Langages pour le

Développement Web Avancé

Généralités sur le langage PHP

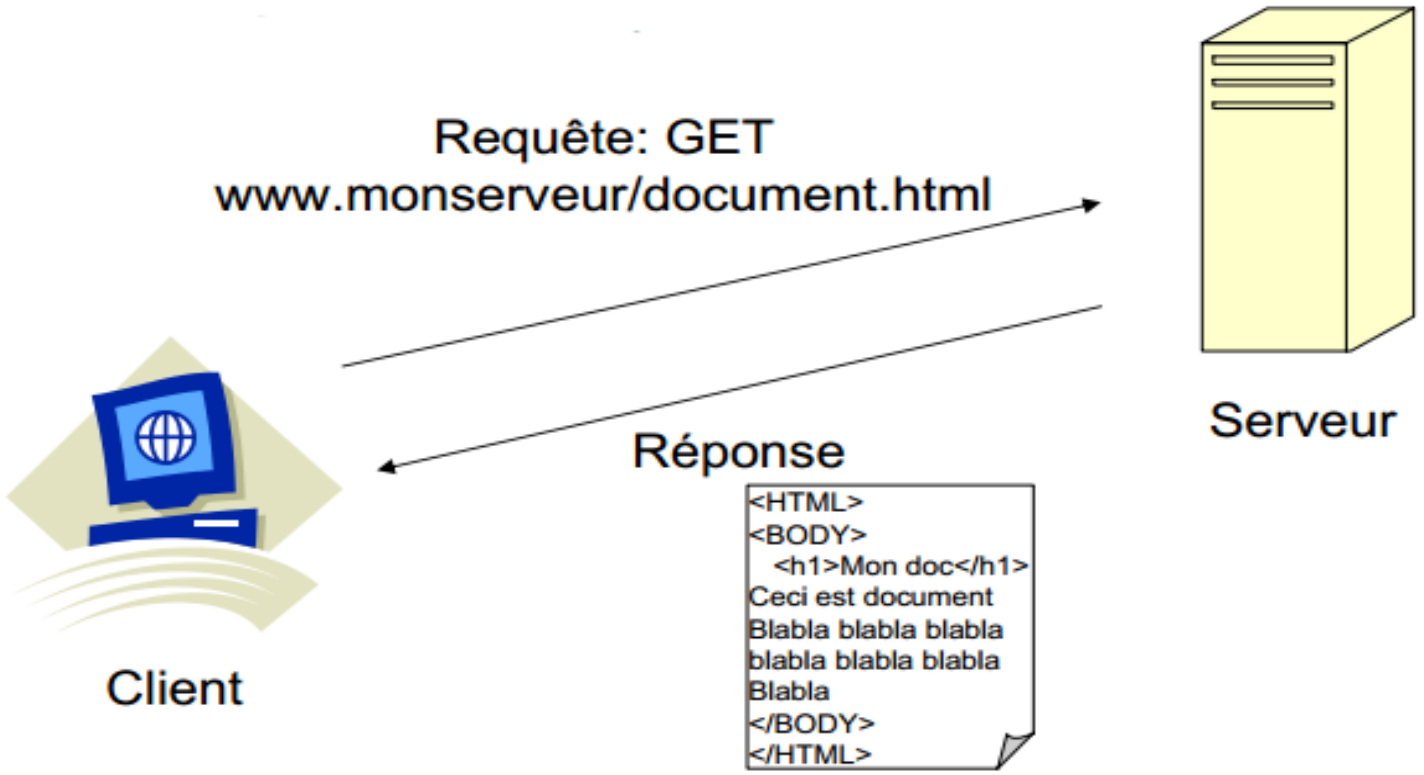
Dr. Yoro DIA

Séquence 1

- Pourquoi PHP?
- Présentation générale
- Premiers pas
  - Environnement
  - Bases du langage
- Activités pratiques

Un rappel: HTTP & HTML

- HTTP (HyperText Transfert Protocol): protocole d'accès au web



Un rappel: HTTP & HTML

- HTML (HyperText Markup Language):

  langage *statique* de description de pages web

- Les pages sont déposés sur le serveur…
- …et appelées à la demande

Oui mais…

- Site de vente en ligne
  - Un utilisateur sélectionne un produit
  - Le met dans un caddie
  - Et souhaite afficher le contenu du caddie
  - … le résultat est *dynamique* (« variable »)
  - … il faut donc être capable de construire des pages *à la volée*

PHP: caractéristiques

- « Personnal Home Pages » puis « PHP Hypertext Preprocessor »
- Langage de scripts proche du C, Perl, Java (mais plus simple!)
  - Qui s'intègre dans une page HTML
  - Traité par le serveur
  - *spécifiquement dédié* à la création dynamique de pages HTML
- Créé en 1994 par Rasmus Lerdof
  - Aujourd'hui: PHP version 8 (La version actuelle de PHP est la 8.2.4. La version 7 est sortie en Décembre 2015 et il n'y a jamais eu de version 6 !)
  - Développé en C

PHP: atouts majeurs

- PHP permet en outre de créer des pages interactives. Une page interactive permet à un visiteur de saisir des données personnelles.
- Ces dernières sont ensuite transmises au serveur, où elles peuvent rester stockées dans une base de données pour être diffusées vers d’autres utilisateurs.
- Un visiteur peut, par exemple, s’enregistrer et retrouver une page adaptée à ses besoins lors d’une visite ultérieure. Il peut aussi envoyer des e-mails et des fichiers sans avoir à passer par son logiciel de messagerie.

PHP: atouts majeurs

- Rapidité, facilité de prise en main et de développement
  - …car directement intégré dans les pages HTML
- Gestion d'outils avancés: sessions, cookies
- S'interface avec tous les SGBD
- Transparent vis-à-vis des OS et serveurs
  - Windows, Unix, Linux, MacOS
  - Apache, Microsoft IIS

PHP: atouts majeurs

- Grande communauté de développeurs
- Pour contenir la masse d'informations collectées, PHP s'appuie généralement sur une base de données, généralement MySQL, et sur des serveurs Apache.
- PHP est utilisé aujourd'hui par plus de la moitié des sites de la planète.

PHP: atouts majeurs

- Un programme PHP n'étant ni plus ni moins qu'un simple fichier texte contenant des lignes de code, il est interprétable sur tout type de machine ou de système d'exploitation sans que cela nécessite la modification de la moindre virgule : on appelle cela la « portabilité ».
- Le marché potentiel d'un programme écrit en PHP est par là même bien plus vaste que celui d'un programme compilé qui, généralement, n'est développé que pour un système d'exploitation et une architecture donnés.

