Module:
Développement Web

Mohamed CHERRADI

UAE / ENSAH

Formateur Web

m.cherradi@uae.ac.ma

Chapitre 4
PHP

Plan

Introduction

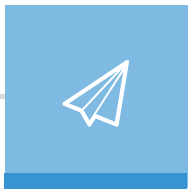
Présentation, Syntaxe de base, PHP5,
Les variables , les constantes, les structures de contrôles,
Les tableaux, les fonctions, …

Les formulaires en PHP

Récupération des données des formulaires, Fichier de traitement, méthodes d'envoie, transfert des fichiers vers le serveur, Redirection et Inclusion, Cookies & Sessions
…

POO PHP

Les classes simple et abstraites, les interfaces, Héritage, Espace de noms, Auto-chargement des classes, Accès à la base de données (PDO), …



L’architecture MVC

Description du design pattern MVC, fichier .htaccess, frameworks PHP (Laravel, Symfony, …)
…

Introduction

Le langage PHP

Introduction

- PHP (*PHP Hypertext PreProcessor)* est un langage de programmation conçu pour permettre la création des applications dynamiques, le plus souvent développées pour le Web.
- Il est à l'origine un langage de script conçu spécifiquement pour agir sur un serveur web. Ce dernier permet d'interpréter le code PHP des pages web.
- Grace à un serveur Web tel que *Apache*, PHP permet, généralement, de réaliser les traitements suivants :
  - Récupérer des données envoyées par le navigateur afin d'être interprétées ou stockées pour une utilisation ultérieure.
  - Récupérer des informations issues d'une base de données, d'un système de fichiers (contenu de fichiers et de l'arborescence)
- Les capacités de PHP ne s'arrêtent pas à la création de pages web. Il est aussi possible de manipuler des images, de créer des fichiers PDF,…etc
- PHP a été utilisé pour *créer un grand nombre de sites web* célèbres, comme Facebook, Wikipédia, etc.

Principe de fonctionnement

- Lorsqu'un visiteur demande à consulter une page de site internet, son navigateur envoie une requête au serveur HTTP correspondant. Si la page est identifiée comme un script PHP (généralement grâce à l'extension .php), le serveur appelle l'interpréteur PHP qui va traiter et générer le code final de la page (constitué généralement d'HTML, mais aussi souvent de CSS et de JS).
- Ce contenu est renvoyé au serveur HTTP, qui l'envoie finalement au client.
- Une étape supplémentaire est souvent ajoutée : celle du dialogue entre PHP et la base de données. Classiquement, PHP ouvre une connexion au serveur de SGBD voulu, lui transmet des requêtes et en récupère le résultat, avant de fermer la connexion.

Syntaxe de base

- PHP appartient à la grande famille des descendants du C, dont la syntaxe est très proche. Il s'agit d'un langage de script qui peut facilement être mélangé avec du code HTML au sein d'un fichier PHP.
- Un script PHP peut être placé n'importe où dans un document Web. Il commence par **<?php** et se termine par **?>** :

```
<?php
// le code PHP s'insère ici
?>
```

- L'extension des fichiers PHP est ".php". Un fichier PHP peut contenir du code de script PHP mais aussi des les balises HTML.
- Exemple: un script PHP qui utilise une fonction PHP intégré "echo" pour afficher le texte "Hello World !" sur une page web:

```
<!DOCTYPE html>
<html>
<body>
<h1>My first PHP page</h1>
<?php
echo "Hello World!";
?>
</body>
</html>
```

PHP5

- Dès qu'on parle de PHP5, on pense implicitement au PHP Orienté Objet.
- Depuis la version 5, PHP intègre les concepts de l'orientée objet pour bénéficier des avantages offerts par cette approche de programmation tels que l'*encapsulation*, l'*abstraction*, *l'héritage* …
- Bien qu'on peut continuer à programmer en procédural, pourtant le concept de l'OO est de plus en plus utilisé par les développeurs. D'ailleurs c'est l'approche utilisé dans les Framework et les CMS (Système de Gestion de Contenu) comme par exemple : Joomla. WordPress, Drupal, Shopify, …
- La POO apporte **quelques avantages** non négligeables au langage PHP à savoir :
  - *Réutilisation du code* : les objets créés peuvent servir de base pour d'autres objets. Cela permet de regrouper les objets similaires pour ne pas avoir à écrire plusieurs fois le même code. De plus, il est possible de réutiliser le code dans différents projets.
  - *Modularité du code*: facilité de rajouter des éléments sans avoir à modifier tout le code.
  - *Clarté du code*: le concept impose un code plus lisible et mieux organisé. Il permet aussi d'avoir un code plus compréhensible pour les autres développeurs amenés à reprendre votre code.

