le cnam
Grand Est

Développement web (2)

Sites dynamiques
et développement côté serveur

NFA017 (4 ECTS)

Séance 03
Le langage PHP
2022

Le Cnam en Grand Est - Centre régional
4 avenue du Docteur Heydenreich CS 65228 F 54052 Nancy Cedex
tél +33 (0)3 83 85 49 00 www.cnam-grandest.fr

Le plus grand soin a été apporté à la réalisation de ce support pédagogique afin de vous fournir une information complète et fiable. Cependant, le Cnam Grand-Est n'assume de responsabilités, ni pour son utilisation, ni pour les contrefaçons de brevets ou atteintes aux droits de tierces personnes qui pourraient résulter de cette utilisation.

Les exemples ou programmes présents dans cet ouvrage sont fournis pour illustrer les descriptions théoriques. Ils ne sont en aucun cas destinés à une utilisation commerciale ou professionnelle.

Le Cnam ne pourra en aucun cas être tenu pour responsable des préjudices ou dommages de quelque nature que ce soit pouvant résulter de l'utilisation de ces exemples ou programmes.

Tous les noms de produits ou autres marques cités dans ce support sont des marques déposées par leurs propriétaires respectifs.

Ce support pédagogique a été rédigé par Alexandre ECUVILLON et mis en page par Simon MAHIEUX, enseignants au Cnam Grand-Est.

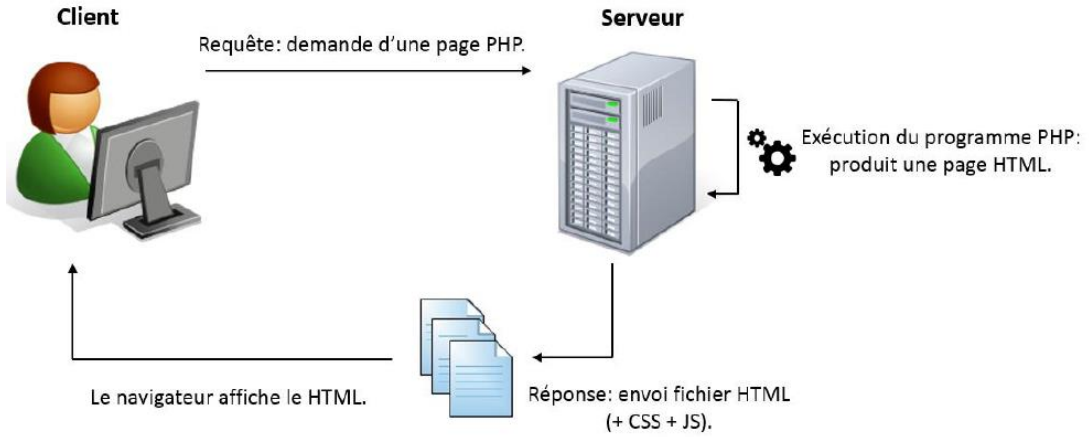
Table des matières

1. Introduction

1.1 Un langage côté serveur

PHP, qui signifie *PHP Hypertext Preprocessor* (l'acronyme est donc récursif), est un langage de programmation qui fonctionne du côté serveur. Jusqu'à maintenant, les langages que nous avons étudiés (HTML, CSS et Javascript) étaient tous interprétés du côté du client. Avec ces langages, le serveur Web n'a aucun rôle (fort) à jouer : il stocke simplement les fichiers HTML (contenant éventuellement du code Javascript) puis les envoie au client lorsque celui-ci les réclame. Il s'agit juste d'un échange de données : les pages HTML, les images, les feuilles de styles, …

L'objectif va maintenant être d'amener le serveur à produire de manière dynamique un document HTML (ou autre…), afin de permettre la prise en compte de données fournies par l'utilisateur côté client. Un fois ce document HTML dynamique généré, c'est lui qui va être envoyé au navigateur. Lorsqu'on utilise un programme PHP, le serveur n'envoie pas le fichier PHP au client mais le résultat du programme PHP. Le serveur et le client ne possèdent donc pas le même document.



Il faut bien comprendre que sur le serveur, le document PHP est un programme écrit en langage PHP alors que sur le client le document PHP n'est en fait que le résultat de l'exécution du programme PHP sur le serveur - c'est-à-dire un document HTML contenant éventuellement du Javascript et/ou du CSS Alors qu'avec le langage HTML, l'internaute peut visualiser la page HTML exactement comme elle a été créée, avec le langage PHP l'internaute ne peut pas consulter le code source du programme PHP.

