

PHP : Partie 1

Apprendre PHP

PHP : Partie 1

Sommaire

- I. Tour d'horizon
- II. Éléments de la programmation
- III. Interaction avec l'utilisateur

I. Tour d'horizon

Tour d'horizon

- Introduction
 - Historique et caractéristiques
 - Pages statiques # pages dynamiques
 - Intérêt du traitement côté serveur
- Besoins
 - Matériels et logiciels nécessaires
 - Installation de EasyPHP
- Première page en PHP
 - Écriture du script PHP
 - Enregistrement de la page
 - Afficher le résultat

Historique et caractéristiques

- Langage créé en 1994 par Ramus Lerdrof
 - S'appelait Personal Home Page
 - Utilisé pour des besoins personnels (compteurs d'accès, livres d'or, ...)
- En 1997, il est réécrit par Zeev Suraski & Andi Gutmans
 - Version PHP3
 - Signifiait HyperText Preprocessor
- En 2000 PHP4
- Actuellement PHP5

Historique et caractéristiques

- S'intègre dans une page HTML
- S'exécute sur le serveur
- Permet d'accéder aux bases de données
- C'est un produit 'Open source' (code accessible)
- Il est gratuit au même titre que Linux, Apache, MySQL

Pages statiques # pages dynamiques

Lorsque vous visitez une page statique,
<http://www.unserveur.com/index.htm>, par exemple :

- Le client situe le serveur Web spécifié dans la première partie de l'URL (<http://www.unserveur.com>, par exemple)
- Le serveur Web demande ensuite la page statique spécifiée dans la deuxième partie de l'URL (/index.htm, par exemple)
- Le serveur Web envoie le contenu de ce fichier au format HTML
- Le client reçoit alors le fichier HTML envoyé par le serveur Web
- Le client traduit les balises HTML du fichier et affiche ensuite le résultat.

Pages statiques # pages dynamiques

Lorsque vous visitez une page dynamique,
<http://www.unserveur.com/index.php>, par exemple :

- Le client situe le serveur Web spécifié dans la première partie de l'URL (<http://www.unserveur.com>, par exemple)
- Le serveur Web demande ensuite la page dynamique spécifiée dans la deuxième partie de l'URL (/index.php, par exemple)
- Le serveur lit le fichier et traite le code
- Le résultat du traitement est envoyé au client, au format HTML
- Le client reçoit alors le fichier HTML envoyé par le serveur
- Le client traduit les balises du fichier HTML reçu et l'affiche

Pages statiques # pages dynamiques

Page statique (côté client)

- Le navigateur exécute le script après avoir chargé complètement la page
- Le code du script fait partie de la page et peut donc être lu par l'utilisateur (Menu affichage → code source)
- Le code du script doit être pris en charge par le navigateur et les différences entre navigateurs rendent parfois difficile l'écriture des scripts

Page dynamique (côté serveur)

- Le code du script est exécuté par le serveur avant de transmettre au navigateur le résultat
- Le code du script n'est pas transmis au navigateur, il est donc invisible par l'utilisateur
- Le script est indépendant du navigateur, mais impose un équipement adéquat sur le serveur Web

Intérêt du traitement côté serveur

- Diminution du trafic réseau
- Absence de problèmes de compatibilité des navigateurs
- Accès aux bases de données
- Code invisible (confidentialité)

Matériels et logiciels nécessaires

- Un serveur Web :
 - Apache sous Unix, Linux, Windows
 - IIS sous Windows NT/2x
 - PWS sous Windows 95/98
- Un simple éditeur de texte pour la saisie
- Un navigateur pour exécuter les pages
- Éventuellement une base de données (MySQL ou autres)

Installation de EasyPHP

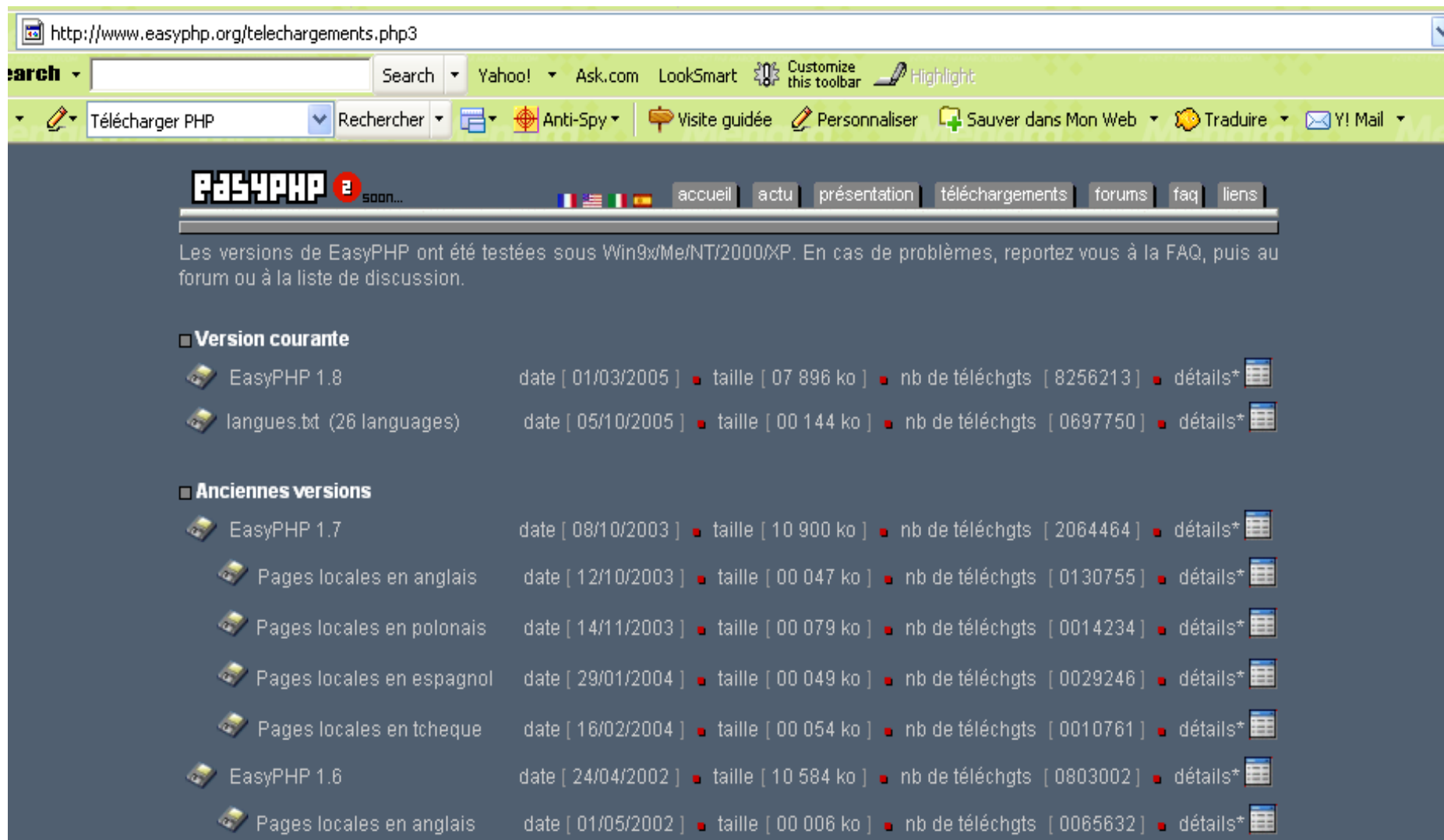
- Téléchargement :

