password

Première version

Objectif

Donner l'accès à une page qui contient des informations cachées pour ceux qui ont le mot de passe. Si on n'a pas le bon mot de passe, on revient sur la page d'accueil.

Méthode

D'abord réfléchir sur papier !

Quelles sont les pages en jeu ? Une, plusieurs ?

Comment les pages communiquent-elles entre elles ?

Première analyse

- Une page d'accueil : un formulaire de saisie d'un mot de passe. En HTML.
- Le mot de passe saisi dans le formulaire est envoyé dans le tableau \$_POST à la page action, en PHP, qui vérifie que le mot de passe est bon avant d'afficher les informations cachées.
- Si le mot de passe est faux, il faut afficher un message d'erreur
- Dans tous les cas, la page appelée permet de revenir sur la page appelante.

Elément de solution

Dans la page action on trouve ce code :

```
if(isset($_POST['password']) AND  
htmlspecialchars($_POST['password'])=='password') {
```

Remarque : crypter les mots de passe

Circulation sur le réseau : https

Le mot de passe apparaît donc dans la page action. Toutefois, il est dans le code php (dans un if), il ne sera donc jamais visible sur navigateur du poste client. C'est donc correctement sécurisé si le serveur est lui aussi sécurisé !

MAIS : si on fait circuler le mot de passe sur le réseau, il y a un risque. C'est le https qui permet de sécuriser la circulation.

Enregistrement dans la BD

En pratique, le mot de passe sera récupéré dans la BD :

MAIS : les mots de passe doivent être enregistré crypté dans la BD.

Pour crypter, on utilise la fonction crypt :

```
$ hashed_password = crypt($password)
```

Pour décrypter, on écrit le code suivant, password étant le mot de passe saisi, hashed_password le mot de passe récupéré dans la BD.

```
if (crypt($password, $hashed_password) == $hashed_password) {
```

```
        echo "Mot de passe correct !";  
    }  
    else{  
        echo "Mot de passe incorrect !";  
    }
```

Deuxième version : monofichier

On souhaite tout coder dans un seul fichier.

La page action est la page d'accueil elle-même.

Il va falloir tester les différents cas au début de la page d'accueil :

- Est-on dans le cas où il faut afficher le formulaire ? `$_POST['password']` n'est pas seté
- Est-on dans le cas où le mot de passe est le bon ? `$_POST['password'] = « password »`
- Est-on dans le cas où le mot de passe est mauvais ? `$_POST['password'] != « password »`

Exemple 7 : saisie d'un utilisateur

Structure du code

On veut obtenir la page suivante :

Affichage des utilisateurs

Nom	Mail	Mot De passe	Age
Sia PEI	ji@gmail.com	jipei	23
Yawei CAI	jawei@yahoo.com	yaweicai	22
Zikeng PENG	zikeng@china.com	zikeng	24
Jiawen LI	jiawen@orange.fr	jiawen	23
Xiaoyu LIU	xiowyu@gmail.fr	liu	22
Olivier TRAN	tran@gmailcom	olivier	21
tootot	tptpt	ttt	2006
toto	tt	ttt	1918
toto	ttt	ttt	1918

Entrez les informations dun utilisateurs

Tous les champs sont obligatoires

Entrez le bon mot de passe pour accéder aux codes secrets

Nom et Prénom

mail

mot de passe

année naissance

Utilisation de fonctions

On va utiliser la fonction « printLesUsersHTML() » qu'on avait défini dans les exercices sur les tableaux d'utilisateur.

En cas d'erreur :

Affichage des utilisateurs

Nom	Mail	Mot De passe	Age
Sia PEI	ji@gmail.com	jipei	23
Yawei CAI	jawei@yahoo.com	yaweicai	22
Zikeng PENG	zikeng@china.com	zikeng	24
Jiawen LI	jiawen@orange.fr	jiawen	23
Xiaoyu LIU	xiowyu@gmail.fr	liu	22
Olivier TRAN	tran@gmailcom	olivier	21
tootot	tptpt	ttt	2006
toto	tt	ttt	1918
toto	ttt	ttt	1918

Entrez les informations dun utilisateurs

Tous les champs sont obligatoires

Entrez le bon mot de passe pour accéder aux codes secrets

Nom et Prénom

nom

mail

mail

mot de passe

password

année naissance

annee

Valider

Tous les champs sont obligatoires

Si on ne saisit pas tous les champs, on affiche le message qu'on voit en jaune

Structure de la page HTML

La page HTML va utiliser 2 variables PHP : \$lesUsers et \$msgErr.

