Programmation avancée pour le Web

PHP5

Réaliser par : Mr Chouha Adel
E-mail : adch05@yahoo.fr

Introduction à PHP

- **PHP** est un langage de programmation de scripts côté serveur permettant de produire des pages web dynamiques.
- **PHP** : "Personnal Home Page" inventé par Rasmus Lerdorf en 1994
- **PHP** : Hypertext Preprocessor
- Langage de script HTML interprété coté serveur
  - Il est différent du **Perl** qui ne peut être insérer dans du code HTML
  - Il est différent du **JavaScript** qui s'exécute coté client
  - Soit une page HTML avec un script PHP
  - Soit un fichier PHP avec des lignes de HTML

Créer des scripts PHP

Les scripts PHP :

- sont de simples fichiers ”texte” (extension .php) à créer avec un éditeur de texte.
- contiennent du code PHP et du code HTML.
- sont exécutés côté serveur par un “interpréteur” PHP.

Généralement, PHP sert :

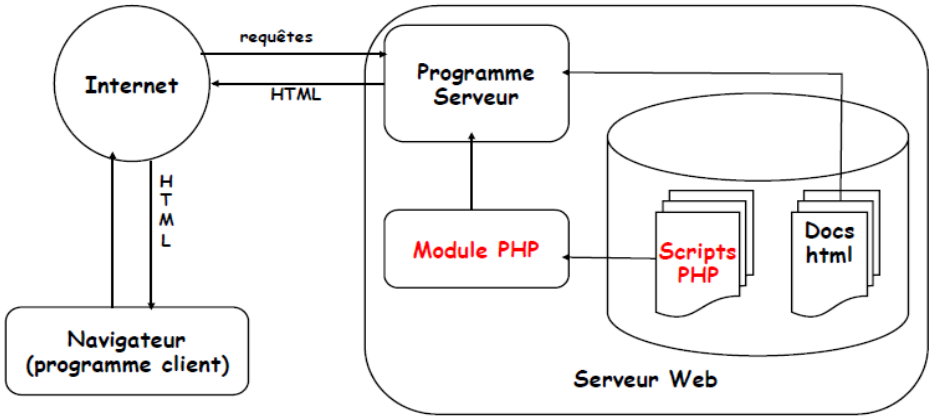
- à produire des pages web dynamiques et donc à récupérer et traiter des informations issues d’une base de données, d’un système de fichiers ou plus simplement des données envoyées par le navigateur.

Exemple

- Le **code PHP** doit être inséré entre des **balises <?php** et **?>**. Le **script** doit porter l’extension **.php**.
- **Le script helloworld.php :**

```
<HTML>
<HEAD>
<!-- Un titre -->
<TITLE>Exemple 1</TITLE>
</HEAD>
<BODY>
<?php
// forme la plus simple, recommandée
echo '<P>Hello World!</P>';
?>
</BODY>
</HTML>
```

Architecture d'un site web avec PHP



Syntaxe de PHP

- Un fichier **PHP** a une extension "**.php**", il est toujours interprété au niveau du serveur.
- tout code entre les balises : **<?php et ?>**
- **séparateur d'instructions** : « **;** »
- **bloc d'instructions** délimité par des accolades **{}**
- **commentaires** : comme en C : « **/*** » suivi de « ***/** » Ou // juste sur une ligne

Exemple : un contenu dynamique

// URL : http://localhost/helloworld-2.php?lang=fr

```
$lang = strtolower($_GET['lang']); // appel de la fonction strtolower()
if ($lang === 'fr') // test du type ET de la valeur
{
$message = 'Bonjour le monde !';
}
elseif ($lang == 'en') // test de la valeur seulement
{
$message = "Hello"." World!"; // une concaténation de chaînes de caractères
}
else
{
$message = "Je ne vois pas quelle est votre langue $lang !";
}
echo $message; // affiche le contenu de la variable message
```


La commande echo

- Permet d'insérer du texte (écrire la page au format HTML résultant de l' interprétation de PHP)
  - **echo "Bonjour" ;**
  - **$nom="adel" ; echo "Bonjour $nom" ;**

Syntaxe de PHP

Variables :

