NFA017
2016

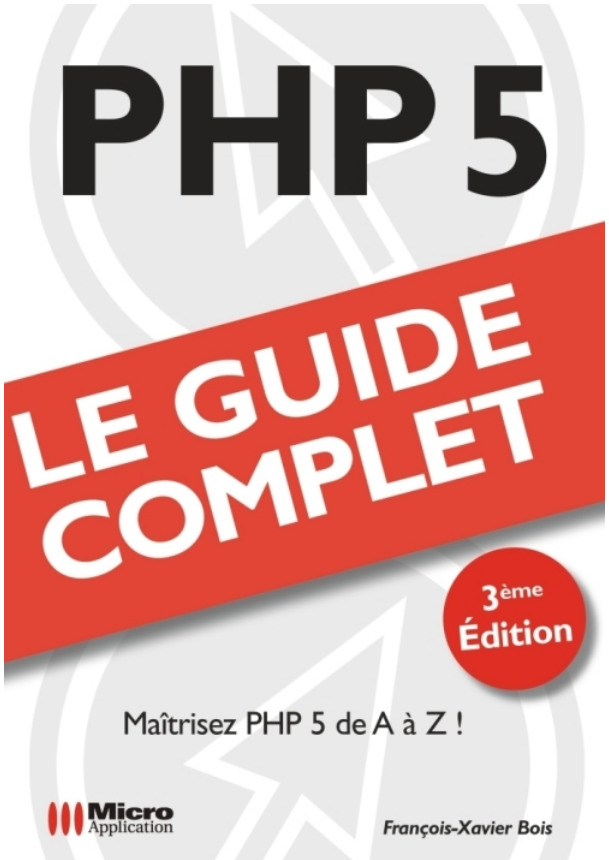
Guide de survie: PHP

CNAM le 19 février 2016

O. Pons S. Rosmorduc

Quelques documents de references sur PHP

- Le manuel php :
  `http://www.php.net/manual/fr/index.php`
- Un tutoriel "pour grands debutants" :
  `http://php.developpez.com/cours/`
- Un petit livre :



Principe général de fonctionnement

- page HTML *classique* : **statique**
  - Contenu : page html stockée sur le serveur
  - A chaque requête : envoyer exactement le contenu au serveur
- page PHP : **dynamique** :
  - contenu : page php (code php + code html) stockée sur le serveur web
  - A chaque requête :
    1. interpréter le contenu (fonction des paramètres de la requête).
    2. envoyer au serveur le code html produit

Exemple Trivial

- Fichier sur le serveur :

```
<p>
  Il est actuellement <?php print date("H:i:s"); ?>.
</p>
```

- Code généré et envoyé au client :

```
<p>
  Il est actuellement 8:45:34.
</p>
```

Généralités

- **Extension du fichier** : .php
- **Délimiteurs** : `<?php ... ?>`
- **Les Commentaires** :

```
/* premier style de
        commentaire sur plusieurs lignes! */

// l'autre style sur une seule ligne

#  encore un autre sur une ligne
```

- **fin/séparation d'instruction** : ;
- **Affichage** :`echo` ou `print`

Généralités (suite)

Variables :

- précédées de “$”
- pas de typage explicite, dépendant contexte d’exécution
- les types scalaires
  - **entiers** : int
  - **réels** : float, double
  - **chaînes de caractères** : string
  - **booleen** : boolean
- les types composés
  - **les tableaux** : array
  - **les objets** : object
- les types spéciaux
  - **NULL**
  - **ressource**

Exemples de variables

```
<?php
$i = 42; // entier
$pi = 3.14159; // réel
$nom = "olivier"; // chaîne
$saluer  = "bonjour $nom"; // chaîne avec interprétation
                           //des variable
$prix = 'le $us est bas'; // chaîne sans interprétation
$vraiOuFau =false; //boolean

echo "i est un "; echo gettype($i);echo " valant ";
echo $i ; echo "<br/>";
echo "pi vaut $pi <br/>";
echo $saluer;
?>
<br/>
<?php
echo $prix; ?>
```