Langages interprétés et langages compilés

- Il serait bien évidemment inutile et fastidieux de connaître plusieurs langages si ces derniers proposaient tous les mêmes fonctionnalités.
- Comme vous pouvez vous en douter, ce n’est pas du tout le cas : chaque langage dispose de ses particularités, de ses avantages et de ses défauts.
- La première grande différence à observer entre les différents langages se situe dans leur mode de fonctionnement : certains sont à classer parmi les langages interprétés et d’autres parmi les langages compilés.

Langages interprétés et langages compilés

- La différence entre ces deux types de langages se situe au niveau de leur mode d'exécution.
- Pour exécuter un programme écrit dans un langage non interprété, il est nécessaire de compiler le code source pour en faire un binaire.
- Le compilateur est le programme qui se charge de cette opération.
- Chaque langage dispose ainsi d'un compilateur qui lui est propre : celui du C s'appelle gcc, celui du Java se nomme javac.

Langages interprétés et langages compilés

- Pour exécuter un script écrit avec un langage interprété, il faut, comme son nom l'indique, passer par un interpréteur.
- Cet interpréteur lit le code pas à pas et le convertit au fur et à mesure en instructions pouvant être traitées par le processeur.
- Quand on parle de PHP, on parle donc à la fois du langage et de l'interpréteur.

Langages interprétés et langages compilés

Langages interprétés	Langages compilés
PHP	C
Perl	C++
Python	Java
JavaScript	Pascal

Langages interprétés et langages compilés

- **Script**
  - ➢Le terme « script » est souvent utilisé lorsque l'on souhaite faire référence à un programme écrit dans un langage interprété.
  - ➢On parle ainsi de script PHP.

Premier contact avec PHP

- **Ma première page PHP**

  - Un script ou code de PHP commence toujours par <?PHP et se termine par ?>. Un script(code) de PHP peut être placé n'importe où dans le document(page web).
  - Un bloc script ou code de PHP peut aussi commencer par <? et se terminer par ?>.
  - La première forme est la plus recommandée pour une compatibilité maximum, donc nous vous recommandons d’employez le format standard (<?PHP) plutôt que la forme (<? ?>)

Premier contact avec PHP

- La connaissance du langage HTML est utile pour se lancer dans l'écriture de scripts PHP.
- Il est utile de connaître la structure des fichiers HTML car une page dynamique PHP est bien un document HTML envoyé par le serveur vers le poste client.

Premier contact avec PHP

- PHP est à l'origine un langage de script conçu spécifiquement pour agir sur les serveurs web.
- En ajoutant quelques lignes de PHP à une page HTML, le serveur exécute les instructions correspondantes pour écrire du code HTML à la place.
- Le résultat (le code HTML initial ajouté à celui produit par PHP) est envoyé au navigateur.
- Cela permet par exemple d'afficher la date du jour à un endroit bien précis du visuel. On parle alors de page dynamique.

Premier contact avec PHP

- Dans l’exemple suivant, PHP ajoute une chaîne de caractères au milieu du code HTML

```
<html>
        <head>
                <title>Exemple</title>
        </head>
<body>
        <p>
                <?php
                        echo "Ceci est une syntaxe PHP";
                ?>
        </p>
</body>
</html>
```

Premier contact avec PHP

- Le script doit avoir une extension particulière pour être reconnu en tant que script PHP.
- L'extension la plus répandue est *.php* (par exemple *test.php*).
- Il est néanmoins possible de rencontrer d'autres extensions : *.php3* ou *.phtml*.
- L'hébergeur précise généralement quelle extension doit être utilisée.
- Si le choix vous est offert, utiliser l'extension *.php* semble plus logique car nous en sommes aujourd'hui à la version 8 de PHP.
- De plus, cette extension risque fort de devenir la norme.

Remarque

- PHP n’est qu’un langage de programmation parmi d’autre.
- PHP fait partie des langages interprétés.
- L’interprétation des scripts PHP est réalisée le plus souvent au niveau du serveur web dont le meilleur représentant est Apache.
- PHP est particulièrement adapté aux développements web.
- PHP fonctionne sur une multitude de plateformes.
- PHP est libre et gratuit.

Séquence 2

- **Les fondamentaux**
- **Les structures de contrôle**

Les fondamentaux

- **HTML et PHP**
  - Il serait tout à fait envisageable qu'un fichier *.php* ne contienne que du HTML.
  - La page serait retournée sans aucune modification.
  - L'intérêt est cependant bien mince car vous ajoutez, dans ce cas, une étape entre le serveur web et vous.
  - En effet, bien que la page ne contienne aucun bloc d'instructions, l'interpréteur sera quand même mis à contribution.

Les fondamentaux

- **Les commentaires**

  Un bloc de code peut contenir des commentaires. Ces derniers peuvent être ajoutés au code de différentes manières :

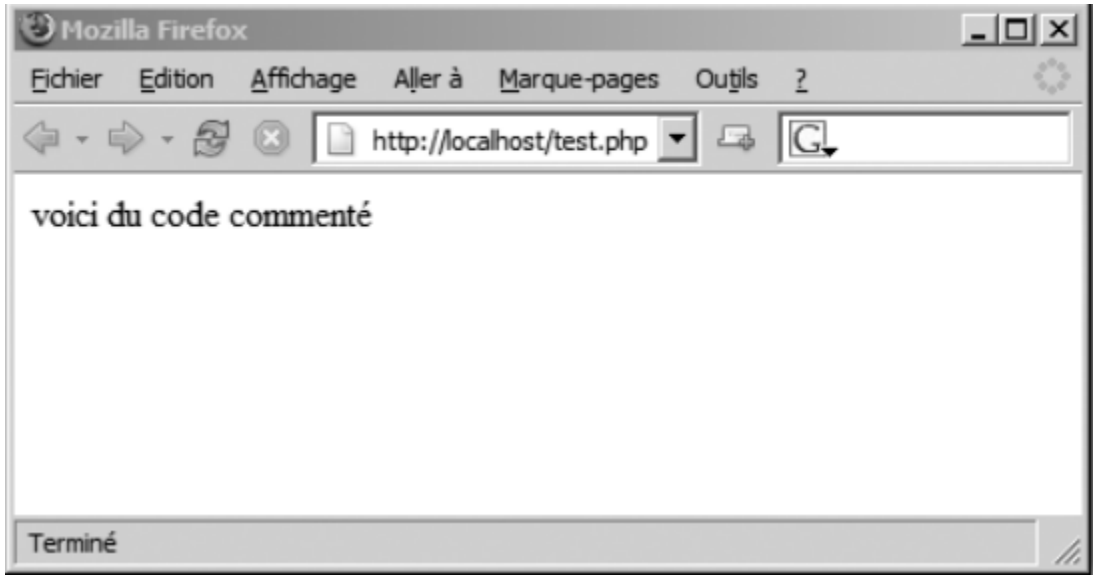
```
<?php
/* commentaires hérités du C */
// commentaires hérités
// du C++
# commentaires hérités
# du SHELL
print("code commenté"); // commentaire de fin de ligne
?>
```

Les fondamentaux

- **Les commentaires**

  ➢ Ajouter des commentaires ne ralentit en rien l'exécution de votre code.