Il est possible de créer un fichier qui ne contient que du code PHP mais il est également possible d'insérer du code PHP dans un document HTML. Le code PHP est délimité par les balises <?php et ?>.

Voici un premier script PHP :

```
<html>
   <body>
      <?php print"<b>Voici un premier programme PHP !</b>"; ?>
   </body>
</html>
```

Ce script PHP génère le document HTML suivant :

```
<html>
   <body>
      <b>Voici un premier programme PHP !</b>
   </body>
</html>
```

Vous pouvez d'ores et déjà repérer la fonction *print* qui permet d'écrire une chaîne de caractères dans le document HTML (un peu comme *document.write* en Javascript).

Le code PHP et le code HTML peuvent même être fortement imbriqués mais attention à la lisibilité du programme. Voici un exemple :

```
<?php $manuel="http://www.php.net/manual/fr/"; ?>
<html>
   <body>
      <a href="<?php print $manuel; ?>" target="_blank">Consulter le manuel de PHP en
ligne</a>
   </body>
</html>
```

Ce script PHP génère le document HTML suivant :

```
<html>
   <body>
      <a href="http://www.php.net/manual/fr/" target="_blank">Consulter le manuel de PHP en
ligne</a>
   </body>
</html>
```

Vous pouvez ici constater comment on réalise une affectation dans une variable et la manière dont on utilise son contenu (notez bien le $ qui doit être placé devant son nom). Au passage vous pourrez noter l'adresse du manuel officiel du langage PHP, qui s'avère bien utile…

Remarquez que l'on utilise le terme script PHP, programme PHP ou code PHP pour désigner la même chose. De la même manière, le côté client, le navigateur et l'utilisateur font référence à des notions similaires.

PHP versus ASP, ASP.net, JSP, etc. ?

Notons dès à présent que PHP est un langage de programmation côté serveur … parmi d'autres. On peut citer pour principaux concurrents : ASP et son évolution plus récente ASP.net (monde Microsoft), JSP (plateforme Java), Ruby, Python, etc. Chaque langage a ses propres particularités, ses propres forces et faiblesses. Mais globalement les grands principes restent similaires.

1.2 A quoi sert PHP ?

Alors qu'avec HTML et CSS on ne peut produire que des sites statiques, PHP permet la réalisation de sites dynamiques. Une page statique propose toujours le même contenu à l'internaute. A l'inverse, une page dynamique affiche un contenu changeant sans l'intervention d'un développeur. Typiquement, un site dynamique sera utilisé dès lors qu'il faut présenter des informations susceptibles de changer. De nos jours, la très grande majorité (si ce n'est la quasi-totalité) des sites sont dynamiques !

PHP versus HTML ?
Non : PHP ne remplace pas HTML et pire, il sera généralement utilisé en complément. D'une certaine façon, PHP « s'ajoute » donc au HTML, CSS et Javascript.

Les usages de PHP sont nombreux ; citons quelques applications courantes :

- La manipulation de bases de données.
- Le traitement de formulaires soumis par le client.
- La génération / lecture de fichiers.
- L'envoi d'emails.
- L'utilisation de cookies et de sessions.

1.3 Histoire de PHP

Le langage PHP fut créé en 1994 par Rasmus Lerdorf - à la base, il s'agissait d'une bibliothèque C utilisée pour conserver les traces des visiteurs qui consultaient son site. Cette bibliothèque se complexifia jusqu'à être publiée en 1995. Peu de temps après, deux étudiants (Andi Gutmans et Zeev Suraski) redéveloppèrent le coeur du langage jusqu'à aboutir à une nouvelle version du moteur, appelé Zend Engine, et la naissance de la version 4 de PHP. L'usage de PHP s'est alors grandement répandu jusqu'à devenir de nos jours un des langages principaux du web.

PHP est salué pour ses performances, sa fiabilité et l'extensibilité du moteur. On peut, sans trop de controverses, le qualifier de langage simple et souple. On lui reproche souvent d'être incohérent, désorganisé et/ou archaïque. Quoi qu'il en soit, sa popularité ne souffre d'aucun débat.

La version la plus récente de PHP est la version 7, bien que la version 5 soit encore très employée. Notons que la version 6 n'a jamais existé, PHP est directement passé de la version 5 à la version 7.

