Webitech nextgroup

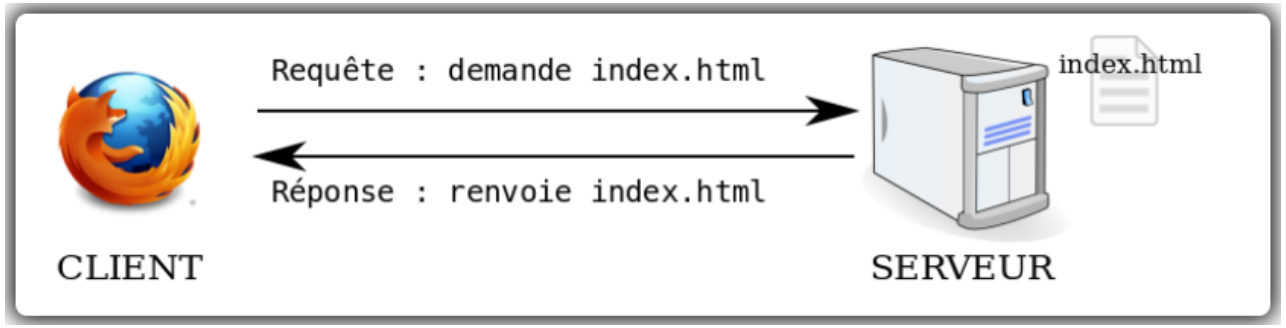
BTS SIO Option SLAM

# Module 3 - Développement Web Coté Serveur : PHP et MySQL

## CM1: L'essentiel de PHP

Sarah Malaeb
2022-2023

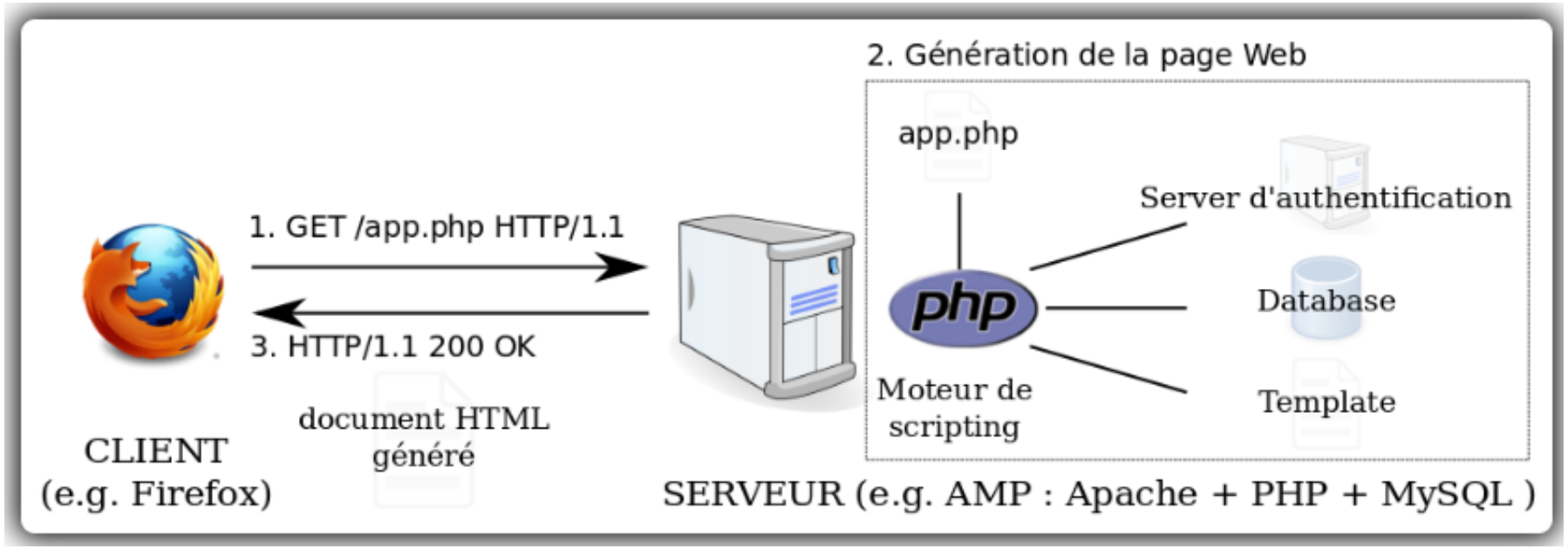
# Rappel: Architecture Web Client / Serveur




Le client fait une requête que serveur, qui répond en donnant la page Web.

- **Le client**
  - c'est le visiteur d'un site Web. Il demande la page Web au serveur. En pratique, vous êtes des clients quand vous surfez sur le Web.
  - Plus précisément c'est votre navigateur Web (Firefox, Chrome, Safari, IE, Edge…) qui est le client, car c'est lui qui demande la page Web.
- **Le serveur**
  - ce sont les ordinateurs qui délivrent les sites Web aux internautes, c'est-à-dire aux clients.