  ➢ Au moment de l'interprétation, PHP va tout simplement supprimer ces zones commentées.



Les parties commentées n'apparaissent pas

Les fondamentaux

- **Les variables**
  - Quel que soit le langage de programmation, l'élément principal d'un programme est la variable.
  - Une variable est un élément qui peut prendre différentes valeurs au cours de l'exécution d'un programme.
  - En PHP, les noms des variables sont précédés du caractère $: $abc correspond à la variable abc.
  - $abc est ici le nom de la variable, à ne pas confondre avec ce que contient la variable ($abc peut par exemple contenir la valeur 3).

Les fondamentaux

- **Les variables**

- Précisément, pour donner la valeur 3 à la variable $abc, écrivez l'affectation suivante :

```
<?php
$abc = 3;
$abc += 5; // affecte la valeur 8 à la variable $abc correspond à l'instruction '
$abc = $abc + 5';
$z = "Bonjour ";
$z .= " tout le monde!";  // affecte la valeur "Bonjour tout le monde!" à
                          //  la variable $z
                          //  identique à $z = $z." tout le monde!";


?>
```

Les fondamentaux

- **Affectation par référence**

L'affectation par référence est aussi supportée, au moyen de la syntaxe "$var = &$othervar;".

L'affectation par référence signifie que les deux variables pointent vers le même conteneur de donnée, rien n'est copié nulle part.

```php
<?php
$a = 3;
$b = &$a; // $b est une référence à $a

print "$a\n"; // affiche 3
print "$b\n"; // affiche 3

$a = 4; // change $a

print "$a\n"; // affiche 4
print "$b\n"; // affiche 4 aussi, car $b est une référence à $a, qui a été changée
?>
```

Les fondamentaux

- **Les variables**
- Il existe différents moyens permettant d'affecter une valeur à une variable :
  - soit directement : $abc = 2;
  - soit en lui faisant recevoir le résultat d'une opération : $abc = 1 + 3;
  - soit en lui faisant recevoir le résultat d'une fonction : $abc = pow(2,4); (c'est-à-dire la puissance 4 de 2).

Les fondamentaux

- Il existe une notation alternative et plus complexe qui permet d'avoir accès aux variables :

${'nom_de_la_variable'}.

- **Exemple:**

```
$x = 0;
${'x'} = 10; // assigne la valeur 10 à la variable x
print(${'x'}); // affiche : 10
print($x); // affiche : 10
$x = 5; // assigne 5 à la variable x
print(${"x"}); // affiche : 5
$y = "x";
print(${$y}); // affiche 5 car $x vaut 5
```

Les fondamentaux

- L’instruction suivante est tout à fait valable, malgré la présence d’un espace dans le nom de la variable :

  ${"ma variable"} = "bonjour monde";

- **Majuscules et minuscules**

  Le nom des variables est sensible à la casse (*case sensitive*). Cela signifie que PHP fait une différence entre majuscules et minuscules. Une variable $a sera donc différente de la variable $A et pourra donc cohabiter avec la première en disposant de sa propre valeur.

Les fondamentaux

- **Les constantes**
  - Les constantes peuvent être assimilées à des variables dont le contenu ne peut être pas modifié durant l’exécution du programme.
  - La déclaration d’une constante fait appel à la fonction define() dont le premier argument correspond au nom de la constante, et le second, à sa valeur.

Les fondamentaux

- **Les constantes**
  - **Exemple: Création de la constante VERSION_SITE**

    define("VERSION_SITE","v3.1");

    À la différence des variables, les constantes n’ont pas à être précédées du caractère $.

Les fondamentaux

- **Les constantes**
  - **Exemple 2: Utilisation de constantes au sein d'un site**

```
<?php
define("NOM_SITE","Cool Site");
print(NOM_SITE);
define("VERSION_SITE",3.1);
$a = VERSION_SITE + 1;
?>
```

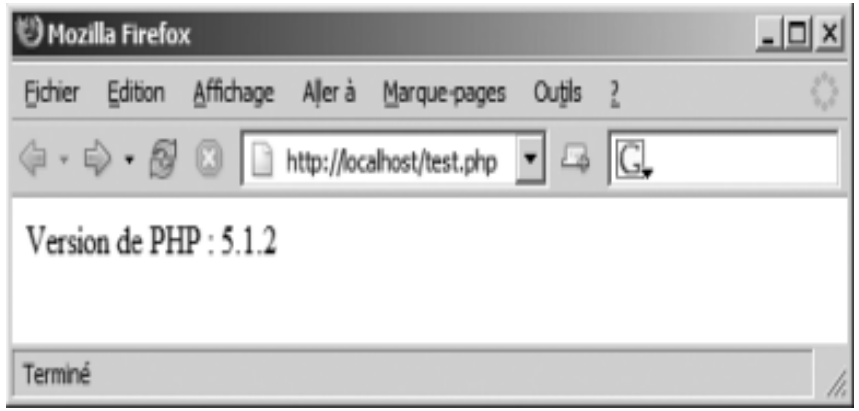
Les fondamentaux

- **Les constantes**

  ➢ PHP dispose d'un certain nombre de constantes définies par défaut. La constante PHP_VERSION contient par exemple la version de l'interpréteur PHP.

  Exemple : Utilisation d'une constante par défaut

```
<?php
print("Version de PHP : ");
print(PHP_VERSION);
?>
```



Les fondamentaux

- **Les constantes**

  ➢ L’exécution du script suivant vous permet d’obtenir la liste de toutes les constantes définies par défaut par PHP.

  NB : Plus de 700 constantes définies par PHP

```
<pre>
    <?php
        print_r(get_defined_constants());
    ?>
</pre>
```

Les fondamentaux

- **Les constantes**
- **Les types de données**

  En informatique, les données sont souvent typées, en ce sens qu'elles contiennent une certaine catégorie d'information.

- Il existe globalement deux principaux types de données : les variables contenant des chiffres et les variables contenant des lettres.

