Cours de php

Sommaire

- Environnement WAMPSERVER
- Les bases du langage PHP
- Fonctions
- MySQL - PHPMyAdmin
- Connexion PHP <-> MySQL

  Echanges de données entre PHP et MySQL
- Projet

➢ liens utiles

♦ PHP

- http://www.php.net
- http://www.phpinfo.net
- http://www.phpfrance.com
- http://www.developpez.com/php/

♦ MySQL

- http://www.mysql.com/
- http://dev.nexen.net/docs/mysql/

♦ Exemple concret

- http://www.miag-rezo.net

➢ documentation en ligne

- La documentation officielle (en français):

  http://fr.php.net/

- Renseignements Commandes:

  rajouter en fin d'URL le nom de la commande:

  Exemple :

  http://fr.php.net/echo

Partie 1 :

Les bases de PHP

➢ les sites statiques et dynamiques

Sites statiques

- **réalisé uniquement à l'aide des langages (X)HTML et CSS**
- **ne peut pas être mis à jour automatiquement**

Sites dynamiques

- **utilisent d'autres langages, en plus de ()HTML et CSS, tels que PHP et MySQL.**
- **le contenu de ces sites web est dit "dynamique" parce qu'il peut changer sans l'intervention du webmaster**

➢ les sites dynamiques

Exemples d'applications

Un espace membres

s'inscrire au site et avoir accès à des sections réservées aux membres

Un compteur de visiteurs

compter le nombre de visiteurs connectés dans la journée ou en train de naviguer .

Une newsletter :

envoyer un Email à tous vos membres régulièrement pour leur présenter les nouveautés.

Des actualités

automatiser l'écriture d'actualités, en offrant au visiteurs la possibilité d'en rédiger, de les commenter, etc

Un forum :

proposition d'un forum de discussion.

historique

- Créé en 1994 par Rasmus Lerdorf

  PHP signifiait *Personnal Home Page*

- En 1997, PHP devient un projet collectif et son interpréteur réécrit: PHP3

  PHP : Hypertext Preprocessor

- A la fin de l'année 1999 la version PHP4 est apparue.

- PHP en est aujourd'hui à sa cinquième version: PHP5

➢ introduction

- PHP est un langage de programmation web: il introduit du dynamisme dans la page web.
- **HTML et CSS:** ne sont pas des langages de programmation, mais des langages de ***simple affichage statique***.
- La syntaxe du langage provient de celles du langage C, du Perl et de Java.

définitions

- PHP est un langage de script interprété qui est exécuté du côté **serveur** et non du côté **client**.

  c.a.d.: le code est exécuté sur le serveur. Celui-ci va lire le code PHP et l'interpréter.

- Un **script** est une suite d'instructions simples, peu structurées, permettant d'automatiser certaines tâches.

- Un **langage de script** est langage permettant de réaliser des programmes généralement petits et interprétés.

- Un **serveur** est un ordinateur détenant des ressources particulières qu'il met à la disposition d'autres ordinateurs par l'intermédiaire d'un réseau.

introduction

- Le code PHP traité par le serveur produit du code HTML qui sera ensuite interprété et affiché par le navigateur.
- Comme PHP s'exécute côté serveur, on ne trouve aucune trace du code PHP lorsqu' on regarde le code source de la page dans le navigateur.
- Le code PHP inséré dans les pages agit à chaque chargement (et donc rafraîchissement) de page web: première arrivée sur une page web, bouton actualiser …
- Ce code HTML va changer en fonction des circonstances qui ont été programmées.

➢ le script php

- **Un script PHP** est un fichier texte qui contient des instructions incluses dans un code HTML à l'aide de balises spéciales, et stocké sur le serveur.

  Ce fichier doit avoir l'extension **« .php »** pour pouvoir être interprété par le serveur.

exemple:

code1.php

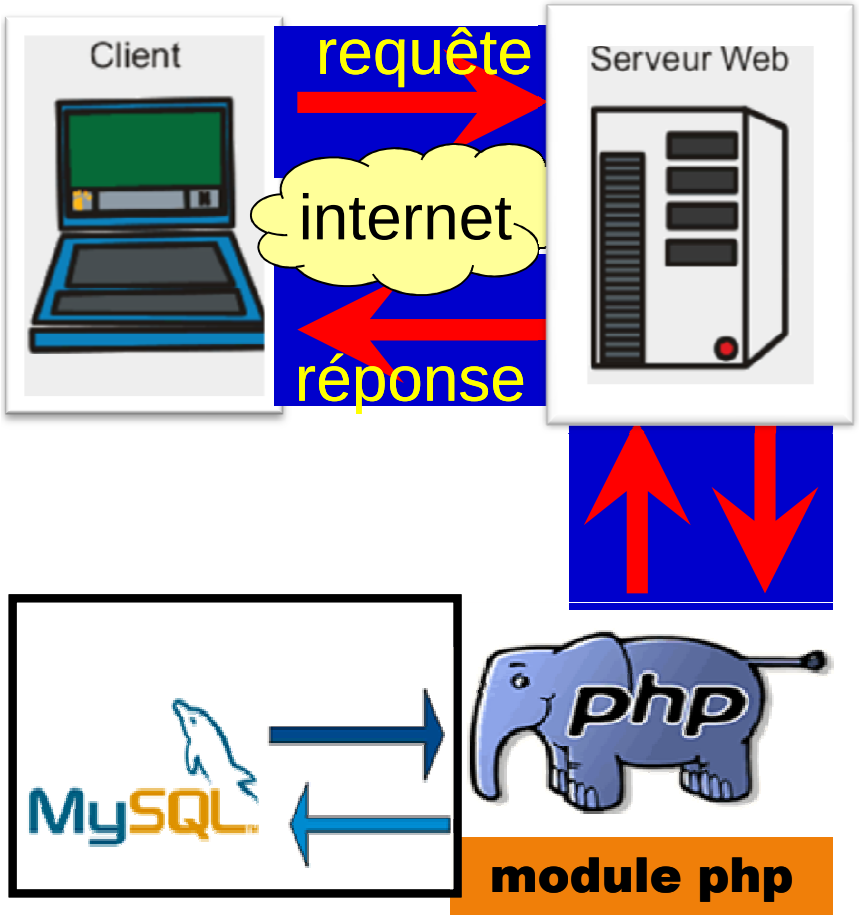
➢ accès à une page php

==> PHP est un langage incrusté au HTML et interprété côté serveur.

dés lors

- Comment se déroule une requête http ?
- Que se passe-t-il lorsqu'on tape une adresse dans son navigateur ?
- Que se passe-t-il lorsque votre page HTML contient du code PHP ?
- Où interviennent PHP et MySQL (la base de données) ?

➢ Comment se déroule une requête http ?



- Le navigateur envoi l'adresse saisie par l'internaute …
- Le serveur reçoit et analyse la page .
- Les parties entre balises PHP *(<?php ..... ?>)* sont envoyées au module PHP pour y être interprétées.
- PHP **parse** le fichier, c'est-à-dire qu'il va analyser et exécuter le code PHP qui se trouve entre les balises *<?php et ?>*.
- Si ce code contient des requêtes vers une base de données MySQL, PHP envoie la requête SQL.
- Le module base de données renvoie les données .
- PHP continue de parser la page, puis retourne le fichier dépourvu du code PHP au serveur web.

- Le serveur web renvoie donc un fichier ne contenant plus du PHP, mais seulement du HTML au navigateur qui l'interprète et l'affiche.

➢ serveur local

- PhP est un langage exécuté coté serveur :

  ==> Son apprentissage ou son utilisation pour la création de pages web nécessitent l'installation d'un serveur sur son PC

  ==> Ceci va permettre de travailler "en local".

  Le serveur local s'appellera "localhost" par défaut.

➢ serveur local

Pourquoi installer un serveur web sur son ordinateur?

- Pour tester des pages web html en local :

  ==> ouvrir le fichier dans un navigateur

- Pour tester des pages web php en local :

  ==> installer un serveur local capable d'interpréter le langage PHP.

  Tester les scripts PHP en local est plus souple que les envoyer sur l'hébergeur, sur Internet.

L'envoi répétitif de ces fichiers par FTP est très lourd.

➢ serveur local

Pourquoi installer un serveur web sur son ordinateur?

- Avec ce serveur local le PC sera à la fois client et serveur

  ==> On pourra donc faire de la programmation en PHP sans avoir à être connecté à Internet.

- Il existe plusieurs utilitaires pour installer un serveur local. Ils permettront d'installer les outils nécessaires à cette utilisation: **Apache**, **PHP**, **MySQL** ...