# Rappel: Architecture Web Client / Serveur

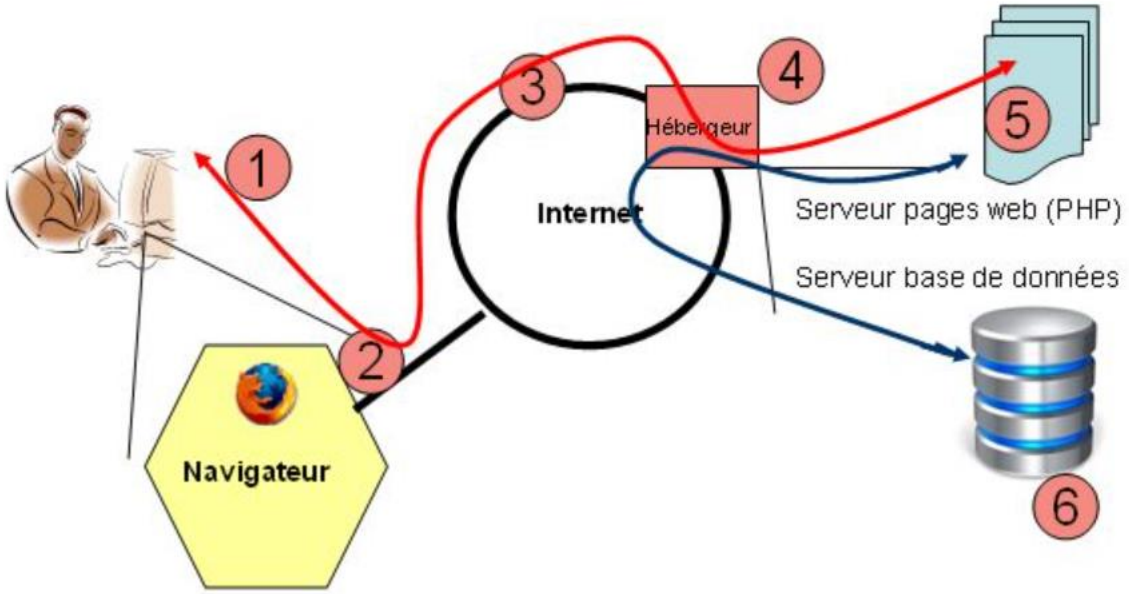



**Site Web dynamique**

- Le client demande au serveur à voir une page Web (requête HTTP) ;
- Le serveur crée la page spécialement pour le client (en suivant les instructions du PHP) ;
- Le serveur lui répond en lui envoyant la page qu'il vient de générer (réponse HTTP).

# Principe des échanges avec le serveur de bases de données

- L'internaute (**1**) se connecte (via **2, 3, 4** sur la page WEB hébergée (**5**).
- Les instructions php contenues dans la page WEB nécessite des demandes de données contenues dans le serveur de base de données (**6**).
- La demande et la réception des données s'effectuent à l'aide d'une requête **SQL**. Les donnéees récupérées sont utilisées pour mettre à jour la page WEB qui est envoyée à l'utilisateur.




Source: monwebfacile.free.fr/Tutoriaux/PHP/PHP_MYSQL00.php

# C’est quoi PHP?


- PHP (Hypertext Preprocessor) est un langage de programmation open source utilisé pour créer des pages web.

- Le langage PHP peut s'exécuter sur différentes plates-formes et est compatible avec presque tous les serveurs.

- Les fichiers PHP ont l'extension **.php**.

# C’est quoi PHP?


- PHP peut effectuer des fonctions telles que la création, l'ouverture, la lecture, l'écriture et la fermeture de fichiers sur un système.

- Vous pouvez ajouter, supprimer et modifier des éléments dans votre base de données à l'aide de PHP.

- Vous pouvez également restreindre l'accès de certains utilisateurs à certaines pages de votre site web en utilisant PHP.

# C’est quoi PHP?


- PHP est le langage de programmation web côté serveur le plus populaire et le plus largement utilisé pour le développement web.

- Cependant, il nécessite un serveur web pour exécuter même une page web développée localement.

- Il existe différents logiciels de serveur web pour configurer notre serveur web local. Parmi eux, PHP XAMPP et WampServer sont les plus populaires. Alors que WampServer est uniquement disponible pour la plateforme Windows, XAMPP est une application multiplateforme qui peut s'exécuter sur Windows, Linux et Mac OS.

- Par conséquent, durant ce semestre, vous apprendrez PHP en utilisant XAMPP.

# Avantages de PHP

Les développeurs web utilisent de nombreux autres langages, mais la plupart préfèrent utiliser PHP en raison de ses avantages. Voici quelques avantages de PHP :

- **Simplicité et facilité d'apprentissage** - PHP est considéré comme le langage de script le plus facile, car il ne nécessite pas d'études intensives. Les commandes sont très faciles à comprendre pour les nouveaux apprenants et les développeurs également.

- **Compatibilité** - Le langage PHP est compatible car il peut s'exécuter sur de nombreux systèmes d'exploitation. Il peut facilement fonctionner sur des plateformes telles que Windows, LINUX et UNIX.

- **Flexibilité** - Le langage PHP est très flexible pour les développeurs car il vous permet de modifier un projet existant ou terminé.

# Avantages de PHP

- **Moins coûteux** - Comme PHP est un langage open-source, vous pouvez le télécharger gratuitement. Vous n'avez pas besoin d'acheter de licence ou de logiciel.