- distinction **minuscules/majuscules**
- **nom** de variable toujours précédé de « **$** » et au moins un caractère non numérique
- le **type** de la valeur associée à une variable peut **changer**
- **pas** de **déclaration** de variable en **PHP**, c'est à la 1ère référence :

```
$mavariable = 1; echo "$mavariable";
```


Syntaxe de PHP

- **Constantes :**
  - association symbole/valeur
  - ne peut jamais être modifié
  - **define** (PI , 3.14159);
- **Les types :**

1. Types numériques (entiers et flottants) et booléens

```
$i = 1;
$i = 3.14159;
$f = 0.3e-3;
```

- pas de type booléen explicite : la valeur **faux** est le **0**, la chaîne **vide** ou la chaîne "**0**", tout le reste est **vrai**
- Constantes **TRUE** et **FALSE** prédéfinies

Syntaxe de PHP

Les types :

2. Chaînes de caractères

entourées de « " » ou de « ‘ »

- **guillemets simples « ‘ » :**
  - on ne peut pas mettre de variable, de caractère d’échappement, mais sauts de ligne acceptés
- **guillemets doubles « " » :**
  - ces chaînes peuvent inclure des variables. Plus les caractères d’échappement (\n, \r, \t, \\, \$, \")

Syntaxe de PHP

Les tableaux

- suite de valeurs référencées par une unique variable
- gestion dynamique de la taille
- tableaux multi-dimensionnels
- tableaux indicés :
  - référencement par la position qui commence à 0
    - $tab[0] = "chaîne1";
    - $tab[1] = 1;
  - affectation automatique d’un indice :
    - $tab[] = "chaîne1";
    - $tab[] = 1;
  - instruction array pour initialiser un tableau
    - $tab = array("chaîne1", 1);

Syntaxe de PHP

Tableaux associatifs :

- au lieu d'utiliser un indice, on utilise une clé, la notion d'ordre disparaît
- on accède à un élément par sa clé
  - **initialisation** : $mois["Jan"] = "Janvier" ;

    $mois["Fev"] = "Février";
  - **initialisation par array :**
    - $mois = array ("Jan" => "Janvier" ,"Fev" => "Février");
    - accès à une valeur : $mois["Jan"]; donne "Janvier"

Syntaxe de PHP

Conversion et typage :

- le type d'une variable est déterminé selon le contexte
- possibilité de tester le type de la valeur référencée par une variable :
  - **is_string**, **is_long** (pour un entier), **is_double** (pour un flottant), **is_array**, **is_object**
- fonction **gettype**() renvoie le type
- possibilité de convertir le type d'une variable :
  - $v = "3 petits"; → 3 petits
  - $v = (integer) $v; → 3
  - $v = (double) $v; → 3

Syntaxe de PHP

Les opérateurs :

- opérateurs arithmétiques **(+ , - , * , / , %)**
- concaténation de chaîne par « **.** »
- incrémentation avec **$i++** ou **++$i** pour décrémentation
- opérateurs de bits **(& (ET binaire), | (OU binaire), ^ (OU exclusif))**
- opérateurs logiques **(&& (ou and), || (ou or), xor** et **!)**
- opérateurs de comparaison **(= =,!=,<,>,<= et >=)**

Syntaxe de PHP

Les structures de contrôle : test

- **if… else :**

```
if (expression)
{…}
else {…}
```

- **switch :**

```
switch (expression)
{ case valeur1 :    …
                    break;
  case valeur2 :    …
                    break;
  default :         …
}
```

Syntaxe de PHP

Les structures de contrôle : boucles

- **while :**

```
while (expression){
    actions;
}
```

- **do-while :**

```
do {
   actions;
} while (expression);
```

- **for :**

```
for (expression1; expression2; expression3){
    actions;
}
```


Syntaxe de PHP

Les fonctions :

- les fonctions doivent être définies avant leur appel
- **syntaxe** :

```
function Nom ([$arg1, $arg2, …. ])
    { …. }
```

- renvoi une seule valeur avec **return**
- passage des arguments par **valeur** par défaut
- on indique un passage par **adresse** avec « & »

Syntaxe de PHP

Les variables d'environnement :