<http://www.easyphp.org/telechargements.php3>

Installation et démarrage

- Double click sur easyphp_xx.exe
- Clic droit sur son icône dans la barre des tâches :
 - Fichier Log : erreurs générées par Apache et MySQL
 - Configuration : configuration de EasyPHP
 - Web Local : ouvre la page <http://localhost>
 - Démarrer/Arrêter : démarre et arrête Apache et MySQL
 - Quitter : ferme EasyPHP
 - ...

Installation de EasyPHP




The screenshot shows the EasyPHP website with the following content:

URL: <http://www.easyphp.org/telechargements.php3>

Search: Search

Navigation: [Télécharger PHP](#) [Rechercher](#) [Anti-Spy](#) [Visite guidée](#) [Personnaliser](#) [Sauver dans Mon Web](#) [Traduire](#) [Y! Mail](#)

EasyPHP  [accueil](#) [actu](#) [présentation](#) [téléchargements](#) [forums](#) [faq](#) [liens](#)

Les versions de EasyPHP ont été testées sous Win9x/Me/NT/2000/XP. En cas de problèmes, reportez vous à la FAQ, puis au forum ou à la liste de discussion.

Version courante

Version	date	taille	nb de téléchgts	détails*
EasyPHP 1.8	01/03/2005	07 896 ko	8256213	détails*
langues.bt (26 langues)	05/10/2005	00 144 ko	0697750	détails*

Anciennes versions

Version	date	taille	nb de téléchgts	détails*
EasyPHP 1.7	08/10/2003	10 900 ko	2064464	détails*
Pages locales en anglais	12/10/2003	00 047 ko	0130755	détails*
Pages locales en polonais	14/11/2003	00 079 ko	0014234	détails*
Pages locales en espagnol	29/01/2004	00 049 ko	0029246	détails*
Pages locales en tcheque	16/02/2004	00 054 ko	0010761	détails*
EasyPHP 1.6	24/04/2002	10 584 ko	0803002	détails*
Pages locales en anglais	01/05/2002	00 006 ko	0065632	détails*

Écriture du script PHP

Page contenant le PHP pur :

```
<?php
//Script PHP pour afficher le contenu d'une variable
$Message= "Bonjour Tout le monde !";
echo $Message;
?>
```

Écriture du script PHP

Page contenant le PHP dans une page HTML :


```
<html>
<body> <b>
<?php
//Script PHP pour afficher le contenu d'une variable
$Message= "Bonjour Tout le monde !";
print $Message;
?>
</b>
</body>
</html>
```

Enregistrement de la page

Si l'installation de base de EasyPHP n'est pas modifiée :

- Créer un dossier dans le rep. www
- Enregistrer la page avec l'extension .php

Afficher le résultat

- S'assurer, d'abord que Apache est démarré (son icône  est actif dans la barre des tâches)
- Deux façons pour afficher la page :
 - Dans le menu Web Local (clic droit sur l'icône de EasyPHP dans la barre des tâches)
 - Invoquer la page avec :
http://localhost/votre_dossier/votre_page.php



II. Éléments de programmation

Éléments de programmation

- Les variables et constantes
 - Déclaration et initialisation
 - Variables et types de données
 - Constantes (utilisateurs et prédéfinies)
 - Tableaux

Éléments de programmation

■ Les instructions

- ❑ commentaires
- ❑ Opérateurs
- ❑ Tests : if, switch
- ❑ Boucles : for, while, do ...while
- ❑ Instructions break et continue

■ Fonctions

- ❑ Définition
- ❑ Déclaration
- ❑ Porté des variables : locales, globales et statiques

Déclaration et initialisation des variables

- Zone ayant un nom pouvant stocker des valeurs qui peuvent changer
- Précédée par le symbole '\$'
- Pas de déclaration explicite
- Initialisée à l'aide du symbole '=' (\$variable = expression)

Variables et types de données

- PHP faiblement typé (variable peut varier au fur et à mesure des affectations successives)
- Exemples
 - Variable entière
 - `$N=5;` (Décimal)
 - `$z=0123` (Octal)
 - `$t=0XFF34` (Hexadécimal)
 - Variable réelle
 - `$x=7.25;`
 - `$y=1.4e10;`
 - Variable texte
 - `$var='jour';`
 - `$var="bon";`

Variables et types de données

- Les variables sont sensibles à la casse : les majuscules sont différents des minuscules
- Une variable peut être interprétée dans une chaîne délimitée par les guillemets (") et non pas les quotes (') :
 - `$user="Yassine";`
 - `$mess1="Bonjour $user";` —► Bonjour Yassine
 - `$mess2='Bonjour $user';` —► Bonjour \$user
- On peut utiliser `\n`, `\'`, `\"`, `\\`, ...

Variables et types de données

Quelques fonctions utiles pour 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 d'une variable
- `settype($var, "type")` convertie le type d'une variable
- `is_long()`, `is_double()`, `is_string()`, `is_array()`, `is_bool()`, `is_object()`, `is_float()`, `is_numeric()`, `is_integer()`, `is_int()`, ...

constantes

- Zones dont le contenu reste fixe
- Ne sont pas précédées par "\$"
- Définies par : `define("constante",valeur)`
`define("author","Ramus");`
`echo author;`
`define("annee",1994);`
`echo annee;`
- Les identificateurs de constantes (et aussi ceux des fonctions, voir plus tard) ne sont pas sensibles à la casse (#variables)

Tableaux

- Variable de type array
- Accepte des éléments de tout type
- Les éléments peuvent être de types différents et sont séparés par des virgules

Tableaux

- Tableau initialisés avec la syntaxe array
 - ❑ `$colors=array("red","green","blue");`
 - ❑ `$tab("M. Yassine",12,7500.75);`
- Tableau initialisé au fur et à mesure
 - ❑ `$colors[]="red";` ou `$colors[0]="red";`
 - ❑ `$colors[]="green";` ou `$colors[1]="green";`
 - ❑ `$colors[]="blue";` ou `$colors[2]="blue";`