Les fondamentaux

- **Les types de données**

  - PHP rend les choses extrêmement simples au niveau des types car il n'impose pas d'associer explicitement un type à une variable (dans la plupart des autres langages, les variables doivent être associées à un type dès leur initialisation au début du programme).
  - Si vous affectez 3 à $abc, celle-ci devient de type numérique.
  - Si en revanche vous lui affectez la phrase "bonjour monde", elle devient ce que l'on appelle une « chaîne de caractères » (souvent appelée *string*).

    Arrêtons-nous quelque temps sur ces deux principaux types de données.

Les fondamentaux

- **Les types de données**

  - **Les données numériques**

    Vous trouvez, parmi les données numériques, les nombres entiers (2, 4, 233, -12) et les nombres flottants (0.5, 23.8, -123.4).

    - Il existe un certain nombre d'opérateurs mathématiques qui peuvent être utilisés avec les numériques :

Opérateur	Rôle
+	L'addition
–	La soustraction
*	La multiplication
/	La division
%	Le modulo

Les fondamentaux

- **Les types de données**
  - **Les données numériques**

```
<?php
$a = 10 ;
$b = 5 ;
$c = 2;
$x = $a + $b; // la variable $x contient 15
$x = $x - $c ; // $x vaut 13
$x = $x * $x ; // $x vaut 169
$x = ($a / $b) + $c; // $x vaut 4
$x = 11 % $a ; // $x vaut 1,
// le modulo correspond au reste de la division 11 / 10
?>
```

NB:
L'ordre de priorité des opérateurs doit être respecté : les opérations * / % sont traitées avant les opérations + –.

Les fondamentaux

- **Les types de données**

  ➢ **Les données numériques**

Il existe un type de numérique à part : les booléens. Seules deux valeurs booléennes existent : true (vrai), false (faux).

➢ **Exemple:**

```
<?php
        $var = true ; // $var contient la valeur booléenne true
?>
```

NB: Il est courant que la valeur 0 soit équivalente à false et que la valeur 1 soit équivalente à true. Cet abus peut cependant se révéler dangereux lors de certains tests.

Les fondamentaux

- **Les types de données**

  - **Les chaînes de caractères**

    Une chaîne de caractères (souvent appelée *string*) doit être délimitée par des guillemets (") ou des primes (’).

    - **Les guillemets et les primes**

      - Lorsque les caractères de délimitation sont des guillemets ("), la chaîne de caractères peut contenir des caractères ainsi que des variables.
      - La variable est alors remplacée par son contenu.

Les fondamentaux

- **Les types de données**

  ➢ **Exemple: Une variable dans une chaîne de caractères**

```
<?php
$n = 2;
$s = "valeur de la variable n = $n";
print($s); // affiche : "valeur de la variable n = 2"
?>
```

Remarque: Quand les primes (’) sont utilisées comme caractères de délimitation, les variables ne sont plus remplacées par leur contenu :

```
<?php
$n = 2;
$s = ’valeur de la variable n = $n’;
print($s); // affiche : ’valeur de la variable n = $n’
?>
```

Les fondamentaux

- **Les types de données**
  - **Les chaînes de caractères**
    - L'affichage d'une chaîne de caractères peut également être réalisé avec la fonction echo(). Les syntaxes suivantes pourront être trouvées indifféremment dans la littérature:

```
echo "bonjour";
echo("bonjour");
print("bonjour");
print "bonjour";
```

    - **Syntaxe spéciale de** echo

      echo permet l'utilisation d'une syntaxe spéciale : echo $var1, $var2; pour afficher la $var1 puis $var2. Aucune limitation n'existe au niveau du nombre de variables à afficher.

Les fondamentaux

- **Les types de données**
  - **Échappement de caractères**
  - Une question peut alors se poser : comment une chaîne de caractères peut contenir le guillemet (") quand ses délimiteurs sont précisément des guillemets ?
  - Il est dans ce cas nécessaire d’utiliser un caractère dit d’échappement : \. Ce caractère est nommé « barre oblique inversée » ou *antislash* (également *backslash*). Pour obtenir la phrase bonjour " monde, il est nécessaire d’écrire "bonjour \" monde". Le principe est le même pour la prime (’) avec des strings délimitées par des primes.

Les fondamentaux

- **Les types de données**
  - **Exemple: Échappement de caractères**

```
print("bonjour \" monde"); // affiche : bonjour " monde
print(’bonjour \’ monde’); // affiche : bonjour ’ monde
print(’bonjour \" monde’); // affiche : bonjour \" monde
```

Les fondamentaux

- **Les types de données**
  - **Les chaînes de caractères : L'opérateur de concaténation**
    - Il existe un opérateur pour les chaînes de caractères : l'opérateur de concaténation.
    - Il permet de réunir deux chaînes : $str = $str1 . $str2 ; et signifie que la variable $str contient la variable $str1 suivie de $str2.

Les fondamentaux

- **Les types de données**

  - **Exemple: L'opérateur de concaténation**

```
<?php
$var1 = "ui";
$var2 = "le temps est " . "beau aujourd'h" . $var1;
print($var2); // affiche : "le temps est beau aujourd'hui"
print("cou" . 'cou'); // affiche : "coucou"
?>
```

  - **Remarque**

La version raccourcie existe aussi : .=.

$abc = $abc . " coucou" ; est l'équivalent de $abc .= " coucou";.

Les fondamentaux

- **Les types de données**

  - **Le type NULL**

    Ce type, composé d'un seul élément (NULL), permet d'indiquer qu'une variable n'a pas de valeur.

    Exemple : Une fois utilisée, la variable $str est passée à NULL

```
<?php
$str = "coucou";
print($str);
$str = NULL;
?>
```

Les fondamentaux

- **Changement de type**

  PHP a ceci de sympathique qu'il permet aux variables de changer de type si vous leur affectez des données de types différents au cours du programme :

  - **Exemple : Changement de type**

```
<?php
        $abc = "1"; // $abc est une string contenant la chaîne "1"
        $abc += 2; // $abc est maintenant un numérique contenant la valeur 3
?>
```

Dans cet exemple, la variable est d'abord une chaîne de caractères contenant la chaîne "1". Après l'opération d'incrémentation, elle devient de type numérique et contient la valeur 3.

Les structures de contrôle

Vous disposez, avec PHP, de toutes les structures de contrôle standard.

Celles-ci sont généralement regroupées en deux catégories…

- **Les conditions :**

  SI test est vrai ALORS FAIRE action1 SINON FAIRE action2

  FAIRE DANS LE CAS 1 action1, DANS LE CAS 2 action 2 etc.