1.4 PHP, Apache et MySQL

Les sites écrits en PHP se reposent très fréquemment sur Apache et MySQL, au point que ce triplet semble parfois (à tort) indissociable.

1.4.1 Le serveur web : Apache

Le protocole HTTP implique qu'une page HTML soit délivrée par une machine : un serveur. Sur cette machine s'exécute un logiciel (nommé également « serveur web ») dont le rôle est de traiter les requêtes et d'envoyer les réponses. Dans le cadre de PHP, il faut que ce serveur web soit capable d'exécuter du code PHP pour générer les ressources demandées. Apache est le serveur web le plus souvent utilisé pour héberger des sites PHP, mais il en existe d'autres : nginx, lighttpd, IIS, etc.

En production, les machines clientes et le serveur sont bien entendu différentes. Cependant dans le cadre du développement, on installera fréquemment le serveur web sur la même machine que celle sur laquelle on développe et lance le navigateur.

1.4.2 La base de données : MySQL

Une base de données se définit comme « un ensemble de données interrogeable par son contenu ». Concrètement, il s'agit d'un outil capable de stocker des données en rapport avec un thème ou une activité, ces informations étant structurées. Ce type de logiciel offre des fonctionnalités qui permettent la collecte, le stockage, le retraitement et l'utilisation des données. Dans le monde PHP, MySQL est un type de base de données très fréquemment utilisé ; mais d'autres existent : Oracle, PostgreSQL, SQL Server ...

L'interaction avec une base de données en PHP fait l'objet d'un cours ultérieur dédié.

1.4.3 Les packages WAMP, LAMP, etc.

LAMP est un acronyme pour « Linux, Apache, MySQL, PHP » qui désigne un ensemble de composants « nécessaires » pour construire un environnement PHP. Des variantes existent pour les différentes plateformes : WAMP pour Windows, MAMP pour Macintosh, etc.

Divers logiciels proposent ainsi sous une forme packagée ces outils, ce qui simplifie grandement leur installation, en particulier pour préparer un environnement de développement. On pourra ainsi citer des packages comme XAMPP, WampServer ou EasyPHP - que vous pouvez télécharger sur le web.

2. Syntaxe et fondements du langage PHP

PHP est un langage de programmation à part entière. Sa syntaxe, classique, est proche de celle du C ou de Java. Nous allons passer en revue de façon synthétique l'ensemble des commandes et structures de contrôles qui peuvent être utilisées.

2.1 Un exemple commenté

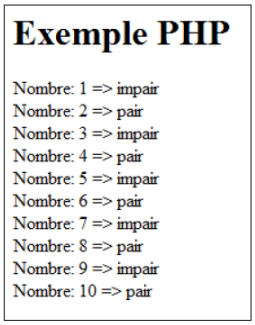
Voici un script PHP :

```
<!DOCTYPE html>
<html>
   <body>
      <h1>Exemple PHP</h1>
      <?php
         $txt = "";
         // On compte de 1 à 10
         for ($i=1; $i<=10; $i++)
         {
            $txt = $txt . "Nombre: " . $i . " => ";
            if ($i%2 == 0)
            {
               $txt = $txt . "pair<br>";
            }
            else
            {
               $txt = $txt . "impair<br>";
            }
         }
         echo $txt;
      ?>
   </body>
</html>
```

L'exécution de ce script produit la page HTML suivante :

```
<!DOCTYPE html>
<html>
   <body>
      <h1>Exemple PHP</h1>
      Nombre: 1 => impair<br>
      Nombre: 2 => pair<br>
      …
      Nombre: 10 => pair<br>
   </body>
</html>
```

Dont le rendu est similaire à cela :



Ce script PHP imbrique du HTML classique et du code PHP. Le code PHP est délimité entre les balises <?php et ?>. On commence par déclarer une variable nommée « txt » (notez la présence obligatoire du $), à laquelle on affecte une chaîne de caractères vide (le typage de la variable n'est donc pas explicite). Suit immédiatement un commentaire, qui commence par « // ».

Puis avec l'instruction « for », on boucle sur une variable « i » pour des valeurs de 1 à 10. A chaque itération de cette boucle, on commence par ajouter du texte à notre variable « txt » en y concaténant (ajoutant) un texte fixe puis la valeur de la variable « i » ; la concaténation se fait avec l'opérateur « . ».