Tableaux

- Accès aux éléments du tableau
 - Utilisation des indices (commençant par 0)
- Parcours d'un tableau
 - ```
for ($i=0;$i<count($tab);$i++){
echo $tab[$i]."\n"
}

foreach($tab as $elem){
echo $elem."\n";
}
```

# Tableaux

Quelques fonctions utiles pour les tableaux

- `count($tab)` ou `sizeof($tab)` : retourne le nombre d'éléments
- `in_array($var,$tab)` : TRUE si \$var existe dans \$tab
- `range($i,$j)` : retourne un tableau contenant un intervalle de valeurs
- `shuffle($tab)` : mélange les éléments d'un tableau
- `sort($tab)` : trie un tableau par ordre croissant
- `rsort($tab)` : trie un tableau par ordre décroissant

# Opérateurs

- Arithmétique :  
+, -, \*, /, %, ++, --, +=, -=, \*=, /=, %=
- Logique :  
&& (and), || (or), !, xor
- Comparaison :  
==, <=, >=, !=
- Conditionnel :  
(condion)?(exp1):(exp2)
- Opérateur de concaténation  
. (point)

# commentaires

- Partie ignorée par le compilateur
- Permettent de commenter des programmes
- Trois types :
  - `//` : commentaire en ligne (C++)
  - `#` : commentaire en ligne (Perl)
  - `/*...*/` : commentaire sur plusieurs lignes (C)

---

# Affichage des données

## Trois fonctions

- `echo()`
- `print()`
- `print_r()`



# Affectation

- Consiste à affecter la valeur d'une expression à une variable.
- **\$variable = expression**
- L'expression peut être :
  - ❑ Constante
  - ❑ Variable
  - ❑ Combinaison de constantes et/ou variables

# Tests : if, switch

```
$passe=$_POST["passe"];
if($passe=="mot"){
 echo "Mot de passe correct";
}
else{
 echo "Mot de passe incorrect";
}
```

# Tests : if, switch

```
switch ($choix_menu) {
case 1 :
case 2 : appel_fct_menu1_2();
 // si menu=1 ou 2
 break;
case 3 : appel_fct_menu3();
 break;
default : echo ("vous n'avez pas choisi de menu");
}
```

# Boucles : for, while, do ...while

```
$somme=0;
for ($i=1 ; $i <= 10 ; $i++)
//pour i variant de // 1 à 10 (inclus) par pas de 1
{
 $somme = $somme + $i;
}
echo $somme;
```

# Boucles : for, while, do ...while

```
$n = 10 ;
$somme = 1 ;
while ($n > 0) //tant que n > 0 (inclus) par pas de 1
{
 $somme = $somme * $n;
 $n-- ;
}
echo $somme;
```

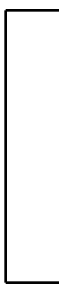
# Boucles : for, while, do ...while

```
$somme=0;
do{
 $somme+=$n;
 $n++;
}
while (n!=0);
echo $somme;
```

# Instructions break et continue

L'instruction break permet d'interrompre prématurément une boucle for ou while.

```
$compt=1;
while ($compt<50) {
 if ($compt / 4 == 10)
 break ; // sortie de boucle
 $compt++;
}
echo ("compt vaut $compt
"); // = 40
```



# Instructions break et continue

L'instruction continue permet de sauter à l'occurrence suivante de la boucle for ou while.

```
for ($i=1 ; $i < 100 ; $i++)
{
 if ($i%4 != 0)
 continue ;//si reste de la division différent de 0
 $multiple = $multiple . "," . $i ;
}
echo ("Liste multiples $multiple"); // = 4,8,12,16,...
```



---

# Fonctions

- Regroupent un ensemble d'instructions réutilisables
- Elles peuvent accepter des arguments et/ou retourner une valeur
- Les identificateurs de fonctions ne sont pas sensibles à la casse

# Fonctions

## Exemple 1

- Déclaration de la fonction

```
function somme($a,$b){
 $s=$a + $b;
 return($s);
}
```

- Appel de la fonction

```
$x=12.5;
$y=15;
$som=somme($x,$y);
```

## Exemple 2

- Déclaration

```
function afficher($a){
 echo "a : $a
";
}
```

- Appel

```
$vnom=$_POST["name"];
$vpass=$_POST["pass"];
$vcoord="Bonjour ";
$vcoord.=vnom." ".vpass
Afficher($vcoord);
```

# III. Interaction avec l'utilisateur

---

# Interaction avec l'utilisateur

Deux façons pour interagir :

- Utilisation des liens hypertexte
- Utilisation des formulaires

# Utilisation des liens hypertexte

## Index.htm

```
<HTML><HEAD><TITLE>Utilisation des liens
hypertexte</TITLE><BODY>
```

Veuillez choisir un livre :

```
Livre A

Livre B

Livre C

</BODY></HTML>
```

## recup.php

```
<?php
```

```
$livre=$_GET["livre"];
```

```
echo "vous avez choisi le livre $livre";
```

```
?>
```

# Utilisation des liens hypertexte

## **Remarque**

*Il est possible de passer plusieurs valeurs dans un lien:*

**`<a href="recup.php?livre=a&chap=1">`**

**`Livre A - Chapitre 1`**

**`</a>`**

# Utilisation des formulaires

Pour utiliser un formulaire en PHP, deux étapes sont nécessaires:

- La première est l'écriture de la page Web de création de formulaire. Il s'agit de la page PHP ou HTML qui comprend de code HTML permettant d'afficher le formulaire.
- La seconde étape est la création du script de traitement de formulaire. Il s'agit de la page PHP qui est mentionnée dans la propriété **action** du formulaire.

---

# Utilisation des formulaires

Le script du traitement du formulaire a deux tâches principales :

- ❑ Stocker les données de champs de formulaire dans des variables
- ❑ Traiter ces variables (informations)



# Utilisation des formulaires

## Remarques

- ❑ Les données envoyées par la méthode **Post** seront récupérées dans la variable **\$\_POST**
- ❑ Les données envoyées par la méthode **Get** seront récupérées dans la variable **\$\_GET**.

# Lire les valeurs des zones de textes

## Page.htm

```
<form method="Post" action="recup.php">
<input type="text" name="nom">
<input type="text" name="prenom">
<input type="submit" value="Valider">
<input type="Reset" value="Annuler">
</form>
```

## recup.php

```
<?php
$vnom=$_POST["nom"];
$vprenom=$_POST["prenom"];
echo "Bonjour $vnom $vprenom";
?>
```

# Lire les valeurs d'une liste

- Une liste est créée avec deux balises HTML `<select>` et `<option>`.
- La balise `<select>` possède une propriété importante, **Name**, qui indique le nom de la liste.
- La propriété **Value** de la balise `<option>` indique la valeur de l'option sélectionnée de la liste qui sera envoyée pour être traitée par une page PHP.