- **Les boucles :**

  FAIRE action TANT QUE test est vrai

  TANT QUE test est vrai ALORS FAIRE action

  POUR TOUS LES CAS SUIVANTS FAIRE action

Les structures de contrôle

- **Les conditions**

Nous présenterons dans cette partie les expressions **if… else, if… elseif… else et switch… case**.

➢ **IF… ELSE**

L'équivalent de *SI test est vrai ALORS FAIRE action1 SINON FAIRE action2* **est l'instruction** *if... else* (*if* signifie « si » et *else* signifie « ou bien »).

➢ **Exemple:** Écrivez un petit programme qui affiche "*i est plus grand que 5*" si la variable **$i** est plus grande que 5 et la phrase "*i est plus petit que 5*" dans le cas contraire.

Les structures de contrôle

- **Résultat:**

```
<?php
$i = 4;
if ($i > 5)
    print("i est plus grand que 5");
else
    print("i est plus petit que 5");
?>
```

Remarque: Dans ce cas, "i est plus petit que 5" apparaît à l’écran car vous avez initialisé la variable $i à 4.

Les structures de contrôle

Exercice :

1) Quelle est la sortie après l'execution de ce code ?

2) Reproduisez le code en utilisant **elseif**

```
<?php
  $a=1;
  if($a==1)
    echo "Un";
  else{
    if($a==2)
      echo "Deux";
    else
      echo "Autre";
  }
?>
```

Les structures de contrôle

- **Les conditions**
  - ➢Les opérateurs de comparaison

Opérateur	Résultat
`$a == $b`	Vrai si `$a` est égal à `$b`
`$a === $b`	Vrai si `$a` est égal à `$b` et si ces deux variables sont de même type
`$a != $b`	Vrai si `$a` est différent de `$b`
`$a !== $b`	Vrai si `$a` est différent de `$b` (en type ou en valeur)
`$a > $b`	Vrai si `$a` est supérieur à `$b`

Les structures de contrôle

- **Les conditions**
  - ➢Les opérateurs de comparaison

Opérateur	Résultat
`$a < $b`	Vrai si `$a` est inférieur à `$b`
`$a >= $b`	Vrai si `$a` est supérieur ou égal à `$b`
`$a <= $b`	Vrai si `$a` est inférieur ou égal à `$b`

Les structures de contrôle

- Remarque
  - ➢ Il est important de comprendre que ces opérateurs renvoient une valeur booléenne, false ou true, suivant le résultat de la comparaison.
  - ➢ Le test est en fait réalisé sur cette valeur de retour.
    Écrire if ($a > $b) est en fait la même chose qu'écrire if (($a > $b) == true).
  - ➢ L'expression if ($i) est identique à if ($i != NULL). Cette version condensée est très souvent utilisée en informatique.

Les structures de contrôle

- **Les conditions**
  - ➢ Groupe d'instructions
    - ➢ Exemple: Modifiez votre script afin qu'il affiche deux lignes :

```
<?php
$i = 4;
if ($i > 5){
                print("nous sommes dans le IF ... ");
                print("i est plus grand que 5");
}
else{
                print("nous sommes dans le ELSE ... ");
                print("i est plus petit que 5");
}
?>
```

Les structures de contrôle

- **Les conditions**
  - If imbriqué

Pour l'instant, les tests étudiés sont simples car vous ne testez qu'une seule valeur. Si vous devez écrire le test suivant :

SI la personne a plus de 18 ans ALORS écrire "homme majeur" SI elle est de sexe masculin

```php
<?php
	$age = 22;
	$sexe = "masculin";
	if ($age > 18) {
		if ($sexe == "masculin")
			print("homme majeur");
	}
?>
```

Les structures de contrôle

- **Les conditions**
  - La façon logique d'exprimer l'exemple précèdent est :

    SI la personne a plus de 18 ans ET qu'elle est de sexe masculin ALORS écrire "homme majeur "

  - Il s'agit là d'une double condition. L'opérateur (&&) peut être utilisé pour signifier l'opérateur ET :

```
<?php
$age = 22;
$sexe = "masculin";
if (($age > 18) && ($sexe == "masculin"))
print("homme majeur");
// affiche bien "homme majeur"
// car : (22 > 18) ET (masculin == masculin)
?>
```

Les structures de contrôle

- **Les conditions**
  - ➢Opérateurs logiques

Opérateur	Résultat
`$a and $b`	Vrai si `$a` ET `$b` sont vraies
`$a && $b`	Vrai si `$a` ET `$b` sont vraies (identique à and)
`$a or $b`	Vrai si `$a` OU `$b` sont vraies
`$a || $b`	Vrai si `$a` OU `$b` sont vraies (identique à or)
`$a xor $b`	Vrai si `$a` OU `$b` sont vraies, mais pas les deux
`! $a`	Vrai si `$a` est fausse

Les structures de contrôle

- **Les conditions**
  - **IF… ELSEIF…**

    Essayez maintenant d’écrire :

    si la couleur est rouge, jaune ou bleue, **écrire** "primaire"

    si la couleur est noire, **écrire** "noire"

    si la couleur est blanche, **écrire** "blanche"

    sinon écrire "mélange"

Les structures de contrôle

- **Les conditions**
  - **Résultat**

```
<?php
if (($couleur=="rouge") || ($couleur=="jaune") ||
($couleur=="bleue"))
                print("primaire");
elseif ($couleur == "noire")
                print("noire");
elseif ($couleur == "blanche")
                print("blanche");
else
                print("mélange");
?>
```

Les structures de contrôle

- **Les conditions**

  - **SWITCH**

    Supposez désormais que vous deviez écrire le test suivant :

    si la couleur est rouge, écrire "R"

    si la couleur est bleue, écrire "B"

    si la couleur est jaune, écrire "J"

    sinon écrire "?"

Les structures de contrôle

Dans ce cas, il est dommage d'utiliser un if… elseif….

Imaginez, en effet, que la couleur soit jaune : il faudrait alors tester si la couleur est rouge, puis tester si la couleur est bleue et enfin tester si la couleur est jaune.