On a donc une page HTML classique, avec un tout petit peu de PHP, en gris :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="style.css" />
    <title>Tableau</title>
  </head>

  <body>
    <?php
      echo '<h2>Affichage des utilisateurs</h2>';
      printLesUsersHTML($lesUsers);
    ?>

    <h2>Entrez les informations dun utilisateurs</h2>
    <p>Tous les champs sont obligatoires</h2>
    <form action="index-tp-users-formulaire.php" method="POST" >
      <fieldset>
        <legend>Entrez le bon mot de passe pour accéder aux
codes secrets</legend>
        <p><label for="nom">Nom et Prénom</label>
          <input type="nom" name="nom" id="password"
placeholder="nom"></p>
        <p><label for="mail">mail</label>
          <input type="mail" name="mail" id="mail"
placeholder="mail"></p>
        <p><label for="mail">mot de passe</label>
          <input type="password" name="password" id="password"
placeholder="password"></p>
        <p><label for="annee">année naissance</label>
          <input type="annee" name="annee" id="password"
placeholder="annee"></p>
          <input type="hidden" name="lesUsers"
value='<?php echo serialize($lesUsers) ?>' >
        <p><input type="submit" name="submit"
value="Valider"></p>
      </fieldset>
    </form>
    <?php
      if (isset($msgErr)){
        echo $msgErr;
      }
    ?>
  </body>
</html>
```

Code PHP

Le code PHP est placé tout du début.

Il y a 4 étapes :

- 1) Le debug : on affiche \$_POST
- 2) On inclut les fonctions
- 3) On initialise les variables : ici \$lesUsers
- 4) On traite le cas où on a validé la saisie.

```
<?php
// 1 : debut
echo '$_POST : '; print_r($_POST);echo '<br>';

// 2 : inclusion des fonctions
include_once("unUser.php");
include_once("lesUsers.php");

// 3 : on initialise le tableau des users
$lesUsers=initLesUsers();

// 4 : si on on a validé le formulaire
if( isset($_POST['submit']) ) {
    // on charge le bon $lesUsers
    $lesUsers = unserialize($_POST['lesUsers']);

    // si la saisie n'est pas complète
    if ( !isset($_POST['nom']) OR ltrim($_POST['nom'])==' '
        OR !isset($_POST['mail']) OR ltrim($_POST['mail'])==' '

        OR !isset($_POST['annee']) OR ltrim($_POST['annee'])==' '
        OR !isset($_POST['password']) OR
        ltrim($_POST['password'])==' '
    ){
        $msgErr='<h2 style="background-color:yellow">Tous les
champs sont obligatoires</h2>';
    }
    // si la saisie est complète, on récupère le $lesUsers et on
ajoute la saisie
    else {

        $lesUsers[]=newUser($_POST['nom'],$_POST['mail'],$_POST['passwo
rd'],$_POST['annee']);
    }
}
?>
```

Conclusion

Avec cette organisation, on a séparé le PHP du HTML. C'est une première étape vers le MVC.

Les pages HTML-PHP vont souvent ressembler à

```
< ?php
    //Partie PHP
    session_start();// si on ouvre une session, c'est fait en 1er
    etc.
?>

<DOCTYPE html>
    <!-- Partie HTML -->

</html>
```

\$_SESSION

Présentation

- Une « session » c'est une instance de fonctionnement d'un programme.
- Elle a un début et une fin.
- Pour un site web, une session, c'est un fonctionnement du site pour un client, c'est-à-dire un navigateur.
- Chaque session contient des informations qui peuvent être accessibles à toutes les pages. Ces informations sont rangées dans le tableau associatif \$_SESSION

Précision technique

Comment fait le serveur pour savoir qu'il s'agit de tel ou tel utilisateur ? C'est le cookie de session qui permet de garantir cela. Ce cookie n'a de sens que pour le serveur web et ne donne aucune information sur l'utilisateur. On y reviendra dans le chapitre sur les cookies.

<https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/session-cookies>

Utilisation du tableau associatif \$_SESSION

Le démarrage d'une session doit se faire avant tout code HTML.

session_start()

Pour pouvoir utiliser la variable \$_SESSION il faut appeler la fonction session_start(). Toutes les pages qui veulent utiliser \$_SESSION doivent appeler session_start()

```
< ?php
session_start();
?>
<DOCTYPE html>
```

L'appel de la fonction doit se faire au tout début de la page, avant le <DOCTYPE html>

\$_SESSION

\$_SESSION s'utilise comme un tableau associatif.

Pour enregistrer une donnée dans \$_SESSION on écrira :

```
$_SESSION['administrateur']='Bertrand';
```

Une fois la clé « administrateur » créée, on peut tester si elle existe :