On teste ensuite si le restant de la division de « i » par 2 vaut zéro (modulo), ce qui revient à tester si « i » est pair. Si « i » est pair, on ajoute le texte « pair » sinon on ajoute le texte « impair ».

Une fois sorti de la boucle, on écrit le contenu de la variable « txt » dans le fichier HTML via la fonction native « echo » (qui est très similaire à « print »).

2.2 Syntaxe de base

Les instructions se terminent par un point-virgule : *instruction* ;

Les expressions sont placées entre parenthèses : *(expression)* et renvoient une valeur booléenne *true* ou *false*.

Les commentaires sur une ligne sont placés après le symbole # ou // et les commentaires sur plusieurs lignes sont délimités par les symboles /* et */.

Lorsqu'une instruction peut afficher un message d'erreur, il est possible de brider cet affichage en utilisant le symbole @ : @instruction ;

2.3 Les types de données

Il n'est pas nécessaire (ni possible !) de déclarer le type des données que l'on manipule, celui-ci est affecté dynamiquement par PHP. Cependant il est intéressant de connaître les différents types existants :

- Les booléens : ils expriment une valeur soit vrai (*true*) soit fausse (*false*).
- Les entiers : il s'agit d'un nombre appartenant à l'ensemble des entiers naturels que l'on peut exprimer en base décimale (c'est presque toujours le cas) ($a = 1234 ;), en base octale en utilisant le préfixe 0 ($a = 02322 ;) ou en base hexadécimale en utilisant le préfixe 0x ($a = 0x4D2 ;).
- Les réels : il s'agit d'un nombre à virgule que l'on peut exprimer avec la notation simple ($a = 1.234 ;) ou avec la notation scientifique ($a = 1.2e3 ;).
- Les chaînes de caractères : elles peuvent être exprimées de manière simple à l'aide du symbole ` (simple cote) pour préciser une chaîne de caractères dans laquelle les variables et les séquences d'échappement ne seront pas interprétées ou à l'aide du symbole " (guillemet) qui permet d'interpréter les séquences d'échappement et les variables. Notez que l'opérateur « . » permet de réaliser la concaténation de chaînes de caractères.
- Les tableaux : il ne s'agit pas réellement d'un type de données mais d'un ensemble de variables. Pour créer un tableau, vous disposez du mot-clé array. Les tableaux sont indexés à partir de 0 et il faut utiliser les crochets pour accéder à l'indice d'un tableau ($a[3] = 5 ;).
- Les tableaux associatifs : vous avez la possibilité de ne pas utiliser les indices numériques mais des chaînes de caractères ($a["toto"] = 5 ;).

Note : le typage en PHP est faible mais il est possible à partir de PHP 7 d'activer un typage strict. Ceci cependant se limite au typage des paramètres de fonction.

```
// Exemple de typage strict PHP 7
declare(strict_types=1);
function additionner(int $foo, int $bar)
{
   echo $foo + $bar;
}
echo additionner(1, 1);
```

2.4 Les opérateurs

Les opérateurs sont des fonctions spéciales qui fournissent les opérations primitives du langage, à l'instar des opérations arithmétiques.

2.4.1 Les opérateurs arithmétiques

Les opérateurs arithmétiques permettent de manipuler les valeurs : addition, soustraction, etc.

Symbole	Exemple	Commentaire
+	`$a + $b`	additionne `$a` et `$b`
-	`$a - $b`	soustrait `$b` de `$a`
*****	`$a * $b`	multiplie `$a` par `$b`
/	`$a / $b`	divise `$a` par `$b`
%	`$a % $b`	reste de la division de `$a` par `$b` (modulo)

2.4.2 Les opérateurs de comparaison

Les opérateurs de comparaison permettent de tester deux valeurs entre elles et sont utilisés dans les structures conditionnelles.

Symbole	Exemple	Expression vraie (true) si ...
==	`$a == $b`	`$a` est égal à `$b`
===	`$a === $b`	`$a` est égal et du même type que `$b` (PHP4 seulement)
!=	`$a != $b`	`$a` est différent de `$b`
<	`$a < $b`	`$a` est inférieur à `$b`
>	`$a > $b`	`$a` est supérieur à `$b`
<=	`$a <= $b`	`$a` est inférieur ou égal à `$b`
>=	`$a >= $b`	`$a` est supérieur ou égal à `$b`

2.4.3 Les opérateurs logiques

Ils permettent les opérations booléennes : OU, ET, etc.