- Les plus connus sont : **EasyPHP**, **MOV'AMP**, **WAMP** …

  ==> **wampserver**

➢ serveur local

WampServer

➢ serveur local

Qu'est ce que wampserver ?

- **utilitaire/package** qui installe et configure automatiquement un environnement de travail complet permettant de mettre en œuvre la programmation web dynamique avec PHP.

- C'est une plate-forme de développement Web sous Windows.

- Il installe le serveur http Apache, le module PHP et le gestionnaire de bases de données MySQL.

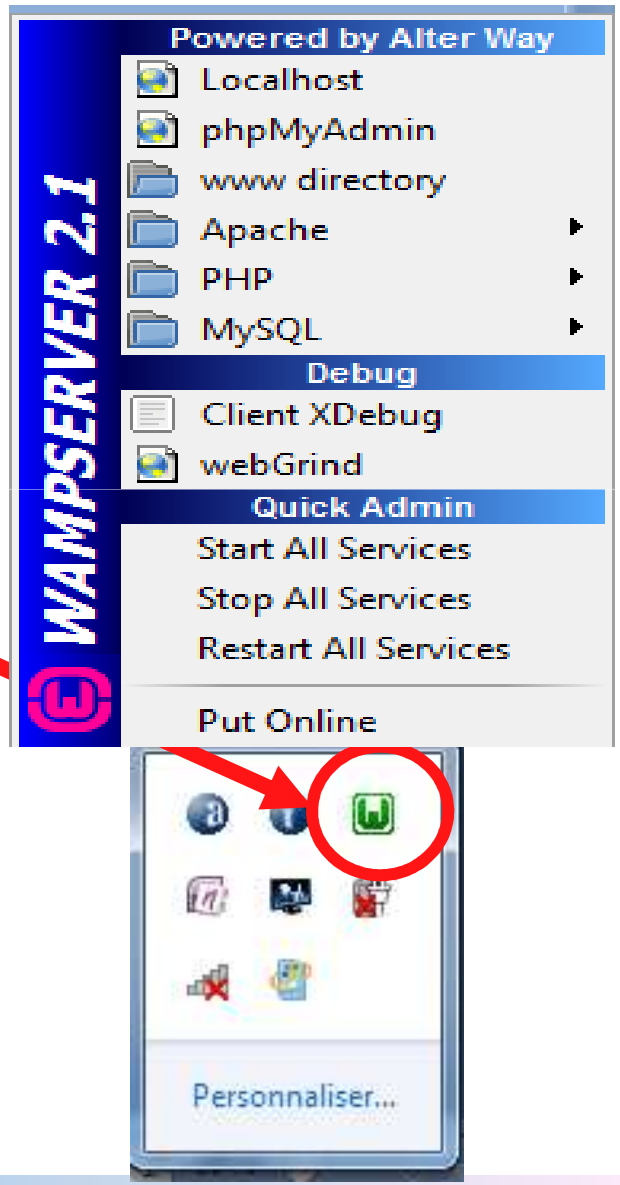
- Il installe également PHPMyAdmin et SQLite Manager, pour une gestion plus facile de nos bases de données.

http://www.wampserver.com

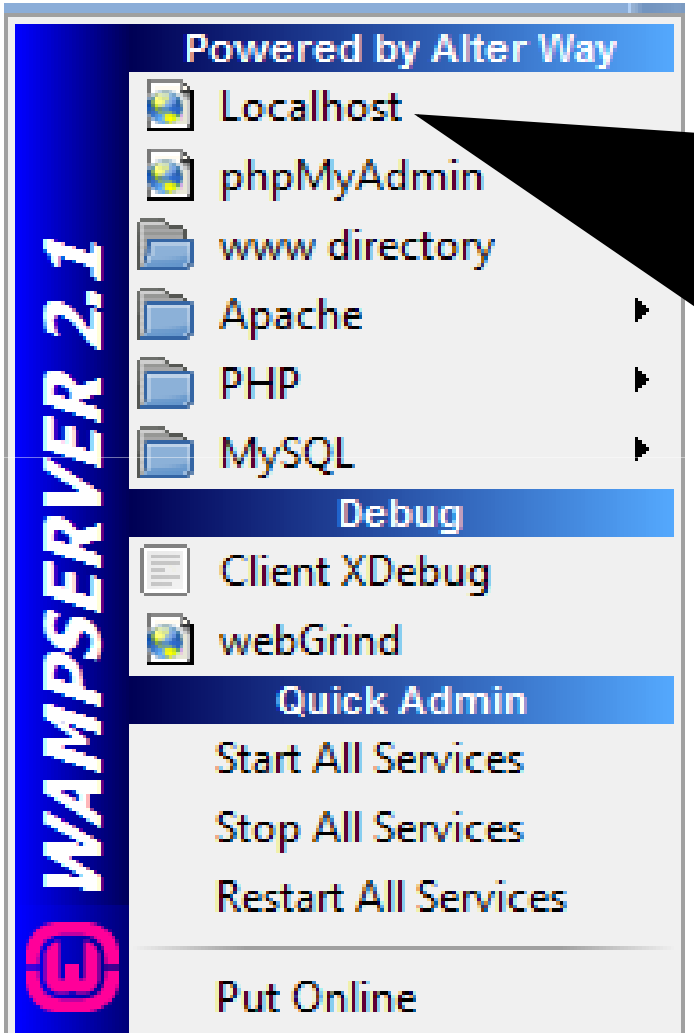
➢ serveur local

Wampserver :

- Après installation un raccourci de démarrage s'installe sur le bureau.
- Après démarrage de wampserver une icône se place dans la barre des tâches du PC. Elle indique que le serveur est actif.
- Un clic droit sur cette icône permet d'accéder à un menu contextuel permettant entre autre de démarrer ou arrêter Apache et MySql, ouvrir l'index de votre serveur en cliquant sur "localhost", qui est en fait simplement un raccourci vers http://localhost/.



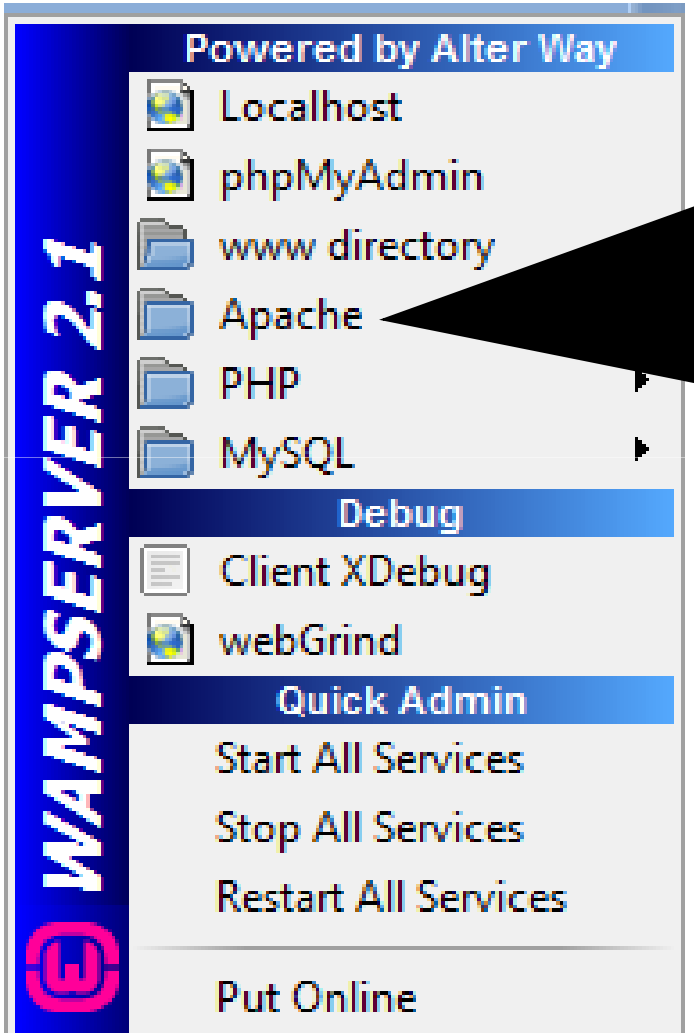
➢ serveur local



- serveur local qui va permettre d'accéder aux scripts et applications PHP et de les exécuter sur notre machine.

