

IFT1147  
Programmation Serveur Web avec PHP

Introduction à PHP



Plan

- ♦ Fonctionnement de base
- ♦ Définition du langage
- ♦ Bonnes pratiques
- ♦ Programmation par objets

IFT1147 - Introduction à PHP 2

Introduction

et fonctionnement de base d'une page PHP

PHP

- ♦ PHP a été créé en 1995 par Rasmus Lerdorf comme compteur du nombre de visiteurs d'un site Web.
- ♦ Les instructions PHP sont intégrées dans le code HTML et exécutées par le serveur, avant que le code HTML ne soit envoyé au navigateur: PHP est un langage de programmation côté serveur.

IFT1147 - Introduction à PHP 4

Documentation

- ♦ Le site principal de PHP est <http://www.php.net>
- ♦ La documentation de chaque fonction peut être accédée à l'URL <http://www.php.net/nomDeLaFonction>
- ♦ Pour d'autres sites de référence consultez la section Liens du site du cours.

IFT1147 - Introduction à PHP 5

Exemple de page PHP

```
<html>
<body>
<h1>Test de Php</h1>
<?php
    echo "<em>Bonjour</em>";
?>
tout
<?php echo "le monde"; ?>
</body>
</html>
```

IFT1147 - Introduction à PHP 6

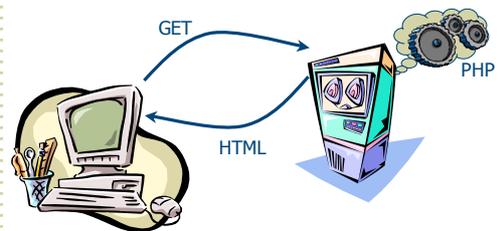
## Fonctionnement d'une page PHP

- ♦ Le serveur Web reçoit une requête pour un fichier .php
- ♦ L'extension .php indique au serveur qu'il faut compiler et exécuter du code PHP
- ♦ Le serveur traite le fichier et retourne ensuite le résultat de l'exécution au navigateur: le code source n'est jamais transmis au navigateur !

IFT1147 - Introduction à PHP

7

## Fonctionnement d'une page PHP



IFT1147 - Introduction à PHP

8

## PHP versus serveurs d'applications

- ♦ Contrairement à certains serveurs d'applications, le code PHP reste, à moins d'utiliser un *opcode cache*, sous forme de code source sur le serveur; le code PHP est donc compilé et exécuté à chaque requête pour une page.
- ♦ Chaque requête génère une instance d'exécution séparée et indépendante.

IFT1147 - Introduction à PHP

9

## Définition du langage PHP

fonctions, variables etc.

## Début et fin du code PHP

- ♦ Le début et la fin du code PHP doivent être indiqués par `<?php` et `?>`
- ♦ D'autres façons d'indiquer le début et la fin du code sont possibles, mais doivent être configurées par l'administrateur du site. L'abréviation `<?(` (pour le début du code PHP) est très populaire (et disponible sur [www-desi](http://www-desi)).

IFT1147 - Introduction à PHP

11

## Exemple: bas de page standard

- ♦ On peut créer un bas de page standard:
  - en plaçant le code HTML du bas de page dans le fichier `bas.html`
  - en utilisant la fonction PHP `require_once` dans toutes les pages à l'endroit où doit apparaître le bas de page  
`<?php require_once("bas.html");?>`

IFT1147 - Introduction à PHP

12

## Variables

- ♦ Tout nom de variable doit commencer par le symbole \$
- ♦ En PHP, le type d'une variable n'est pas déclaré explicitement, mais déterminé, lors de l'exécution, en fonction de l'assignation. Il peut changer !

```
$ligne = "abc";  
$ligne = 123;
```

IFT1147 - Introduction à PHP

13

## Echo

- ♦ echo permet d'imprimer rapidement
  - Une ou plusieurs chaînes de caractères
  - Le contenu de variables
- ♦ echo accepte un nombre variable d'arguments

```
echo $bonjour;  
echo "Bienvenue $bonjour\n";  
echo "Bienvenue ", $bonjour, "\n";
```

IFT1147 - Introduction à PHP

14

## Structures de contrôle

- ♦ Les structures de contrôle de PHP sont identiques à celles des autres langages de programmation populaires.
- ♦ if ( ) { } else { }
- ♦ for ( ; ; ) { }
- ♦ while ( ) { }
- ♦ do { } while ( )

IFT1147 - Introduction à PHP

15

## Comparaisons

- ♦ Comme dans d'autres langages, les opérateurs == et != testent pour l'(in)égalité de deux valeurs.
- ♦ PHP effectue automatiquement certaines conversions de types (p.e. "1" == 1); si vous voulez l'en empêcher, utilisez plutôt les opérateurs === et !==



IFT1147 - Introduction à PHP

16

## Commentaires

- ♦ // indique les commentaires d'une seule ligne
- ♦ /\* \*/ permet de composer un commentaire de plusieurs lignes
- ♦ Les commentaires sont invisibles pour le visiteur de votre site ! Pourquoi ?  
Le code PHP est exécuté côté serveur ...