Symbole	Exemple	Expression vraie (true) si …
&&	`$a && $b`	`$a` et $b retournent *true*
and	`$a and $b`	`$a` et $b retournent *true*
||	`$a || $b`	`$a` ou $b retourne *true*
or	`$a or $b`	`$a` ou $b retourne *true*
xor	`$a xor $b`	`$a` ou $b (exclusivement) retourne *true*
!	`! $a`	`$a` retourne *false*

2.4.4 Les opérateurs d'incrémentation / décrémentation

Comme en C, on dispose des opérateurs d'incrémentation (++) et décrémentation (--) qui sont des raccourcis syntaxiques. Ils s'utilisent avant ou après le nom de la variable.

Exemple	Effet
$a++;	Retourne `$a` puis l'incrémente de 1.
++$a;	Incrémente `$a` de 1 puis retourne `$a`.
$a--;	Retourne `$a` puis décrémente de 1.
--$a;	Décrémente `$a` de 1 puis retourne `$a`.

2.4.5 Opérateurs de chaines de caractères

On concatène (« ajoute ») des chaînes de caractères avec l'opérateur « . ». Attention, en PHP on n'utilise pas le « + » !
Exemple :

```
$s1 = "Pierre";
$s2 = "Bonjour, " . $s2;
```

2.5 Les structures de contrôle

La plupart des programmes ne suivent pas un déroulement linéaire : ils ont besoin de s'adapter à des cas particuliers, d'utiliser des conditions, de répéter plusieurs fois une même portion de code. Les principales structures de contrôles sont les conditions (structures alternatives) et les boucles (structures répétitives).

2.5.1 if / else / elseif

Le « if » est la forme le plus simple de structure conditionnelle.

```
if ( expression )
{
…
}
```

On peut la complexifier avec un *else* pour le cas où la condition testée est fausse (false) :

```
if ( expression )
{
   …
}
else
{
   …
}
```

Enfin, on peut imbriquer plusieurs conditions successives avec « elseif » :

```
if ( expression )
{
   …
}
elseif ( expression )
{
   …
}
else
{
   …
}
```

2.5.2 switch / case

Le « switch » est une structure conditionnelle un peu particulière qui permet de comparer une variable à plusieurs valeurs.

```
switch ( expression )
{
   case ( expression ) :
      …
      break ;
   case ( expression ) :
      …
      break ;
   default :
      …
      break ;
}
```

2.5.3 while

La « while » (« tant que » en algorithme) est la structure itérative la plus classique :

```
while ( expression )
{
   …
}
```

2.5.4 do / while

Le « do … while » est très similaire au « while », mais l'instruction est toujours exécutée une première fois avant de tester la condition :

```
do
{
   …
}
while ( expression ) ;
```

2.5.5 for

La boucle « for » est une alternative au « while », le plus souvent employée lorsque le nombre d'itérations est connue à l'avance :

```
for ( initialisation ; test ; incrémentation )
{
   …
}
```

2.5.6 foreach

Le « foreach » est une forme spécifique de boucle qui permet via une syntaxe raccourcie de traiter chaque élément d'un tableau.

```
foreach ( tableau as variable )
{
   …
}
```

2.6 Les fonctions

Les fonctions sont des sous algorithmes admettant des paramètres et retournant un seul résultat.

La déclaration d'une fonction en PHP est réalisée à l'aide du mot-clé *function* suivi du nom de la fonction et éventuellement d'une liste de paramètres. En ce qui concerne les paramètres, il n'y a aucun typage explicite, il suffit donc de donner un nom de variable au paramètre. Si la fonction retourne un paramètre, il faut utiliser la fonction *return*.

Voici un exemple :

```
function f($x)
{
   return 2+$x;
}
function afficher($texte)
{
   print "Voici le texte à afficher : " . $texte;
}

$a = f(5);
afficher("a=".$a);
```

2.7 Les variables prédéfinies

L'environnement de PHP contient des variables prédéfinies qui peuvent être utilisée à n'importe quel moment pour obtenir diverses informations.