# Lire les valeurs d'une liste

## Index.htm

```
<form method="Post" action="recup.php">
Indiquez le jour de la semaine :<select name="jours_sem">
<option value="Dim">Dimanche
<option value="Lun">Lundi
<option value="Mar">Mardi
<option value="Mer">Mercredi
<option value="Jeu">Jeudi
<option value="Ven">Vendredi
<option value="Sam">Samedi
</select>
<input type="submit" value="valider"><input type="reset" value="Annuler">
</form>
```

# Lire les valeurs d'une liste

## recup.php

...

```
<?php
```

```
$jour=$_POST["jour_sem"];
```

```
echo "Nous sommes le $jour";
```

```
?>
```

...

# Lire les boutons d'option

- Les boutons d'options sont utilisés lorsque vous voulez donner à l'utilisateur la possibilité de sélectionner une et une seule option dans un groupe de boutons.
- Les boutons d'un même groupe doivent avoir le même nom (propriété **Name**) et des valeurs différentes (propriété **Value**).
- La valeur transmise à la page PHP, et qui indiquera le choix de l'utilisateur, est la propriété **Value** du groupe des boutons d'option.

# Lire les boutons d'option

## Index.htm

```
<form method="Get" action="recup.php">
Quelle tranche d'âge avez-vous ?

<input type="radio" name="age" value="Moins de 18 ans" checked>

<input type="radio" name="age" value="entre 19 et 25 ans">

<input type="radio" name="age" value="Plus que 26 ans">

<input type="submit" value="valider"><input type="reset" value="Annuler">
```

## recup.php

```
<?php
$vage=$_GET["age"];
?>
Votre tranche d'âge est : <?php echo $vage ?>
```

# Les valeurs de cases à cocher

- Les cases à cocher sont les contrôles les plus difficiles à gérer dans un formulaire, et ceci est dû au fait qu'un utilisateur peut ne cocher aucune case, comme il peut en cocher plusieurs.
- Un groupe de cases à cocher est créé en attribuant la valeur **checkbox** à la propriété **type** et la même valeur pour la propriété **Name sous forme de tableau**. Seule la valeur de **Value** permet de différencier entre les cases à cocher d'un même groupe.
- Dans un script de traitement de formulaire, les valeurs des cases qui sont cochées sont stockées sous forme de tableau.



# Les valeurs de cases à cocher

```
...
<input type=checkbox name=c[] value=rouge>Couleur rouge

<input type=checkbox name=c[] value=vert>Couleur Verte

<input type=checkbox name=c[] value=bleu>Couleur bleue

```

```
...
alors
```

```
...
$couleur=$_POST["c"] est le tableau contenant les valeurs : rouge, bleu
 (ceci suppose que seule les deux cases roue et bleu sont cochées)
```

# Les valeurs de cases à cocher

## Index.htm

```
<form method="Post" action="recup.php">
```

Choisir les cours qui vous intéresse : <br>

```
<input type=checkbox name=cours[] value=HTML>HTML
```

```
<input type=checkbox name=cours[] value=Javascript>Javascript
```

```
<input type=checkbox name=cours[] value=PHP>PHP
```

```
<input type=checkbox name=cours[] value=Java>Java
```

```
<input type=checkbox name=cours[] value=AJAX>AJAX
```

```
<input type=submit>
```

```
<input type=reset>
```

```
</form>
```

# Les valeurs de cases à cocher

## recup.php

```
<?php
$cours=$_POST["cours"];
$n= count($cours);
if ($n>=1){
 echo "Liste des cours :
";
 for($i= 1;$i<$n;$i++)
echo "$cours[$i]
";
}
else
 echo "Vous n'avez choisi aucun cours";
?>
```

# PHP : Partie 2


## Gestion d'une base de données

---

# Gestion d'une base de données

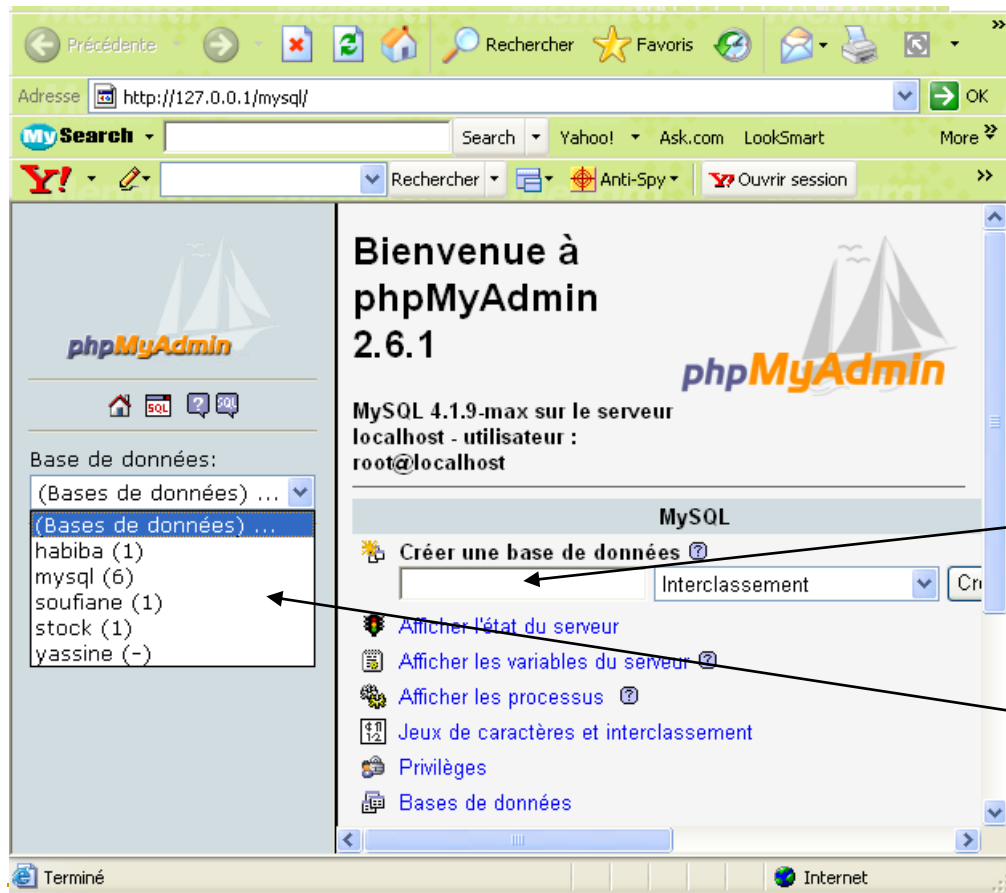
- Création d'une base de données MySQL
- Interrogation d'une base MySQL
- Accès à d'autres types de base
- Administration MySQL

