

Bases PHP

Architecture logicielle

Il faut un serveur Web, faisant tourner PHP, pour interpréter les scripts côté client de la page. Vous pouvez utiliser :

Toutes les solutions web intégrant (Apache/PHP/MariaDB) : Wampp, Lampp, Xampp, Mampp EasyPHP...

- Sous Windows, préférer une solution portable indépendante du système (Xampp)
- Et sinon, PHP peut aussi servir de serveur Web :


```
php -S 127.0.0.1:8000
```

Dans tous les cas, ne pas oublier de démarrer le serveur Http (Apache ou PHP).

-- phpInfo

- Récupération des informations liées à la configuration de PHP

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>phpInfo</title>
</head>
<body>
<?php
phpinfo();
?>
</body>
</html>
```



| | |
|--|--|
| System | Windows NT PC-DELL 6.2 build 9200 (Unknown Windows version Home Premium Edition) i586 |
| Build Date | Sep 12 2012 23:44:56 |
| Compiler | MSVC9 (Visual C++ 2008) |
| Architecture | x86 |
| Configure Command | cscrip /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo" |
| Server API | Apache 2.4 Handler Apache Lounge |
| Virtual Directory Support | enabled |
| Configuration File (php.ini) Path | C:\Windows |
| Loaded | C:\xampp\php\php.ini |

-- Variables

- Les variables php n'ont pas besoin d'être déclarées ce qui ne facilite généralement pas le débogage...
- Le type est déterminé par affectation du contenu de la variable
- Une variable commence par le signe \$

Nommage : Un nom de variable

- doit commencer par une lettre majuscule ou minuscule ou par un tiret bas "_" (touche 8 de votre clavier)
- peut comporter des lettres, des chiffres et des tiret bas "_"
- ne doit pas comporter d'espaces.
- ne doit pas commencer par un chiffre
- ne doit pas comporter de caractères spéciaux comme: @ ou # ...
- ne doit pas comporter de tiret, soit le signe (-) touche 6 du clavier

Commenter le source suivant en utilisant

//comment sur 1 ligne et /*comment sur plusieurs lignes*/

pour en expliquer la signification

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Variables</title>
</head>
<body>
<?php
//un entier
```

```

$i=5;
echo($i);

//un booléen
$b=true;
echo($b);
?>
<br>
<?php
$str="1 chaîne";
echo "\$str est : ".gettype($str)."<br>\n";
echo $str."<br>";
echo $i." accessible depuis un autre bloc de script<br>\n";
echo "Ajout d'un entier et d'une chaîne : ".$i+$str."<br>\n";
$i="devenu une chaîne";

echo "La variable $i est devenue une chaîne : ".$i."<br>\n";
echo 'La variable $i est devenue une chaîne : '.$i."<br>\n";
?>
</body>
</html>

```

- Utiliser **gettype(\$variable)** qui retourne le type d'une variable et **var_dump(\$variable)** qui permet d'en afficher le contenu détaillé pour mieux comprendre le rôle de l'affectation en php.
- Afficher le code source de votre page dans le navigateur
- Ajouter le caractère \n après les
 et afficher le code source de votre page dans le navigateur

-- Opérateurs

Arithmétiques

| Opérateur | Rôle | Type de contenu |
|-----------|----------------|-----------------------------------|
| \$a + \$b | addition | Numériques : entiers, décimaux... |
| \$a - \$b | soustraction | |
| \$a * \$b | multiplication | |
| \$a / \$b | division | |
| \$a % \$b | modulo | |
| - \$a | opposé | |

Sur les chaînes

| Opérateur | Rôle | Type de contenu |
|--------------|--------------------------|-----------------|
| \$s1 . \$s2 | Concaténation | Chaînes |
| \$s1 .= \$s2 | affectation concaténante | |

Affectation

| Opérateur | Équivalent |
|------------|-----------------|
| \$a += \$b | \$a = \$a + \$b |
| \$a -= \$b | \$a = \$a - \$b |
| \$a *= \$b | \$a = \$a * \$b |
| \$a /= \$b | \$a = \$a / \$b |
| \$a %= \$b | \$a = \$a % \$b |

| Opérateur | Équivalent |
|--------------------------|-------------------------------|
| <code>\$s .= \$s1</code> | <code>\$s = \$s . \$s1</code> |

Tester tous les opérateurs dans une page **opérateurs.php** en utilisant des variables et la fonction [sprintf](#) pour l'affichage

-- Structures de contrôle

-- Condition

if

```
<?php
if ($a > $b) {
    echo "a est plus grand que b";
}
?>
```

Si le code à exécuter ne comporte qu'une seule ligne, le bloc défini par { et } est facultatif.