- **Modèle MVC** - Le modèle Modèle-Vue-Contrôleur en PHP vous aide à organiser les codes.

- **Temps de chargement** - PHP est plus rapide que d'autres langages de programmation. Il peut être chargé même lorsque votre connexion réseau est lente.

- **Support de bibliothèque** - PHP dispose également d'une collection de nombreux codes avancés déjà écrits que vous pouvez utiliser de manière répétée. Vous pouvez également l'utiliser chaque fois que vous souhaitez exécuter un programme.

# Comment configurer et installer PHP pour votre projet ?

- Étape 1 : Trouvez un serveur web qui prend en charge PHP et MYSQL.
- Étape 2 : Ensuite, installez PHP à partir de son site web.
- Étape 3 : Ensuite, installez la base de données MYSQL sur votre ordinateur.

- Durant ce semestre, vous allez utiliser le serveur XAMPP et l'interface phpMyAdmin.

# phpMyAdmin

- phpMyAdmin  est un outil de gestion de base de données open-source, conçu pour faciliter l'administration et la manipulation des bases de données MySQL via une interface web conviviale.

# XAMPP

- XAMPP est un package de solution de serveur web open-source.
- Il est principalement utilisé pour tester des applications web sur un serveur web local (localhost).
- XAMPP est un serveur web local qui comprend Apache (le serveur web), PHP (le langage de script côté serveur) et MySQL (le système de gestion de base de données), offrant ainsi une solution prête à l'emploi pour le développement et le déploiement de sites web basés sur PHP et MySQL.

XAMPP

XAMPP signifie :

X = Cross-platform (Multiplateforme)

A = Serveur Apache

M = MariaDB

P = PHP

P = Perl

Sarah Malaeb

# Langage PHP / Plan :

- Syntaxe de base PHP
- Fonctions en PHP
- Classes en PHP
- Héritage
- Objets
- Tableaux associatifs

# Intégration d’un script dans une page

Les pages web sont au format html. Les pages web dynamiques générées avec PHP4 sont au format php. Le code source php est directement insérer dans le fichier html grâce au conteneur de la norme XML :

**<?php … ?>**

*Exemple:*

```
<html>
<body>
<?php
        echo ’’Bonjour’’;
?>
</body>
</html>
```

*Autres syntaxes d’intégration :*

- **<? … ?>**
- **<script language=‘’php’’> … </script>**
- **<% … %>**

# Exemple de script

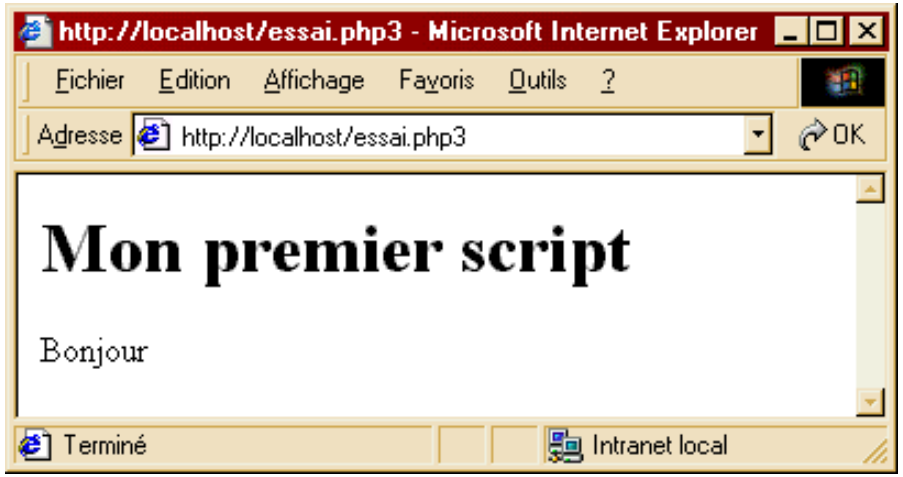
*Exemple de script, code source (côté serveur) :*

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

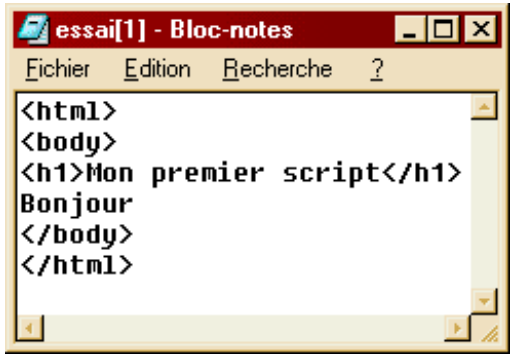
*Autre écriture du même script :*

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

*Résultat affiché par le navigateur :*




*Code source (côté client) de la page essai.ph3 résultant du script*



# Commentaires

Un script php se commente :
*Exemple :*

```
<?php

// commentaire de fin de ligne

/* commentaire
sur plusieurs
lignes */

# commentaire de fin de ligne comme en Shell
?>
```

Tout ce qui se trouve dans un commentaire est ignoré. Il est conseillé de commenter largement ses scripts.

# Variables, types et opérateurs

Le typage des variables est implicite en php. Il n'est donc pas nécessaire de déclarer leur type au préalable ni même de les initialiser avant leur utilisation.