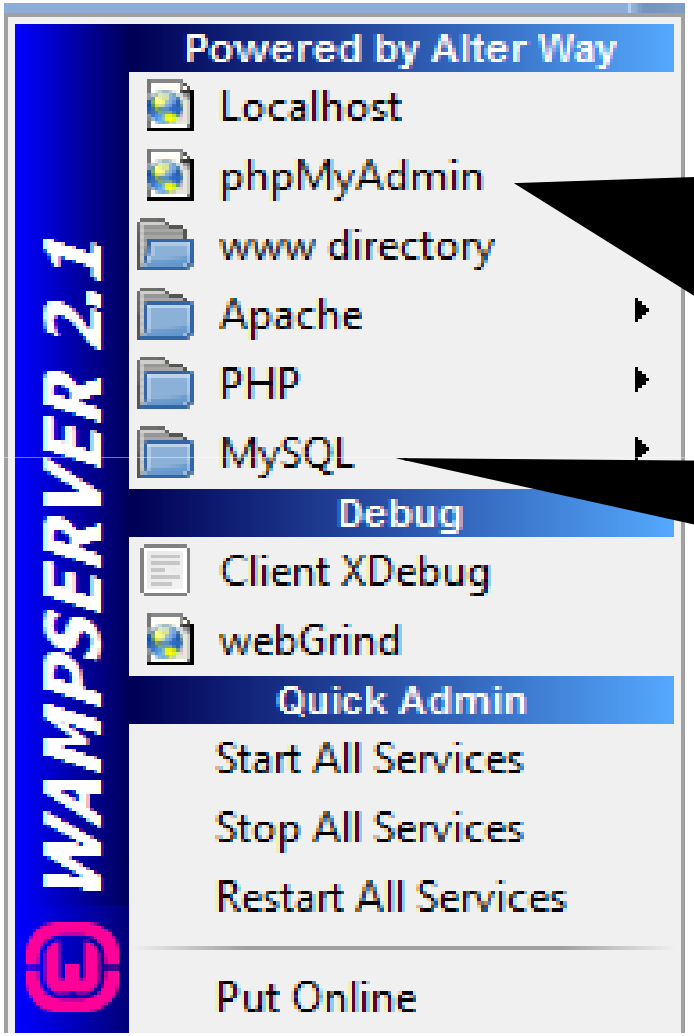
- Mais pour cela, il est indispensable de placer les fichiers php dans le répertoire nommé www.

➢ serveur local



- Apache est le serveur web/http
  Il permet de tester les scripts en local sur son PC.
  Un serveur HTTP est un logiciel qui met des fichiers à la disposition des utilisateurs d'un réseau

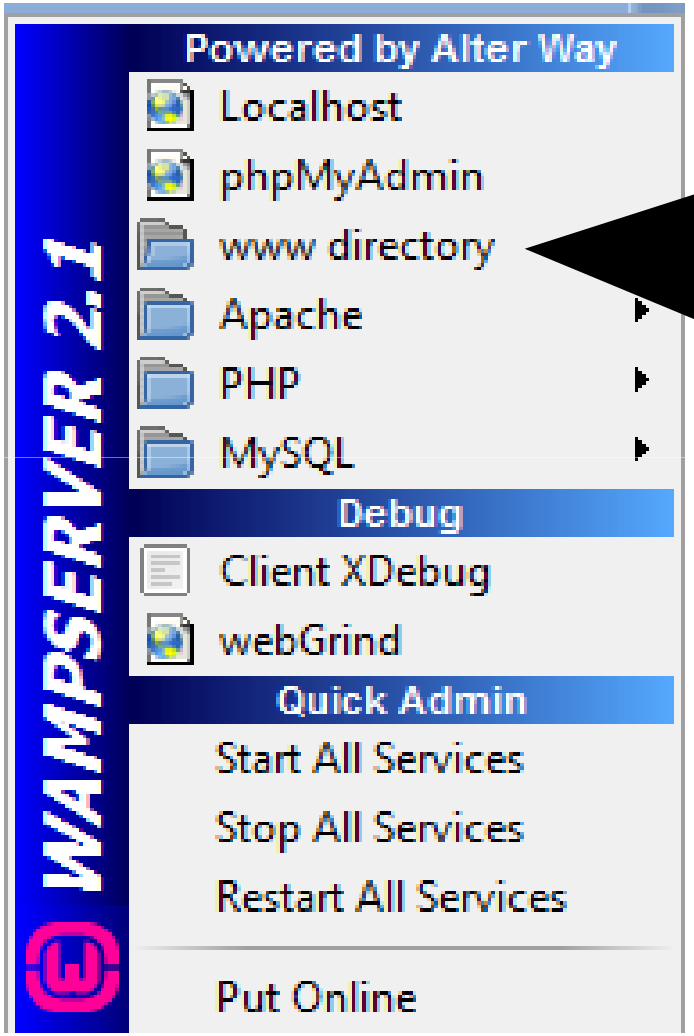
➢ serveur local



- phpMyAdmin: script PHP qui permet d'administrer nos bases de données.

- MySQL : serveur de bases de données relationnelles.

➢ serveur local



- www directory : permet d'accéder au répertoire où sont déposés les projets php.

➢ serveur local

Principaux points à connaître après l'installation :

♦ **Où vont se trouver les fichiers et dossiers du site ?**

- Par défaut, le dossier du site se nomme **www**

- Emplacement sous Windows: **c:/wamp/www/**

- Vos fichiers doivent être mis dans un dossier de travail (nommé tests par exemple), qui doit être placé dans le dossier www. On a alors le chemin physique suivant :

c:/wamp/www/tests/

➢ serveur local

Principaux points à connaître après l'installation :

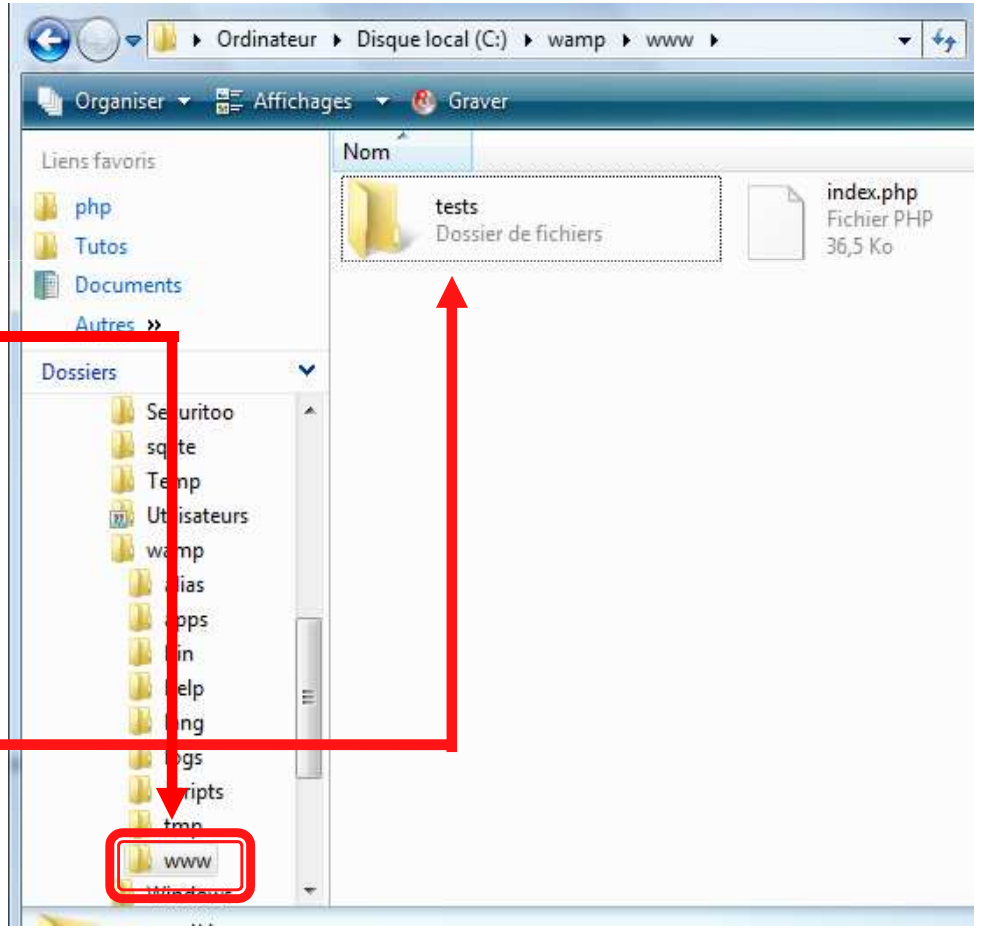
♦ **Où vont se trouver les fichiers et dossiers du site ?**

- Par défaut, le dossier du site se nomme **www** et il est placé dans:

c:/wamp/www/

- le chemin physique du site est :

c:/wamp/www/tests/



➢ serveur local

Principaux points à connaître après l'installation :

♦ **Comment voir le site avec le serveur ?**

- A l'aide du navigateur: Taper : http://localhost/.../ ou http://127.0.0.1/ sur la barre d'adresse.
- A partir de l'icône wampserver de la barre des tâches : cliquer sur localhost puis dossier du site.