Exemples de variables (suite)

S'affichera :

```
i est un integer valant 42
pi vaut 3.14159
bonjour olivier
le $us est bas
```

Les opérateurs

- Opérateurs usuels :

```
+ - * / == && || > < >= <= !=
++ : pre ou post incrémentation
-- : pre ou post décrémentation
+= : incrémentation + affectation
-= : décrémentation + affectation
```

- Opérateurs sur les chaînes

```
. : concatenation
.= : concatenation + affectation
```

Exemples de tableau

- **Initialisation des tableaux :**
  `array(val1,...,valn)` ou élément par élément.

```
$tab = array("un", "2", "III") ; //tableau de chaine
$tab1=array(1,3,4);                 //tableau d'entier
$tab3 =array(1,"trois",5.5);        //tableau polymorphe

$tab4[0]=1; //indice de 1er élément est 0
$tab4[1]=2;
$tab3[3]=6; //ajout dans la case d'indice 3  de tab3
$tab3[]="neuf"; //ajout dans la derniere case
```

- **accès aux elements** :

```
echo $tab3[1];//affiche le 2em element de tab3 :trois
echo $tab3[2];//affiche le 3em element de tab3 :5.5
```

Tableaux (suite)

- **savoir si une variable est un tableau :** `is_array($tab)`
- **affichage de debug** `print_r`

  `print_r($tab3)`

  affiche :

```
Array(
    [0] =>1 [1] => eee [2] => 5 [3] => 6 [4] => neuf
      )
```

Tableau associatifs

tableaux dont les indices sont des clefs.

```
$tab = array("clef1"=>"val 1", "clef2"=>2);
$tab2["fraise"]="rouge";
$tab2["pomme"]="verte";

echo $tab2["pomme"];//affiche verte
```

Pour le debug,s'affichent aussi avec `print_r`.

```
 print_r($tab2)
```

s'affiche

```
Array
(
    [fraise] => rouge
    [pomme] => verte
)
```

Debug

Pour inspecter tout type de variable : utiliser `var_dump`
Avec les declaration suivantes :

```
$a=100;
$b="je suis le plus beau !";
$c=[30,"trente",30.0,[1,2,3]];
$d=["key1"=>"valeur 1",2=>"deux","trois"=>3];
```

`var_dump($a);`, Produit

```
int(100)
```

`var_dump($b);` Produit :

```
string(22) "je suis le plus beau !"
```

Debug (suite)

```
var_dump($c);
```

```
array(4) {
  [0]=>
  int(30)
  [1]=>
  string(6) "trente"
  [2]=>
  float(30)
  [3]=>
  array(3) {
    [0]=>
    int(1)
    [1]=>
    int(2)
    [2]=>
    int(3)
  }
}
```

Debug (suite)

```
var_dump($d);
```

```
array(3) {
  ["key1"]=>
  string(8) "valeur 1"
  [2]=>
  string(4) "deux"
  ["trois"]=>
  int(3)
}
```

structures de contrôle

- **if**(c1) {inst si c1 }
  elsif (c2) {inst si c2} ...
  else{inst}
  Exemple :

```
$h=date("H");//recupere l heure

if ($h<8){
    echo "il est trops tot...";
    }
    elsif ($h<12) {
    echo "c'est le matin";
    } else {
    echo "c'est l aprem";
}
```

structures de contrôle

- **switch**(val){
  case v1:inst v1;
  ...
  case vn:inst vn ;
  default : inst par defaut}

```
switch($h){
  case "1":echo "une heure du mat et quelques";break;
  case "2":echo "II";
  case "3":echo "2 ou 3";
  case "4":echo "2 ou 3 ou 4";break;
  case "5":echo "5";break;
  default:
      echo " plus de 6h";
}
```