Eléments de base - PHP

- Comme tout langage de programmation, PHP manipule des données. Cette manipulation nécessite l'utilisation des variables.
- Ces variables peuvent être de différents types, les plus employés sont :
  - Sous forme de chaîne de caractères,
  - Sous forme de nombres entiers ou décimaux,
  - Sous forme de valeurs booléennes vrai ou faux (TRUE ou FALSE).
- Cependant, il en existe d'autres, qui peuvent être des types composés comme:
  - Les tableaux
  - Les objets
  - Des types particuliers comme NULL
  - …

Les variables, constantes et leur déclaration

- **Les variables:**
  - Une variable est le conteneur d'une valeur d'un des types utilisés par PHP (entiers, flottants, chaînes de caractères, …).
  - En PHP, une variable commence toujours par le caractère dollar **($)** suivi du nom de variable. Exemples : $var, $_var, $var2, $M1, $_123
  - PHP c'est un langage peu typé, la déclaration des variables n'est pas obligatoire en début de script.
- **Les constantes:**
  - Les constantes sont souvent utilisés en PHP pour conserver des données répétitive dans toutes les pages d'un même site.
  - Pour définir une constate, PHP fournit la fonction *define(),* dont la syntaxe est la suivante : *boolean define(string nom_cte, divers valeur_cte, boolean casse).*

```php
<?php
define("PI",3.1415926535,TRUE);
echo "La constante PI vaut ",PI,"<br />";
if(define("site","http://www.ensah.ma",FALSE))
{
echo "<a href=\" ",site," \">Lien vers le site de ENSAH</ a>";
}
?>
```

Affectations par valeur et par référence

- En PHP, l'opération d'affectation se faite de la même façon que d'autres langages de programmation comme C/C++ ou java. Il possible de faire affecter une variable par valeur ou par référence. L'opérateur égale (=) est utilisé pour un telle affectation. Exemples :
  - *Affectation par valeur :* $var = valeur;
  - *Affectation par référence :* $var2=&$var1;
- PHP accepte aussi les opérateurs d'affectation combinée comme += ($x += $y équivaut à $x = $x + $y), *= , … En plus des opérations arithmétique, il existe un autre operateur bien particulier au langage PHP qui est le point (.), il permet d'effecteur une concaténation. Il possible de l'utiliser de la façon suivante (.=) .
- Exemples:

```
<!DOCTYPE html>
<html>
<body>
<?php
$x=2;
$y=3;
echo "La somme : $x+$y =",$x+$y,"<br />";
?>
</body>
</html>
```

```
<?php
$nom="Meachel";
$prénom="Jean";
echo "ton nom et prénom est :",$prénom," ",$nom,"<br />";
?>
```

Les opérateurs de comparaison

- == Teste l'égalité de deux valeurs.
- != ou <> Teste l'inégalité de deux valeurs sans prend en compte
- === Teste l'identité des valeurs et des types de deux expressions.
- !== Teste la non-identité de deux expressions.
- < Teste si le premier opérande est strictement inférieur au second.
- <= Teste si le premier opérande est inférieur ou égal au second.
- > Teste si le premier opérande est strictement supérieur au second.
- >= Teste si le premier opérande est supérieur ou égal au second.
- OR Teste si l'un au moins des opérandes a la valeur TRUE.
- || équivaut à l'opérateur OR mais n'a pas la même priorité.
- XOR Teste si un et un seul des opérandes a la valeur TRUE.
- AND Teste si les deux opérandes valent TRUE en même temps.
- && Équivaut à l'opérateur AND mais n'a pas la même priorité.
- ! Opérateur unaire de négation, qui inverse la valeur de l'opérande.

Les instructions conditionnelles

- Comme tout langage, PHP dispose d'instructions conditionnelles qui permettent d'orienter le déroulement d'un script en fonction de la valeur de données.
- La syntaxe utilisée est identique à celle utilisée langages de programmation comme C, C++, Java…
- Exemples:

```php
<?php
$cat="PC";
$prix=900;
if($cat=="PC") {
 if($prix >= 1000)
 { echo "<h3> La remise est de 15 %, le prix de PC est de : ",$prix*0.85, "&euro; </h3>"; }
 else
 { echo "<h3> La remise est de 10 %, le prix de PC est de : ",$prix*0.90, "&euro; </h3>"; }
 }
elseif($cat=="Livres")
 { echo "<h3> La remise est de 5 %, le prix de livre est de : ",$prix*0.95, "&euro; </h3>"; }
 else
 { echo "<h3> la remise est de 2 %, le prix de livre est de : ",$prix*0.98, "&euro; </h3>"; }
?>
```

Les instructions répétitives (Boucles)

- De même pour les boucles la syntaxe est inspirée aussi de langage C.
- En plus des boucles habituelles *(for, while, do…while)*, PHP ajoute à partir de la version 4 une autre boucle qui est *foreach* .
- La boucle *foreach* permet de parcourir rapidement l'ensemble des éléments d'un tableau, sans avoir besoin de connaître ni le nombre d'éléments ni les clés. Sa syntaxe est donnée comme suit: *foreach($tableau as $cle=>$valeur) { }*
- Exemples:

```php
<?php
//Création d'un tableau associatif
$i=0;
while($i<=4){
    $tab["nombre aleatoire ".$i] = rand(100,1000);
    $i++;
}
//Lecture des clés et des valeurs
foreach($tab as $cle=>$val)
{echo " Le <b>$cle</b> egale à <b>$val</b><br />";}
?>
```

Le **nombre aleatoire 0** egale à **408**
Le **nombre aleatoire 1** egale à **841**
Le **nombre aleatoire 2** egale à **243**
Le **nombre aleatoire 3** egale à **362**

```php
<?php
//Création du tableau de 9 éléments
for($i=0;$i<=8;$i++)
{
$tab[$i] = pow(2,$i);
}
//Lecture des valeurs du tableau
echo"Les puissances de 2 sont :";
foreach($tab as $val)
{echo $val." ";}
?>
```