Les identificateurs de variable sont précédés du symbole « **$** » (dollars). Exemple : **$toto**.

Les variables peuvent être de type entier (**integer**), réel (**double**), chaîne de caractères (**string**), tableau (**array**), objet (**object**), booléen (**boolean**).

Il est possible de convertir une variable en un type primitif grâce au ***cast*** (comme en C).

*Exemple :*

```
$str = ''12'';          // $str vaut la chaîne ''12''
$nbr = (int)$str;       // $nbr vaut le nombre 12
```

Le ***cast*** est une conversion de type. L'action de caster consiste en convertir une variable d'un type à un autre.

# Variables, types et opérateurs

Quelques fonctions :

- **empty($var)** : renvoie vrai si la variable est vide
- **isset($var)** : renvoie vrai si la variable existe
- **unset($var)** : détruit une variable
- **gettype($var)** : retourne le type de la variable
- **settype($var, ''type'')** : convertit la variable en type **type** (cast)
- **is_long()**, **is_double()**, **is_string()**, **is_array()**, **is_object()**, **is_bool()**, **is_float()**, **is_numeric()**, **is_integer()**, **is_int()…**

Une variable peut avoir pour identificateur la valeur d'une autre variable.

*Syntaxe :* **${$var} = valeur;**

*Exemple :*

```
$toto = ''foobar'';
${$toto} = 2002;
echo $foobar;          // la variable $foobar vaut 2002
```

# Variables, types et opérateurs

La portée d'une variable est limitée au bloc dans lequel elle a été crée. Une variable locale à une fonction n'est pas connue dans le reste du programme. Tout comme une variable du programme n'est pas connue dans une fonction. Une variable créée dans un bloc n'est pas connue dans les autres blocs, mêmes supérieurs.

**Opérateurs arithmétiques :**

**+** (addition), **-** (soustraction), ***** (multiplié), **/** (divisé), **%** (modulo), **++** (incrément), **--**(décrément). Ces deux derniers peuvent être pré ou post fixés

**Opérateurs d'assignement :**

**=** (affectation), ***=** (**$x*=$y** équivalent à **$x=$x*$y**), **/=**, **+=**, **-=**, **%=**

**Opérateurs logiques :**

**and**, **&&** (et), **or**, **||** (ou), **xor** (ou exclusif), **!** (non)

**Opérateurs de comparaison :**

**==** (égalité), **<** (inférieur strict), **<=** (inférieur large), **>**, **>=**, **!=** (différence)

# Variables, types et opérateurs

**L'opérateur de concaténation** qui est le point **(.)** va nous permettre de mettre bout à bout deux chaines de caractères.

```php
<?php
$a = "Bonjour ";
$b = $a . "Monde !"; // $b contient "Bonjour Monde !"

$a = "Bonjour ";
$a .= "Monde !"; // $a contient "Bonjour Monde !"
?>
```

# Variables, types et opérateurs (IV)

Signalons un opérateur très spécial qui équivaut à une structure conditionnelle complexe *if then else* à la différence qu'il renvoie un résultat de valeur pouvant ne pas être un booléen : l'opérateur ternaire.

*Syntaxe :*

**(condition)?(expression1):(expression2);**

Si la **condition** est vrai alors évalue et renvoie l'**expression1** sinon évalue et renvoie l'**expression2**.

*Exemple :*

**$nbr = ($toto>10)?($toto):($toto%10);**

Dans cet exemple, la variable **$nbr** prend **$toto** pour valeur si **$toto** est strictement supérieur à **10**, sinon vaut le reste de la division entière de **$toto** par **10**.

# Constantes

L'utilisateur peut définir des constantes dont la valeur est fixée une fois pour toute. Les constantes ne portent pas le symbole **$** (dollars) en début d'identificateur et ne sont pas modifiables.

**define(''var'',valeur)** : définit la constante **var** (sans **$**) de valeur **valeur**

*Exemple 1 :*

```
define(''author'',''Foobar'');
echo author;                    // affiche 'Foobar'
```

*Exemple 2 :*

```
define(MY_YEAR,1980);
echo MY_YEAR;                   // affiche 1980
```

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

# Références

On peut à la manière des pointeurs en C faire référence à une variable grâce à l'opérateur **&** (ET commercial).

*Exemple 1 :*

```
$toto = 100;          // la variable $toto est initialisée à la valeur 100
$foobar = &$toto; // la variable $foobar fait référence à $toto
$toto++;              // on change la valeur de $toto
echo $foobar;         // qui est répercutée sur $foobar qui vaut alors 101
```

*Exemple 2 :*

```
function change($var) {
        $var++;    // la fonction incrémente en local l'argument
}
$nbr = 1;  // la variable $nbr est initialisée à 1
change(&$nbr);     // passage de la variable par référence
echo $nbr;         // sa valeur a donc été modifiée
```

# Fonctions mathématiques

Il existe une myriade de fonctions mathématiques.