- (cond)?valSiVrai:valSiFaux

```
echo ($h<12)?"matin":"apres midi";
```

boucles

- **while**(cond){instr}

```
<?php
//tant que cond est vrai
$i = 1;
while ($i <= 10) {
    echo $i++;  /* La valeur affiche est $i
      avant l'incrémentation  (post-incrémentation)  */
}
?>
```

- **do**{inst}**while**(cond)

```
<?php
//au moins une fois meme si cond est faux
$i = 0;
do {
    echo $i;
} while ($i > 0);
?>
```

boucles (suite)

- **for** `(expr1; expr2; expr3){inst}`

```
<?php
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
?>
```

Parcours des tableaux

- **foreach**(tableau **as** $value){inst}

```
<?php
$arr = array(1, 2, 3, 4);
$i=0;
foreach ($arr as $value) {
   echo "l'element d'indice $i++ vaut: $value;
}
?>
```

- **foreach**(tableau **as** $key => $value){inst}

```
<?php
$arr=array("cle1" => 2, "untruc" => "et son bidule",
                          7 => "un sept");
foreach ($arr as $key => $value) {
    echo "la clef  $key est associee a la  valeur :
                                  $value<br />\n";
}
?>
```

transformation chaine<->tableau

- **split**(`$sep, $chaine`) ; transforme une chaîne en tableau (`$sep` est le séparateur).

```
<?php
$machaine="il fait beau, c'est cool";
$arr1=split(" ",$machaine);
print_r($arr1);
echo "<br />";
$arr2=split(",",$machaine);
print_r($arr2);
?>
```

Affiche :

```
Array (
  [0] => il [1] => fait [2] => beau, [3] => c'est
      [4] => cool )
Array ( [0] => il fait beau [1] => c'est cool )
```

sur split/explode

- pour **split**, séparateur=une expression régulière. deprecated(Obsolète) à partir de php 5.3 (utiliser par exemple **preg_split**)
- pour **explode**, séparateur=une chaîne de caractères

Obsolète

transformation chaine<->tableau

- **join**(`$sep, $tab`) : transforme un tableau en chaîne de caractères (**implode** est synonyme)

```
<?php
$array = array('nom', 'email', 'telephone');
$separer_par_diese = join("#", $array);
$separer_par_plus = join("+", $array);
echo $separer_par_diese;
echo "<br />";
echo $separer_par_plus;
?>
```

Affiche :

```
nom#email#telephone
nom+email+telephone
```

autres fonction sur les tableaux

- **count**(`$tab`) : nombre d'éléments du tableau
- **reset**(`$tab`), **end**(`$tab`), **prev**(`$tab`), **next**(`$tab`) :parcours lineaire du tableau
- **current**(`$tab`) :accès a l'élément courant
- **key**(`$tab`) : acces a la clé de l'élément courant.
- **each**(`$tab`) : acces la paire (clé, valeur) suivante du tableau associatif.

exemples

```
$tab = array("a", "b", "c") ;
$c = count($tab) ; // $c vaut 3
$res = end($tab) ; // $res vaut "c"
$res = prev($tab) ; // $res vaut "b"

$tab = array("un"=>1, "deux"=>2, "trois"=>3) ;
// premier parcours
for(reset($tab) ; $cle = key($tab) ; next($tab)) {
                  $val = current($tab);
                  print("$cle = $val< br />\n");
}
// second parcours
reset($tab);
while(list($cle, $valeur) = each($tab)){
                  print "$cle = $valeur<br />";
}
```

Les fonctions

- Sans typage, comme les variables
- Passage de paramètres :
  - par défaut, par valeur (pas de modif des arguments)
  - passage par référence possible avec &.