```
If (isset($_SESSION['administrateur'])){
    echo 'Bonjour'. $_SESSION['administrateur'];
    echo 'Vous êtes connecté et pouvez accéder au back-office';
}
```

Terminer une session

Plusieurs techniques peuvent terminer une session.

1 : unset(\$_SESSION['administrateur'])

Un « unset » ne finit pas vraiment la session, mais supprime une clé dans \$_SESSION. Un test avec « isset » donnera donc une réponse fausse.

2 : \$ SESSION = array()

Cette instruction supprime toutes les variables de session.

3 : patienter ! timeout

On bout d'un certain temps sans activité, le « timeout », les variables de session sont supprimées.

4 : se déconnecter : fermer le client, donc le navigateur.

- Si on quitte le navigateur, les variables de session seront supprimées.
- A noter que fermer la page sans fermer le navigateur ne supprime pas les variables de session.
- A noter aussi que certains sites conservent les variables de session (la connexion typiquement) même quand on a fermé le navigateur. Dans ce cas, ils utilisent les cookies. On y reviendra.

5 : session_destroy();

<http://php.net/manual/fr/function.session-destroy.php>

La fonction bloque l'effet du session_start() sans supprimer les données.

Un nouveau session_start() permet d'y réaccéder.

TP 2 : Synthèse : site users

Objectif :

Consulter les utilisateurs

Se connecter comme admin pour pouvoir ajouter un utilisateur.

Coder en séparant le PHP du HTML.

On reprend les corrigés des exercices tableau user étape 2-fonction et étape 3-tri.

Etape 1 : affichage des utilisateurs avec un Header et un Nav

L'objectif est d'obtenir la page suivante : index-users.php

POST : Array ()

GET : Array ()

URL : /PHP_2018/Partie-3/04-TP-projet-user/indexUsers.php

Site Users

Les Users

Les Users :

Nom	Mail	Mot De passe	Age
Sia PEI	ji@gmail.com	jipei	23
Yawei CAI	jawei@yahoo.com	yaweicai	22
Zikeng PENG	zikeng@china.com	zikeng	24
Jiawen LI	jiawen@orange.fr	jiawen	23
Xiaoyu LIU	xiowyu@gmail.fr	liu	22
Olivier TRAN	tran@gmailcom	olivier	21

tri par nom desc

tri par nom asc

CSS

Pour obtenir ce résultat, le CSS est fourni.

header - nav

Pour qu'il fonctionne, on gère un header HTML qui contient un nav avec le lien « les Users ».
Le nav a un id=nav.

Le mieux est de mettre le header dans un fichier séparé : header.php, le tout dans un dossier « include ».

initLesUsers(), printLesUsers()

Le tableau des utilisateurs est obtenu en utilisant les fonctions initLesUsers() et printLesUsersHTML qui sont fournies dans le fichier lesUsers.php qu'il faut donc inclure.

triParNomDesc(), triParNomAsc()

On ajoute deux boutons : « tri par nom desc » et « tri par nom asc ». On va utiliser les fonctions triParNomDesc() et triParNomAsc() qui sont définies dans le fichier tri.php.

Après avoir utiliser la fonction initLesUsers(), on devra donc vérifier s'il y a un tri à faire.

Affichage debug

Au debut du fichier, on écrit :

```
echo'POST : ';print_r($_POST);echo'<br/>';  
echo'GET : ';print_r($_GET);echo'<br/>';  
echo'URL : ';print_r($_SERVER['PHP_SELF']);echo'<br/>';
```

STRUCTURE DU CODE : très important !!!

Dossiers : on se dote d'un dossier css et d'un dossier include. On ne laisse à la racine que indexUser.php.

Dans le fichier indexUser.php, on commence par du code php :

Le debug

L'initialisation du tableau

Le tri du tableau

Ensuite, on met le code HTML : <!DOCTYPE html> etc.

Etape 2 : ajout du formulaire admin-mot de passe

Objectif :

On ajoute le formulaire de connexion à la page d'administration qui permettra de saisir un nouvel utilisateur.

POST : Array ()
GET : Array ()
URL : /PHP_2018/Partie-3/04-TP-projet-user/indexUsers.php

Site Users

Les Users

admin = admin

password=admin

ok

Les Users :

Nom	Mail	Mot De passe	Age
Sia PEI	ji@gmail.com	jipei	23
Yawei CAI	jawei@yahoo.com	yaweicai	22
Zikeng PENG	zikeng@china.com	zikeng	24
Jiawen LI	jiawen@orange.fr	jiawen	23
Xiaoyu LIU	xiowyu@gmail.fr	liu	22
Olivier TRAN	tran@gmailcom	olivier	21

tri par nom desc

tri par nom asc

Le formulaire

Pour obtenir ce résultat avec notre CSS, on ajoute dans le nav le formulaire « admin, password, ok ».
Le formulaire un id = admin.

Le bouton ok amène sur la page index-admin-users.php, qu'on va écrire à la page suivante.
Les variables du formulaire s'appellent « admin » et « password ».

Etape 3 : page d'administration

L'objectif est d'obtenir la page suivante : index-admin-users.php

POST : Array ([admin] => admin [password] => admin)

GET : Array ()

URL : /PHP_2018/Partie-3/04-TP-projet-user/indexAdminUsers.php

Administration du Site Users

Les Users Admin Les Users

Administrateur connecté. [deconnexion](#)

Les Users :

Nom	Mail	Mot De passe	Age
Sia PEI	ji@gmail.com	jipei	23
Yawei CAI	jawei@yahoo.com	yaweicai	22
Zikeng PENG	zikeng@china.com	zikeng	24
Jiawen LI	jiawen@orange.fr	jiawen	23
Xiaoyu LIU	xiowyu@gmail.fr	liu	22
Olivier TRAN	tran@gmailcom	olivier	21

Entrez les informations d'un utilisateur

Tous les champs sont obligatoires

Entrez le bon mot de passe pour accéder aux codes secrets

Nom et Prénom

mail

mot de passe

année naissance

➤ *header*

On crée d'un nouveau header : header-admin.php (on duplique header.php).

On fait les mises à jour en ajoutant un lien : « Admin Les Users » et en modifiant le formulaire de saisie.

Pour gérer l'utilisateur connecté, on aura enregistré les valeurs admin et mot de passe dans \$_SESSION.

➤ *Fichier index-admin-users.php*

On duplique le fichier index-user.php et on supprime ce qui concerne les tris.

➤ *Formulaire de saisie*

Pour afficher le formulaire de saisie d'un nouvel utilisateur, on utilise **l'exercice de l'étape 3 : 03-exercice-tableau-users-formulaire**

➤ ***Gérer toutes les entrées dans les deux pages***

Dans un premier temps, on ne gère aucune conséquence des boutons autre que l'accès à la page : on ne vérifie pas le mot de passe, on n'enregistre pas le nouveau user.

Etape 4 : vérifier le login

Objectif :

On vérifie de mot de passe.

En cas d'erreur, on affiche une page d'erreur.

En cas de réussite, on fait en sorte que l'utilisateur reste connecté même s'il clique sur le menu « Les Users ».

On gère la déconnexion.

Page d'erreur



The screenshot shows a web page with a blue header containing the title 'Site Users'. Below the header is a white navigation bar with the text 'Les Users' on the left and a login form on the right. The login form has two input fields: 'admin = admin' and 'password=admin', followed by an 'ok' button. Below the navigation bar, a red error message states: 'vous n'avez pas l'autorisation d'afficher cette page'.

Pour obtenir la page d'erreur, on crée une nouvelle page : indexErreur.php

C'est une page HTML qui inclut le header.php et affiche le message d'erreur.

On peut tester directement cette page.

Vérification du login – mot-de-passe

Dans la page d'admin, on vérifie au début si on a bien reçu le bon login – motDePasse. On vérifie que les deux champs sont settés et qu'ils contiennent, par exemple « admin » et « admin ».

Si c'est le cas, on enregistre la valeur de \$_SESSION['admin']. On commence donc par faire un session_start() en début de page.

Ensuite on teste si \$_SESSION['admin'] est setté. Si ce n'est pas le cas, on include indexErreur.php et on quitte la page.

On ajoute un