♦ **fichier index.php**

- C'est le fichier par défaut ou prioritaire du site.
- Il s'exécute directement: lorsqu'on clique dans localhost sur le dossier du site, ou si on tape les noms du serveur et du répertoire dans le navigateur.

http://localhost/tests/ <=> http://localhost/tests/index.php

➢ intégration de PHP dans une page HTML

- Les pages web sont au format html.
- Les pages web dynamiques générées avec PHP sont au format php. Le code source php est directement inséré dans le fichier html grâce aux balises php.

  Lorsque PHP traite un fichier, il cherche les balises d'*ouverture* et de *fermeture*, qui délimitent le code qu'il doit interpréter.
- De cette manière, PHP peut être intégré dans toutes sortes de documents, car tout ce qui se trouve en dehors des balises ouvrantes / fermantes de PHP est ignoré.

➢ Balises du code PHP

- Tout script PHP doit être entouré de balises spécifiques. Elles permettent à l'interpréteur de savoir à quel endroit dans le fichier se trouve du code PHP à exécuter.

- **P**lusieurs paires différentes de balises ouvrantes / fermantes pour **référencer du code php**:

  - *<?php* et *?>*

  - *<script language="php">* et *</script?>* (rarement utilisés)

  - *<?* et *?>* (tags courts)

  - *<%* et *%>* (tag issus du langage ASP-rarement utilisés)

  - *<?=* et *?>* (très rarement utilisés)

➢ Balises de code PHP

► Les moyens de référencer du code php

- Deux de ces balises sont toujours disponibles:

❖ standard:

<?php ……. ?>

assurent une portabilité totale sur tous les serveurs et toutes les versions de PHP.

Ce sont les tags par défaut du langage PHP

❖ marqueur script html:

<script language="php"> ……. </script>

➢ Balises de code PHP

► Les moyens de référencer du code php

❖ les balises courtes: (*short-tags*)

<? ..... ?>

pourraient empêcher l'exécution des scripts du fait :

- La directive *short_open_tag* du php.ini est placée à la valeur *off*, ce qui désactive l'utilisation de ces balises.

- Confusion avec la balise d'ouverture d'un fichier XML:
<?xml version="1.0" encoding="utf-8" standalone="yes"**?>**

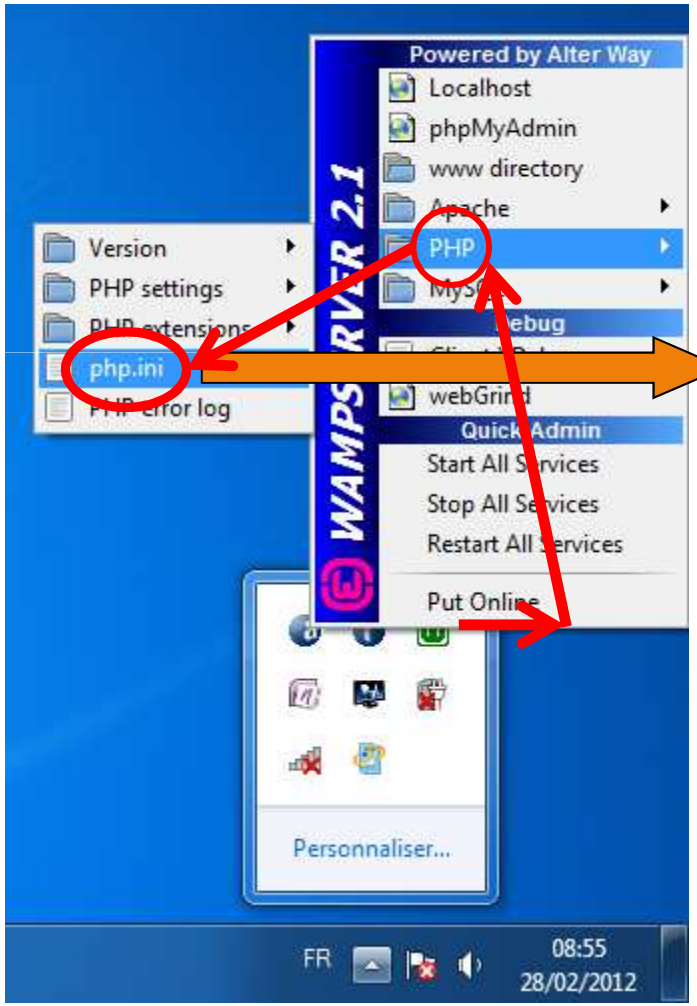
Message d'erreur renvoyé:

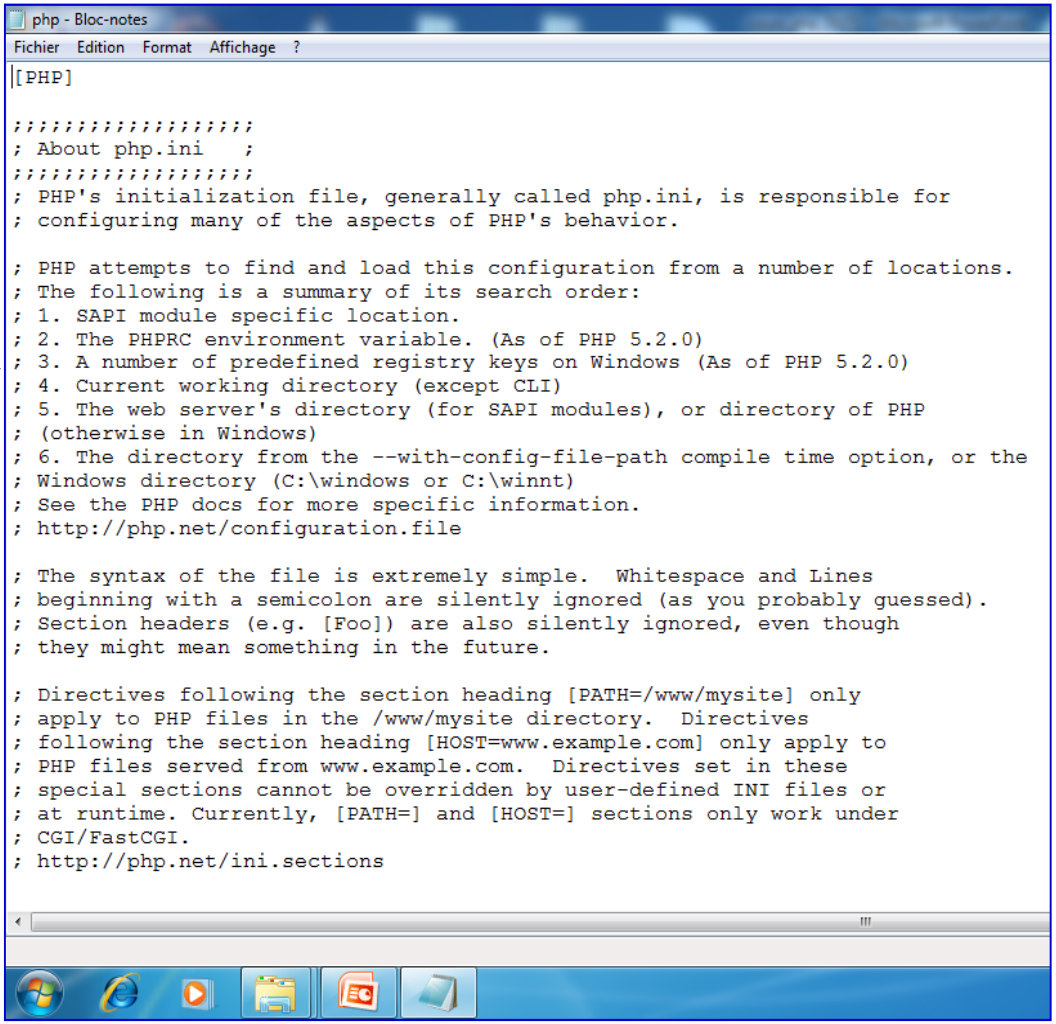
Parse error: syntax error, unexpected T_STRING in