- **QUERY_STRING** : récupère l'ensemble des informations transmises depuis un formulaire (visibles au niveau du navigateur en utilisant la méthode **GET**)
- **REQUEST_METHOD** : le résultat est **GET** ou **POST**
- **HTTP_USER_AGENT** : permet d'avoir des informations sur le type de navigateur utilisé par le client, ainsi que son système d'exploitation La récupération se fait grâce à **$_SERVER**[**'variable_environnement**']
- **$_POST** ou **$_GET** : tableau associatif global contenant les données transmises exemple : **$_GET**[**'var**']

Syntaxe de PHP

Fonctions permettant d'envoyer du texte au navigateur :

La fonction echo :

- permet d'envoyer au navigateur la chaîne de caractères ou une expression que l'interpréteur évalue

```
echo "Bonsoir à tous";
```

La fonction print :

- similaire à la fonction **echo**, l'expression à afficher est entre parenthèses :

```
print(expression);
print("Bonsoir à tous");
```

La fonction printf :

- celle du langage C, rarement utilisée, permet un formatage des données affichée à l'écran

Syntaxe de PHP

Les Fichiers :

- Par le nom relatif : "../../images/aaa.gif"
- Par le nom absolu : "/users/work/ddd/file.txt"
- Par une URL : http://www.ccc.dz/index.html

Les commandes :

- **require(string nom_fichier)** : cette commande se remplace dans le script toujours par le contenu du fichier
- **include(string nom_fichier)** : inclus et évalue le fichier spécifié en argument

Syntaxe de PHP

Ouvrir un fichier :

- **int fopen(string nom_fichier, string "mode")**
- **Modes possibles :**
  - r : ouverture en lecture seulement
  - w : ouverture en écriture seulement (la fonction crée le fichier s'il n'existe pas)
  - a : ouverture en écriture seulement avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)
  - r+ : ouverture en lecture et écriture
  - w+ : ouverture en lecture et écriture (la fonction crée le fichier s'il n'existe pas)
  - a+ : ouverture en lecture et écriture avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)

Fermer un fichier :

- Un fichier ouvert avec la fonction **fopen** doit être fermé par la fonction **fclose** en lui passant en paramètre "**int**" retourné par la fonction **fopen**
  - **bool fclose (int nom_fichier)**

Les classes et les objets

- Une classe peut contenir ses propres constantes, variables (appelées "propriétés" ou "attributs"), et fonctions (appelées "méthodes"),
- Une définition de classe commence par le mot-clé **class**, suivi du nom de la classe puis d'une paire d'accolades contenant la définition des attributs et des méthodes appartenant à la classe.
- La pseudo-variable $this est disponible lorsqu'une méthode est appelée depuis un objet : **$this** est une référence à l'objet appelant. **$this** est obligatoire pour accéder aux membres de l'objet en utilisant l'opérateur **→**.
- Pour créer une instance d'une classe, le mot-clé **new** doit être utilisé. Un objet sera alors systématiquement créé, à moins qu'il ait un constructeur défini qui lance une exception en cas d'erreur.
- Une **classe** est une déclaration des propriétés et des méthodes :
  - **Propriété** = variable de classe
  - **Méthode** = fonction de classe

Les classes et les objets

- Un **objet** est une **instance** d'une **seule** et **unique** classe
- Depuis **PHP 5**, les objets sont des références

  $objet1 = $objet2;
- ne duplique pas l'objet **$objet1**, mais crée une **deuxième référence** vers le même objet **$objet1**
- Une **classe** permet de créer des **objets** au moyen d'un "**constructeur**" appelé, par défaut, "**new**" suivi du nom de la classe => **PHP** créé l'objet en mémoire et référence son emplacement en mémoire dans la variable :

  $object = new MaClasse();
- La variable **$object** contient une référence à un objet de la classe **MaClasse,** mais pas l'objet lui-même
- Pour **détruire** cet objet en mémoire, il faut **détruire toutes les références** vers cette instance de la classe :

  unset($object);
- toutes les références sont détruites, l'objet n'existe donc plus en mémoire

Les classes et les objets

- Une **classe** (mot clé : **class**) est la définition d'un ensemble d'objets et est composée de :
  - **attributs** : (mot clé : **public/protected/private**) données caractérisant l'état de l'objet
  - **méthodes** : (mot clé : **fonction**, précédé du mot clé : **public/protected/private**) opérations applicables aux objets