```
echo'SESSION : ' ; print_r($_SESSION);echo'<br/>'; pour le debug.
```

Gestion de la déconnexion

Dans la page user, on teste si \$_POST['deconnexion'] est setté (on vient du bouton déconnexion). Si c'est le cas, on unset \$_SESSION['admin']

Dans la page d'admin, si on clique sur Les Users, on revient à la page user. Il faudrait rester sur admin. Pour ça, quand on affiche la page admin, il faut sélectionner le header ou le headerAdmin selon que le \$_SESSION['admin'] est setté ou pas.

Etape 5 : enregistrer le nouveau user

Objectif :

On reprend l'exercice de l'étape 3 : 03-exercice-tableau-users-formulaire

Cette exercice permettait d'ajouter un utilisateur dans le tableau en gérant un champs hidden et la fonction serialize().

Solution :

Dans cette exercice, il faut faire circuler le tableau des utilisateurs entre tous les pages, dans tous les cas.

Le plus simple est de le mettre dans \$_SESSION.

On a donc une instruction :

```
$_SESSION['lesUsers'] = serialize($lesUsers);
```

Dans la page indexUser.php

Avant d'initialiser le tableau \$lesUsers, on teste si \$_SESSION['lesUsers'] est setté. Si c'est le cas, \$lesUsers vaudra le contenu de \$_SESSION['lesUsers'].

```
$lesUsers = unserialize($_SESSION['lesUsers']);
```

Dans la page indexAdminUser.php

Pour initialiser le tableau \$lesUsers, on le récupère dans \$_SESSION :

```
$lesUsers = unserialize($_SESSION['lesUsers']);
```

Ensuite on fait l'insertion du nouveau user, s'il y a lieu

Ensuite met à jour le \$_SESSION :

```
$_SESSION['lesUsers'] = serialize($lesUsers);
```

Insertion du nouveau user

On peut reprendre l'exercice de l'étape 3 : 03-exercice-tableau-users-formulaire en l'adaptant un peu.

Etape 6 : création d'un fichier index.php – fonction header(URL)

Objectif

On veut que le projet démarre tout seul quand on arrive dans le dossier.

Idée

On pourrait choisir de renommer la page indexUser.php en index.php et de mettre à jour tous les appels à cette page dans le code. Ce serait long et à refaire si on change de page d'accueil !

Solution : la fonction header(URL)

<http://php.net/manual/fr/function.header.php>

Header permet de demander une URL, un peu comme un href.

On pourra donc passer des paramètres, comme pour toute URL. Ce sera utile quand on passera au MVC.

Elle doit impérativement être utilisée avant tout code HTML.

➤ *Exemple de fichier avec header*