… /WAMP/www/Test.php on line 1

➢ Balises de code PHP

► accès au fichier php.ini





➢ Balises de code PHP

► Les moyens de référencer du code php

❖ les balises du style ASP:

<%php ……. %>

doivent être activées depuis le fichier php.ini. (asp_tags=On)

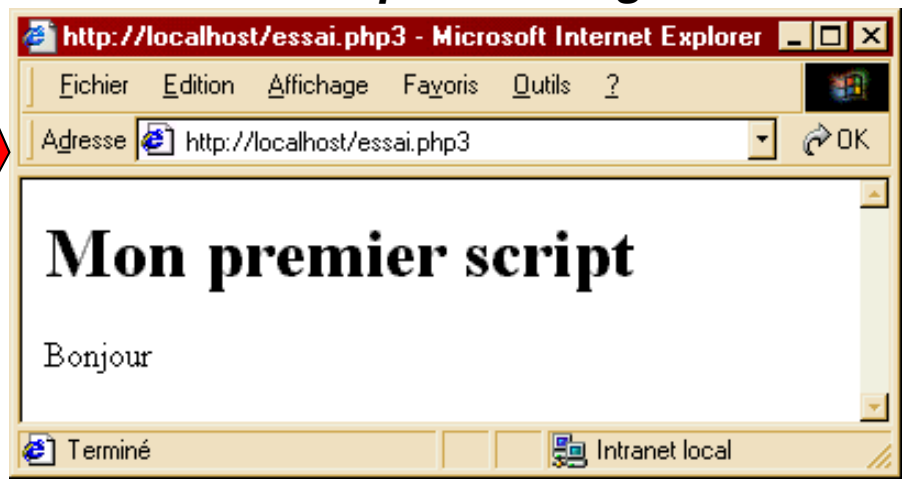
Elles sont moins portables donc, non recommandées.

➢ Balises de code PHP

Exemple de script, code source (côté serveur) :

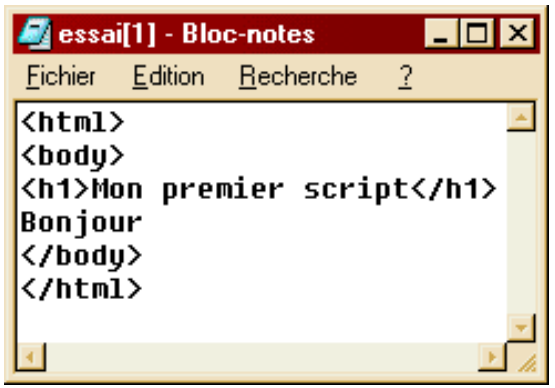
```
<html>
  <body>
     <h1>Mon premier script</h1>
     <?php echo ''Bonjour\n''; ?>
  </body>
</html>
```

Résultat affiché par le navigateur :



Code source de la page

Le code php a généré du html



intégration d'un script php dans une page html

- Il est tout à fait possible de mélanger, au sein d'une même page WEB, des instructions HTML et des instructions PHP.
- avec PHP, on écrit une page HTML avec du code PHP inclus à l'intérieur afin de réaliser une action précise
- Lorsque le serveur web rencontre dans une page web les balises **<?php … ?>**, il passe en "mode PHP".
- Lorsqu'on insère le moindre petit bout de code PHP dans une page HTML, on doit changer l'extension de ce fichier en .php (Si on dispose d'une page nommée **page.htm** et qu'on y insère du code PHP, il faut la renommer en **page.php**).

➢ intégration d'un script php dans une page html

Exemple de script intégré dans du html

```
<html>
  <body>
     <h1>Mon premier script</h1>
     <?php echo ''Bonjour\n''; ?>
  </body>
</html>
```

Autre écriture du même script :

```
<?php
    echo "<html>\n<body>\n";
    echo "<h1>Mon premier script</h1>\n";
    echo "Bonjour\n";
    echo "</body>\n</html>\n";
?>
```

➢ intégration d'un script php dans une page html

Exemple de script intégré dans du html

```
<?php
    $title = "Bonjour tout le monde!";
    $text = "Bon courage.";
?>
<html>
   <head>
        <title><?php echo $title; ?></title>
   </head>
   <body>
        <h1><?php echo $title; ?></h1>
        <p><?php echo $text; ?></p>
   </body>
</html>
```

➢ principes de base

- PHP utilise certaines notions de programmation : variables, séquentialité, conditions, boucles, tableaux, fonctions …
- Toute instruction (sauf les structures de contrôle et sauf la dernière ligne de code) doit se terminer par un " ; "
- Un script php se commente comme en C :
  - // : pour une seule ligne
  - /* */ : multilignes
  - # : pour une seule ligne

➢ principes de base

♦ Sensibilité à la casse. En règle générale :

- Les noms de variables sont sensibles à la casse
- Les autres éléments du langage (fonctions, instructions...) ne le sont pas.

exemple: print , Print , pRint , prinT … toutes acceptées

Important**:** il est fortement recommandé de respecter la casse dans :

► tous les liens HTML

► les noms des tables MySQL (ce sont des fichiers)

♦ Les blocs d'instructions (une à plusieurs instructions) peuvent être identifiés, délimités par des accolades " { } " .

➢ variables, types et opérateurs

- **informatique:** manipulation de l'information
- **Pour manipuler cette information ?**

  stockée en mémoire ------> variable
- **Une variable est une entité qui contient une information:**
  - elle possède un nom, on parle d'**identifiant**
  - elle possède une **valeur**
  - elle possède un **type** qui caractérise l'ensemble des valeurs que peut prendre la variable

Une variable possède un identifiant, une valeur et un type

- L'ensemble des variables sont stockées dans la mémoire de l'ordinateur

➢ variables, types et opérateurs

- Il existe différents types de variables en PHP:
  - les variables définies par le programmeur ;
  - les variables d'environnement ;
  - les variables de sessions etc …
- Les variables commencent obligatoirement par un **$** puis une lettre ou un _ , puis une suite de lettres, chiffres (_).

exemple :

$1var : écriture fausse $_1var : écriture juste

- Par convention, un nom de variable ne commence pas par une majuscule. S'il faut plusieurs mots pour composer le nom, ils sont habituellement séparés par des soulignés (_).

➢ variables, types et opérateurs

Deux types de données élémentaires ...

- **variables**:

  sa valeur peut changer durant le déroulement d'un algorithme: donnée élémentaire

  ex : $somme_initiale

- constantes:

  elle ne prend qu'une unique valeur durant l'exécution de l'algorithme.

  ex : taux de TVA unique
  Const_I=10

➢ variables, types et opérateurs

identifiant et valeur d'une variable ...

- **identifiant**:

  C'est le nom que le programmeur a donné à la variable:

  simple, parlant, sans accent ni caractères spéciaux.

- **valeur**:

  - une variable possède une **valeur**
  - elle est de même type que la variable

- **Le type d'une variable caractérise** :
  - l'ensemble des **valeurs** que peut prendre la variable
  - l'ensemble des **actions** que l'on peut effectuer sur une variable

➢ variables, types et opérateurs

Déclaration de variables ...

En programmation, les variables doivent être déclarées avec leur type pour être utilisée.

La déclaration d'une variable permet de réserver de l'espace mémoire pour stocker les données (espace dépend du type de ces données: entiers, réels, caractères, ...)

- Une variable est une **zone mémoire** qui porte un nom choisi par le programmeur pour faciliter sa programmation.
- Le **nom** de la variable est une **adresse mémoire**.

➢ variables, types et opérateurs

Déclaration de variables :

PHP ne contient de partie déclarative clairement définie. Pour déclarer une variable, il suffit de l'initialiser. Dés lors elle est accessible.

- En PHP la déclaration de variables est implicite, c'est-à-dire la déclaration se fait en attribuant une valeur à la variable.
- En PHP, les variables ne sont pas typées.

  C'est le contenu de la variable, en cours de traitement, qui permettra d'assigner à la variable le type le plus approprié.
- On parle de ***chargement*** de variables, et on appelle cette technique le "**typage dynamique**".