Les classes et les objets

```php
<?php
    class nouvelle_classe {
    private $atr1; private $atr2; private $atr3;
    public function fonct1() {
    …}};
    //création d'un objet
    $objet1 = new nouvelle_classe();
    //affectation des propriétés de l'objet
    $objet1->atr1=…; $objet1->$atr2=…; $objet1->$atr3=…;
    //utilisation des propriétés
    echo "l'attribut de l'objet1 est ", $objet1->atr1, "<br />";
    //appel de la méthode fonct1()
    $objet1->fonct1() ;
?>
```

Exemple :

```
Class Voiture {                  // déclaration de la classe
var $couleur;                    // déclaration d'un attribut
var $belle= TRUE;                // initialisation d'un attribut
Function voiture() {             // constructeur (_ _construct)
$this->couleur = ''noire'';
}                                // le mot clé $this faisant référence à l'objet est obligatoire
Function Set_Couleur($couleur) {
$this->couleur = $couleur;
}
}
$mavoiture= newVoiture();        // création d'une instance
$mavoiture->Set_Couleur(''blanche'');   // appel d'une méthode
$coul= $mavoiture->couleur;      // appel d'un attribut
```


Les classes et les objets

Manipulation des classes et des objets

- **Instanciation de la classe** : Constructeur (par défaut) de même nom que la classe et appelé par l'opérateur new
  - $Nom_de_l_objet = new Nom_de_la_classe;
- **Accéder aux propriétés d'un objet**
  - $Nom_de_l_objet->Nom_de_la_donnee_membre = Valeur;
- **Accéder aux méthodes d'un objet**
  - $Nom_de_l_objet->Nom_de_la_fonction_membre(par1,par2,…);
- **La variable $this** permet d'appeler à un objet de s'appeler lui-même :
  - $this->atr1 = $atr1;

Les classes et les objets

Héritage :

- Instruction **extends**

Exemple :

```
Class Voituredeluxe extends Voiture {          // déclaration de la sous classe
var $couleur;
function voituredeluxe() {                     // constructeur
$this->Voiture();
}
function Set_Couleur($couleur) {
$this->couleur = $couleur;
}
functionGet_Couleur() {
return$this->couleur;
}
}
```


Les classes et les objets

Fonctions sur les classes et les objets :

- **get_class()** Retourne la classe d'un objet
- **get_parent_class()** Retourne les super-classes d'un objet
- **method_exists()** Vérifie si la méthode existe dans un objet
- **class_exists()** Vérifie si la classe existe
- **is_subclass_of()** Vérifie si une classe est une sous classe d'une autre
- **get_class_methods()** Retourne les méthodes d'une classe dans un tableau
- **get_declared_classes()** Retourne les classes déclarées dans un tableau
- **get_class_vars()** Retourne les variables de classe dans un tableau
- **get_object_vars()** Retourne les variables d'un objet dans un tableau

Les classes et les objets

Les méthodes magiques sont des méthodes qui, si elles sont déclarées dans une classe, ont une fonction déjà prévue par le langage; elles sont appelées automatiquement :

- _construct() : Constructeur de la classe
- _destruct() : Destructeur de la classe
- _set() : Déclenchée lors de l'accès en écriture une propriété de l'objet
- _get() : Déclenchée lors de l'accès en lecture une propriété de l'objet
- _call() : Déclenchée lors de l'appel d'une méthode inexistante de la classe (appel non statique)
- _callstatic() : Déclenchée lors de l'appel d'une méthode inexistante de la classe (appel statique, depuis PHP 5.3)
- _isset() : Déclenchée si on applique isset() une propriété de l'objet
- _unset() : Déclenchée si on applique unset() une propriété de l'objet
- _sleep() : Exécutée si la fonction serialize() est appliquée à l'objet
- _wakeup() : Exécutée si la fonction unserialize() est appliquée à l'objet
- _toString() : Appelée lorsque l'on essaie d'afficher directement à l'objet :
  echo $object;
- _set_state() : Méthode statique, lancée lorsque l'on applique la fonction var_export() à l'objet
- _clone() : Appel lorsque l'on essaie de cloner l'objet