```
<?php
    header('Location: mapage.php');
?>
```

LES VARIABLES SUPERGLOBALES

Présentation

Ce sont des tableaux associatifs

<http://php.net/manual/fr/language.variables.superglobals.php>

Les variables superglobales sont des tableaux associatifs.

Elles sont accessibles à tout moment dans la page, quel que soit le contexte (particulièrement dans les fonctions).

`$_GET` - `$_POST` - `$_SESSION`

On a déjà abordé 3 variables superglobales.

`$_COOKIE` et `$_REQUEST`

`$_COOKIE` est une variable utilisée pour conserver des informations sur a machine client via des cookies. **Cf. chapitre suivant.**

`$_REQUEST` contient par défaut `$_COOKIE`, `$_GET` et `$_POST`.

`$_FILE`

`$_FILES` : est une variable utilisée pour les téléchargements de fichier via HTTP par la méthode POST. **Cf. chapitre suivant.**

`$_SERVER` et `$_ENV`

`$_SERVER` : variables concernant le serveur et le programme.

On y trouve par exemple : `$_SERVER['REMOTE_ADDR']` qui est l'adresse IP du client qui demande la page courante.

`$_ENV` : variables concernant les variables d'environnement du serveur. Cette superglobale est rarement utilisée.

Afficher les superglobales : `print_r()` ou `var_dump()`

Les fonctions `print_r` et `var_dump` affiche les tableaux associatifs en ligne. Avec les balise `<PRE>` `</PRE>`, on peut avoir un affichage avec une ligne par couple key-value.

\$_FILE

Objectif : upload

Transmettre un fichier à un script PHP.

➤ ***Gestion des chargements de fichiers***

- <http://php.net/manual/fr/features.file-upload.php>

➤ ***Chargements de fichiers par méthode POST***

- <http://php.net/manual/fr/features.file-upload.post-method.php>

➤ ***\$_FILES***

- <http://php.net/manual/fr/reserved.variables.files.php>
- <https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/transmettre-des-donnees-avec-les-formulaires>

Technique : un formulaire spécial

- Une balise FORM avec un attribut method="POST" et un attribut enctype="multipart/form-data"
- Une balise INPUT avec un attribut type="file" et un attribut name="la clé du nom du fichier dans \$_FILES"

Exemple 8 : \$_FILE

Le fichier de la balise <FORM>

Il contient le formulaire suivant :

```
<form method="post" action="action.php"
enctype="multipart/form-data">
    Sélectionner un fichier :
    <input type="file" name="userfile">
    <input type="submit" value="Envoyer">
</form>
```

L'attribut name donne le nom de la clé dans le tableau associatif \$_FILES qui aura comme valeur le fichier qu'on aura uploadé.

Le input type= « file » génère une interface de sélection de fichier sur l'ordinateur client.

Le fichier d'action

➤ ***Résultats - \$_FILE***

Dans la page action.php, la variable \$_FILE['userfile'] sera définie, avec les champs suivants : name, type, size, tmp_name, error.

```
Array
(
    [userfile] => Array
        (
            [name] => exo.php
```



```

        [type] => text/php
        [tmp_name] => /Applications/MAMP/tmp/php/phpYpRqlW
        [error] => 0
        [size] => 269
    )
)

```

➤ *move_uploaded_file*

Dans la page action, on va mettre le fichier chargé dans le répertoire qu'on veut.

Le fichier est chargé dans un répertoire temporaire : \$_FILES['userfile']['tmp_name'].

Pour le mettre où on veut, on utilise la fonction :

```

move_uploaded_file (
    $nomDuFichierTmpSurLeServeur,
    $nomDuFichierSurLeSite
)

```

```

$nomFile=$_FILES['userfile']['name'];
echo 'Fichier chargé : '.$nomFile.'<br>';

$tmpDir=$_FILES['userfile']['tmp_name'];
echo 'Fichier stocké temporairement ici : '.$tmpDir.'<br>';

// on upload le fichier temporaire $tmpDir : le nouveau fichier
// s'appelle $nomFile, il se trouve dans le repertoire courant,
// le repertoire du fichier php qui s'exécute
$res=move_uploaded_file($tmpDir, $uploadDir.$nomFile);