➢ variables, types et opérateurs

exemple :

```
<?php
    $var=1 ;
    print ($var);          ==>  affiche 1 (var de type int)
    $var="salam" ;
    print ($var);          ==>  affiche salam (ch. de caract.)
    $var=1.66 ;
    print $var;            ==>  affiche 1.66 (type double)
?>
```

➢ variables, types et opérateurs

♦ Les types des variables en PHP :

boolean : "vrai" ou bien "faux", "1" ou "0" ... ;

integer : valeur numérique entière ;

float-double : valeur numérique flottante (à virgule) ;

string : chaîne de caractères (texte) ;

array : tableau (ensemble de valeurs) ;

object : objet (instance de classe) utilisé en POO ;

resource : une ressource (type abstrait, inutilisable par le programmeur, utilisé uniquement pour des fonctions). Par exemple une variable permettant d'identifier une connexion à une base de données;

NULL : désigne l'absence de valeur : valeur/variable vide ou inexistante

➢ variables, types et opérateurs

- On peut incruster une variable dans une autre variable:

```
$var1 = 'wa sahlane!';
$var2 = "Ahlane $var1";
echo $var2;    // Affiche: Ahlane wa sahlane!
```

- Une variable peut avoir pour identificateur (nom) la valeur d'une autre variable.

Syntaxe :

${$var} = valeur;

exemple :

```
$Mois = "Avril";
${$Mois} = 4;
echo $Avril;    // la variable $Avril vaut 4
```

➢ variables, types et opérateurs

Les chaînes de caractères:

- Une variable chaîne de caractères n'est pas limitée en nombre de caractères. Elle est toujours délimitée par des simples quotes ou des doubles quotes.

exemple :

```
$nom = "Botoel";
$prenom = 'walid';
```

- Comme on a vu les doubles quotes permettent l'évaluation des variables et caractères spéciaux contenus dans la chaîne alors que les simples ne le permettent pas.

➢ variables, types et opérateurs

Les Opérateurs :

- Un **opérateur** est un symbole d'opération qui permet d'agir sur des variables ou de faire des "calculs"

- Une **opérande** est une entité (variable, constante ou expression) utilisée par un opérateur

- Une **expression** est une combinaison d'opérateur(s) et d'opérande(s), elle est évaluée durant l'exécution de l'algorithme, et possède une valeur (son interprétation) et un type

➢ variables, types et opérateurs

Les Opérateurs :

Les opérateurs sont les mêmes que ceux utilisés dans C:

- Les opérateurs arithmétiques : + , - , * , / , %
- Les Opérateurs unaires : ++ , --
- Les opérateurs logiques : && , || , !
- Les opérateurs de comparaison : == , != , > , < , >= , <=
- L'Opérateur ternaire/conditionnel :

(condition) **?** (expr1) **:** (expr2)

- Les opérateurs d'affectation …

➢ variables, types et opérateurs

Les Opérateurs :

- Les Opérateurs unaires : **++ , --**

En programmation, on utilise souvent des expressions d'incrémentation et de décrémentation qui augmentent ou diminuent de 1 la valeur d'une variable.

exemple:

i = i + 1, n = n – 1

Ces actions peuvent être réalisées par les opérateurs "++ et -- " :

++ incrémentation

- - décrémentation

➢ variables, types et opérateurs

Les Opérateurs :

- Les Opérateurs unaires : **++ , --**

Ces opérateurs incrémentent / décrémentent de 1 la valeur opérande

i++; ou ++i; correspondent à i=i+1;

i-- ; ou --i; correspondent à i=i-1;

➢ variables, types et opérateurs

Les Opérateurs :

Les opérateurs sont les mêmes que ceux utilisés dans C:

- Les opérateurs logiques : **&&** , **||** , **!**

&&	**ET logique (AND)**
||	**OU logique (OR)**
!	**négation logique (NOT)**

➢ variables, types et opérateurs

Les Opérateurs :

Les opérateurs sont les mêmes que ceux utilisés dans C:

- Les opérateurs logiques : **&&** , **||** , **!**

table de vérité:

! NON

F	V
V	F

||

	F	**V**
F	F	V
V	V	V

&&

	F	**V**
F	F	F
V	F	V

➢ variables, types et opérateurs

Les Opérateurs :

Les opérateurs sont les mêmes que ceux utilisés dans C:

- L'Opérateur ternaire/conditionnel :

(condition) ? (expr1) : (expr2)

Si la condition est vraie (non nulle), le résultat est expr1, sinon le résultat est expr2.

exemple:

Max = (A>B) **?** A **:** B ;

si condition (A>B) vrai alors Max=A sinon Max=B

➢ variables, types et opérateurs

Les Opérateurs :

exemples:

```
<?php
    $a=20 ; $b=5 ; $c=-10 ; $d=2 ; $x=12 ;
    $a += ($x+5) ;
    echo($a!=($c*=(-$d)));
    $a*=$c+($x-$d) ;
    $a %= $d++ ;
    $a %= ++$d ;
   echo '$a='.$a.' $c='.$c.' $x='.$x.' $d='.$d.'<br>';
?>
```

➢ variables, types et opérateurs

➢ La postincrémentation et la préincrémentation

* Postincrémentation (postdécrémentation):

$j = $i++ ; $j = $i-- ;

affecte d'abord la valeur de i à j, et ensuite incrémente i

* Préincrémentation (prédécrémentation):

$j = ++$i ; $j = --$i ;

la variable est incrémentée ou décrémentée avant l'affectation.

C'est à dire que:

$i++ ($i--) permet de retourner la valeur de $i puis d'ajouter (enlever) 1 à $i

++$i (--$i) va ajouter (enlever) 1 à $i puis retourner la nouvelle valeur de $i

➢ variables, types et opérateurs

➢ La postincrémentation et la préincrémentation

* Postincrémentation (postdécrémentation):

$j = $i++ ; $j = $i-- ;

affecte d'abord la valeur de i à j, et ensuite incrémente i

* Préincrémentation (prédécrémentation):

$j = ++$i ; $j = --$i ;

la variable est incrémentée ou décrémentée avant l'affectation.

$i = 2 ;
$j = ($i++) + 3 ;
➔ $j contient 5 et $i contient 3
$i = 2 ;
$j = (++$i) + 3 ;
➔ $i contient 3 et $j contient

➢ variables, types et opérateurs

➢ exemples :

Soient les valeurs: a=20; b=5; c=-10; d=2; x=12; y=15;

Quels résultats donnent les instructions suivantes:

1) a += (x+5) ;

a=a+(x+5) ⇔ a=20+(12+5)=37

2) a != (c *=(-d))

a=37 et c=c*(-d)=-10*-2=20 ⇔ a != (c *= (-d)) →1 (vrai)

3) a *= c+(x-d) ;

a=a*(c+(x-d)) ⇔ a=37*(20+(12-2))=37*(30)=1110

4) a %= d++ ;

a=a%d puis d=d+1 ⇔ a=0 , d=3

5) a %= ++d ;

d=d+1 puis a=a%d ⇔ d=4 , a=0

➢ variables, types et opérateurs

exercice

Soit un programme qui initialise les variables A, B, C, D, X et Y à 2, 3, 4, 5, 6 et 7. Quels résultats donnent les relations suivantes:

1) A+=(--X+2) ;	A=9	B= 3	C= 4	D=5	X=5	Y=7
2) A!=(C*=(-D))) ;	A=9	B= 3	C=-20	D=5	X=5	Y=7
3) B*= C+(X++-D);	A=9	B=-60	C=-20	D=5	X=6	Y=7
4) C/=++D ;	A=9	B=-60	C=-3	D=6	X=6	Y=7
5) (X++)*(A+C)) ;	A=9	B=-60	C=-3	D=6	X=7	Y=7
6) X=++A*(B<C)+Y--*!(B<C)	A=10	B=-60	C=-3	D=6	X=10	Y=6

➢ variables, types et opérateurs

L'Opérateur de concaténation de chaînes : . (point)

Il est utilisé pour concaténer des chaînes, variables etc …

exemples :

- print("Il est $date".":"gmt.");

```
// $date est une variable et gmt. un texte
```

- <?php

```
$nom = " Said "; $i=1;
echo 'Mon nom est '.$nom.<br>;
// affiche : Mon nom est Said puis saut de ligne
print '<br>test numero: '.++$i.'<br>' ;
// incrémentation, saut de ligne puis affichage
```

?>

➢ variables, types et opérateurs

♦ fonctions utiles sur les variables :

empty($var) : renvoie vrai si la variable est vide

isset($var) : renvoie vrai si la variable existe

unset($var) : détruit une variable

gettype($var): retourne le type de la variable

settype($var, "type") : permet de modifier le type d'une variable pendant l'exécution du programme au type **type** (cast)

var_dump() : affiche le type d'une variable et son contenu (ainsi que sa taille si c'est une chaîne)

➢ variables, types et opérateurs

♦ fonctions utiles sur les variables :

is_numeric()	La variable est de type numérique ?
is_int()	La variable est un entier ?
is_integer()	
is_long()	
is_double()	La variable est un nombre décimal ?
is_real()	
is_float()	
is_string()	La variable est un chaîne de caractères ?
is_array()	La variable est une liste ?
is_object()	La variable est un objet ?
is_bool()	La variable est un booléen ?
is_null()	La variable est nulle?
is_resource()	indique si la variable est une ressource.
is_scalar()	La variable est scalaire ?