# Création d'une base de données MySQL

- La création d'une base de données peut être faite à l'aide de **PhpmyAdmin**
- Pour démarrer PhpmyAdmin :
  - ❑ Clic droit sur l'icône
  - ❑ Choisir configuration, puis  nyAdmin



# Création d'une base de données MySQL



Le navigateur propose de :

- créer une nouvelle base de données
- choisir parmi les bases déjà créées

# Création d'une base de données MySQL

The screenshot shows the phpMyAdmin interface in a web browser. The address bar shows 'http://127.0.0.1/mysql/'. The browser's search bar and various toolbars are visible at the top. In the main content area, the server is 'localhost', the database is 'stock', and the table is 'Emp'. Below this, a table defines the structure of 'Emp' with the following columns:

Champ	Type ?	Taille/Valeurs*	Interclassement
EmpNo	INT		
EName	VARCHAR	20	
Sal	FLOAT		
Comm	FLOAT		
DeptNo	INT		

On y trouve les colonnes suivantes :

- **Champ** : nom du champ
- **Type** : type de données
- **Longueur** : longueur de la données
- **Attributs** : type de valeurs (binaire, ...)
- **Null** : la valeur peut être nulle ou pas
- **Défaut** : valeur par défaut

- **Extra** : numérotation automatique
- **Primaire** : clé primaire
- **Index** : champ index (utilisé lors de la recherche)
- **Unique** : les doublons sont interdits dans ce champ



# Interrogation d'une base MySQL

L'accès à la base se fait en plusieurs étapes:

- ❑ Ouvrir la connexion sur le serveur MySQL
- ❑ Ouvrir la connexion sur la base de données
- ❑ Définir la requête à exécuter
- ❑ \* Exécuter la requête. On obtient un recordset
- ❑ \* Extraire les données de chaque ligne du recordset
- ❑ \* Traiter les données de chaque ligne
- ❑ Fermer le recordset pour libérer la mémoire
- ❑ Fermer la connexion

\* : Dans le cas d'une requête SELECT

# Ouvrir la connexion sur le serveur MySQL

- `int mysql_connect("server","user","password")`
- Retourne TRUE (si la connexion a réussie) ou FALSE (sinon)
- La connexion sera fermée automatiquement à la fin du script, à moins d'utiliser `mysql_close()`

# Ouvrir la connexion sur le serveur MySQL

```
//connexion locale, login root et pas de mot de passe
$con=mysql_connect("localhost","root","");
if($con==FALSE){
//connexion échouée, affiche un message et arrêt du script
die("connexion impossible");
}
echo "connexion établie avec succès";
//suite du traitement
```

# Ouvrir la connexion sur la base de données

- `int mysql_select_db("base de données")`
- Retourne TRUE (si connexion à la base établie avec succès), FALSE (sinon)
- Les requêtes exécutées avec `mysql_query()` seront faite dans la base active

# Ouvrir la connexion sur la base de données

```
//connexion à la base de données "Stock"
$base=mysql_select_db("Stock");
if($base==FALSE){
//si connexion a échouée
die("Connexion à la base a échouée");
//affiche un message et arrête le script
}
echo "Connexion à la base a réussie";
//Suite du traitement
```

# Création et exécution de la requête

- La fonction `mysql_query()` envoie une requête à la base de données active
- `Resultat mysql_query(requête)`
- Retourne :
  - ❑ FALSE (si la requête a échouée)
  - ❑ Le recordset contenant le résultat de la requête (si la requête est exécutée avec succès)

# Création et exécution de la requête

```
//Requête
$req="select * from Emp where comm is null";
//exécution de la requête
$resultat=mysql_query($req);
if ($resultat==FALSE){
//si echec de la requête, affichage d'un message et arrête la script
die("Requête incorrecte : $req");
}
//suite du traitement
echo "Requête exécutée avec succès";
```

# Extraction des données de chaque ligne du recordset

- mysql\_affected\_rows(\$resultat) : pour connaître le nombre de lignes affectées par les instructions : `delete`, `insert`, `update` et `replace`
- mysql\_num\_rows (\$resultat) : indique le nombre d'enregistrements qui répondent à la requête `select`
- mysql\_fetch ... (\$resultat) : retourne une ligne du recordset retourné par la fonction `mysql_query()`



# Extraction des données de chaque ligne du recordset

- mysql\_fetch\_row(\$resultat) : affiche le résultat selon la position (indice) du champ
- mysql\_fetch\_object(\$resultat) : affiche le résultat selon le nom donné au champ
- mysql\_fetch\_array(\$resultat) : affiche le résultat selon la position (indice) ou le nom donné au champ

# Extraction des données de chaque ligne du recordset

```
$req="select * from Emp where comm is null";
$resultat=mysql_query($req);
if ($resultat==FALSE){
 die("échec de la requête");
}
else{
 for($i=1;$i<=mysql_num_rows ($resultat);$i++){
 $lig=mysql_fetch_row($resultat);
 echo $lig[0]." ".$lig[1];
 }
}
```

# Extraction des données de chaque ligne du recordset

```
$req="select * from Emp where comm is null";
$resultat=mysql_query($req);
if ($resultat==FALSE){
 die("échec de la requête");
}
else{
 for($i=1;$i<=mysql_num_rows($resultat);$i++){
 $lig=mysql_fetch_object($resultat);
 echo $lig->empno." ".$lig->ename;
 }
}
```

# Extraction des données de chaque ligne du recordset

```
$req="select * from Emp where comm is null";
$resultat=mysql_query($req);
if ($resultat==FALSE){
 die("échec de la requête");
}
else{
 for($i=1;$i<=mysql_num_rows($resultat);$i++){
 $lig=mysql_fetch_array($resultat);
 echo $lig[0]." ".array["ename"];
 }
}
```

# Libération de la mémoire et fermeture de la connexion

- mysql\_free\_result() : efface le résultat de la mémoire

Exemple :

```
mysql_free_result($resultat);
```

- mysql\_close() : ferme la connexion mysql

Exemple

```
mysql_close($con);
```

# Exemple Complet (1)

```
<?php
//connection au serveur MYSQL
$con=mysql_connect("localhost","root","");
if($con==false){
 die("Impossible de se connecter au serveur !");
}
//connection à la base de données STOCK
$mabase=mysql_select_db("stock");
if($mabase==false){
 die("Impossible d'ouvrir la base de données !");
}
```

# Exemple Complet (2)

*//envoi de la requête SQL*

```
$req="select * from produits";
$result=mysql_query($req);
if($result==false){
 die("Erreur dans la requête : $req");
}
```