- **abs($x)** : valeur absolue
- **ceil($x)** : arrondi supérieur
- **floor($x)** : arrondi inférieur
- **pow($x,$y)** : x exposant y
- **round($x,$i)** : arrondi de x à la ième décimale
- **max($a, $b, $c …)** : retourne l'argument de valeur maximum
- **pi()** : retourne la valeur de Pi

Et aussi : **cos**, **sin**, **tan**, **exp**, **log**, **min**, **pi**, **sqrt**…

*Quelques constantes :*
M_PI : valeur de pi (3.14159265358979323846)
M_E : valeur de e (2.7182818284590452354)

# Fonctions mathématiques

**Nombres aléatoires :**

- **rand([$x[,$y])** : valeur entière aléatoire entre 0 et RAND_MAX si x et y ne sont pas définis, entre x et RAND_MAX si seul x est défini, entre x et y si ces deux paramètres sont définis.
- **srand()** : initialisation du générateur aléatoire
- **getrandmax()** : retourne la valeur du plus grand entier pouvant être généré

- L'algorithme utilisé par la fonction **rand()** - issu de vieilles bibliothèques libcs - est particulièrement lent et possède un comportement pouvant se révéler prévisible. La fonction **mt_rand()** équivalente à **rand()** est plus rapide et plus sûre puisque l'algorithme utilisé se base sur la cryptographie.
- En cas d'utilisation de la fonction **mt_rand()**, ne pas oublier d'utiliser les fonctions de la même famille : **mt_rand([$x[,$y]])**, **mt_srand()** et **mt_getrandmax()**.

# Rappel…Une fonction

- Morceau de code réutilisable.
- Possède sa propre « portée locale » (*local scope*).

```
function my_func($arg1,$arg2) {
    << function statements >>
}
```

# Rappel…Une fonction

## Conceptuellement, que représente une fonction ?

…donnez quelque chose à la fonction (arguments), elle
en fait quelque chose, puis renvoie un résultat…

Action (*procédure*)
ou
Méthode

# Qu'est-ce qu'une classe ?

Conceptuellement, une classe représente un **objet**, avec des méthodes et des variables associées

# Définition d’une classe

```php
<?php
class dog {
  public $name;
  public function bark() {
     echo 'Woof!';
  }
}
?>
```

Un exemple de définition de classe pour un chien. L'objet chien a un seul attribut, le nom, et peut effectuer l'action d'aboyer.

# Définition d'une classe

```php
<?php
class dog {
  public $name;
  public function bark() {
    echo 'Woof!';
  }
}
?>
```

Définissez le nom de la classe.

# Définition d’une classe

```php
<?php
class dog {
  public $name;
  public function bark() {
    echo 'Woof!';
  }
}
?>
```

Définissez un attribut d'objet (variable), le nom du chien.

# Définition d’une classe

```php
<?php
class dog {
  public $name;
  public function bark() {
      echo 'Woof!';
  }
}
?>
```

Définir une action (fonction) d'objet, l'aboiement du chien

# Définition d’une classe

```
<?php
class dog {
  public $name;
  public function bark() {
     echo 'Woof!';
  }
}
?>
```

Terminer la définition de classe

# Définition d’une classe

Semblable à la définition d'une fonction.

*La définition **ne fait rien par elle-même**.*

*C'est un plan, ou une description, d'un objet.*

*Pour faire quelque chose, vous devez utiliser la classe…*

# Utilisation d’une classe

```php
<?php
require('dog.class.php');
$puppy = new dog();
$puppy->name = 'Rover';
echo "{$puppy->name} says ";
$puppy->bark();
?>
```

# Utilisation d'une classe

```php
<?php
require('dog.class.php');
$puppy = new dog();
$puppy->name = 'Rover';
echo "{$puppy->name} says ";
$puppy->bark();
?>
```

Inclure la définition de classe

# Utilisation d'une classe

```php
<?php
require('dog.class.php');
$puppy = new dog();
$puppy->name = 'Rover';
echo "{$puppy->name} says ";
$puppy->bark();
?>
```

Créez une nouvelle **instance** de la classe.

# Utilisation d’une classe

```php
<?php
require('dog.class.php');
$puppy = new dog();
$puppy->name = 'Rover';
echo "{$puppy->name} says ";
$puppy->bark();
?>
```

Définissez la variable *name* de **cette instance** sur "Rover".

# Utilisation d'une classe

```
<?php
require('dog.class.ph
$puppy = new dog();
$puppy->name = 'Rover';
echo "{$puppy->name} says ";
$puppy->bark();
?>
```

Utilisez la variable *name* de cette instance dans une instruction echo.

# Utilisation d’une classe

```php
<?php
require('dog.class.php');
$puppy = new dog();
$puppy->name = 'Rover';
echo "{$puppy->name} says ";
$puppy->bark();
?>
```

Utilisez la méthode d'aboiement de l'objet chien.

# Utilisation d’une classe

```php
<?php
require('dog.class.php');
$puppy = new dog();
$puppy->name = 'Rover';
echo "{$puppy->name} says ";
$puppy->bark();
?>
```

# Un dollar and un seul…

```
$puppy->name = 'Rover';
```

erreur la plus courante consiste à utiliser plus d'un signe dollar lors de l'accès aux variables. Ce qui suit signifie quelque chose de complètement différent…

~~`$puppy->$name = 'Rover';`~~

# Utilisation des attributs dans la classe..

- Si vous devez utiliser les attributs de classe dans des actions de classe, utilisez la variable spéciale **$this** dans la définition :

```
class dog {
   public $name;
   public function bark() {
    echo $this->name.' says Woof!';
   }
  }
```

# Méthodes constructeur

- Une méthode qu'on appelle **constructeur** est une fonction qui est automatiquement exécutée lors de la première instanciation de la classe.