Version contractée :

```
<?php
$max=($a > $b)?$a:$b;
?>
```

if/else

```
<?php
if ($a > $b) {
    echo "a est plus grand que b";
} else {
    echo "a est plus petit ou égal à b";
}
?>
```

if/else/elseif

```
<?php
if ($a > $b) {
    echo "a est plus grand que b";
} elseif ($a < $b) {
    echo "a est plus petit que b";
} else {
```

```
echo "a est égal à b";
}
?>
```

-- Boucles

while

```
$i = 1;
while ($i <= 10) {
    echo $i++; /* La valeur affichée est $i avant l'incrémentacion
                (post-incrémentacion) */
}
?>
```

for

La même boucle avec un for

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

-- Fonctions

Déclaration

Une fonction se déclare de la manière suivante :

```
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n){
    echo "Exemple de fonction.\n";
    return $retval;
}
?>
```

Aucun typage des variables, donc les arguments ne sont pas plus typés que le retour éventuel.

```
<?php
function concat($s1, $s2){
    return $s1.$s2;
}
?>
```

Les fonctions peuvent prendre des paramètres optionnels (à la fin)

Dans l'exemple qui suit, **\$arg2** est un paramètre optionnel qui prend la valeur **true** s'il est omis

```
<?php
function getValue($arg1, $arg2=true){
    if($arg2)
        return $arg1;
    else
        return "";
}
?>
```

Appels de fonctions

Une fonction peut être appelée dans la page où elle est déclarée (ou incluse on le verra plus tard), avant même sa déclaration.

Exemple d'appels :

```
<?php
$a=getValue(5,true);
$b=getValue(2);
$s=getValue("Chaîne",false);
echo(getValue("Chaîne"));
?>
```

-- Requête HTTP

-- Récupération des données passées dans l'URL : méthode GET

Les données passées dans l'URL, par la méthode **get**, sous la forme :

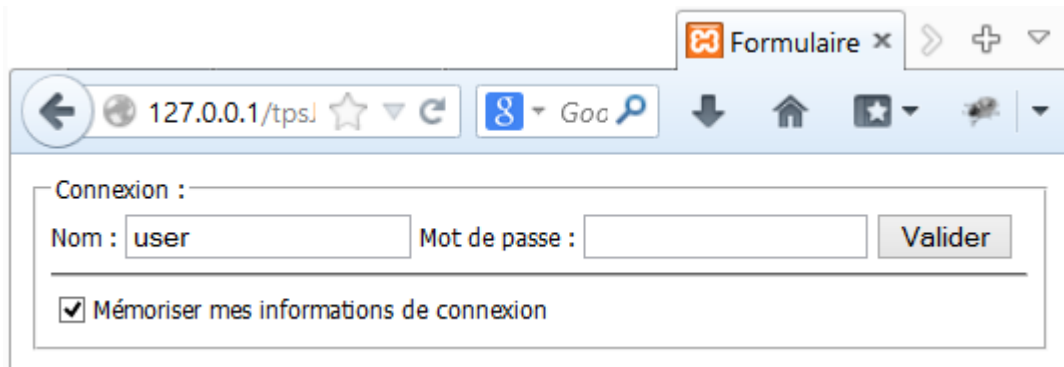
```
http:127.0.0.1/tps/tp1/maPage.php?module=M2105&dut=RT
```

sont récupérées grâce au tableau associatif **\$_GET** dans la page cible de la requête :

```
<?php
echo $_GET["dut"];
echo $_GET["module"];
?>
```

-- Récupération des données d'un formulaire : méthode POST (GET déconseillé)

Un formulaire



```

<div>
  <form method="post" action="<?php echo $_SERVER['PHP_SELF']?>">
    <fieldset style="vertical-align: middle;"><legend>Connexion :</legend>
      <label for="nom">Nom : </label><input type="text" name="nom" id="nom">
      <label for="password">Mot de passe : </label><input type="password"
name="password" id="password">
      <input type="submit" value="Valider">
      <hr>
      <div id="remember"><input type="checkbox" id="ckRemember"
name="ckRemember"><label for="ckRemember">Mémoriser mes informations de
connexion</label></div>
    </fieldset>
  </form>
</div>