if($res) echo 'Upload réussi <br>';
else echo 'L\'upload a échoué <br>';

```

Test

On teste avec un upload d'un fichier qui peut se trouver n'importe où sur la machine du navigateur.

On peut uploader n'importe quel fichier.

Le programme de test affiche le contenu des fichiers texte, des fichiers php et sql, des images.

Les fichiers

Présentation

<http://php.net/manual/fr/ref.filesystem.php>

<https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/lire-et-ecrire-dans-un-fichier-5>

Les fichiers permettent d'enregistrer des informations sur le serveur pour les consulter ou les modifier ultérieurement.

Le fonctionnement général est celui du langage C.

Autoriser l'écriture de fichier sur le serveur : CHMOD

CHMOD est une commande linux pour donner des droits en lecture, écriture et exécution sur les fichiers et les dossiers.

Pour écrire dans un fichier sur le serveur, il faut que le dossier qui contient les fichiers donne le droit aux utilisateurs de lire et d'écrire (et éventuellement d'exécuter). C'est la valeur 777.

La manipulation se fait directement sur le serveur, à travers à logiciel FTP par exemple.

Syntaxe générale de la manipulation des fichiers

Il y a 3 étapes dans la manipulation des fichiers :

1 : Créer et ouvrir un fichier

- fopen()

2 : Les traitements

➤ *Ecrire dans le fichier :*

- fputs() pour écrire une ligne,
- fputc() pour écrire un caractère,
- fprintf() pour écrire une chaîne formatée,
- etc.

➤ *Lire dans le fichier :*

- fgets() pour lire une ligne,
- fgetc() pour lire un caractère, fseek pour se déplacer dans le fichier,
- fscanf() pour lire une chaîne formatée,
- etc.

➤ *Se déplacer dans le fichier,*

- fseek()

4 : Fermer un fichier

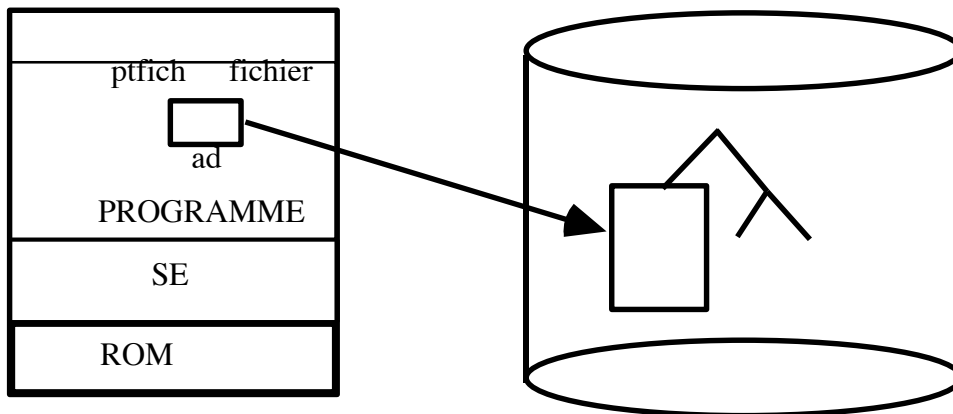
- fclose()

Le type fichier

Ce type est essentiellement un pointeur qui donne l'adresse d'un élément du fichier (d'abord l'adresse du premier élément du fichier).

Mais il va aussi contenir d'autres informations : ce sera en fait une structure dont l'un des champs contient l'adresse de l'élément courant du fichier.

Il faut bien comprendre que dans le programme, il y a une variable habituelle (une structure comme n'importe quelle structure) qui contient l'adresse d'un élément d'un fichier.



Ouvrir un fichier et le créer si nécessaire : fopen

<http://php.net/manual/fr/function.fopen.php>

```
$ptfich = fopen("nomDuFichierSurLeDisque", "mode d'ouverture");
```

Le nom du fichier sur le disque peut être absolu ou relatif.

Il y a plusieurs mode d'ouverture du fichier : r, r+, w, w+, a+, etc.

« r » permet la lecture seule, « r+ » la lecture et l'écriture, « w » l'écriture seule avec création si le fichier n'existe pas, « w+ », la lecture et l'écriture avec création si le fichier n'existe pas, etc.

Voir la documentation de référence pour le détail.

Fermer un fichier : fclose

Il faut fermer les fichiers qu'on utilise pour permettre à d'autres de les utiliser.

```
$booléen fclose ($ptFich) ;
```

lire dans un fichier

```
$string fgets ($ptFich) ;
```

fgets permet de lire la ligne courante.

Il y a d'autres fonctions fgets, fscanf, fread, etc.

Voir par exemple : <http://php.net/manual/fr/function.fscanf.php> et son chapitre voir aussi.

écrire dans un fichier

```
$int fwrite ($ptFich, $string) ;
```

fwrite écrit la \$string passée en paramètre. fputs est un alias de fwrite.

La fonction retourne le nombre d'octets écrits. 0 = false, si 0 octet écrit.

La fonction fprintf est une autre fonction permettant d'écrire dans un fichier avec un système de formatage.

<http://php.net/manual/fr/function.fwrite.php>

Se déplacer dans un fichier

On peut se déplacer dans un fichier d'un nombre d'octets précisés. Si le nombre est négatif, on part de la fin du fichier.

```
fseek ($ptFich, nombreDOctets);
```

<http://php.net/manual/fr/function.fseek.php>

Repérer la fin du fichier

On peut vérifier à tout moment si le \$ptfich est arrivé à la fin du fichier.

```
$booléen feof ($ptFich);
```

<http://php.net/manual/fr/function.feof.php>

Exemple 9 – tableau-users-Etape-4 : codage avec fichier

On reprend l'étape 2 ou l'étape 3 et on va rajouter des fonctions de traitements de fichier.

On se dote d'une fonction d'écriture du tableau \$lesUsers dans un fichier :

function ecrireLesUsersDansFichier(\$nomFichier, \$lesUsers)

Et d'une fonction de lecture d'un tableau \$lesUsers à partir d'un fichier :

function lireLesUsersDansFichier(\$nomFichier, &\$lesUsers)

Fonction ecrireLesUsersDansFichier

```
function ecrireLesUsersDansFichier($nomFichier, $lesUsers){
    $ptfich=fopen($nomFichier, "w");

    foreach ($lesUsers as $unUser){
        foreach ($unUser as $element){
            echo $element.' -- ';
            fprintf($ptfich, "%s -- ", $element);
        }
        echo '<br/>' ;
        fprintf($ptfich, "\n"); //\n tout seul pour le saut de ligne
    }
    fclose($ptfich);
}
```

La fonction commence par un fopen qui retourne un \$ptfich : chercher l'usage de fopen. Pour écrire dans un fichier, on passe « w » au fopen.

Ensuite on écrit dans le fichier avec un fprintf : chercher l'usage de fprintf.

Pour passer à la ligne, on écrira : fprintf(\$ptfich, "\n");

Enfin, un fclose ferme le fichier.

Fonction lireLesUsersDansFichier