Définition de fonction

- **function** nomDeFunction ($param$$_1$,...$param$$_n$){inst}

```
function double($i) {
            return $i*2;
}
$e=2;
$d = double($e);
echo $d; //affiche 4
```

Exemples de fonction

```
<?php
 function plus_un($x) {
        return $x+1 ; }
?>
```

Passage par référence à la déclaration:

```
<?php
 function incremente(&$x) { $x += 1 ; }
 $a = 1;
 incremente($a);
 print $a; //affiche 2
?>
```

Passage par référence à l'appel (deprecated a partir de php

```
<?php
 function incremente($x) { $x += 1 ; }
 $a = 1;
 incremente(&$a);
 print $a;
?>
```

Portée des variables

- La page ou fonction englobante
- Accès aux variables globales depuis une fonction : `"global $var"`
- Durée de vie : chargement de la page.
- Les supers-globales (visible quelque soit le contexte)

Inclusion de fichier et bibliotheque

- `include 'fichier'` : warning si le fichier n'est pas présent
- `require 'fichier'` : erreur fatale si fichier n'est pas présent
- `include_once 'fichier'` : comme include mais pas rechargé si deja fait
- `require_once 'fichier'` : comme require mais pas rechargé si deja fait

fichier : `olivier.php`

```
<?php
echo "olivier";
?>
```

fichier : `saluer.php`

```
<?php
echo "bonjour ";
require_once('olivier.php');
?>
```

affichera : bonjour olivier

Plusieurs syntaxe sont possibles :

```
<body>
<?php
 $a = 3 ;
 if ($a ==3) {
   echo "A vaut 3 <br />'' ;
 } else {
   echo "A ne vaut pas 3 <br />" ;
 }
?>
</body>
Ou bien:
<body>
<?php $a = 3 ; if ($a ==3) : ?>
A vaut 3 <br />
<?php else : ?>
A ne vaut pas 3 <br />
<?php endif; ?>
</body>
```

Variables predefinies, les superglobales

- `$_SERVER` : Variables fournies par le serveur web
- `$_GET` ; `$_POST` ; Variable pour la transmition de paramétres
- `$_COOKIE` : pour la gestion des cookies.
- `$_FILES` : Variables pour la transition de fichiers
- `$_ENV` : Les variables fournies par l'environnement.
- `$_SESSION` : Variable de la sessions

Toutes les infos : `phpinfo();`

Traitement de formulaire

Accès au donné d'un formaulaire par `$_GET` et `$_POST`

- le fichier html : `monform.html`

```
<html>
  <head><title>exemple formulaire </title></head>
  <body>
   <form action="traite.php" method="GET"/>
      Nom:<input type=text name="name" id="name"  >
          <input type="submit">
        </form>
  </body></htm>
```

- le fichier de traitement php : `traite.php`

```
Le nom est <?php echo $_GET['name'] ?>
```

Envoie de données sans formulaire...

- Si `traite.php` est sur la machine : `mserveur.cnam.fr`
- Lors du chargement de l'url : `http:\\mserveur.cnam.fr/traite.php?nom=baba&prenom=ali` dans le code de `traite.php` , `$_GET[nom]` contient baba
- Et si on utilise `$_GET[nom]` mais qu'on ne le passe pas dans l'url ?
  - `isset` et `empty`

```
if(isset($_GET["nom"])){
  //on peut l'utiliser
  $nom=$_GET["nom"];
}
else{
 //code n'utilisant pas $_GET["nom"];
}
```

upload de fichier

Utilisation d'un formulaire :

- type MIME `"multipart/form-data"`
- côté serveur, utilisation de la variable `$_FILE`
- le fichier est automatiquement décodé et stocké par PHP
  Contenu du fichier envoyé dans une requète POST

Exemple d'upload de fichier

Exemple de formulaire :

```
<html>
  <head><title>exemple d'upload </title></head>
  <body>
  <form enctype="multipart/form-data"
     action="upload.php" method="POST">
     Fichier à envoyer :
         <input name="userfile" type="file" />
         <input type="submit" value="Envoyer" />
  </form>
  </body>
</html>
```

Exemple d'upload de fichier

Côté serveur : Reception de la reqête HTTP :