A titre d'exemple, nous pouvons détailler la variable $_SERVER… Cette variable offre différentes informations concernant le serveur Web :

- *$_SERVER["SERVER_NAME"]* : il s'agit du nom du serveur.
- *$_SERVER["DOCUMENT_ROOT"]* : il s'agit du chemin du répertoire racine du serveur.
- *$_SERVER["PHP_SELF"]* : il s'agit du nom complet (chemin inclus) du script PHP.
- *$_SERVER["REMOTE_ADDR"]* : il s'agit de l'adresse IP de la machine qui demande le résultat du script.
- *$_SERVER["HTTP_USER_AGENT"]* : il s'agit d'une description du navigateur du client.

D'autres variables sont disponibles… Nous allons, par la suite, découvrir celles qui sont en rapport avec divers mécanismes de PHP comme l'accès aux cookies, la manipulation de formulaires, etc.

3. Utilisations classiques de PHP

Ce chapitre aborde divers usages « habituels » de PHP.

3.1 Inclure des portions de page

Il n'est pas rare lors de la conception d'un site web d'avoir besoin de réutiliser un « morceau de page ». C'est par exemple typiquement le cas lorsque toutes les pages du site partagent une structure commune, par exemple un menu toujours identique positionné sur la partie gauche.

En HTML, on est obligé de dupliquer ce morceau de page autant de fois que nécessaires avec tous les problèmes que cela implique (en particulier le fait de devoir modifier toutes les pages à la moindre modification). En PHP, on peut factoriser cet élément commun avec le mécanisme d'inclusion, un peu comme on le ferait avec une fonction.

L'inclusion se pratique avec la fonction « include(nom_de_la_page_à_inclure) ». Voici un exemple : « Menu.php » est le morceau de page qu'on veut réutiliser ; « Page.php » est une des pages qui référence notre menu.

Menu.php :

```
<nav id="menu">
   <div class="menu">
      Menu:
      <ul>
         <li><a href="Page1.php">Lien 1</a></li>
         <li><a href="Page2.php">Lien 2</a></li>
         <li><a href="Page3.php">Lien 3</a></li>
      </ul>
   </div>
</nav>
```

Page.php :

```
<!DOCTYPE html>
<html>
   <body>
      <?php include("Menu.php"); ?>
      ... contenu de la page ...
   </body>
</html>
```

3.2 Gestion des cookies

Un cookie est un fichier stocké sur le disque-dur de l'usager et qui permet au serveur de reconnaître le visiteur la prochaine fois qu'il revient sur le site de telle façon à, par exemple, connaître ses préférences et ainsi éviter qu'il ait à les ressaisir.

Le « problème » de ces cookies est qu'ils stockent des informations qui vous concernent. Par exemple, lorsque vous vous connectez à un site commercial, celui-ci va mémoriser vos choix de façon à dresser votre profil, puis stocker ces données dans un cookie. Selon le site sur lequel vous vous connectez cela sera à votre avantage ou non. De manière générale, ces cookies améliorent l'ergonomie des sites, mais posent la question de la confidentialité des données…

Ainsi, un site de vente en ligne peut récolter des informations sur les préférences des utilisateurs par le biais d'un questionnaire, afin de leur proposer ultérieurement des articles pouvant les intéresser. Par exemple, en sachant si vous êtes un homme ou une femme il pourra vous aiguiller directement au rayon approprié pour vous faire économiser du temps (et augmenter ses chances de vendre …), et s'il sait que vous êtes amateur de tennis il vous proposera les derniers articles en la matière. En revanche, vous devriez refuser de céder toute information personnelle à un site qui ne vous inspire pas confiance.

En réalité un cookie n'a rien de dangereux en soi car c'est le navigateur qui le gère en écrivant dans un fichier des paires clés/valeurs. Par ailleurs, les données stockées dans un cookie sont envoyées par le serveur, ce qui signifie qu'il ne peut en aucun cas contenir des informations sur l'utilisateur que celui-ci n'a pas donné.

Pour créer un cookie vous devez utiliser la fonction *setcookie* en passant en paramètre son nom, sa valeur ainsi que sa date d'expiration comme dans l'exemple suivant :

```
<?php
   setcookie("rubrique", "informatique", time()+24*3600); // expire dans 24h
?>
```

Pour réutiliser ce cookie ou un autre, vous disposez de la variable *$_COOKIE*. Il s'agit d'un tableau associatif qui contient la valeur des cookies.

```
<?php
   print "Voici la valeur du cookie \"rubrique\".$_COOKIE["rubrique"];
?>
```

3.3 Gestion des formulaires

3.3.1 Rappel : requêtes HTTP en GET et POST

Les méthodes GET et POST sont les méthodes de requêtes HTTP les plus couramment utilisées. Elles impliquent des différences importantes quant à la façon dont les paramètres des pages sont transmis :

- Avec la méthode GET, les paramètres et leurs valeurs sont passés directement dans l'URL du programme appelé.
- Avec la méthode POST, les paramètres ne sont pas concaténés à l'URL, ils sont passés dans le corps de la requête HTTP.