- Créez un constructeur en incluant une fonction dans la définition de classe avec le **__construct *name***.

- N'oubliez pas... si le constructeur requiert des arguments, ils doivent être passés lors de son instanciation !

# Exemple de constructeur

```php
<?php
class dog {
  public $name;
  public function __construct($nametext) {
     $this->name = $nametext;
  }
   public function bark() {
      echo 'Woof!';
   }
}
?>
```

constructeur

# Exemple de constructeur

```
<?php

    …

$puppy = new dog('Rover');

    …

?>
```

Les arguments du constructeur sont passés lors de l'instanciation de l'objet.

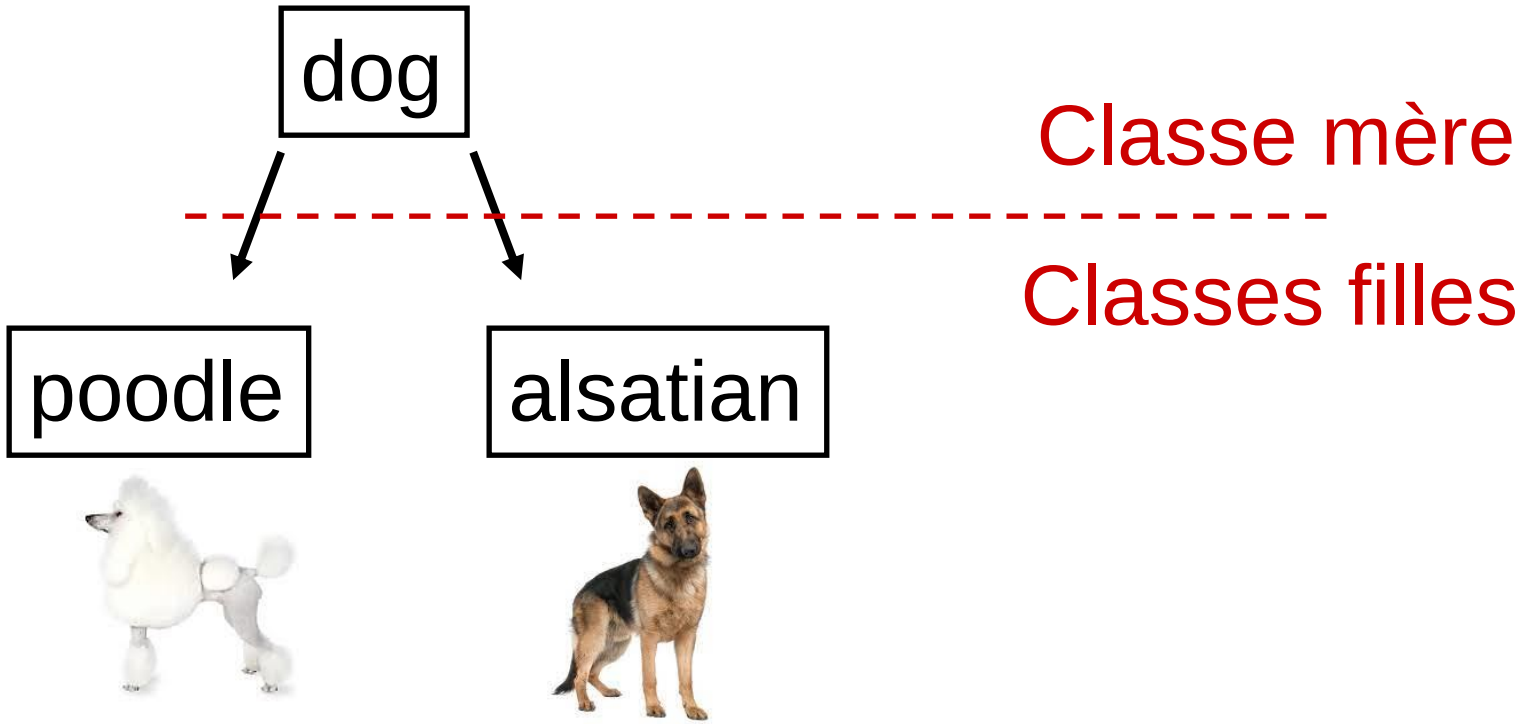
# La portée de classe

- Comme les fonctions, **chaque objet instancié** a sa propre portée locale.

- par exemple : si 2 objets chiens différents sont instanciés, **$puppy1** et **$puppy2**, les deux noms de chien **$puppy1->name** et **$puppy2->name** sont entièrement indépendants.