```

Récupération des données postées

Les données envoyées par la méthode **post**, généralement depuis un formulaire sont récupérées grâce au tableau associatif **\$_POST** dans la page cible de la requête :

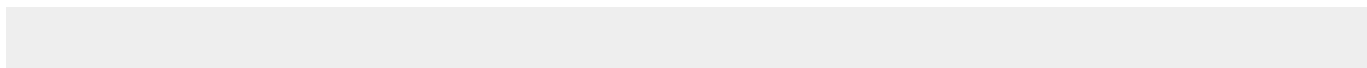
```

<?php
var_dump($_POST);
?>

```

| Expression | Rôle |
|------------------------------------|--|
| \$_SERVER['PHP_SELF'] | URL de la page en cours (le formulaire est ici posté vers la page qui le contient) |
| \$_POST | Tableau associatif des données postées |
| \$_POST["nom"] | Permet l'accès à la variable nom postée |
| isset(\$_POST["nom"]) | Teste l'existence de la variable nom dans les en-têtes de la requête |
| \$_SERVER['REQUEST_METHOD'] | Variable server donnant la méthode utilisée (POST ou GET) |

Traitement du formulaire



```
<?php
//Traitement du POST
if (strtoupper($_SERVER['REQUEST_METHOD']) == 'POST'){
    echo "Votre nom : ".$_POST["nom"]."<br>";
    echo "Votre mot de passe (hasché en sha1) : ".sha1($_POST["password"])."<br>";
    if(isset($_POST["ckRemember"]))
        echo "Mémorisation des infos :
".$_POST["ckRemember"]=="on"? "oui":"non");
}
else {
    //Affichage du Formulaire
    ?>
    <div>
        <form method="post" action="<?php echo $_SERVER['PHP_SELF']?>">
            <fieldset style="vertical-align: middle;"><legend>Connexion :</legend>
                <label for="nom">Nom : </label><input type="text" name="nom"
id="nom">
                <label for="password">Mot de passe : </label><input type="password"
name="password" id="password">
                <input type="submit" value="Valider">
                <hr>
                <div id="remember"><input type="checkbox" id="ckRemember"
name="ckRemember"><label for="ckRemember">Mémoriser mes informations de
connexion</label></div>
            </fieldset>
        </form>
    </div>
<?php }?>
```

Rq :

A noter que le mot de passe n'est haché en sha1 qu'à sa réception sur le serveur, il passe donc en clair sur le réseau.

The screenshot shows a web browser window with the address bar displaying `127.0.0.1/tpsJc/tp1/formulaire.php`. The page content shows a login form with the following text: `http://127.0.0.1/tpsJc/tp1/formulaire.php`, `Votre nom : admin@local.fr`, `Votre mot de passe (crypté en md5) : e10adc3949ba59abbe56e057f20f883e`, and `Mémorisation des infos : oui`. The developer tools network tab is open, showing a request to `http://127.0.0.1/tpsJc/tp1/formulaire.php` with a status of `200 OK`. The request headers include `Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8`, `Accept-Encoding: gzip,deflate,sdch`, `Accept-Language: fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4`, `Cache-Control: max-age=0`, `Connection: keep-alive`, `Content-Length: 50`, `Content-Type: application/x-www-form-urlencoded`, and a `Cookie` containing `__utma=96992031.1760393890.1383006835.1384903547.1385169990.57; __utmz=96992031.1383006835.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none)`. The form data is `nom: admin@local.fr`, `password: 123456`, and `ckRemember: on`. The response headers include `Connection: Keep-Alive`, `Content-Length: 569`, `Content-Type: text/html`, `Date: Sun, 19 Jan 2014 16:59:31 GMT`, `Keep-Alive: timeout=5, max=100`, `Server: Apache/2.4.3 (Win32) OpenSSL/1.0.1c PHP/5.4.7`, and `X-Powered-By: PHP/5.4.7`.

From:
<http://slamwiki2.kobject.net/> - **SlamWiki 2.1**

Permanent link:
<http://slamwiki2.kobject.net/php-rt/bases>

Last update: **2019/08/31 14:21**