La réponse à toutes ces questions sera "Vrai" ou "Faux".

➢ variables, types et opérateurs

♦ fonctions utiles sur les variables :

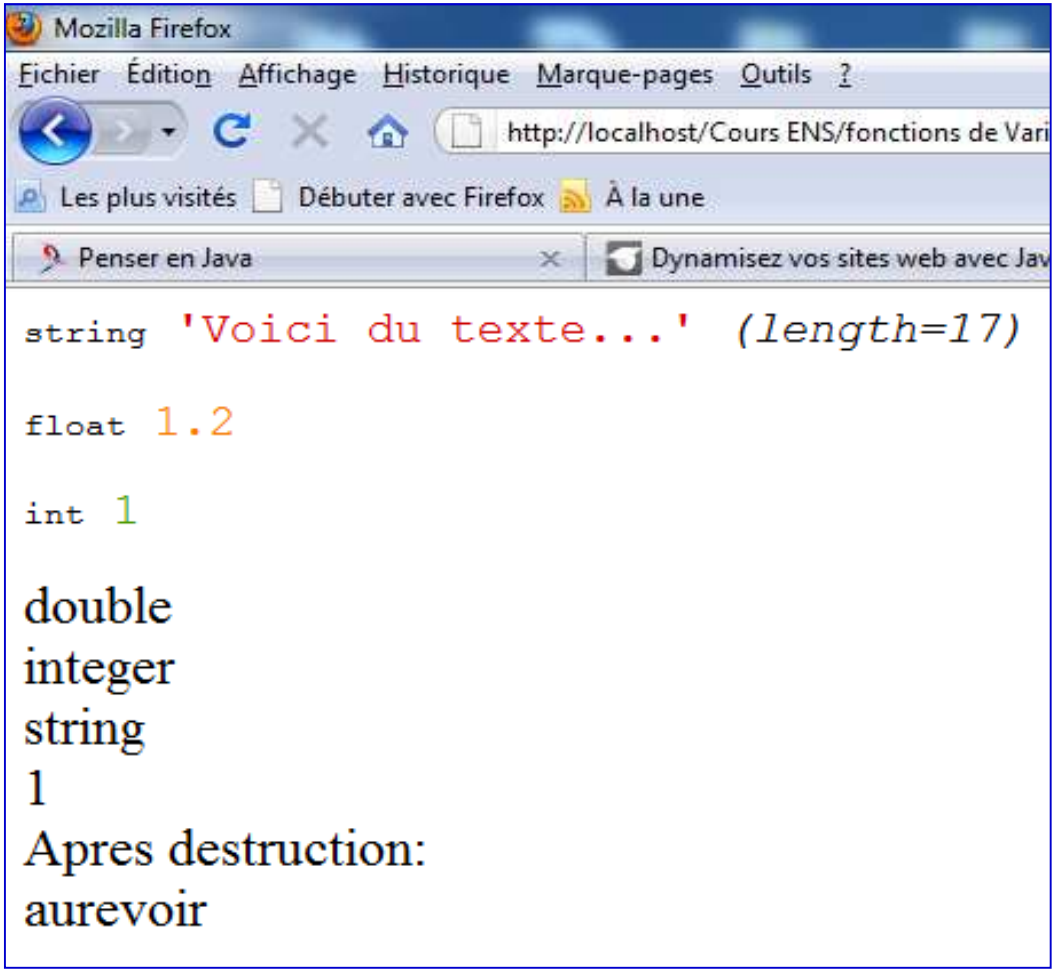
exemples:

```
<?php
    echo var_dump("Voici du texte...").'<br>';
    echo(var_dump(1.2).'<br>');
    print(var_dump(1)).'<br>';
    $var=8.9;
    echo gettype($var).'<br>';
    $var=9;
    echo gettype($var).'<br>';
    $var='texte';
    echo gettype($var).'<br>';
    echo isset($var);
    unset ($var);
    echo ("<br> Apres destruction: ".isset($var).'<br>');
    echo "aurevoir";
?>
```

➢ variables, types et opérateurs

♦ fonctions utiles sur les variables :

exemples:



➢ variables, types et opérateurs

♦ fonctions sur les chaînes :

On dispose de plusieurs fonctions prédéfinies pour effectuer diverses opérations sur les chaînes de caractères.

- Recherche.
- Comparaison.
- Extraction.
- Remplacement.
- Suppression.
- Ajout.

➢ variables, types et opérateurs

♦ fonctions sur les chaînes :

$resultat= chop($chaine);

Supprime les espaces blancs en fin de chaîne.

$liste = explode($delimiteur, $chaine);

Transforme une chaîne de caractères en une liste.

$chaine_result = implode($delimiteur, $liste);

Concatène tous les éléments d'une liste dans une chaîne.

$nombre = strpos($chaine, $caractere [, $depart]);

Retourne la position de la première occurrence d'un caractère dans une chaîne.

$nombre = strrpos($chaine, $caractere);

Position de la dernière occurrence d'un caractère dans une chaîne.

$chaine_result = strrev($chaine);

Inversion de l'ordre des caractères d'une chaîne.

➢ variables, types et opérateurs

◆ Variables spéciales php:

- $_POST: Les valeurs envoyées par formulaire ;
- $_GET: Les valeurs provenant de l'URL ;
- $_FILES: Les fichiers envoyés par formulaire ;
- $_ENV: Les variables d'environnement (syst. exploitation)
- $GLOBALS
- $_SERVER
- $_COOKIE
- $_REQUEST
- $_SESSION

➢ constantes

La déclaration des constantes se fait au moyen de la fonction define()

Syntaxe:

```
define("var",val)
```

définit la constante **var** de valeur **val**

- Les constantes ne portent pas le symbole **$** (dollars) en début d'identificateur et ne sont pas modifiables.

- A la déclaration, si la valeur d'une constante est une chaîne de caractères, il faut l'entourer d'apostrophes ou de guillemets doubles.

➢ constantes

exemple 1 :

```
define("Nom","Mansouri");

echo Nom;                    // affiche Mansouri
```

exemple 2 :

```
define('Naissance',1990);

echo Naissance;              // affiche 1990
```

➢ constantes

- Contrairement aux variables, les identificateurs de constantes (comme ceux des fonction) ne sont pas sensibles à la casse.

- Les constantes sont définies et accessibles à tout endroit du code, globalement.

➢ constantes

exemple :

Utilisation des constantes dans des expressions

```
<?
    define("pi",3.14159265);
    define('e',2.71828183);
    define('unite',' metres');
    $diametre = 10 ;
    $perimetre = $diametre * pi;
    print ("Perimetre = ".$perimetre.unite.".\n");
    $surface = ($diametre * $diametre) / 4 * pi;
    print ("Surface = ".$surface.unite." carres.\n");
?>
```

constantes

Constantes prédéfinies:

php dispose d'un certain nombre de constantes prédéfinies.

```
<?php
    print "Constantes predefinies: <br>";
    print ('- __FILE__ est le nom du fichier programme: '.__FILE__);
    print ('- __LINE__ est le numero de la ligne courante: '.__LINE__);
    print ('- PHP_VERSION est le n°de la version PHP: '. PHP_VERSION );
    print ('- PHP_OS est le système sous lequel on travaille: '.PHP_OS);
    print ('- TRUE est toujours la valeur vraie: '.TRUE."<br>");
    print ('- FALSE est toujours la valeur fausse: '.FALSE);
?>
```

constantes

Constantes prédéfinies:

Constantes predefinies :

- __FILE__ est le nom du fichier programme : C:\wamp\www\Cours ENS\Var-Cste-echo.php\Constantes_Predefinies.php

- __LINE__ est le numero de la ligne courante : 4

- PHP_VERSION est le numero de la version PHP : 5.2.0

- PHP_OS est le systeme sous lequel on travaille : WINNT

- TRUE est toujours la valeur vraie : 1

- FALSE est toujours la valeur fausse :

➢ Les variables d'environnement

- PHP propose toute une série de variables qui sont déjà implantées dans le langage sans qu'on ait à les créer. On les appelle les **variables d'environnement**.
- Ces variables sont des données stockées dans des variables permettant au programme d'avoir des informations sur son environnement (serveur et client).
- Elles permettent notamment d'avoir des informations sur le type de serveur, son administrateur, la date à laquelle le script a été appelé, l'adresse IP et le type de navigateur du client…….