Les puissances de 2 sont :1 2 4 8 16 32 64 128 256

Les tableaux

- Pour la création des tableaux , PHP fournit deux méthodes:
  - En utilisant la méthode classique : la déclaration commence par dollar suivi de nom de tableau puis par des crochets [] . Exemple : $tab[0] , $tab[1] = 2004; $tab[2] = "PHP5";
  - En utilisant la fonction array() *:* elle permet de créer de manière rapide des tableaux indicés ou associatifs. Il s'agit de la méthode le plus souvent utilisée pour la création de tableaux. La syntaxe est: $tab= array(valeur0,valeur1,…,valeurN). Un tableau multidimensionnels peut être défini comme étant un ensemble d'array: array(array(), array(), … , array()).
- Cependant, il en existe d'autres, qui peuvent être des types composés comme:
- La création d'un tableau associatif consiste à associer pour chacun de ses éléments une clé et une valeur. La syntaxe de la fonction array() dans ce cas, est la suivante: $tabasso=array("clé1"=>valeur1,"clé2"=>valeur2,… "cléN"=>valeurN).
- Exemples:

```php
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Peter is 35 years old.

Les fonctions

- Comme d'autres langages, PHP permet bien entendu d'écrire ses propres fonctions. Pour déclarer une fonction en PHP, il suffit d'utiliser le mot-clef *function*.
- Comme le langage n'est pas typé, une fonction peut retourner n'importe quel type de valeur (chaîne, entier…) ou ne rien retourner du tout !!!
- Enfin, ses arguments peuvent avoir des valeurs par défaut.
- Exemple:

```php
<?php
function foo($x,$y=0){
    print 'La fonction foo('.$x.','.$y.') donne :';
    return $x+$y;
}
echo foo(2); /*La fonction foo('.$x.','.$y.') donne :2 */
echo foo(3,4) /* La fonction foo('.$x.','.$y.') donne :7 */
?>
```

Traitement des chaînes de caractères

- De très nombreuse fonctions de traitement des chaines de caractères, dont voici les principales:
  - int strlen($s): retourne la taille d'une chaîne de caractère.
  - int strcmp($s1,$s2): compare deux chaînes, elle retourne 0 si identique, -1 si $s1<$s2, 1 sinon (il existe aussi strcasecmp() insensible à la casse).
  - string substr($s, int $indice, int $n): retourne une sous-chaîne de taille $n à partir de $indice.
  - int substr_count($s1, $s2): retourne le nombre d'occurrences de $s2 dans la chaîne $s1.
  - int strpos($s1, $s2, int offset): trouve la position d'un caractère dans une chaîne. (stripos() version insensible à la casse)
  - mixed str_replace($s1,$s2,$s, $var): remplace toutes les occurrences de $s1 par $s2 dans une chaîne $s. Le nombre de remplacement est contenu dans $var. (str_ireplace() version insensible à la casse)
  - Array explode($sep, $s): retourne un tableau de chaînes, chacune d'elle étant une sous-chaîne du $s extraite en utilisant le séparateur $sep.

Traitement des chaînes de caractères

- string implode($sep, array $tab): fait le contraire de explode(). Elle réunit tous les éléments d'un tableau dans une chaîne en les séparant par le caractère $sep.
- string trim($s, $sc) (resp rtrim()): retire la chaine $sc de $s (resp de la fin de $s).
- int similar_text($s1, $s2, $pourcent): calcule la similarité de deux chaînes en nombre de caractères ou en pourcentage retourné dans la variable $pourcent.
- mixed str_word_count($s): retourne le nombre de mots présents dans une chaîne.
- htmlspecialchars(): utilisée pour convertir des caractères spéciaux (par exemple <, >) aux entités HTML (& amp; , &#039, < , > , …). Une fonction identique est htmlentities().
- string sprintf(string "format" , $s1, s2,... $sN) : Retourne une chaîne formatée contenant $s1 à $sN.
- divers sscanf($s , string "format", string $s1,...,$sN]) : Décompose une chaîne selon un format donné et retourne ses éléments dans un tableau ou dans les variables $s1 à $sN.

Exemples

```php
<?php
$ch1 ="ENSAH est un établissement public d'enseignement supérieur";
echo substr($ch1 , 0,5); //affiche : ENSAH
echo substr($ch1, 10);
//affiche:un établissement public d'enseignement supérieur
$ch2 ='p';
$nb=substr_count($ch1, $ch2); //
echo "le caractere $ch2 est presenté $nb fois dans la pharase";
// affiche : le caractere p est presenté 2 fois dans la pharase
$ch3=str_replace('ENSAH', 'FST', $ch1);
echo $ch3;
//affiche: FST est un établissement public d'enseignement supérieur
$ch4= 'FST';
$nb=strpos($ch3,$ch4);
echo "Le mot $ch4 comme à la position $ndans la phrase";
// Le mot FST comme à la position 0 dans la phrase
?>
```

```php
<?php
$ch1="Blanc";
$ch2="Bleu";
$ch3="blanc";
echo strcmp ($ch1,$ch2);//Affiche -1
echo strcasecmp ($ch1,$ch3);//Affiche 0
echo strncasecmp ( $ch1, $ch2,2);//Affiche 0
?>
```

```php
<?php
$s1="MySQL";
$s2="PgSQL";
echo similar_text($s1,$s2,$pourc), "caractères communs";
//3 caractères communs
echo "Similarité : ",$pourc,"%";  //Similarité : 60%
?>
```

```php
<?php
$lien  = "ensah.ma/eservices/etudiant/afficher/1";
$query = explode("/", $lien);
echo $query[0]; // ensah.ma
echo $query[1]; // eservices
echo $query[2]; // etudiant
echo $query[3]; // afficher
echo $query[4]; // 1