IFT1147 - Introduction à PHP

17

## Débugger



- ♦ Vérifiez que l'extension de fichier est .php et que le fichier est en lecture.
- ♦ Vérifiez que PHP est disponible sur le serveur et qu'il est bien configuré  
<?php phpinfo() ?>
- ♦ Utilisez la fonction print\_r afin d'examiner le contenu de vos variables.
- ♦ error\_reporting(E\_ALL);

IFT1147 - Introduction à PHP

18

## Chaînes de caractères

- ♦ Vous pouvez créer une chaîne de caractères de trois façons
  - Entre simple guillemets  
Le plus rapide, mais le moins flexible
  - Entre double guillemets  
Les variables sont évaluées
  - Avec heredoc (<<<)  
Surtout pour de longues chaînes

IFT1147 - Introduction à PHP

19

## Chaînes de caractères - exemple

```
$nom = 'toto';  
  
$bienvenue = "Bonjour {$nom}\n";  
  
$message = <<< END_CHAINE  
Nous vous souhaitons la  
bienvenue sur le site.  
END_CHAINE;
```

IFT1147 - Introduction à PHP

20

## Chaînes de caractères - fonctions

- ♦ L'opérateur de concaténation est .
- ♦ `strcmp`, ainsi que `stricmp`, `strnatcmp` etc., permettent de comparer deux chaînes
- ♦ `strpos` permet de déterminer si une chaîne de caractères est une sous-chaîne d'une autre.

IFT1147 - Introduction à PHP

21

## Tableaux

- ♦ Les tableaux en PHP fonctionnent selon le concept des mappes (associations entre clés et valeurs).
- ♦ Les clés d'un tableau peuvent être des chiffres (tableaux indexés) ou des chaînes de caractères (tableaux associatifs).

IFT1147 - Introduction à PHP

22

## Tableaux indexés

- ♦ La tableau indexé est le tableau usuel: on accède chaque élément en spécifiant son indice.
- ♦ Le premier élément se trouve à la position 0.
- ♦ Un tableau peut contenir des variables de types différents, et même d'autres tableaux.

IFT1147 - Introduction à PHP

23

## Tableaux indexés - exemple

```
$couleur = array("rouge", "vert");  
$couleur[] = 28;  
$couleur[3] = "jaune";
```

0	1	2	3
rouge	vert	28	jaune

IFT1147 - Introduction à PHP

24

## Tableaux indexés - parcours

```
for ($i = 0; $i < count($t); $i++) {
    echo $t[$i];
}

// legerement plus rapide
for ($i=0, $cnt=count($t); $i<$cnt; $i++) {
    echo $t[$i];
}
```

## Tableaux associatifs

- ◆ Dans un tableau associatif (table de hachage) on peut accéder chaque élément par un nom (chaîne de caractères).
- ◆ Tout comme le tableau indexé, le tableau associatif peut contenir des valeurs de types différents et sa taille est gérée dynamiquement.

## Tableaux associatifs - exemple

```
$resultat = array("nom" => "dift1147",
                 "intra" => 18,
                 "devoir1" => 9.5,
                 "final" => 90);
```

nom	intra	devoir1	final
dift1147	18	9.5	90

## Tableaux associatifs - parcours

```
$cles = array_keys($t);
for ($i = 0; $i < count($cles); $i++) {
    echo $cles[$i], $t[$cles[$i]];
}

//ou avec foreach
foreach ($t as $key => $value) {
    echo $key, $value;
}
```

## Tableaux - tri

- ◆ PHP offre une multitude de fonctions de tri. Dans un tableau associatif, on peut même choisir si on souhaite trier selon la clé ou la valeur.
- ◆ Un bon point de départ pour explorer les fonctions de tri est la documentation pour `sort`.

## Chaînes de caractères et tableaux

- ◆ La fonction `implode` transforme un tableau en chaîne de caractères. On peut spécifier un séparateur pour les éléments.
- ◆ `echo implode(", ", $tableau);` imprime tous les éléments de `$tableau`, séparés par des virgules.

## Chaînes de caractères et tableaux

- ♦ `explode` est une fonction inverse de `implode`. Elle prend en argument un séparateur et une chaîne de caractères et retourne un tableau.
- ♦ `$t=explode(",","tp1,tp2,tp3");`  
`$t` est alors un tableau avec trois éléments.