Interfaces

- Pour éviter l'héritage multiple, les langages orientés-objet utilisent les interfaces d'objet de la classe pour spécifier quelles méthodes une classe doit implanter de l'autre classe
- Pour définir une classe qui implante des interfaces on les déclarer par:

  Implements interface1, interface2, …, interfaceN!
- Toutes les méthodes de ces interfaces doivent être définies dans des classes
- Pour éviter toute ambiguïté, les divers noms de méthodes doivent différents

Interfaces

Exemple d'interface :

```php
interface SpeedRegulator
{
    public function accelerer();
    public function freiner();
}
```

```php
class Voiture implements SpeedRegulator
{
    public $name;
    public $vitesse=0;
    function accelerer()
    {
        if($this->vitesse + 10 < 120)
        $this->vitesse = $this->vitesse + 10;
    }
    function freiner()
    {
        if($this->vitesse - 10 > 0)
        $this->vitesse =  $this->vitesse -10;
    }
}
```

```php
class Camion implements SpeedRegulator
{
    public $name;
    public $vitesse=0;
    function  accelerer()
    {
        if($this->vitesse + 5 < 100)
        $this->vitesse = $this->vitesse + 5;
    }
    function freiner()
    {
        if($this->vitesse - 5 > 0)
        $this->vitesse =  $this->vitesse - 5;
    }
}
```


Les formulaires

Page formulaire

```php
<form action="confirm.php" method="post">
<input type="text" name="nom">
<input type="text" name="adresse">
<input type="submit" value="Terminé">
</form>
```

confirm.php

```php
<p>
  <?php
    echo "Bienvenue <b>".$_POST["nom"]."</b>";
    echo "votre adresse est  <b>".$_POST["adresse"]."</B>";
  ?>
</p>
```

Accès aux champs d'un formulaire

- Si le formulaire est envoyé avec la méthode « **get** », alors on utilise :
  - **$_GET["nom"]**
- Si la méthode d'envoi est « post », alors
  - **$_POST["nom"]**

Sessions

- Les sessions sont un moyen de sauvegarder et de modifier des variables tout au cours de la visite d'un internaute sans qu'elles ne soient visibles dans l'URL et quelque soient leurs types (tableau, objet…).
- Cette méthode permet de sécuriser un site, d'espionner le visiteur, de sauvegarder son panier (e-commerce), etc.
- Les informations de sessions sont conservées en local sur le serveur.

Sessions

Quelques fonctions :

- **session_start()** : démarre une session
- **session_destroy()** : détruit les données de session et ferme la session
- **session_register("var")** : enregistre la variable $var dans la session en cours, attention!!!, ne pas mettre le signe $ (dollars) devant le nom de variable
- **session_unregister("var")** : détruit la variable $varde la session en cours

Les cookies

```
<?php
  setcookie("NomToon", "Roger", time()+3600);
 ?>
<html>
…
```

- Comme pour les sessions, il est **très important** de placer le cookie avant tout autre chose et ne pas laisser d'espace avant la balise *<?php* et le début du fichier.
  - **Warning**: Cannot modify header information - headers already sent

Lecture d'un cookie

- **PHP** place les **cookies** reçus dans un tableau appelé **$_COOKIE[]**
  - Ex.: **$_COOKIE["nom"]**
- Pour vérifier si une variable globale existe utiliser la fonction **isset ( )**
- **Syntaxe :**

  int isset(mixed var)
  - Retourne **true** si la variable existe.
- Exemple : if(isset($_COOKIE["usager"]))…

PHP et les bases de données

PHP offre un interfaçage très simple entre plusieurs base de données :
- – Oracle, MySQL, SQLServer
- – Sybase, Empress, FilePro
- – Interbase, mSQL, PostgreSQL

- La communication avec les bases de données se fait à l'aide du langage SQL.
  - CREATE TABLE : Pour la création d'une table
  - DELETE : Pour la suppression de lignes d'une table
  - INSERT : Pour l'insertion d'une nouvelle ligne
  - SELECT : Pour récupérer des lignes d'une table ou d'une vue
  - UPDATE : Pour modifier les valeurs dans des champs

MySQL

- MySQL est une base de données implémentant le langage de requête SQL un langage relationnel très connu.
- Il existe un outil libre et gratuit développé par la communauté des programmeurs libres : **phpMyAdmin** qui permet l'administration aisée des bases de données **MySQL** avec **php**. Il est disponible sur : http://sourceforge.net/projects/phpmyadmin/ et http://www.phpmyadmin.net.
- Avec **MySQL** vous pouvez créer plusieurs bases de données sur un serveur. Une base est composée de tables contenant des enregistrements.