$lien = implode("/", $query);
echo $lien; // ensah.ma/eservices/etudiant/afficher/1
?>
```

```php
<?php
$a = "...Jean " ;
$b = "Dupont___";
echo $a,$b,"<br />"; // Affiche : ...Jean Dupont___
echo trim($a,' .')," ",rtrim($b,' _');
// Affiche : Jean Dupont
?>
```

```php
<?php
$personne = "1685-1750 Jean-Sébastien Bach";
$format="%d-%d %s %s";
$nb = sscanf($personne,$format,$ne,$mort,$prenom,$nom);
echo "$prenom $nom né en $ne, mort en $mort <br />";
echo "Nous lisons $nb informations";
?>
```

```php
<?php
$ch1 = "Monsieur " ;
$ch2 = "Rasmus" ;
echo sprintf ("Bonjour %s %s, bravo !",$ch1,$ch2);
//Affiche: Bonjour Monsieur Rasmus, bravo !
?>
```

Traitement sur les tableaux

- PHP fournit plusieurs fonctions prédéfinies pour faire des traitements sur les tableaux :
  - *int count(array $tab, mode)*: retourne le nombre d'éléments de *$tab*. *mode* prend 0 ou 1, *mode=1* pour compter les éléments d'un tableau multidimensionnel.
  - *mixed array_sum (array $tab)* : retourne la somme des éléments d'un tableau.
  - *$sous_tab = array_slice(array $tab,int $debut, int $fin) :* crée un sous-tableau avec $debut à $fin .
  - *int array_push($tab, valeur1, valeur2,…, valeurN)* : ajoute en une seule opération les N éléments passés en paramètres à la fin du tableau *$tab* et retourne la taille.
  - *int array_unshift($tab, valeur1, valeur2,…, valeurN) :* à l'encontre de *array_push()* cette fonction ajoute des éléments au début d'un tableau.
  - *array_pop($tab) resp (array_shift($tab)):* supprime le dernier (resp le premier) élément du tableau. *$tab.*
  - *array array_values($tab) resp. array_keys* : retourne un tableau ne contenant que les valeurs du tableau associatif *$tab*. Crée des indices numériques de 0 à *N*.

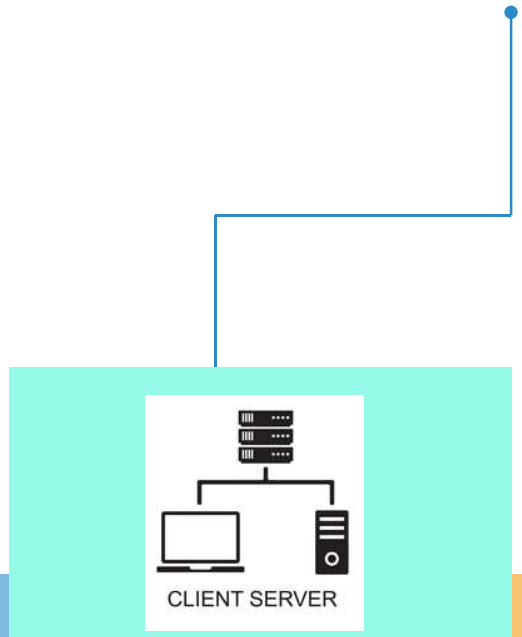
Traitement sur les tableaux

- *void shue($tab) :* mélange les valeurs de *$tab.*
- *array array_unique(array $tab) :* élimine les doublons et retourne un nouveau tableau.
- *array array_reverse(array $tab , bool $cle)* : Inverse l'ordre des éléments du tableau. Préserve les clés si le paramètre *$cle* vaut TRUE.
- *array_count_values(array $tab)* : compte le nombre d'occurrences des éléments de *$tab* et retourne un tableau dont les clés sont les valeurs du tableau $tab et les valeurs le nombre d'occurrences.
- *bool in_array(divers $val, array $tab , bool $type)* : recherche l'élément *$val* dans le tableau *$tab*. Si le paramètre type vaut TRUE les types de la valeur recherchée et de l'élément doivent être identiques.
- *divers array_rand (array $tab , int $N)* : choisit *$N* éléments au hasard dans *$tab* et retourne un tableau contenant les clés des éléments choisis. Si *$N* vaut 1 la fonction retourne une chaîne contenant sa clé.
- void sort($tab) : tri croissant (décroissant avec rsort) selon les valeurs. Il existe aussi ksort($asso), asort($asso), void usort($tab, moncmp).

Traitement sur les variables

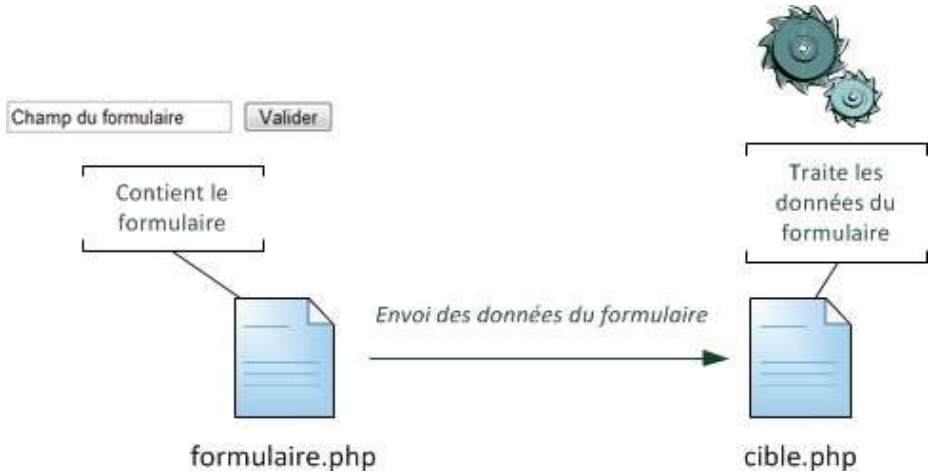
- ***Déterminer le type:*** avant de manipuler des variables, en utilisant, par exemple, des opérateurs, il peut être utile de connaître leur type.
  - La principale fonction permettant de le faire est *gettype()*, dont la syntaxe est la suivante : *string gettype($var)*, elle retourne une chaîne de caractères contenant le type de la variable en clair.
  - Les fonctions suivantes permettent de vérifier si une variable est d'un type précis : *is_null($var)*, *is_integer($var)* ou *is_int($var), is_double($var), is_string($var), is_bool($var), is_array($var), is_object($var), is_resource($var).*
- ***Convertir le type*:** parfois, il peut être indispensable de convertir explicitement une variable d'un type dans un autre. Ce qui le cas des variables issues d'un formulaire qui sont toujours de type *string*. Pour convertir une variable d'un type dans un autre, voici comment : *$result=(typeDésiré)$var;*
- ***Contrôler l'état*:** lors de l'envoi de données d'un formulaire vers le serveur, le script qui reçoit les informations doit pouvoir détecter l'existence d'une réponse dans les champs du formulaire. Les fonctions *isset()* et *empty()* permettent ce type de contrôle.

Les formulaires en PHP



Récupération des données d'un formulaire en PHP

- Rappelons que pour créer un formulaire en HTML on utilise l'élément <form>. Ce dernier possède deux attributs obligatoires qui sont *method* et *action*.
  - Le premier sert à préciser la méthode avec laquelle les données seront envoyées.
  - Le deuxième (action) sert à préciser le fichier récupérant les données.
- <form method=" " action=" " >



Fichier de traitement

- Grâce à l'attribut *action=" "*, l'endroit de fichier de traitement est précisé selon les cas suivants :
  - Au cas où le fichier de traitement et le document contenant le formulaire se trouvent séparés, il faut donner comme valeur à l'attribut *action* l'adresse où se trouve le fichier. Exemples : *action="nom_de_fichier.php"* (même endroit), *action= "http://www.ensah.ma/dossier/fichier.php"*.
  - Au cas où le fichier de traitement contient au même temps le script de traitement et le code de formulaire, la meilleur solution c'est de donner à action la variable *$_SERVER["PHP_SELF"]* comme valeur, qui contient le nom du fichier en cours d'exécution comme valeur de l'attribut action.
    - $_SERVER est un tableau contenant des informations comme les en-têtes, dossiers et chemins du script, etc.... (Pour voir son contenu vous pouvez exécuter la fonction *var_dump($_SERVER)*)
- Il est recommandé que ce fichier soit présent dans le même répertoire que celui contenant le formulaire, mais ce n'est pas obligatoire.

Méthode d'envoie

- Les deux méthodes (method) d'envoi sont *GET* et *POST*.
  - Pour la première méthode (*GET*), les données transiteront par l'URL comme on l'a appris précédemment se récupèrent grâce à l'array $_GET.