3.3.2 Formulaire GET

La variable *$_GET* contient l'ensemble des champs envoyés par un formulaire utilisant la méthode GET. Dans votre formulaire HTML, le nom donné à un champ (balise <input>) permet de récupérer sa valeur.

Voici un exemple de formulaire GET :

```
…
<form method="get" action="script.php">
   Votre prénom : <input name="prenom" type="text"><br>
   Votre nom : <input name="nom" type="text"><br>
</form>
…
```

Et voici comment récupérer les champs du formulaire dans le fichier script.php :

```
…
$prenom = $_GET["prenom"] ;
$nom = $_GET["nom"] ;
…
```

3.3.3 Formulaire POST

La variable *$_POST* contient l'ensemble des champs envoyés par un formulaire utilisant la méthode POST. Dans votre formulaire HTML, le nom donné à un champ (balise <input>) permet de récupérer sa valeur.

Voici un exemple de formulaire POST :

```
…
<form method="post" action="script.php">
   Votre prénom : <input name="prenom" type="text"><br>
   Votre nom : <input name="nom" type="text"><br>
</form>
…
```

Et voici comment récupérer les champs du formulaire dans le fichier script.php :

```
…
$prenom = $_POST["prenom"] ;
$nom = $_POST["nom"] ;
…
```

A partir d'un formulaire POST, il est également possible d'envoyer des fichiers sur le serveur Web. Pour cela vous devez utiliser un type de balise particulier, il s'agit du type « file ». La balise form doit également utiliser un paramètre particulier pour l'encodage du fichier avec le paramètre « enctype » qui doit être positionné à « multipart/form-data ». Pour brider la taille du fichier il est possible de définir un champ caché (type « hidden ») de nom MAX_FILE_SIZE et dont la valeur correspond au nombre maximum d'octets à accepter.

Dans le programme PHP, le fichier peut être manipulé par la variable *$_FILES* de la manière suivante :

- *$_FILES["fichier1"]["tmp_name"] :* nom temporaire du fichier après la copie sur le serveur.
- *$_FILES["fichier1"]["name"] :* nom du fichier du côté client.

Voici un exemple de formulaire d'envoi de fichier :

```
…
<form enctype="multipart/form-data" action="script.php" method="post">
   <input type="hidden" name="MAX_FILE_SIZE" value="1024">
   <input type="file" name="fichier1"><br>
   <input type="submit" value="Télécharger">
</form>
…
```

Voici le script PHP qui traite le fichier envoyé :

```
…
<?php
   $rep_dest="../fichiers/";
   $ok=move_uploaded_file($_FILES['fichier']['tmp_name'],$rep_dest .
$_FILES['fichier']['name']);
   if ($ok)
   {
      print "Le fichier \"".$_FILES["fichier1"]["name"]."\" a bien été téléchargé.";
   }
   else
   {
      print "Echec du téléchargement.";
   }
?>
…
```

La fonction *move_uploaded_file* permet de déplacer un fichier uploadé dans un répertoire spécifique car par défaut tous les fichiers uploadés sont placés dans un répertoire temporaire.

3.4 Gestion des sessions

Une session est un mécanisme qui permet au serveur de reconnaître un utilisateur tout au long de sa navigation sur un site Web. Il est ainsi possible d'attacher des variables à une session et de les retrouver ou les exploiter sur chacune des pages visitées. Après un certain temps d'inactivité, une session est automatiquement fermée par le serveur Web.

Pour créer une session il faut utiliser la fonction *session_start*. Attention, cette fonction permet également de récupérer sur une page une session existante. Elle doit donc être présente sur chaque page qui souhaite pouvoir utiliser les variables de session. Il faut appeler cette fonction avant « d'écrire » la page HTML, c'est-à-dire avant d'envoyer le moindre octet de résultat au navigateur car elle modifie les en-têtes HTTP. Par la suite, pour stocker une variable en session, il suffit de l'affecter à *$_SESSION*. Les différentes variables de session peuvent être ensuite consultées et modifiées à l'aide de la variable *$_SESSION*. Une variable de session peut être retirée de la session à l'aide de la commande unset. Pour détruire la session, il faut utiliser la fonction *session_destroy*. Elle détruit l'ensemble des variables de session.