## Définition de fonctions

- ♦ Des fonctions peuvent être déclarées par `function nomFct (par1, par2)`
- ♦ Aucun type de retour n'est spécifié et une fonction peut retourner des valeurs de types différents.
- ♦ Tous les paramètres sont assignés (par défaut) par valeur.

## Définition de fonctions - exemple

```
function factorielle($n) {  
    if ($n < 0) {  
        return -1;  
    }  
    if ($n == 1) {  
        return 1;  
    }  
    return $n * factorielle($n - 1);  
}  
  
echo factorielle(5);
```

## Paramètres par référence

- ♦ Il peut être utile de passer certains paramètres par référence:
  - Consommation de mémoire
  - Nécessité de modifier le paramètre
- ♦ Il faut alors déclarer le paramètre avec le symbole `&`  
`function f(&$parametreRef)`

## Portée des variables

- ♦ La portée de toute variable est, à priori, limitée au corps de la fonction.
- ♦ Aucune variable globale n'est immédiatement accessible dans la fonction. S'il faut accéder une variable globale, il faut la déclarer dans la fonction avec `global`  
`global $maVar;`



## Librairies de code

- ♦ Il est très pratique de placer les fonctions les plus utilisées dans un fichier PHP à part. On utilise souvent l'extension `.inc` (« à inclure ») pour ces fichiers.
- ♦ Le fichier `.inc` doit commencer par `<?php` et finir par `?>`
- ♦ Un fichier PHP peut y faire référence par `require_once("fichier.inc");`

## Lecture de fichiers

- ♦ La façon la plus simple de lire le contenu complet d'un fichier est la fonction `file()`.
- ♦ Attention: cette fonction place le contenu complet du fichier en mémoire ... son utilisation est donc déconseillée si le fichier est volumineux !



## file

- ♦ `file` prend en argument le nom d'un fichier et retourne un tableau indexé contenant une case par ligne du fichier.

```
Ligne1
...
Ligne3
4
```

Ligne1	...	Ligne3	4
--------	-----	--------	---

## file et trim

- ♦ `file` n'enlève pas les retours à la ligne: chaque case du tableau retourné se termine par un retour à la ligne.
- ♦ Si on a besoin d'enlever ces retours, on peut utiliser la fonction `trim`.
- ♦ Des variantes de `trim` sont `ltrim` et `rtrim` qui enlèvent respectivement les blancs à gauche et à droite de la chaîne.

## fopen

- ♦ La fonction `fopen` constitue une autre façon d'ouvrir un fichier.
- ♦ On peut spécifier si on souhaite ouvrir le fichier en lecture ou en écriture.
- ♦ Attention: si vous voulez ouvrir un fichier en écriture, il faut s'assurer qu'il n'y a pas deux instances du code PHP qui l'accèdent en même temps (`flock`).



## Bonne pratiques

Comment écrire du code PHP qui fonctionne, pourra être modifié et dont vous serez encore fier dans le futur ...

## Bonnes pratiques

- ♦ Documentez votre code, par exemple avec `phpDocumentor`.
- ♦ Utilisez un logiciel de gestion de versions (CVS ou `subversion` par exemple).
- ♦ Adoptez un style d'indentation et de nomenclature des variables.
- ♦ Définissez des fonctions.

## Bonnes pratiques - encore

- ◆ Utilisez les fonctions prédéfinies de PHP
  - Elles sont écrites en C, donc plus rapides
  - Elles sont déjà testées et ne devraient pas contenir (trop) de bogues.
  - De nouveaux développeurs peuvent plus facilement lire votre code.
- ◆ Générez du HTML simple à lire pour un humain, contenant des retours à la ligne.

IFT1147 - Introduction à PHP

43

## Sites web et Applications web

- ◆ PHP peut être utilisé aussi bien
  - pour la création de sites web simples avec des pages assez indépendantes (p.e. l'inclusion d'un bas de page) que
  - pour la création d'applications complexes comportant des accès à des bases de données et des communications avec d'autres programmes (p.e. commerce électronique).

IFT1147 - Introduction à PHP

44

## PHP pour sites web

- ◆ PHP est utilisé essentiellement afin de générer rapidement du code HTML.
- ◆ Le code PHP et le code HTML sont le plus souvent entremêlés.
- ◆ Une demande de changement peut nécessiter la réécriture d'une partie importante du code PHP.

IFT1147 - Introduction à PHP

45

## PHP pour applications web

- ◆ Conception de l'application en couches
  - Vue (HTML)
  - Logique d'application
  - Base de données
- ◆ Chaque couche devrait être la plus indépendante possible et communiquer uniquement avec la couche directement en dessous d'elle.