  - Pour la deuxième (POST), les informations transiteront de manière masquée par cette méthode se récupèrent grâce à l'array $_POST.
- *$_GET* et *$_POST* sont des tableaux de données associatifs et superglobaux. Voici leurs principales caractéristiques :
  - Ils sont générés à la volée par PHP avant même que la première ligne du script ne soit exécuté.
  - Ce sont des *tableaux associatifs* comme ceux que l'on déclare traditionnellement. Leur manipulation est exactement semblable à ces derniers. **Les clés** correspondent aux noms des variables transmises et **les valeurs** à celles associées à ces variables.
  - Ils sont *superglobaux*, c'est à dire visibles de partout dans le programme (même à l'intérieur d'une fonction utilisateur).
  - Ils sont accessibles en lecture et en écriture. Il est donc possible de les modifier.

Quelle méthode d'envoi à utiliser ?

- Les deux méthodes ne permettent pas de sécuriser l'envoi des données. Le fait de ne pas afficher les données par la méthode POST ne signifie en rien qu'elles sont cryptées.
- Le choix d'une méthode alors **dépend du contexte**:
  - *Pour la méthode GET*: il possible de l'utiliser dans le cas suivant : si par exemple, nous souhaitons mettre en place un moteur de recherche alors nous pourrons nous contenter de la méthode GET qui transmettra les mots-clés dans l'url. Cela nous permettra aussi de fournir l'url de recherches à d'autres personnes. C'est typiquement le cas des URLs de Google. Exemple: une URL du moteur de recherche Google :*http://www.google.fr/search?q=php*
  - *Pour la méthode POST*: c'est préférée lorsqu'il y'a un nombre important de données à transmettre ou bien lorsqu'il faut envoyer des données sensibles comme des mots de passe. Dans certains cas, seule la méthode POST est requise : un upload de fichier par exemple.

Comment récupérer les données ?

- Le tableau $_GET contient tous les couples variable / valeur transmis dans l'url.
- Le tableau $_POST contient tous les couples variable / valeur transmis en POST, c'est à dire les informations qui ne proviennent ni de l'url, ni des cookies et ni des sessions.
- **Exemple** : Pour récupérer la valeur d'un élément input de type texte dont le nom est *prenom* <input type = "text" name= "prenom" />
  - **<?php** echo $_GET['prenom']; **?>**
  - **<?php** echo $_POST['prenom'];**?>**
- NB:
  - La casse des variables est importante. Il faut bien penser à mettre *$_GET* et *$_POST* en majuscules. Dans le cas contraire, il sera impossible d'obtenir une valeur et une erreur de type undefined variable sera retournée.
  - A noter aussi qu'il existe d'autres tableaux associatifs *superglobal* *$_REQUEST* qui fonctionne exactement comme tous les autres tableaux.

Les valeurs uniques et multiples

- **Les valeurs uniques** proviennent des champs de formulaire dans lesquels l'utilisateur ne peut entrer qu'une valeur, un texte par exemple, ou ne peut faire qu'un seul choix (bouton radio, liste de sélection à choix unique, …). Exemple : <input type="radio" name="choix" />
- **Les valeurs multiples** proviennent des champs de formulaire qui peuvent permettre aux visiteurs de saisir plusieurs valeurs sous un même nom de composant. Cela peut concerner un groupe de cases à cocher ayant le même attribut *name*, dont il est possible de cocher une ou plusieurs cases simultanément. Dans ce cas, il ne s'agit pas d'une valeur scalaire mais un tableau qui est récupéré côté serveur. Il faut pour cela faire suivre le nom du composant de crochets, comme pour créer une variable de type array. Exemple :
  - DUT :<input type="checkbox" name="choix[]" value="DUT" />
  - DEUG : <input type="checkbox" name="choix[]" value="DEUG" />
  - L'utilisateur peut cocher les deux cases simultanément. **Les valeurs sont récupérées comme suit** :
    - $_POST["choix"][0] contient la valeur "DUT"
    - $_POST["choix"][1] contient la valeur "DEUG"

Transfert des fichiers vers le serveur

- Les deux tableaux $_GET et $_POST permettent le transfert de données simples et ils sont incapable de **transférer des documents ou des fichiers de taille importante**.
- Pour permettre le transfert de fichier de grande taille, le langage PHP, depuis sa version 4.1, a fournit un autre tableau ***associatif et multidimensionnel*** qui est $_FILES. Si, par exemple le nom de l'élément <input type="file" name=« fichier" .. />, il possible de lire les valeurs suivantes :
  - $_FILES["fichier"]["name"]: contient le nom de fichier
  - $_FILES["fichier"]["type"]: le type de fichier (exemple application/pdf)
  - $_FILES["fichier"]["size"]: la taille de fichier en nombre d'octet.
  - $_FILES["fichier"]["tmp_name"]: le nom temporaire sur le serveur.
  - $_FILES["fichier"]["error"] : pour controler s'il y avait des erreurs de transfert. La valeur 0 indique la bonne réception.
- Contrairement aux exemples précédents, l'élément <form> doit avoir l'attribut *method* à la valeur *post* et l'attribut *enctype* à la valeur *multipart/form-data*.

Transfert des fichiers vers le serveur

- Consignes de sécurité: pour éviter des problèmes de sécurité voici quelques précautions à faire :
  - Définir le type MIME de fichiers acceptés, par exemple "image/jpg", "image/gif", … ceci peut être réalisé grâce à l'attribut *accept* de l'élément <input />.
  - Limiter la taille des fichiers à télécharger en ajoutant au formulaire un champ caché nommé *MAX_FILE_SIZE*. Cette valeur est récupérée dans la variable *$_POST["MAX_FILE_SIZE"]* lorsque le champ caché est défini par le code suivant :

    <input type="hidden" name="MAX_FILE_SIZE" value="1000" />
- Après Avoir transférer le fichier sur le serveur, il nécessaire de la placer dans le dossier correspondant. Cela peut se faire en utilisant la fonction: *move_uploaded_file($tmp_name, $new_name).*
- Il est recommandé de renommer le fichier et ne jamais garder son nom de départ, car deux visiteurs différents pourraient avoir un fichier du même nom.

Exemples