Dans ce type de cas, où l'action à réaliser ne dépend que de la valeur d'une variable, il est préférable d'utiliser le switch :

```
switch ($couleur){
        case "rouge":
                print("R");
        break;
        case "bleue":
                print("B");
                break;
        case "jaune":
                print("J");
                break;
        default:
                print("?");
                break;
}
```

Les structures de contrôle

- **Les conditions**

  - WHILE

WHILE est la boucle qui permet d'écrire ceci :

TANT QUE test vrai FAIRE action

À partir de là, il devient évident pour cet exemple que le test va être réalisé sur une variable ($i < 5) et que l'action va consister à l'afficher et à l'incrémenter :

```
$i = 0;
while ($i < 5){
        print("$i<br/>");
        $i++;
}
print("<br/>fin de la boucle");
```

Les structures de contrôle

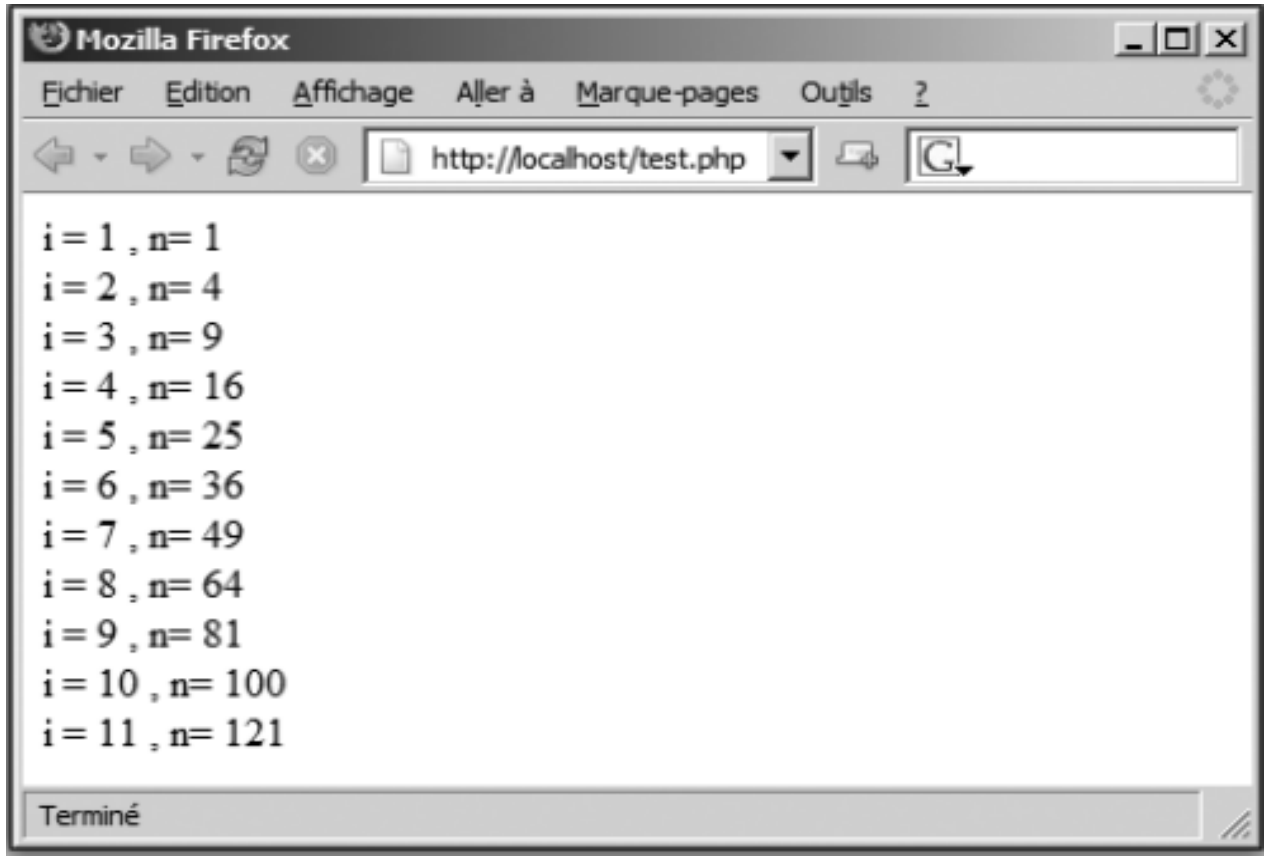
- **Le danger du** while

Faites attention, quand vous travaillez avec une boucle while, à ne pas vous retrouver dans une boucle infinie.

Cela aurait été le cas, si vous aviez oublié la ligne $i++. En effet, la variable aurait gardé la valeur 0 et la boucle while aurait affiché "0" indéfiniment.

Les structures de contrôle

- **Exercice:** En utilisant la boucle **while**, écrivez le code qui permet d'obtenir la sortie suivante :



NB: Dans cet exemple, la condition d'arrêt de la boucle est double : la variable $n doit être différente de 121 et la variable $i doit être inférieure à 100.

- Résultat !

```
$i = 1;
while (($n != 121) && ($i < 100))
{
        $n = $i * $i;
        print("i = $i , n= $n<br/>");
        $i++;
}
```

Les structures de contrôle

- **DO… WHILE**
  Une variante existe au while : le do… while. Il ne s'agit plus d'un tant que… faire, mais d'un faire… tant que. Dans ce cas, le code de la boucle est donc au moins exécuté une fois :

```
$i = 6;
do
{
print($i);
}
while ($i < 5);
```

Bien que $i contienne la valeur 6 et que le test soit $i < 5, le code est bien exécuté une fois.

Les structures de contrôle

- **FOR**
  La syntaxe de la boucle for est la suivante :
  for (expression 1; expression 2; expression 3)

  - expression 1 est l'instruction initiale exécutée avant la première itération ;
  - expression 2 est le test réalisé au début pour chaque itération ;
  - expression 3 est l'instruction exécutée à la fin de chaque itération.

Les structures de contrôle

- Exemple:

Ainsi, l'équivalent de cette boucle while :

```
$i = 0;
while ($i <= 10)
{
        print("$i <br/>");
        $i++;
}
```

… est la boucle for suivante :

```
for ($i = 0; $i <= 10; $i++)
{
        print("$i <br/>");
}
```

Séquence 3

- **Les tableaux**
- **Formulaires et transmissions de données**

Les tableaux

- Nous abordons ici un aspect très important du PHP : les *arrays*.
- Nous allons voir qu'il s'agit de variables « composées », que l'on peut imaginer sous la forme de tableaux.
- On peut faire énormément de choses avec les arrays et leur utilisation n'est pas toujours très facile.
- Il est indispensable de bien comprendre le fonctionnement des tableaux en PHP.

Si nous y parvenons, nous aurons fait le tour des bases du PHP.

Les tableaux

- **Les tableaux scalaires**

  - **Création**
    $couleur = array("rouge", "vert", "bleu");
  - Dans le cadre d’un tableau scalaire, chaque élément du tableau est accessible par son numéro unique d’enregistrement, aussi appelé index ou indice.
  - Par convention, le premier élément d’un tableau a pour index 0. $couleur[0] correspond donc à la première valeur du tableau, c’est-à-dire à "rouge".