➢ Les variables d'environnement

- Ces variables sont créées par le serveur à chaque fois que le script PHP est appelé, le serveur les lui fournit en paramètres cachés lors de l'exécution de l'interpréteur.

- Ces variables appartiennent à la famille des variables globales $_SERVER

- La liste de ces variables est obtenue à partir de la fonction phpinfo(). (accès à partir du serveur: code ou icône …) :

```
<?php
        phpinfo() ;
 ?>
```

dans la page affichée les variables se trouvent sous le titre: php Variables

➢ Les variables d'environnement

exemples :

$_SERVER["HTTP_HOST"] :

affiche l'hôte, c' est à dire le nom de votre espace web

$_SERVER["HTTP_REFERER"] :

affiche la provenance de votre visiteur, l'url d' ou celui ci arrive

$_SERVER["DOCUMENT_ROOT"] :

le répertoire racine de l'arborescence des document sur le serveur

$_SERVER["REQUEST_METHOD"] :

la méthode utilisée, GET, POST, pratique pour vérifier les variables provenant d'un formulaire.

$_SERVER["REMOTE_ADMIN"] :

donne l'adresse de l'administrateur du serveur.

$_SERVER["REMOTE_ADDR "] :

donne l'adresse IP du client ou visiteur.

➢ Les variables d'environnement

exemple :

```
<?php
  echo'1) <u><b>Variables d\'environnement:   $_SERVER[\"DOCUMENT_ROOT\"]
<br> </b> </u>';
  echo'<blockquote><b>Affiche le répertoire racine de l\'arborescence des document
sur le serveur, ici:  ';
  echo $_SERVER["DOCUMENT_ROOT"].'<br></blockquote>' ;
  echo'2) <u>Variables d\'environnement:   $_SERVER[\"REQUEST_METHOD\"]
<br></u>';
  echo'<blockquote>Affiche la méthode utilisée , GET ,POST , pratique pour vérifier les
variable provenant d\'un formulaire, ici:  ';
  echo $_SERVER["REQUEST_METHOD"].'<br></blockquote>' ;
  echo'3) <u>Variables d\'environnement:   $_SERVER[\"HTTP_REFERER\"]<br></u>';
  echo'<blockquote>Affiche la provenance de votre visiteur, l\'url d\'où il arrive, ici:  ';
  echo $_SERVER["HTTP_REFERER"].'<br></blockquote>' ;
  echo '4) <u>Variables d\'environnement:   $_SERVER[\"HTTP_HOST\"]<br></u>';
  echo '<blockquote>Affiche l\'hôte, c.a.d le nom de votre espace web, ici:  ';
  echo $_SERVER["HTTP_HOST"].'<br></blockquote>' ;
  echo'5) <u>Variables d\'environnement:   $_SERVER[\"QUERY_STRING\"] <br></u>';
  echo'<blockquote>Affiche le contenu de ce qui suit l\'url de la page, par exemple
page.php?url=az-php.com , affichera url=az-php.com, ici:  ';
  echo $_SERVER["QUERY_STRING"] ;
?>
```

➢ Les actions élémentaires

Les ordinateurs, quels qu'ils soient , ne sont capables d'exécuter que quatre opérations logiques.

Ces opérations sont :

- l'affectation de variables
- la lecture / écriture
- les tests ou structures conditionnelles
- les boucles ou structures itératives

➢ Les actions élémentaires

L'affectation : =

C'est l'action élémentaire principale qui permet de donner une valeur à une variable ou de modifier la valeur d'une variable

Syntaxes :

```
nom_variable = expression ;
```

```
nom_variable = valeur ;
```

```
nom_variable = nom_variable ;
```

➢ Les actions élémentaires

L'affectation : =

exemples:

- $a = 0 ;
- $b = $n ;
- $i = $i + 1 ;
- $r = $r * $r ;
- $interet ← $somme_initiale * $taux / 100 ;

➢ Les actions élémentaires

➢ Les opérations d'entrée/sortie

Un programme doit absolument pouvoir Saisir, Lire des données pour les stocker et/ou les utiliser dans des calculs

Comme il doit pouvoir afficher , ecrire les données stockées et/ou les résultats des calculs qu'il a effectué si l'utilisateur en fait la demande.

==➢ C'est le rôle des opérations d'entrée/sortie

➢ les instructions d'affichage

♦ les fonctions: echo - print()

la commande echo() est une construction de php.

Elle permet d'afficher à l'écran des chaînes de caractères, qui peuvent être définies directement par l'utilisateur ou qui peuvent être des contenus de variables.

exemple : *différentes écritures des deux fonctions*

```
$nom=said ;
echo $nom ;                    print $nom ;
echo 'Salam' ;                 print 'Salam' ;
echo "Salam" ;                 print "Salam";
echo 'Salam','said' ;
```

plusieurs paramètres séparées par ,

➢ les instructions d'affichage

♦ les fonctions: echo - print()

On peut aussi les écrire avec des ( )

***exemple* :**

```
<?php
    $nbr=1 ;
    echo($nbr);
    echo('bonjour');
    echo("bonjour");
?>
```

```
<?php
    $nbr=1 ;
    print($nbr);
    print('bonjour');
    print("bonjour");
?>
```

➢ les instructions d'affichage

♦ les fonctions: echo - print()

Utiliser " " ou ' ' ?

- ' ' et " " sont utilisés pour délimiter des chaînes de caractères.

- PHP examinera ce que contient une chaîne entre " ", mais pas une chaîne qui est entre ' ' qu'il affichera directement.
  ==> ' ' annule le remplacement des variables par leurs valeurs.

- echo "Salam $nom"; ==> affiche Salam Said
  la variable est remplacée par sa valeur.

- echo 'Salam $nom' ; ==> affiche Salam $nom

- print 'Salam $var' ; ==> affiche Salam $var

$nom et $var sont interprétées comme des chaînes de caractères.

➢ les instructions d'affichage

♦ la fonction: printf()

- php utilise aussi la fonction d'affichage formatée **printf** du langage c.

➢ les instructions d'affichage

♦ la fonction: printf()

```
<?php
    $val=pi();
    printf($val); echo'<br>';
    printf("%d",-$val); echo'<br>';
    printf("%'>3d",$val); echo'<br>';
    printf("%'+3d",$val); echo'<br>';
    printf("%+03d",$val); echo'<br>';
    printf("%06.2f",$val); echo'<br>';
    printf("%o",(int)($val*10)); echo'<br>';
    printf("%012b",(int)($val*10)); echo'<br>';
    printf("%c",(int)($val*52));
?>
```

```
3.1415926535898
-3
>>3
++3
+03
003.14
37
000000011111
£
```

➢ Portée des variables

- La portée d'une variable dépend de son emplacement.
- Les variables ont une portée globale si elles sont définies en dehors d'une fonction, sinon, elles sont locales.
- Il existe trois niveaux de définition de variables :
  - Le niveau global. Il définit des variables dans l'intégralité du code d'une page PHP. L'accès à une variable globale à l'intérieur d'un bloc n'est pas automatique. Il faut explicitement autoriser son accès dans le bloc.
  - Le niveau local. Il définit des variables propres à une fonction, dont la durée de vie ne dépasse pas le temps de cette fonction.
  - Le niveau static. Il définit des variables propres à une fonction, qui persistent pendant l'intégralité du code de la page PHP.

➢ Portée des variables

exemple

Variable Globale

```
<?php
  $a=1;$b=2;
  function somme(){
    global $a,$b;
    $b=$a+$b;
  }
  Somme();
  echo $b;
?>
```

Pour que $a et $b soient reconnues dans la fonction, il faut les déclarer global dans la fonction où elles vont être utilisées

➢ Portée des variables

exemple

Variable locale

```
<?php
   function compteur() {
      $a=0;  // var locale
      $a++;
   }
?>
```

➢ Portée des variables

exemple

Variable static

```
<?php
   function
   compteur(){
      static $a=0;
      $a++;
   }
?>
```

À chaque retour à la fonction $a ne prendra pas la valeur 0, mais la dernière valeur après incrémentation
0 – 1 – 2 – 3 – 4 …