```html
<form action="FormTraitement.php" method="post" enctype="multipart/form-data">
<fieldset>
<input type="hidden" name="MAX_FILE_SIZE" value="100000" />
<legend><b>Transfert de fichier</b></legend>
Fichier : <input type="file" name="fich" accept="image/jpg" size="100"/><br/>
Clic! <input type="submit" value="Envoi" /><br/>
</fieldset>
</form>
```

Fichier *FormTraitement.php*

```php
<?php
if(isset($_FILES['fich']))
{
echo "Taille maximale autorisée :",$_POST["MAX_FILE_SIZE"],"octets<hr / >";
echo "<b>Clés et valeurs du tableau \$_FILES </b><br />";
foreach($_FILES["fich"] as $cle => $valeur)
{
echo "clé : $cle valeur : $valeur <br />";
}
//Enregistrement et renommage du fichier
$result=move_uploaded_file($_FILES["fich"]["tmp_name"],"imagephp.gif");
if($result==TRUE)
{echo "<hr /><big>Le transfert est réalisé !</big>";}
else
{echo "<hr /> Erreur de transfert n°",$_FILES["fich"]["error"];}
}
?>
```

Affichage :

Taille maximale autorisée :100000 octets

Clés et valeurs du tableau $_FILES
clé : name valeur : ensah.jpg
clé : type valeur :
clé : tmp_name valeur :
clé : error valeur : 2
clé : size valeur : 0

Erreur de transfert n°2