Connexion

- Pour se connecter à une base depuis **php**, il faut spécifier un nom de serveur, un nom d'utilisateur, un mot de passe et un nom de base.
- ***Les fonctions de connexion :***
  - **mysql_connect ($server,$user,$password)** : permet de se connecter au serveur $server en tant qu'utilisateur $user avec le mot de passe $password, retourne l'identifiant de connexion si succès, FALSE sinon
  - **mysql_select_db ($base[,$id])** : permet de choisir la base $base, retourne TRUE en cas de succès, sinon FALSE
  - **mysql_close ([$id])** : permet de fermer la connexion

- **Exemple :**

```
<?
if( $id= mysql_connect("localhost", "root"," ") ) {
if( $id_db= mysql_select_db("gigabase") ) {
echo"Succès de connexion.";
/* code du script … */
} else{
die("Echec de connexion à la base.");
}
mysql_close($id);
} else{
die("Echec de connexion au serveur de base de données.");
}
?>
```


Interrogation

- Pour envoyer une requête à une base de donnée, il existe la fonction : **mysql_query ($str)** qui prend pour paramètre une chaîne de caractères qui contient la requête écrite en SQL et retourne un identificateur de résultat ou FALSE si échec.
- Les requêtes les plus couramment utilisées sont : **CREATE** (création d'une table), **SELECT** (sélection), **INSERT** (insertion), **UPDATE** (mise à jour des données), **DELETE** (suppression), **ALTER** (modification d'une table), etc.
- **Exemple**:

```
$res= mysql_query ("SELECT * FROM Table");
```

Extraction des données

- Une fois la requête effectuée et l'identificateur de résultat acquis, il ne reste plus qu'à extraire les données retournées par le serveur.
- Sous MySQL, l'affichage des résultats d'une requête se fait ligne par ligne. Une boucle permettra de recueillir chacune des lignes à partir de l'identifiant de résultat.
- Une ligne contient plusieurs valeurs correspondants aux différents attributs retournés par la requête. Ainsi, une ligne de résultat pourra être sous la forme d'un tableau, d'un tableau associatif, ou d'un objet.

Extraction des données

- **mysql_fetch_row($result):** retourne une ligne de résultat sous la forme d'un tableau. Les éléments du tableau étant les valeurs des attributs de la ligne. Retourne FALSE s'il n'y a plus aucune ligne.
- **mysql_num_fields($res)** //Nb_champs
- **mysql_field_name($res)** //nom_champs
- **Exemple 1 :**

```
$requet= "SELECT * FROM users";
if($res= mysql_query($requet)) {
while($ligne= mysql_fetch_row($res)) {
        $id= $ligne[0];
        $name = $ligne[1];
        $address = $ligne[2];
        echo"$id -$name, $address <br />";
}
}
else{
echo"Erreur de requête de base de données.";
}
```

- Ici, on accède aux valeurs de la ligne par leur indice dans le tableau.

Extraction des données

- **mysql_fetch_array($result)**: retourne un tableau associatif. Les clés étant les noms des attributs et leurs valeurs associées leurs valeurs respectives. Retourne FALSE s'il n'y a plus aucune ligne.
- **Exemple 2 :**

```
$requet= "SELECT * FROM users";
if($res= mysql_query($requet)) {
while($ligne= mysql_fetch_array($res)) {
$id= $ligne["id"];
$name= $ligne["name"];
$address= $ligne["address"];
echo"$id -$name, $address <br />";
}
} else
{ echo"Erreur de requête de base de données.";}
```

- Ici, on accède aux valeurs de la ligne par l'attribut dans le tableau associatif.

Extraction des données

- **mysql_fetch_object($result):** retourne un objet. Les attributs de l'objet correspondent à ceux de la ligne de résultat. Et les valeurs des attributs de l'objet correspondent à ceux de la ligne de résultat. Retourne FALSE s'il n'y a plus aucune ligne.