- Décodage du corps MIME de la requête
- Création du fichier dans un répertoire temporaire
- Récupération des info par une variable super globale, `$_FILE`.
  C'est un tableau associatif. On peut l'inspecter :

```
print_r($_FILES);
```

```
Array
(
 [userfile] => Array
   (
     [name] => monfichierenvoye.png
     [type] => image/png
     [tmp_name] => /Applications/MAMP/tmp/php/phpM5EbP2
     [error] => 0
     [size] => 118481
   )
)
```

Exemple d'upload de fichier

- Clé : nom du champs du formulaire (ici `userfile`)
- Valeur : un tableau associatif
  - `name` : nom du fichier chez le client
  - `type` : type MIME si positionné par le navigateur
  - `tmp_name` : chemin complet du fichier temporaire uploadé
  - `error` : code d'erreur
  - `size` : taille du fichier uploadé en octets

Exemple de script "upload.php" :

```
<?php
//un repertoir ou le serveur doit pouvoir ecrire
$uploaddir = "./upload/";
//nom du fichier telecharge
$uploadfile = $uploaddir . $_FILES["userfile"]["name"];
//nom du fichier temporaire uploade
$tmp_file=$_FILES["userfile"]["tmp_name"];
//verifions qu'il a ete uploade
if( !is_uploaded_file($tmp_file) )
    { exit("Le fichier est introuvable");}
//deplacons le
if (move_uploaded_file($tmp_file, $uploadfile)) {
       print("Ok $uploadfile cree\n") ;
} else {
       print "Erreur:\n";
       print_r($_FILES);
}
?>
```

Cookies

- Mettre un cookie :
  setCookie `( name , value, expire, path, domain, secure )`
  En début de page car envoyés dans les en-têtes HTTP
- Lire un cookie
  Par le tableau associatif super global `$_COOKIE`

Exemple de manipulation des Cookies

- Un cookie qui expire dans 1 heure

```
<?php
        setCookie("test", 3, time()+3600) ;
?>
<html>
  <head><title>titre</title></head>
  <body><h1>exemple</h1></body>
</html>
```

- Heure d'expiration :
  - Nombre de secondes depuis le 1er Janvier 1970
  - Utilisation de la fonction **time**()
  - Utilisation de **mktime**()
    mktime(h,m,s,mois,jour,année) ;

Stockage de données de type tableau : trois solutions :

- convertir le tableau en chaîne en utilisant join() et split()
- autant de cookies que de cases de tableau
- utiliser la sérialisation (conversion de n'importe quel objet en chaîne de caractères) avec `serialize()` et `unserialize()`

tableau.php –

```
<?php
 $a = array(1,2,3) ; setCookie(test, join(",", $a)) ;
?>
<html>
 <body>
  Verification du <a href="verif-tableau.php">cookie</a>
 </body>
</html>
```

verif-tableau.php –

```
<html>
 <body>
  Valeur brute reçue : <?php echo $_COOKIE['test']; ?> <br /
  Valeur décodée :
  <?php
    $a = split(",",$_COOKIE['test']) ; print_r($a) ;
  ?>
 </body>
</html>
```

Sessions

- Besoin :
  - identifier chaque utilisateur de manière unique
  - utiliser des données persistantes pour chaque utilisateur (ex : login, panier...)
  - fiable et sûr (au contraire des cookies)
- Implémentation :
  - identifiant de session unique, stocké dans un cookie (nommé PHPSESSID par défaut)
  - base de données qui associe un ensemble de variables à un identifiant

Sessions

- **session_start()**
  - démarre une nouvelle session si elle n'existe pas (pas de cookie de session)
  - restaure la session courante si elle existe
- Création et utilisation d'une variable de session
  - Utilisation de la super globale : `$_SESSION['variable']`
  - Vérifier si elle est definie : `isset($_SESSION['variable'])`
  - Suppression d'une variable de session : `unset($_SESSION['variable'])`