```
function lireLesUsersDansFichier($nomFichier, &$lesUsers){
    $ptfich=fopen($nomFichier, "r");
    while (!feof($ptfich)){
        // on lit 5 param car prénom et nom sont 2 chaînes séparées
        // on met les -- car ils sont dans le fichier
    }
}
```

```

    // On considère que les 3 champs ne contiennent pas d'espaces
    $lectureOK=fscanf($ptfich, "%s %s -- %s -- %s -- %s -- ",
        $prenom, $nom, $mail, $motDePasse, $anneeNaissance);
    if(!$lectureOK) { // si on a un pb. de lecture, on sort
        return 0;    // on peut gérer ça plus finement !
    }
    $unUser=newUser($prenom.' '.$nom, $mail,
        $motDePasse, $anneeNaissance);
    printUnUser($unUser);
    $lesUsers[]=$unUser;
}
fclose($ptfich);
return 1;
}

```

La fonction commence par un fopen en mode « r ».

On lit dans le fichier avec un fscanf : on récupère ainsi tous les champs d'un utilisateur.

On crée l'utilisateur et on le met dans \$lesUtilisateurs.

Enfin, un fclose ferme le fichier.

\$ _COOKIES : exemple 10

Présentation

Les cookies sont des petits fichiers (<4ko) stockés par le navigateur sur le poste client pour conserver des informations pour une prochaine fois.

Ils peuvent ainsi garder une trace de la visite d'un site pour la visite suivante.

On les appelle aussi témoin de connexion.

Seul le domaine qui a créé le cookie peut le relire.

Les utilisateurs des navigateurs peuvent désactiver les cookies pour un domaine particulier.

Selon les législations en vigueur, l'utilisateur doit ou pas donner son accord pour recevoir des cookies (c'est le cas en France).

<http://php.net/manual/fr/reserved.variables.cookies.php>

<https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/session-cookies>

Principes de codage

On peut enregistrer les cookies : fonction setcookie.

On peut lire les cookies : dans la variable \$ _COOKIE.

Enregistrer un cookie sur la machine client : setcookie

<http://php.net/manual/fr/function.setcookie.php>

setcookie permet d'enregistrer un cookie sur la machine client.

Un setcookie doit être écrit avant le code HTML : avant le DOCTYPE.

Par exemple, si on arrive dans une page suite à la validation d'un formulaire POST avec une variable nom et une variable prenom dans \$ _POST, on peut enregistrer 2 cookies : le nom et le prénom de l'utilisateur :

```
<?php
if(isset($_POST['nom'])
    setcookie('nom',$_POST['nom'] ,time()+3600*24*7);
if(isset($_POST['prenom'])
    setcookie('prenom',$_POST['prenom'] ,time()+3600*24*7);
?>
<!DOCTYPE html>
...
```

Le cookie en général au minimum 3 caractéristiques :

- un nom
- une valeur
- une date d'expiration : aujourd'hui (time)+ des secondes : ici une semaine.

Accéder au cookie dans le code : \$ _COOKIE

Si on veut maintenant afficher un « bonjour nom prenom » quand l'utilisateur arrive sur notre site et qu'il est déjà venu et qu'on a enregistré les cookies de son nom et de son prenom, on utilise le tableau associatif \$ _COOKIE.

Dans le header, par exemple, on pourra récupérer le nom et le prenom pour afficher un bonjour ou autre chose :

```
if (isset($_COOKIE['nom'])) {  
    $nom = COOKIE['nom'] ;  
}  
if (isset($_COOKIE['nom'])) {  
    $prenom = COOKIE['nom'] ;  
}  
// on peut ensuite faire ce qu'on veut de $nom et $prenom
```

➤ **Sécuriser les cookies**

Attention : les cookies sont modifiables par le client ! Il faut donc les considérer comme des informations non sûres !

Il faut donc utiliser la fonction htmlspecialchars pour protéger le contenu du cookie :

```
if (isset($_COOKIE['nom'])) {  
    $nom = htmlspecialchars(COOKIE['nom']);  
}  
if (isset($_COOKIE['nom'])) {  
    $prenom = htmlspecialchars(COOKIE['nom']);  
}  
// on peut ensuite faire ce qu'on veut de $nom et $prenom
```

Modifier et supprimer un cookie

Pour modifier et supprimer un cookie, on reprend la fonction setcookie.

Pour la modification, on garde le même nom et on peut changer la valeur et/ou la date d'expiration.

Pour la suppression, on garde le même nom et on passe une date d'expiration avant la date et heure actuelle : dont on passe un « time() – 1000 » par exemple.

Précisions

Consulter les cookies sur le navigateur

Sur le navigateur, en affichant les outils de développement ou les ressources de la page on peut accéder aux ressources ou au stockage et visualiser les cookies.

On va voir notre cookie.

On voit aussi le cookie PHPSESSID : PHP SESS ID : c'est un cookie créé automatiquement par PHP avec comme valeur un ID de session qui permet de retrouver l'utilisateur.

Autres paramètres du setcookie

<http://php.net/manual/fr/function.setcookie.php>

Il y aussi 4 autres paramètres à la fonction setcookie qui permettent de sécuriser les cookies.

Le dernier, httponly, s'il est à true peut limiter le risque de faille XSS en interdisant que le Javascript puisse accéder au cookie.