# Exemple Complet (3)

*//affichage des données*

```
echo "Nombre de lignes ".mysql_num_rows($result);
echo "<table border bordercolor=red cellpadding=0
 width=80%><tr><td>Ref</td><td>Désignation</td><
 td>PU</td><td>Qtes</td><td>Cat</td></tr>";
for($i=1;$i<=mysql_num_rows($result);$i++){
 $rs=mysql_fetch_row($result);
?>
```



## Exemple Complet (4)

```
<tr><td><?php echo $rs[0]?></td><td><?php echo
 $rs[1]?></td><td><?php echo $rs[2]?></td><td><?php echo
 $rs[3]?></td><td><?php echo $rs[4]?></td></tr>

<?php
}
echo "</table>";
mysql_free_result($result);
mysql_close($con);
?>
```

# PHP : Partie III

## PHP5 et l'Orienté Objet

# Introduction

- Depuis son apparition, PHP avait favorisé l'approche procédural
- Malgré l'introduction de la notion d'objet dans la version 4, plusieurs concepts n'étaient pas implémenté :
  - ❑ La notion de visibilité : tout est publique
  - ❑ Pas de constructeur, ni de destructeur
  - ❑ Pas de gestion des exceptions
  - ❑ Pas d'interfaces
  - ❑ Etc.

---

# Définition d'une classe

- La définition d'une classe commence par le mot clé 'class' suivi du nom et d'une paire d'accolades
- Les accolades renferment la définition des membres et des méthodes
- La pseudo-variable '\$this' fait référence à l'objet courant

# Définition d'une classe

```
class personne{
 private $nom;
 private $prenom;
 public function __construct($nom,$prenom){
 echo "Création et Initialisation de l'objet
";
 $this->nom=$nom;
 $this->prenom=$prenom;
 }
 public function getInf(){
 return "Bonjour $this->nom $this->prenom";
 }
 public function __destruct(){ echo "destruction de l'objet";}
}
```

# Définition d'une classe

- Trois types de visibilité sont utilisés :
  - (publique, public) : visible par tous
  - (privée, private) : visible seulement dans la classe
  - (protégée, protected) : visible seulement dans la classe et dans les sous-classes directes de la classe.
- Si aucune visibilité n'est précisée → visibilité publique

# Instanciación d'un objet

La création d'un objet à partir d'une classe se fait par le mot clé "new".

```
$p=new personne("Alami","Mourad");
echo $p->getInf();
```

## Attention

Les données (privées) ne sont accessibles qu'à travers les méthodes

# Instanciación d'un objet

//script pour tester la classe personne

<?php

//Création d'un objet de la classe personne

\$p=new personne("Alami","Mourad");

//accès aux données privées → Erreur

echo \$p->code;

//accès aux données publiques

echo \$p->getInf();

?>



# Héritage entre classes

- Une classe peut hériter les membres et les méthodes d'une autre classe
- L'héritage ne peut être que simple (pas d'héritage multiple)
- Méthodes héritées peuvent être surchargées, à moins que la classe parente ait défini une méthode comme 'final'
- 'parent::' permet d'accéder à des membres et méthodes de la classe mère

# Héritage entre classes

```
Class employe extends personne{
 private $salaire;
 public function __construct($nom,$prenom,$salaire){
 parent::__construct($nom,$prenom);
 $this->salaire=$salaire;
 }
 public function getInf(){
 parent::getInf();
 echo "$this->salaire";
 }
}
```

# Héritage entre classes

## Remarque

Le constructeur/destructeur de la classe mère ne sont pas appelés implicitement. Il faut le faire explicitement en utilisant

- ❑ `parent::__construct()`
- ❑ `parent::__destruct()`

# Opérateur de résolution de portée (::)

- Appelé aussi 'Paamayim Nekudotayim' (double deux points en hebreu). Nom choisi lors de l'écriture du Zend Engine 0.5 (fourni avec PHP 3)
- Permet d'accéder aux membres
  - ❑ Constantes
  - ❑ Statiques
  - ❑ Redéfinis par une classe

# Constantes de classes

- Il est possible de définir des constantes dans une classe
- Les constantes diffèrent des variables par le fait :
  - ❑ Leur contenu reste fixe
  - ❑ N'utilisent pas '\$' pour les déclarer/utiliser
  - ❑ Ne sont pas accessibles depuis une instance de l'objet

# Constantes de classes

```
<?php
```

```
class Classe1
```

```
{
 const constant = 'valeur constante';
 public function afficherConstante() {
 echo self::constant . "
";
 }
}
```

'self::' Remplace '\$this->'

```
echo Classe1::constant . "
";
```

```
$classe1 = new Classe1();
$classe1->afficherConstante();
// echo $classe1::constant; n'est pas autorisé
```

```
?>
```

# Données de classes/d'instances

Une donnée (attribut ou méthode) peut être :

- Donnée d'instance, propre à l'objet
- Donnée de classe, partagée entre tous les objets de la classe

# Données de classes/d'instances

## Une donnée d'instances

- Attribut : chaque objet possède sa propre valeur
- Méthode : invoquée en faisant référence à l'objet



# Données de classes/d'instances

Une donnée de classe (déclaration précédée du mot clé "static")

- Attribut : partagée entre tous les objets de la classe.
- Méthode : invoquée en précisant la classe à la place de l'objet

---

# Membres et méthodes statiques

- PHP5 permet de déclarer des membres et des variables statiques
- Permet un accès sans avoir besoin d'une instance de classe
- Le mot clé 'static' vient après l'opérateur de visibilité dans la déclaration

# Membres et méthodes statiques

//définition de la classe

```
<?php class personne{
 private $code;
 private $nom;
 private static $nbre=0;
 public function __construct($code,$nom){
 self::$nbre++;$this->code=$code;$this->nom=$nom;
 }
 public function getInf(){
 echo "Code:". $this->code."
nom:". $this->nom;
 }
 public static function getNbre(){
 echo "nombre:".self::$nbre;
 }
}??>
```

# Membres et méthodes statiques

//utilisation de la classe

```
<?php include "personne.php" ?>
```

```
<?php
```

```
 $t1=new personne(1,"Alami");
```

```
 $t1->getInf(); //méthode d'instance
```

```
 personne::getNbre();//méthode de classe
```

```
 $t2=new personne(2,"Berrada");
```

```
 $t2->getInf();
```

```
 personne::getNbre();
```

```
?>
```

# Gestion des exceptions

- PHP5 gère les exception d'une manière similaire à d'autres langages (Java, C#, ...)
- Chaque «try» doit avoir au moins un bloc «catch» correspondant
- Plusieurs blocs «catch» peuvent être utilisés pour attraper différentes classes d'exception

# Gestion des exceptions

La classe prédéfinie 'Exception' :