Les tableaux

- **Les tableaux scalaires**
  - **Accéder à un élément d'un tableau**

```
$couleur = array("rouge", "vert", "bleu");
print("deuxième élément du tableau : $couleur[1]");
// affiche : "deuxième élément du tableau : «vert »
```

  - La modification d'un élément de tableau se fait en affectant une nouvelle valeur directement à l'élément en question :

```
$couleur = array("rouge", "vert", "bleu");
// modification du premier élément du tableau couleur
// il ne contiendra plus "rouge" mais "jaune"
$couleur[0] = "jaune";
```

Les tableaux

- **Les tableaux scalaires**
  - L’association d’une valeur à une cellule dont l’indice n’existe pas permet de créer cette cellule :

```
$couleur = array("rouge", "vert", "bleu");
// ajoute la couleur orange au tableau couleur qui
//contenait 3 valeurs
$couleur[3] = "orange";
print("$couleur[3]") ; // affiche le quatrième élément du //tableau : orange
```

Les tableaux

- **Les tableaux scalaires**

  - **Ordre des indices**

    PHP ne vous impose pas de suivre un ordre rigoureux dans le choix des indices du tableau. L'exemple suivant est par exemple tout à fait valide :

    ```
    $couleur = array();
    $couleur[0] = "bleu";
    $couleur[2] = "jaune";
    $couleur[4] = "orange";
    ```

    À charge pour vous de trouver une logique à tout cela !

Les tableaux

- **Les tableaux associatifs**
  - Dans le cadre d’un tableau associatif, chaque cellule du tableau est identifiée non plus par un indice numérique mais par une clé de type chaîne de caractères.
  - Cette clé, par nature, doit être unique afin de permettre de sélectionner chaque élément du tableau.

Les tableaux

- **Les tableaux associatifs**
  - La création d'un tableau associatif utilise la syntaxe suivante :

    array(cle1 => valeur1, cle2 => valeur2)
  - La création d'un tableau $personne dont les caractéristiques seraient le nom, le prénom et l'âge peut être réalisée de la manière suivante :

    $personne=array("nom"=>"Dupont","prenom"=>"Paul","age"=>23);
  - La syntaxe $personne["nom"] permet quant à elle d'accéder à la valeur de la cellule identifiée par la clé nom.
  - L'assignation d'une valeur à une cellule fonctionne comme pour les tableaux scalaires, en remplaçant l'indice par la clé : $personne["age"] = 34.

Les tableaux

- **Les tableaux associatifs**
  - Si la clé n’existe pas, une nouvelle cellule est alors automatiquement créée.
  - **Le tableau contient désormais quatre cellules**

```
$tab=array("nom"=>"Dupont","prenom"=>"Paul","age"=>23);
$tab["ville"] = "Paris";
```

Les tableaux

- Remarque:
  - **Majuscules et minuscules pour les clés**

    Les clés sont sensibles à la casse. Ainsi $personne[’nom’] et $personne[’Nom’] feront référence à deux cellules distinctes.

Les tableaux

- La définition d’un tableau sur plusieurs lignes favorise souvent la lecture et la compréhension des sources.

  - **Définition d’un tableau sur plusieurs lignes**

```
$tab = array("nom" => "Dupont",
"prenom" => "Paul",
"age" => 23);
```

Les tableaux

- **Parcours d'un tableau**

  Diverses méthodes permettent de parcourir l'ensemble des valeurs d'un tableau.

  - **Boucle foreach**
    - PHP propose une catégorie de boucle for spécialement dédiée aux tableaux : la boucle foreach.
    - Une boucle foreach pourrait être traduite de la manière suivante : POUR chaque élément contenu dans le tableau FAIRE

Les tableaux

- **Affichage des éléments d'un tableau scalaire**

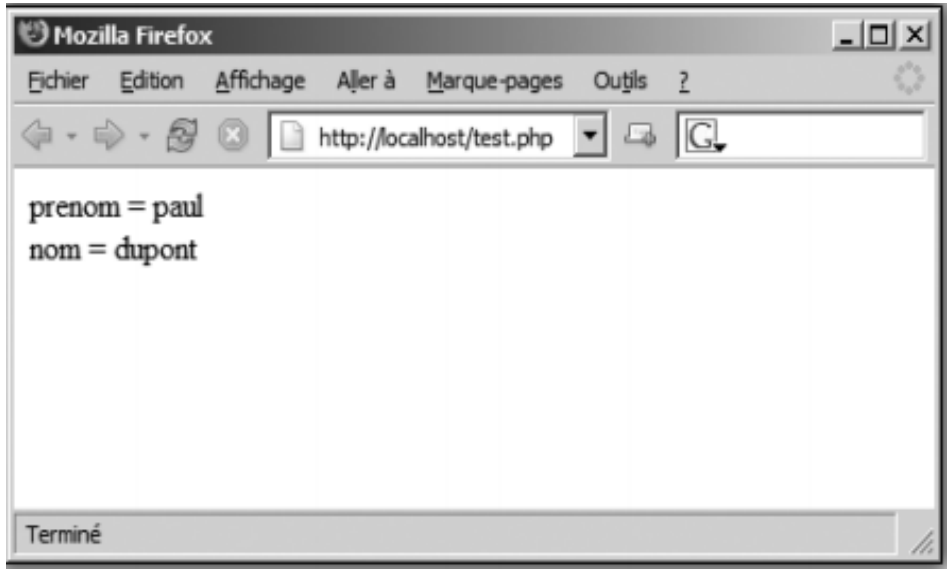
```
$tab = array ("a", "b", "c", "d");
foreach ($tab as $val) {
print("$val<br/>");
}
```

  - ➢ À chaque itération de la boucle, foreach associe la valeur de l'élément du tableau en cours à la variable $val puis déplace son pointeur interne sur l'élément suivant. Le nom de la variable est bien évidemment libre.

Les tableaux

- **Affichage des éléments d'un tableau associatif**

```
$tab = array("prenom" => "paul", "nom" => "dupont");
foreach ($tab as $cle => $valeur) {
print("$cle = $valeur<br/>");
}
```



Les tableaux

- **Taille d’un tableau**
  - ➢La taille d’un tableau peut être obtenue avec la fonction count().
  - ➢Cette taille correspond au nombre d’éléments qu’un tableau contient, qu’il soit scalaire ou associatif.