```php
<?php

if(isset($_POST['submit'])){
    $name=$_FILES['fichier']["name"];
    $extension=substr($name,strpos($name,'.'));
    $new_name=md5(uniqid(rand(), true)).$extension;
    $resultat = move_uploaded_file($_FILES['fichier']['tmp_name'],$new_name);
    if ($resultat) echo "Transfert réussi";
}
?>
```

Bons pratique pour les formulaires

- ***Contrôler l'état des variables*** : lors de l'envoi de données d'un formulaire vers le serveur, il est recommandé de vérifier si chaque valeur envoyée depuis est bien reçu et elle est bien récupérée . Ceci est réalisé grâce à la fonction isset() . ***Exemple*** : <?php if(isset($_POST["prenom"])) echo $_POST["prenom"];?>
- ***Maintenir l'état de formulaire*** :
  - Lorsque le script contenant le formulaire est chargé du traitement des données, l'ensemble de la page est réaffiché après traitement, de même que l'ensemble du formulaire. Le formulaire se retrouve alors dans son état initial, et toutes les saisies effectuées sont effacées.
  - En cas d'erreur de saisie sur un seul champ, l'utilisateur est obligé de recommencer l'ensemble de la saisie. Pour éviter cela, il faut affecter le contenu récupéré grâce aux tableaux ***superglobaux*** (exemple *$_POST*) à la valeur de l'élément de formulaire.
  - Exemple : Pour l'élément suivant: <input type="text" name= "prenom"/>, l'attribut value doit être affecté à "<?php if(isset($_POST["prenom"])) echo $_POST["prenom"];?>". Comme ceci <input type="text" name="prenom" size="40" value="<?php if(isset($_POST["prenom"])) echo $_POST["prenom"]; ?>" />

Valider les éléments du formulaire

- **La validation des données de formulaire** est très nécessaire pour protéger les entrées contre les pirates et les spammeurs!
- Par exemple : la variable $ _SERVER ["PHP_SELF"] peut être utilisée par des pirates! Si PHP_SELF est utilisé dans une page, alors un utilisateur malveillant peut saisir tout un script dans l'URL.
- Pour éviter que $ _SERVER ["PHP_SELF"] soit exploité on peut utiliser la fonction *htmlspecialchars()*. Le code de formulaire doit ressembler à ceci: <form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
- **Les zones de saisit d'un formulaire** (text, textarea, ..) peuvent être aussi exploités par des pirates c'est pourquoi il faut bien les contrôler grâce au différents fonctions suivantes:

```
function test_input($data) {
        $data = htmlspecialchars($data);
        $data = trim($data);
        $data = stripslashes($data);  // Supprime les antislashs d'une chaîne
        return $data;
}
```

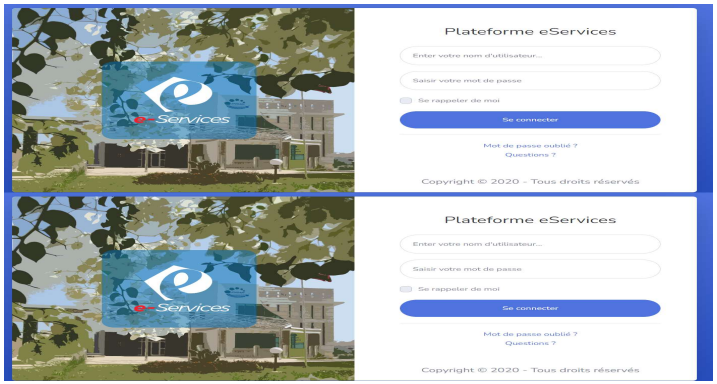
Exemples

```php
<?php
$nom = $email = $genre="";
if (!empty($_POST["nom"])) {  $nom = $_POST["nom"]; }
if (!empty($_POST["email"])) { $email = $_POST["email"]; }
if (!empty($_POST["genre"])) { $genre = $_POST["genre"]; }
if (!empty($_POST["choix"][0])) { $Licence=$_POST["choix"][0]; }
if (!empty($_POST["choix"][1])) { $DUT=$_POST["choix"][1]; }
if (!empty($_POST["choix"][2])) { $DEUG=$_POST["choix"][2]; }
if (!empty($_POST["choix"][3])) { $DEUST=$_POST["choix"][3]; }
?>
<h2>Formulaire exemple </h2>
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
  Nom: <input type="text" name="nom" value="<?php echo $nom;?>"><br><br>
  Mail: <input type="text" name="email" value="<?php echo $email;?>"> <br><br>
  Sexe:
  <input type="radio" name="genre"
  <?php if (isset($genre) && $genre=="F") echo "checked";?> value="F">F
  <input type="radio" name="genre"
  <?php if (isset($genre) && $genre=="M") echo "checked";?> value="M">M
  <br><br>
 <h3> Diplôme obtenu : </h3>
  Licence :<input type="checkbox" name="choix[]"
  <?php if (isset($_POST["choix"][0]))  echo "checked";?> value="Licence" />
  DUT : <input type="checkbox" name="choix[]" <?php if (isset($_POST["choix"][1])) echo "checked";?> value="DUT" />
  DEUG :<input type="checkbox" name="choix[]" <?php if (isset($_POST["choix"][2])) echo "checked";?> value="DEUG" />
  DEUST : <input type="checkbox" name="choix[]" <?php if (isset($_POST["choix"][3])) echo "checked";?> value="DEUST" />
<br><br>
  <input type="submit" name="submit" value="Submit">
</form>
```

L'exemple prend en considération le maintien en état de formulaire

Authentification et redirection

- **Une application web** est un ensemble de pages web crées pour fournir un service donné
- Exemples: service de facturation, e-commerce, gestion scolarité, …
- Dans la plupart des cas, l'accès aux différentes pages d'une application web exige une authentification pour pouvoir utiliser ses différentes services (Exemple de e-services de ENSAH)
- Une telle application utilise un script qui demande un login et un mot de passe pour qu'un visiteur puisse se « connecter » (s'authentifier) et ensuite rediriger vers ses services.



Exemple basique

- Dans cet exemple on va créer un simple script qui va permettre de vérifier si le login et le mot de passe saisit correspondent bien à ceux qui sont prédéfinis à l'aide de *define()* : *define('LOGIN','etudiant');define('PASSWORD','ensah');*