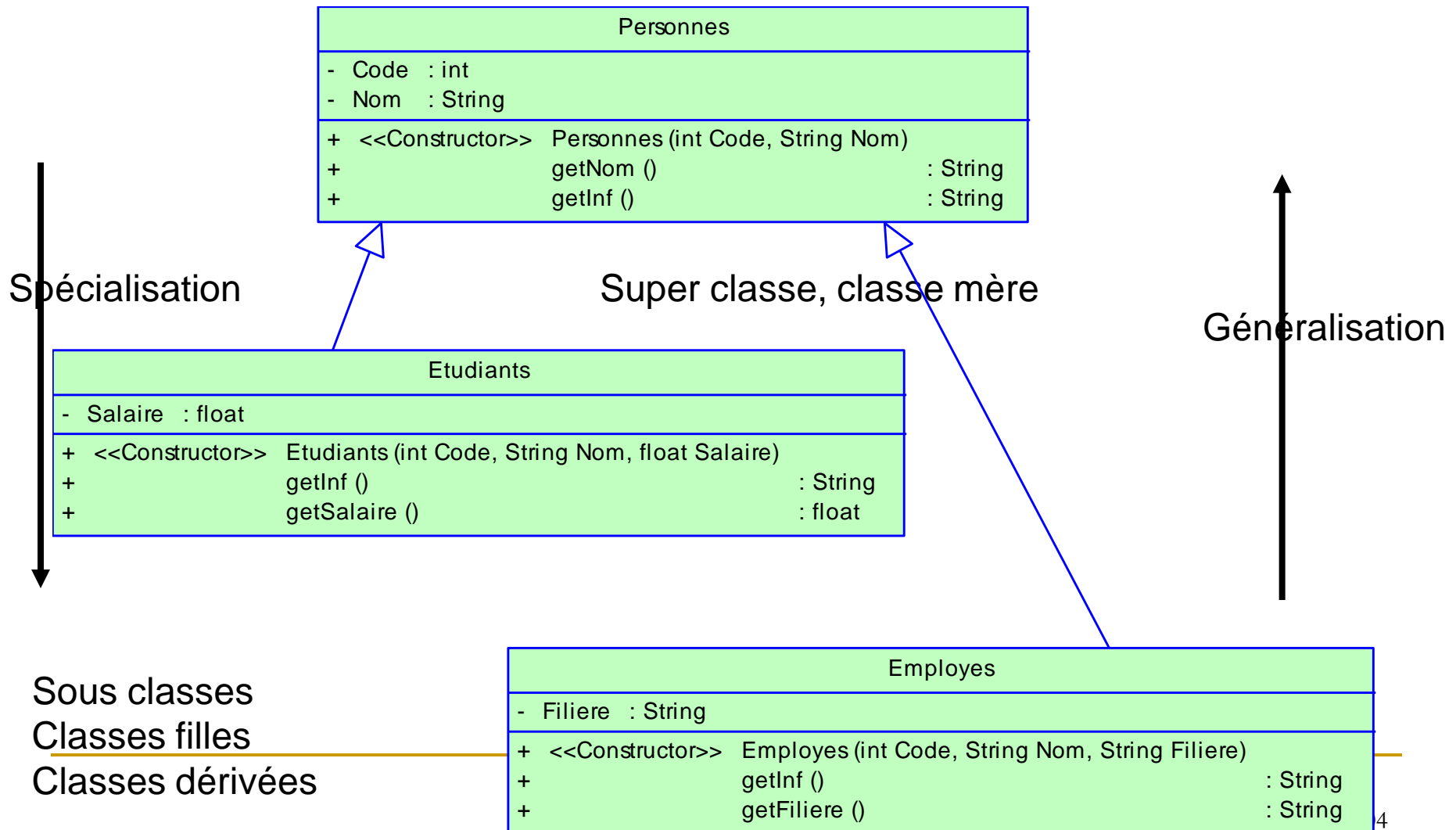
```
class Exception {
 protected $message; // message de l'exception
 protected $code = 0; // code de l'exception défini par l'utilisateur
 protected $file; // nom du fichier source de l'exception
 protected $line; // ligne de la source de l'exception
 function __construct(string $message=NULL, int code=0);
 final function getMessage(); // message de l'exception
 final function getCode(); // code de l'exception
 final function getFile(); // nom du fichier source
 final function getLine(); // ligne du fichier source

}
```

# Gestion des exceptions

```
<?php
try{
 $x=$_POST['nbre1'];$y=$_POST['nbre2'];
 if ($y==0) throw new Exception("Division par 0",2);
 $s=$x/$y;
 echo $s;
}
catch(Exception $e){
 echo $e->getMessage()."
"; //Division par 0
 echo $e->getCode()."
";
 echo $e->getLine()."
";
 echo $e->getFile()."
";
}
?>
```

# Application





# Application

## **Question1**

- Créer 3 fichiers d'inclusion contenant les définitions des classes :
  - Personnes.php
  - Etudiants.php
  - Employes.php
- Créer un fichier "Main.php" qui permet de tester les classes créées

## **Question2**

- Refaire la question1 pour que le code soit «auto incrément»

## **Question3**

- Supposons que nous voulons connaître le nombre de personnes créées. Pour cela, ajouter un attribut statique «nbre» et une méthode statique «getNbre()» permettant de retourner le nombre.

# Fichier de définition de la classe

## Personnes1.php

```
<?php
class Personnes{
 private $code;
 private $nom;
 public function __construct($code,$nom){
 $this->code=$code;
 $this->nom=$nom;
 }
 public function getNom(){
 return "Nom: ".$this->nom;
 }
 public function getInf(){
 return "Code :".$this->code."
Nom :".$this->nom;
 }
}
?>
```

# Fichier de définition de la classe Etudiants1.php

```
<?php
class Etudiants extends Personnes{
 private $filiere;
 public function __construct($code,$nom,$filiere){
 parent::__construct($code,$nom);
 $this->filiere=$filiere;
 }
 public function getFiliere(){
 return $this->filiere;
 }
 public function getInf(){
 return parent::getInf()."
Filière: ".$this->filiere;
 }
}
?>
```

# Fichier de définition de la classe

## Employes1.php

```
<?php
class Employes extends Personnes{
 private $salaire;
 public function __construct($code,$nom,$salaire){
 parent::__construct($code,$nom);
 $this->salaire=$salaire;
 }
 public function getSalaire(){
 return $this->salaire;
 }
 public function getInf(){
 return parent::getInf()."
Salaire: ".$this->salaire;
 }
}
?>
```

# Fichier de test

## Main1.php

```
<?php include "Personnes1.php" ?>
<?php include "Etudiants1.php" ?>
<?php include "Employes1.php" ?>
<?php
 $pers=new Personnes(1,"Madani");
 echo $pers->getNom();
 echo $pers->getInf();
 echo "<hr>";
 $etud=new Etudiants(1,"Alami","Informatique de Gestion");
 echo $etud->getNom();
 echo $etud->getInf();
 echo "<hr>";
 $emp=new Employes(1,"Mouradi",3500.50);
 echo $emp->getNom();
 echo $emp->getInf();
?>
```