```
setcookie('nom', 'Bertrand Liaudet', time()+3600*24*7, null, null,  
false, true);
```

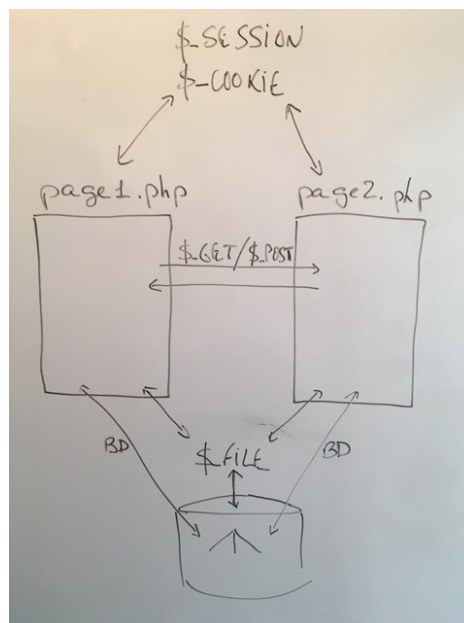
BILAN

Construction d'un site et variables de page

- Un site est constitué de « pages » qui correspondent chacune à un « main ».
- Chaque page gère ses variables de pages, locales ou globales.
- Les variables d'une page ne sont pas visibles dans une autre page.
- Les pages peuvent échanger des informations par les variables superglobales, par les fichiers et par la BD.

Echanges entre les pages

- Quand une page appelle une autre page, elle peut fournir de l'information dans les tableaux \$_GET ou \$_POST.
- Les pages peuvent lire et écrire dans le tableau \$_SESSION qu'elle vont donc partager.
- Les pages peuvent écrire des cookies et lire ces cookies dans le tableau \$_COOKIES.
- Les pages peuvent lire et écrire dans des fichiers qu'elles peuvent partager.
- Les pages peuvent lire et écrire dans une base de données qu'elles peuvent partager.
- Les pages peuvent lire des informations concernant le serveur et le client dans les tableaux \$_ENV et \$_SERVER



Appel d'une page

On peut appeler une page de 3 façons :

< a href= « url » >

Dans ce cas, l'information circule forcément par un **\$_GET** et directement sur l'url.

https://www.w3schools.com/tags/tag_a.asp

< form action= « url » method= « » >

Dans ce cas, l'information circule par **\$_GET** ou **\$_POST** selon la valeur de l'attribut method dans la balise form (GET ou POST).

http://www.w3schools.com/tags/att_form_action.asp

header('location : url')

Dans ce cas l'information circule forcément par un **\$_GET** et directement sur l'url.

<http://php.net/manual/fr/function.header.php>

.HTACCESS ET .HTPASSWD

Présentation

Les fichiers .htaccess et .htpasswd permettent de protéger des dossiers et des fichiers sur un site web. Le « . » au début du nom font que ces fichiers sont cachés sur les OS Linux.

<https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/protéger-un-dossier-avec-un-htaccess>

<http://www.siteduzero.com/tutoriel-3-14649-protéger-un-dossier-avec-un-htaccess.html>

<http://www.siteduzero.com/tutoriel-3-35126-le-htaccess-et-ses-fonctionnalites.html>

Sur la sécurité des sites web

Par exemple :

<https://legissa.ovh/internet-se-protéger-des-pirates-et-hackers.html>

.htaccess

Le fichier .htaccess permet d'interdire l'accès à un dossier.

On met le fichier .htaccess dans le dossier concerné.

On peut préciser beaucoup de choses dans ce fichier.

Au minimum, pour bloquer l'affichage du contenu :

```
Options - Indexes
```

.htpasswd

Le fichier .htpasswd permet de protéger un dossier par un mot de passe.

Il est associé à un fichier .htaccess particulier

Contenu du fichier .htaccess

```
AuthName "Page d'administration protégée"
AuthType Basic
AuthUserFile "/C:/wamp/www/admin/.htpasswd"
Require valid-user
```

- AuthName est le texte qui répond à une tentative d'accès.
- AuthUserFile est le chemin vers le fichier .htpasswd. Ce chemin peut être absolu ou relatif selon les hébergeurs ! En local : absolu. En général : absolu.

Pour connaître le chemin absolu d'un fichier : `<?php echo realpath('monFichier.php'); ?>`

Contenu du fichier .htpasswd

```
user1:password1
```

```
user2:password2
```

Sous WAMP

Le mot de passe est spécifié en clair :

```
bertrand:motDePasseBertrand  
admin:motDePasseAdmin
```

Sur un serveur en ligne

Le mot de passe doit être crypté :

```
bertrand:$1$MEqT//cb$hAvid.qmmSGFW/wDlIfQ81  
admin:$1$/lgP8dYa$sQNXcCP47KhPlsneRIZo00
```

➤ *Crypter le mot de passe*

```
<?php echo crypt('monpasswd'); ?>
```