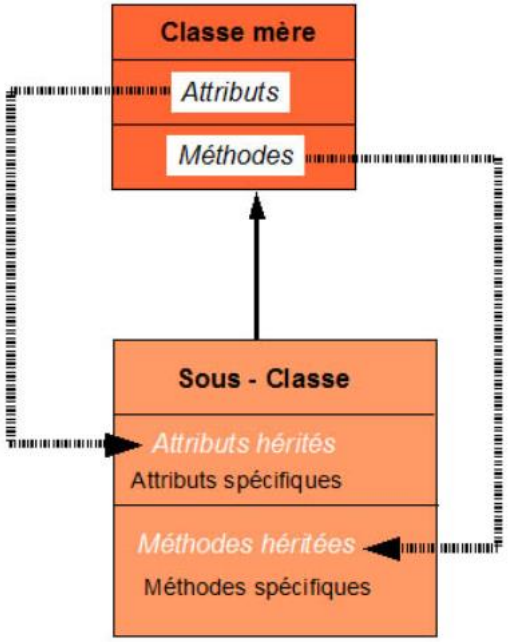
# Héritage

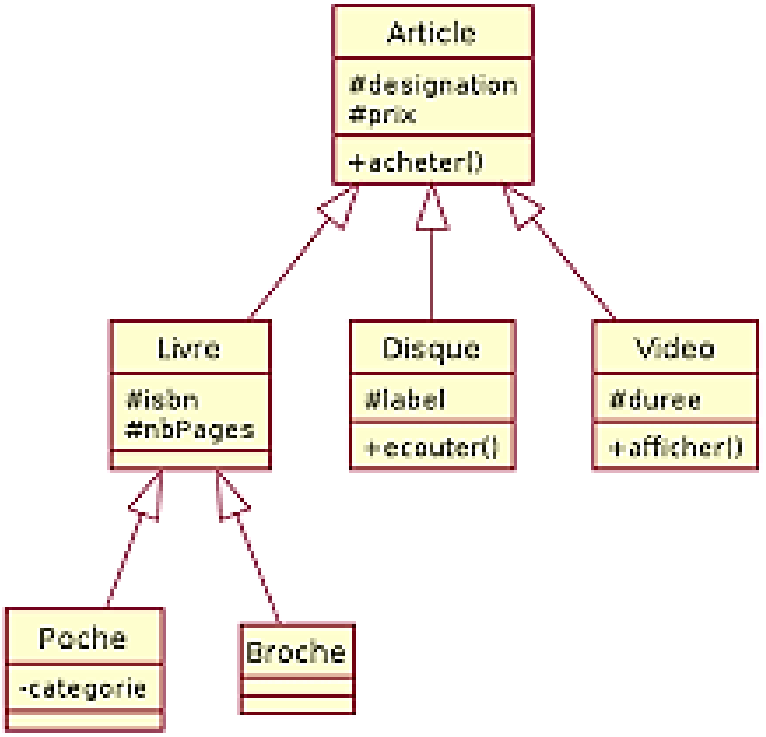
- Le véritable pouvoir de l'utilisation des classes est la propriété de l'héritage - créant une hiérarchie de classes interconnectées.



# Héritage

- Les classes filles "**héritent**" toutes les méthodes et attributs de la classe mère et peuvent en ajouter d'autres.
- par exemple : la classe fille caniche hérite la variable ‘name’ et la méthode ‘bark’ de la classe dog, et peut en ajouter d’autres…



# Example d’héritage

```
class poodle extends dog {
        public $type;
        public function set_type($height) {
                if ($height<10) {
                        $this->type = ‘Toy’;
                } elseif ($height>15) {
                        $this->type = ‘Standard’;
                } else {
                        $this->type = ‘Miniature’;
                }
        }
}
```

*On reconnaît trois tailles de caniche: Standard, Miniature, et Toy…*

# Example d’héritage

*Le American Kennel Club (AKC) reconnaît trois tailles de caniche - Standard, Miniature, and Toy…*

```
class poodle extends dog {
        public $type
        public function se
                if ($height<
                        $this-
                } elseif ($h
                        $this-
                } else {
                        $this-
                }
        }
}
```

Notez l'utilisation du mot clé **extend** pour indiquer que la classe caniche est un enfant de la classe chien…

# Example de héritage

```
…
$puppy = new poodle('Oscar');
$puppy->set_type(12); // 12 inches high!
echo "Poodle is called {$puppy->name}, ";
echo "of type {$puppy->type}, saying ";
echo $puppy->bark();
…
```

# Example de héritage

- Il est possible de remplacer une méthode mère par une nouvelle méthode si elle reçoit le même nom dans la classe fille.

```
class poodle extends dog {
        …
        public function bark() {
                echo 'Yip!';
        }
        …
}
```

**…un caniche (*poodle*) toujours jappe ('Yip!')**

# Constructeurs enfants?

- Si la classe fille possède une fonction ***constructeur***, elle est exécutée et **tout constructeur parent est ignoré**.
- Si la classe fille n'a pas de constructeur, le constructeur de la classe mère est exécuté.
- Si l'enfant et le parent n'ont pas de constructeur, le constructeur grand-parent est tenté...
- etc.

# Objets dans des objets

- Il est parfaitement possible d'inclure des objets dans un autre objet.

```
class dogtag {
    public $words;
}

class dog {
    public $name;
    public $tag;

    public function bark() {
        echo "Woof!\n";
    }
}
```

```
…
$puppy = new dog;
$puppy->name = “Rover";
$poppy->tag = new dogtag;
$poppy->tag->words = “blah”;
…
```

# Suppression d'objets

- Jusqu'à présent, nos objets n'ont pas été détruits jusqu'à la fin de nos scripts.

- Comme les variables, il est possible de détruire explicitement un objet en utilisant la fonction `unset()`.