# Fichier de définition de la classe

## Personnes2.php

```
<?php
class Personnes{
 private static $compteur=0;
 private $code;
 private $nom;
 public function __construct($nom){
 self::$compteur++;
 $this->code=self::$compteur;
 $this->nom=$nom;
 }
 public function getNom(){
 return $this->nom;
 }
 public function getInf(){
 return "Code : ".$this->code."
Nom : ".$this->nom;
 }
}
?>
```

# Fichier de définition de la classe Etudiants2.php

```
<?php
class Etudiants extends Personnes{
 private $filiere;
 public function __construct($nom,$filiere){
 parent::__construct($nom);
 $this->filiere=$filiere;
 }
 public function getFiliere(){
 return $this->filiere;
 }
 public function getInf(){
 return parent::getInf()."
Filière: ".$this->filiere;
 }
}
?>
```

# Fichier de définition de la classe

## Employes2.php

```
<?php
class Employes extends Personnes{
 private $salaire;
 public function __construct($nom,$salaire){
 parent::__construct($nom);
 $this->salaire=$salaire;
 }
 public function getSalaire(){
 return $this->salaire;
 }
 public function getInf(){
 return parent::getInf()."
Salaire: ".$this->salaire;
 }
}
?>
```



# Fichier de test

## Main2.php

```
<?php include "Personnes2.php" ?>
<?php include "Etudiants2.php" ?>
<?php include "Employes2.php" ?>
<?php
 $p1=new Personnes("Madani");
 $p2=new Personnes("Najd");
 echo $p1->getNom(); echo $p2->getNom();
 echo $p1->getInf(); echo $p2->getInf();
 echo "<hr>";
 $etud=new Etudiants("Alami","IG");
 echo $etud->getNom();
 echo $etud->getInf();
 echo "<hr>";
 $emp=new Employes("Mouradi",3500.50);
 echo $emp->getNom();
 echo $emp->getInf();
?>
```

# Fichier de définition de la classe

## Personnes3.php

```
<?php
class Personnes{
 private static $compteur=0;
 private static $nombre=0;
 private $code;
 private $nom;
 public function __construct($nom){
 self::$compteur++;
 self::$nombre++;
 $this->code=self::$compteur;
 $this->nom=$nom;
 }
 public function getNom(){
 return $this->nom;
 }
 public function getInf(){
 return "Code :".$this->code."
Nom :".$this->nom;
 }
 public static function getNombre(){
 return self::$nombre;
 }
}
?>
```

# Fichier de définition de la classe Etudiants3.php

```
<?php
class Etudiants extends Personnes{
 private $filiere;
 public function __construct($nom,$filiere){
 parent::__construct($nom);
 $this->filiere=$filiere;
 }
 public function getFiliere(){
 return $this->filiere;
 }
 public function getInf(){
 return parent::getInf()."
Filière: ".$this->filiere;
 }
}
?>
```

# Fichier de définition de la classe

## Employes3.php

```
<?php
class Employes extends Personnes{
 private $salaire;
 public function __construct($nom,$salaire){
 parent::__construct($nom);
 $this->salaire=$salaire;
 }
 public function getSalaire(){
 return $this->salaire;
 }
 public function getInf(){
 return parent::getInf()."
Salaire: ".$this->salaire;
 }
}
?>
```

# Fichier de test

## Main3.php

```
<?php include "Personnes3.php" ?>
<?php include "Etudiants3.php" ?>
<?php include "Employes3.php" ?>
<?php
 $p1=new Personnes("Madani");
 $p2=new Personnes("Najd");
 echo $p1->getNom(); echo $p2->getNom();
 echo $p1->getInf(); echo $p2->getInf();
 echo "<hr>";
 echo Personnes::getNombre();
 echo "<hr>";
 $etud=new Etudiants("Alami","IG");
 echo $etud->getNom();
 echo $etud->getInf();
 echo "<hr>";
 $emp=new Employes("Mouradi",3500.50);
 echo $emp->getNom();
 echo $emp->getInf();
?>
```

# PHP : Partie IV

## PHP to PDF

---

# Introduction

- FPDF est une classe PHP qui permet de générer des fichiers PDF en pur PHP.
- Le F de FPDF signifie Free : vous êtes libre de l'utiliser et de la modifier comme vous le souhaitez.
- Dans la suite nous présenterons quelques méthodes concernant l'insertion du texte.

# Méthode Text

- Syntaxe

Text(**float** x, **float** y, **string** txt)

- **Description**

Imprime une chaîne de caractères. L'origine est à gauche du premier caractère, sur la ligne de base. Cette méthode permet de positionner précisément une chaîne dans la page.

- **Paramètres**

- ❑ x Abscisse de l'origine.
- ❑ y Ordonnée de l'origine.
- ❑ txt Chaîne à imprimer.



# Méthode write

- **Syntaxe**

Write(**float** h, **string** txt)

- **Description**

Cette méthode imprime du texte à partir de la position courante.

Lorsque la marge droite est atteinte (ou que le caractère \n est rencontré), un saut de ligne est effectué et le texte continue à partir de la marge gauche.

- **Paramètres**

- h Hauteur de la ligne.
- txt Chaîne à imprimer.

# Méthode SetFont

## ■ Syntaxe

SetFont(**string** family [, **string** style [, **float** size]]) **Version**

## ■ Description

Fixe la police utilisée pour imprimer les chaînes de caractères.

## ■ Paramètres

- ❑ family Famille de la police : Courier (caractères de largeur fixe), Helvetica ou Arial (synonymes; sans serif), Times (avec serif), etc
- ❑ style Style de la police. Les valeurs possibles sont (indépendamment de la casse) :
  - ❑ chaîne vide : normal, B : gras, I : italique, U : souligné
- ❑ size Taille de la police en points.

---

# Méthode AddPage

- **Syntaxe**

AddPage([**string** orientation])

- **Description**

Ajoute une nouvelle page au document

- **Paramètres**

orientation : Orientation de la page. P ou Portrait, L ou Landscape. La valeur par défaut est Portrait.

# Exemple

```
<?php
include "phpToPDF.php";
$pdf=new phpToPDF();
$pdf->addpage();
//Début en police normale
$pdf->SetFont('Arial','',14);
$pdf->Write(5,'Visitez ');
//Lien en bleu souligné
$pdf->SetTextColor(0,0,255);
$pdf->SetFont('','U');
$pdf->Write(5,'www.fpdf.org','http://www.fpdf.org');
$pdf->output();
?>
```