Les tableaux

- **Taille d’un tableau**

➢ **Exemple**

```
$tab = array();
$n = count($tab);
print($n); // affiche : 0
$tab = array(1,2);
$n = count($tab);
print($n); // affiche : 2
$tab = array("a"=>"b");
$n = count($tab);
print($n); // affiche : 1
$tab = array("a"=>array(1,2,3),"b"=>array(4,5,6));
$n = count($tab);
print($n); // affiche : 2
```

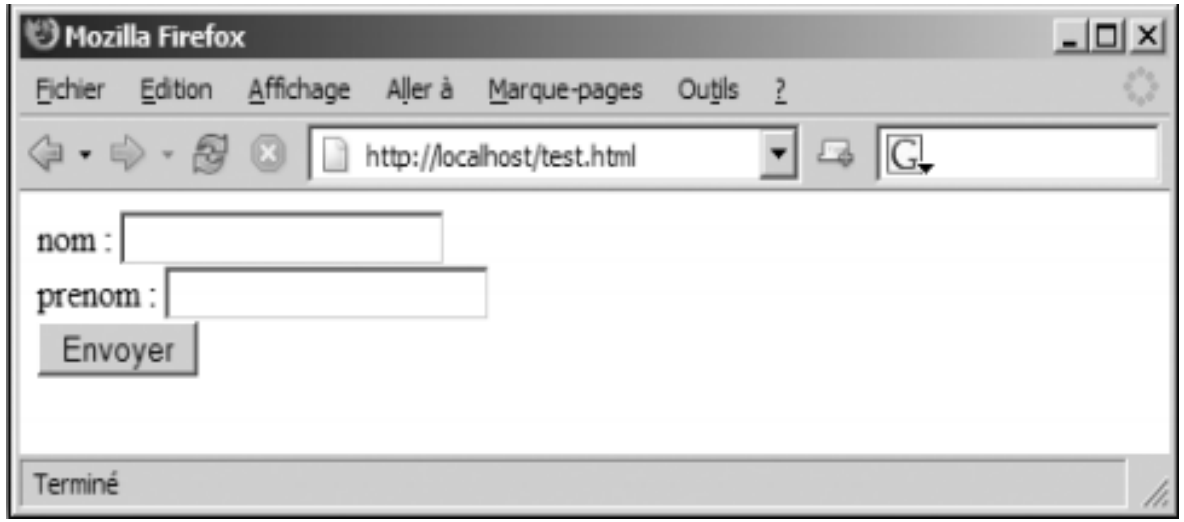
Les formulaires et transmissions de données

- Nous nous intéresserons, dans le cadre de cette partie, aux formulaires HTML. Cette notion est absolument primordiale lors du développement d'un applicatif sur le Web. Ce sont en effet ces formulaires qui vont permettre aux internautes de vous transmettre des informations.
- Nous étudierons donc la technique permettant de récupérer et de traiter au sein d'un script PHP des données issues d'un formulaire.

Les formulaires et transmissions de données

- **Qu'est-ce qu'un formulaire ?**

  ➢Un formulaire peut être envisagé comme un questionnaire permettant aux internautes de transmettre de l'information.



Les formulaires et transmissions de données

- Après la validation d'un formulaire, les données renseignées par l'internaute sont transmises à un script cible.
- Ce dernier peut aussi bien être écrit en PHP qu'en ASP, en Perl, en Python, etc.
- Le traitement des données au sein du script pourra alors consister en un envoi de courriel, un enregistrement dans une base de données, la suppression d'un fichier, l'affichage d'une nouvelle page ou toute autre action pouvant être réalisée par le script.

Les formulaires et transmissions de données

- **Passer des paramètres à un script PHP**
  - Maintenant que nous sommes en mesure de construire nos propres formulaires,
  - Nous allons découvrir les différents moyens de transmettre ces informations à un script PHP qui pourra alors les traiter.

Les formulaires et transmissions de données

- **Passer des paramètres à un script PHP**

  - Si vous souhaitez transmettre les informations du formulaire au script *script.php*, vous devez écrire :

    <form action="script.php">

Les formulaires et transmissions de données

- Remarque

➢Un script situé sur un autre site peut également être appelé via cet attribut action.

➢L'exemple suivant appelle le moteur de recherche Google en lui transmettant le mot-clé correspondant à votre recherche :

```
<form action="http://www.google.fr/search">
<input type="text" name="q" />
<input type="submit" value="nouvelle recherche" />
</form>
```


Mozilla Firefox
Fichier Edition Affichage Aller à Marque-pages Outils ?
http://localhost/test.html
Recherche sur google.fr
php
nouvelle recherche
Terminé

Les formulaires et transmissions de données

- En validant le formulaire avec le mot-clé "PHP", vous basculez sur le site de Google et vous obtenez le résultat de la recherche :

Les formulaires et transmissions de données

- **Application**

Créez maintenant deux fichiers en y plaçant un formulaire (*form.html*) et un script PHP (*script.php*) et faites pointer le formulaire sur le script :

- **form.html**

```
<form action="script.php">
<input type="text" name="x" />
<input type="submit" value="envoyer" />
</form>
```

- **script.php**

```
<?php
print("valeur de x : ");
?>
```

Les formulaires et transmissions de données

- PHP rend la récupération des paramètres transmis extrêmement simple.
- Le principe consiste à passer par la variable prédéfinie $_GET ou $_POST qui transmettent les données respectivement dans la barre de navigation et dans la requête HTTP.

Les formulaires et transmissions de données

- **La méthode GET**

La méthode par défaut est la méthode GET. Les données seront visibles dans l'URL, par exemple : /traitement.php?firstname=Pierre&lastname=Dupont

La page de traitement recevra les données dans la variable $_GET.

Quelques remarques et précautions :

•La longueur des URL peut être limitée (comptez 2048 caractères).

•N'utiliser jamais la méthode GET pour transmettre des données sensibles.

•L'utilisateur pourra ajouter la page de traitement dans ses favoris.

Les formulaires et transmissions de données

- **La méthode GET**
  - Syntaxe:

```
<form action="traitement.php" method="get">
<!-- Champs du formulaire -->
</form>
```

Les formulaires et transmissions de données

- **La méthode POST**

Avec la méthode POST, les données du formulaire seront transmises dans la requête HTTP. Si le site utilise un certificat SSL (domaine commençant par https au lieu de http), les données seront cryptées. Cette méthode est plus sécurisée.

La page de traitement recevra les données dans la variable $_POST.

- Utilisez cette méthode pour la transmission des données sensibles.
- L'utilisateur ne pourra pas ajouter la page de traitement dans ses favoris.

Les formulaires et transmissions de données

- La méthode POST
  - ➢Syntaxe

```
<form action="traitement.php" method="post">
<!-- Champs du formulaire -->
</form>
```

- A suivre: …