# Tableau associatif

- Un tableau associatif est un tableau que ces indexes sont créés par l'utilisateur et qu'il va les associer à chaque valeur.
- Un tableau associatif **est un ensemble de couples (clé, valeur)**.
- Une **clé** est soit un entier soit une chaîne de caractères.
- Une **valeur** est une donnée quelconque (booléen, numérique, chaîne de caractères..). La valeur d'un élément d'une collection dépend donc du couple (nom de la collection, clé).

# Tableau associatif

## Création d'un tableau associative

- Première méthode:

Syntaxe:

$nom_tableau= array(cle1 => valeur1,cle2 => valeur2,cle3 => valeur3…..)

```php
<?php $semaine=array('prmier_jour'=>'lundi', 'deuxième_jour'=>'mardi',
'troisième_jour'=>'mercredi', 'quatrième_jour'=>'jeudi', 'cinquième_jour'=>'vendredi',
'sixième_jour'=>'samedi', 'septième_jour'=>'dimanche'); ?>
```

- Ce tableau est une variable en lui-même puisqu'il commence par $
- Les éléments du tableau peuvent être de même type ou de type différent

# Tableau associatif

## Création d'un tableau associative

- Deuxième méthode:

Syntaxe:

$nom_tableau= array ();

$nom_tableau ['clé1']= valeur1;

$nom_tableau ['clé2']= valeur2;

$nom_tableau ['clé3']= valeur3;

```php
<?php
$semaine=array();
$semaine['prmier_jour']='lundi';
$semaine['deuxième_jour']='mardi';
$semaine['troisième_jour']='mercredi';
$semaine['quatrième_jour']='jeudi';
$semaine['cinquième_jour']='vendredi';
$semaine['sixième_jour']='samedi';
$semaine['septième_jour']='dimanche';
?>
```

- Pour cette écriture les indexes sont implicites et c'est le compilateur qui en déduit les valeurs qui commencent par défaut de 0 et s'incrémentent de 1.
- $semaine=array(); est facultative. Mais le fait de la déclarer est vu comme une bonne habitude.
- Ici on a crée un tableau vide puis on l'a rempli par la suite.

# Tableau associatif

## Création d'un tableau associative

- Troisième méthode:

Syntaxe:

$nom_tableau= [cle1 => valeur1,cle2 => valeur2,cle3 => valeur3…..]

```php
<?php $semaine= ['prmier_jour'=>'lundi’, 'deuxième_jour'=>'mardi’,
'troisième_jour'=>'mercredi’, 'quatrième_jour'=>'jeudi’,
'cinquième_jour'=>'vendredi’, 'sixième_jour'=>'samedi’, 'septième_jour'=>'dimanche'];
?>
```

- C'est l'écriture la plus simplifiée

# Tableau associatif

## Lecture d’un tableau associative :

- **Lecture d’une valeur:**

Pour lire une valeur d’un tableau indexé, on appelle la variable avec, entre les crochets [ ], le numéro de la clé correspondant à la valeur.

```php
<?php $semaine= ['prmier_jour'=>'lundi’,  'deuxième_jour'=>'mardi’,
'troisième_jour'=>'mercredi’, 'quatrième_jour'=>'jeudi’, 'cinquième_jour'=>'vendredi’,
'sixième_jour'=>'samedi’, 'septième_jour'=>'dimanche’];
echo $semaine=['prmier_jour']; //Affiche lundi ?>
```

# Tableau associatif

## Lecture d’un tableau associative :

- **Lecture de plusieurs valeurs:**

Pour lire tous les élément d’un tableau, la solution c’est à utiliser la boucle **for** ou la boucle **while**. Cependant, PHP inclue une structure de contrôle qui s’applique spécialement aux tableaux. Il s’agit de la structure **foreach**.

```php
<?php
$semaine=array('prmier_jour'=>'lundi','deuxième_jour'=>'mardi',
'troisième_jour'=>'mercredi','quatrième_jour'=>'jeudi',
'cinquième_jour'=>'vendredi','sixième_jour'=>'samedi',
'septième_jour'=>'dimanche');

foreach ($semaine as $jour)
{ //affiche jour
echo($jour."</br>");
}
echo "</br></br>";
foreach ($semaine as $jour =>$value)
{ //affiche prmier_jour Lundi
echo ("<b>".$jour."</b> ".$value."</br>");
}
?>
```

# Tableau associatif

## Modifier une valeur dans un tableau associatif

Pour modifier une valeur dans un tableau associatif, il suffit d'affecter une nouvelle valeur au tableau, avec, entre les crochets [ ], la clé sous forme de chaîne correspondant à l'ancienne valeur.

```
$semaine = 'prmier_jour'=>'lundi', 'deuxième_jour'=>'mardi', 'troisième_jour'=>'mercredi',
'quatrième_jour'=>'jeudi', 'cinquième_jour'=>'vendredi', 'sixième_jour'=>'samedi',
'septième_jour'=>'dimanche'];
$semaine['prmier_jour'] =' Lundi';
```

# Cours CM2 // séance prochaine

- PHP, MySQL et JSON

Sarah Malaeb

# Activités

- TD – Exercices PHP
- Activité professionnelle: Site PHP / MySQL