Session

page1.php –

```
<?php
 session_start();//on demare la session
 echo 'Bienvenue à la page numéro 1';
 $_SESSION['nom'] = 'toto';
 $_SESSION['date'] = date();
 // Fonctionne si le cookie a été accepté
 echo '<br /><a href="page2.php">page 2</a>';
?>
```

page2.php --

```
<?php
 session_start();
 echo 'Bienvenue sur la page numéro 2<br />';
 echo bonjour . $_SESSION['nom'];
  echo "vous surfer depuis" $_SESSION['date']);
?>
```

Objets

- Définition des membres d'une classe : propriétés et méthodes

```
class MyClass {
 public $var = 0;
 public function method() {
   ...
 }}
```

- Héritage `class SubClass extends BaseClass { ... }`
- Création / desctruction d'instances
  `$obj = new A(); $obj = null;`
- Accès aux membres : $obj->var = 'bla';obj->method();
- Visibilite
  - `public` : accès par tous
  - `protected` : accès par les classes filles
  - `private` : aucun accès

Objets

```
<?php
 class Personne
  {  // Attributs
      public $nom;
      public $age;

    // Méthodes
    public function __construct($nom,$a) {
         $this->nom=$nom;
        $this->age=$a;
}
    public function viellir(){
      $this->age++;
    }

    public function presentation() {
      echo 'je suis '.$this->nom." ";
      echo "j'ai ".$this->age." ans";
    }
}
```

Objets

```
$toto=new Personne("toto",10);
$toto->viellir();
$toto->presentation();

?>
```

affichera :

```
je suis toto j'ai 11 ans
```

Base de données

Principe :

- émission de requêtes en texte
- réception de ressource en retour

Fonctions principales :

- `mysqli_connect` : connection au serveur MySQL
- `mysqli_select_db` : choix de la base
- `mysqli_query` : émission d'une requête
- `mysqli_fetch_assoc` : récupération d'une ligne résultat dans un tableau associatif
- `mysqli_free_result` : nettoyage
- `mysqli_close` : fermeture de la connection

Base de donées

```
<?php
// Connexion et sélection de la base
$link = mysqli_connect("machine", "login", "motdepasse")
   or die("Impossible de se connecter");
 echo "Connexion réussie";

//selectionner la base courrante
mysqli_select_db($link,"mabase")
       or die("Could not select database");

// Exécuter des requêtes SQL
$sql = "SELECT * FROM unetable";
$result = mysqli_query($link,$sql) or die("Query failed");



...
```

Base de donées

```
...

echo "<table>\n";
 while ($line = mysqli_fetch_assoc($result)) {
   echo "\t<tr>\n";
   foreach ($line as $col_value) {
       echo "\t\t<td>$col_value</td>\n";
   }
   echo "\t</tr>\n";
 }
 echo "</table>\n";
mysqli_free_result($result); // Libération des résultats
mysqli_close($link);         // Fermeture de la connexion

?>
```

Base de donées

- PhpMyadmin : outils graphique de creation/manipulation de BD
- mysqli : bibliotheque de base pour mysql (procèdurale)
- PDO : (PHP Data Objects) bibliotheque générique (mysql,oracle...) object

SQL

- Création / destruction d'une table :

```
CREATE TABLE groupe1.clients (
      numero INTEGER NOT NULL DEFAULT 0 PRIMARY KEY AUT
      nom VARCHAR(255) NOT NULL,
      prenom VARCHAR(255) NOT NULL,
      email VARCHAR(255) NOT NULL );

DROP TABLE groupe1.clients;
```

- Insertion d'une ligne

```
INSERT INTO groupe1.clients (nom, prenom, email)
VALUES ( 'John', 'Smith', 'john@smith.com' );
```

- Sélection

```
SELECT nom, email FROM groupe1.clients
      WHERE nom LIKE '%john%'
      ORDER BY nom ASC;
```