IFT1147 - Introduction à PHP

46

## PHP pour applications web - 2

- ◆ La possibilité de réutiliser le code est très importante; on est donc amené à utiliser (ou à programmer) de grandes bibliothèques (p.e. d'accès aux bases de données).
- ◆ Utilisation quasi omniprésente de la programmation par objets

IFT1147 - Introduction à PHP

47

## Programmation par objets

Ce n'est pas encore ce que ça pourrait être, mais ...

## PHP 4 et PHP 5

- ♦ PHP 4 supporte les rudiments de la programmation par objets:
  - Constructeurs
  - Héritage
- ♦ PHP 5 supporte
  - `private`, `public`
  - Destructeurs
  - *Method overloading*

IFT1147 - Introduction à PHP

49

## Programmation par objets

- ♦ La classe est l'élément de base de la programmation par objets. Elle est le modèle à partir duquel des instances peuvent être créées.
- ♦ Une classe est un conteneur pour des attributs (variables) et des méthodes (fonctions).

IFT1147 - Introduction à PHP

50

## Exemple

```
class NomDeLaClasse {  
    var $attribut1;  
    var $attribut2;  
  
    function NomDeLaClasse () {  
        //constructeur  
    }  
  
    function fonction ( ) {}  
}
```

IFT1147 - Introduction à PHP

51

## Instanciation

- ♦ Afin d'instancier un objet, il faut utiliser le mot clé `new`  
`$maVar =& new NomDeLaClasse();`
- ♦ Afin de faire référence à un attribut à l'intérieur de la classe, il faut utiliser `$this`  
`$this->a = $a;`

IFT1147 - Introduction à PHP

52

## Méthodes statiques

- ♦ Certaines méthodes d'une classe peuvent être exécutées sans qu'on ait à créer une instance de celle-ci.
- ♦ Ces méthodes sont appelées *statiques*.
- ♦ Afin d'exécuter une méthode statique il faut la préfixer par le nom de sa classe, suivie de `::`  
`LaClasse::laMethode();`

IFT1147 - Introduction à PHP

53

## Conventions

- ♦ Le nom d'une classe débute généralement par une majuscule. Ceux des attributs et méthodes par une minuscule.
- ♦ Ne manipulez pas les attributs d'une classe à l'extérieur de celle-ci, mais définissez des méthodes pour l'accès: `getAtt()`, `setAtt($var)`

IFT1147 - Introduction à PHP

54

## Avantages des objets

- ♦ L'implémentation d'une classe peut être changée sans que les appels à celle-ci aient à être modifiés.
- ♦ La programmation par objets permet de créer du code très modulaire et réutilisable.
- ♦ Elle permet de penser en termes d'objets du domaine de l'application.

IFT1147 - Introduction à PHP

55

## Pour aller plus loin

- ♦ La programmation par objets a permis de jeter de nouveaux regards sur la programmation
  - *Extreme Programming*
  - *Test-driven Development*
  - *Refactoring*
  - Catalogues de *Design Patterns*

IFT1147 - Introduction à PHP

56

## Lectures intéressantes

- ♦ *Extreme Programming Explained*  
Kent Beck, Addison Wesley (2000)
- ♦ *Test-Driven Development By Example*  
Kent Beck, Addison Wesley (2003)
- ♦ *Design Patterns*  
Erich Gamma et al., Addison Wesley (1995)
- ♦ *Patterns of Enterprise Application Architecture*  
Martin Fowler, Addison Wesley 2003

IFT1147 - Introduction à PHP

57

## Concrètement: `toHTML()`

- ♦ On programme souvent une méthode `toHTML()` qui permet d'obtenir l'affichage HTML de l'objet.
- ♦ Il est généralement avantageux de laisser cette méthode retourner une chaîne de caractères (donc, on n'utilise pas `echo` dans `toHTML()`).

IFT1147 - Introduction à PHP

58

## Concrètement: `toHTML()`

- ♦ L'utilisation la plus simple de `toHTML()` est donc  

```
echo $monObj->toHTML();
```
- ♦ Cette façon de faire simplifie l'utilisation de bibliothèques de gestion de page qui devient de plus en plus la norme pour des projets d'envergure.

IFT1147 - Introduction à PHP

59

## Concrètement: `$_GET` et `$_POST`

- ♦ Il est généralement avantageux de ne pas accéder directement `$_GET` et `$_POST` à l'intérieur d'objets si ceux-ci ne font pas partie de la couche « interface HTML ».
- ♦ Il vaut mieux ajouter des paramètres aux méthodes.

IFT1147 - Introduction à PHP

60

## Concrètement: \$\_GET et \$\_POST

- ◆ Vous aurez alors des objets qui sont
  - indépendants du protocole HTTP
  - qui peuvent être utilisés dans n'importe quel contexte (web, application indépendante, script, etc.)
  - pour lesquels des tests unitaires sont faciles à concevoir (parce qu'on n'a pas besoin de créer un contexte « web »).