```php
<form action="<?php echo htmlspecialchars($_SERVER['PHP_SELF']); ?>" method="post">
    <legend>Identifiez-vous</legend>
    <label for="login">Login :</label>
    <input type="text" name="login" id="login" value="" />
    <label for="password">Password :</label>
    <input type="password" name="password" id="password" value="" />
    <input type="submit" name="submit" value="Se connecter" />
</form>
```

- Le script de vérification est donné comme suit :

```php
<?php
    $login=isset($_POST['login'])? $_POST['login']:'';
    $password=isset($_POST['password'])? $_POST['password']:'';
    if($login==LOGIN && $password==PASSWORD)
            echo "vous etes bien connecté  $login";
        else
            echo "mot de passe et/ou login sont incorrects";
?>
```

- Ce script permet tout simplement d'afficher « vous êtes bien connecté » si l'utilisateur a bien fait entrer comme login 'etudiant' et comme mot passe 'ensah'.

Redirection

- Il existe **des applications web** pour lesquelles on souhaite rediriger le visiteur en fonction de paramètres. C'est le cas par exemple pour un script d'identification. Si l'internaute fournit les bons identifiants alors il est redirigé automatiquement vers son espace personnel, sinon il est renvoyé vers le formulaire d'authentification.
- Pour créer une redirection avec PHP, on utilise la fonction header() permettant d'envoyer des entêtes de type *Location* (adresse). **<?php** header('Location: pageprotegee.php'); exit(); **?>**
- **NB** : l'appel de cette fonction doit se faire avant tout envoi au navigateur (instruction echo, print, balise html...).
- Exemple:

```php
<?php
    define('LOGIN','etud');
    define('PASSWORD','test');
    $login=isset($_POST['login'])? $_POST['login']:'';
    $password=isset($_POST['password'])? $_POST['password']:'';
    if($login=='')
     header('location: form.php?error=1');
    elseif($password!='test')
        header('location: form.php?error=2&password='.$password);
        else header('location: bienvenue.php');
?>
```

Inclusion

- À la différence de la fonction de redirection, il existe des fonctions permettant d'inclure du contenu sur la même page sans rédiger vers une autre.
- Elles peuvent être utilisé pour créer du contenu tel que : une bannière de page, un bloc d'informations ou un menu dans un fichier de script bien spécifique et de l'inclure en cas de besoin dans n'importe quelle page de l'application.
- Exemple : Selon l'utilisateur connecté, on peut inclure un contenu qui correspond à un tableau ou bien à une liste.
- L'avantage de ceci est lorsqu'on souhaite modifier le contenu, il suffit de l'effectuer dans un fichier unique; celle-ci sera ensuite répercutée sur chaque page dans laquelle apparaît le fichier d'inclusion.
- Il existe deux fonctions d'inclusion de base en PHP : *include()* et *require()*. Les deux se comportent de la même manière mais retournent des erreurs différentes.
  - Si une fonction *include()* n'est pas analysée correctement, elle continue le traitement du reste de la page et affiche un avertissement dans la page où le fichier inclus doit apparaître.
  - Si une fonction *require()* fait référence à un fichier manquant, la fonction arrête le traitement de la page et affiche une page d'erreur dans le navigateur.

Inclusion

- Les fonctions include_once() et require_once() spécifient qu'un fichier d'inclusion ne doit être utilisé qu'une seule fois dans une page. Si deux fonctions **include()** font référence au même fichier d'inclusion, seule la première fonction **include()** s'affiche dans le navigateur.
- Une instruction PHP d'inclusion est un bloc de code qui extrait le contenu d'un fichier externe dans une page Web. Voici un exemple d'instruction PHP d'inclusion : *<?php include('pageBanner.php'); ?>*
- Exemple: Pour l'exemple précédent, au lieu de déclarer des constants directement dans le script consacré à l'authentification, on peut le faire dans un fichier séparé (exemple definition.php) et puis de l'inclure après.

```php
<?php
define('LOGIN','etud');
define('PASSWORD','test');
?>
```

```php
<?php
    include 'definition.php';
    $login=isset($_POST['login'])? $_POST['login']:'';
    $password=isset($_POST['password'])? $_POST['password']:'';
    if($login=='')
     header('location: form.php?error=1');
    elseif($password!='test')
        header('location: form.php?error=2&password='.$password);
        else header('location: bienvenue.php');
```

Cookies & Sessions

POO PHP

L’architecture MVC

Merci pour votre attention !!!