Voici un exemple de formulaire :

```
<html>
   <body>
      <form method="post" action="script.php">
         Votre nom : <input type="text" name="nom"><br>
         Votre prénom : <input type="text" name="prenom"><br>
         <input type="submit" value="Valider">
      </form>
   </body>
</html>
```

Voici le script PHP (script.php) qui stocke des données du formulaire en session :

```
…
<?php
   session_start();
   $_SESSION["nom"]=$_POST["nom"];
   $_SESSION["prenom"]=$_POST["prenom"];
?>
…
```

Voici un deuxième script du même site (script2.php), qui lit les paramètres stockés en session :

```
…
<?php
   session_start();
   print"Voici votre nom : ".$_SESSION["nom"]." Et votre prénom ".$_SESSION["prenom"]."<br>";
?>
…
```

3.5 Fonctions intégrées

PHP propose de nombreuses fonctions prédéfinies intéressantes.

3.5.1 Chaînes de caractères

Plusieurs fonctions sont disponibles pour manipuler les chaînes de caractères. Voici une liste non exhaustive de ces fonctions, suivie d'une courte description :

- *strlen* : cette fonction vous permet de récupérer la taille d'une chaîne de caractères
- *strcmp* : cette fonction permet de réaliser une comparaison binaire de deux chaînes.
- *strpos* : cette fonction permet de trouver la position d'un caractère dans une chaîne.
- *strstr* : cette fonction permet de trouver la première occurrence d'une chaîne dans une autre.
- *strtolower* : cette fonction retourne la même chaîne en minuscules.
- *strtoupper* : cette fonction retourne la même chaîne en majuscules.
- *substr* : cette fonction permet d'extraire une sous-chaîne.
- *trim* : cette fonction supprime les espaces (ou d'autres caractères) en début et fin de chaîne.
- *ucfirst* : cette fonction met une majuscule au premier caractère de la chaîne.

3.5.2 Tests sur les variables

Voici quelques fonctions permettant de réaliser des tests sur les variables :

- *gettype* : cette fonction permet de récupérer le type d'une variable.
- *isset* : cette fonction permet de savoir si une variable est définie (si elle a reçu une affectation).
- *unset* : cette fonction permet de détruire une variable.
- *is_array* : cette fonction permet de savoir si une variable est un tableau.
- *is_bool* : cette fonction permet de savoir si une variable est un booléen.
- *is_int* : cette fonction permet de savoir si une variable est un entier.
- *is_real* : cette fonction permet de savoir si une variable est un réel.

- *is_string* : cette fonction permet de savoir si une variable est une chaîne de caractères.

3.5.3 Manipulation de fichiers

Diverses fonctions permettent de manipuler les fichiers sur le serveur :

- *basename* : permet de récupérer le nom d'un fichier sans le chemin.
- *copy* : permet de réaliser une copie de fichier.
- *fopen* : réalise l'ouverture d'un fichier ou sa création.
- *fclose* : réalise la fermeture d'un fichier.
- *fwrite* : permet d'écrire des données dans un fichier.
- *fread* : permet de lire des données dans un fichier.
- *fgets* : permet de lire une chaîne dans un fichier.
- *feof* : indique si on se trouve à la fin du fichier.

De nombreuses autres fonctions intégrées existent, vous les découvrirez en pratiquant la programmation PHP. N'hésitez surtout pas à vous documenter sur le web (ex : le site www.php.net) pour rechercher des fonctions PHP ou pour obtenir des informations sur le fonctionnement d'une fonction précise que vous connaissez déjà.

4. Aller plus loin

PHP a beaucoup évolué au cours de ses différentes versions. Le langage a la particularité de proposer aussi bien la possibilité de coder de façon intuitive – mais au détriment de l'état de l'art, que de façon structurée et en adéquation avec les bonnes pratiques actuelles. Le web regorge de ressources à ce propos.

Si vous êtes déjà à l'aise avec les bases de PHP, une excellente ressource pour progresser sur les techniques modernes (parfois complexes à appréhender) est le site « PHP the right way », qui a le mérite d'être traduit en français : http://eilgin.github.io/php-the-right-way/