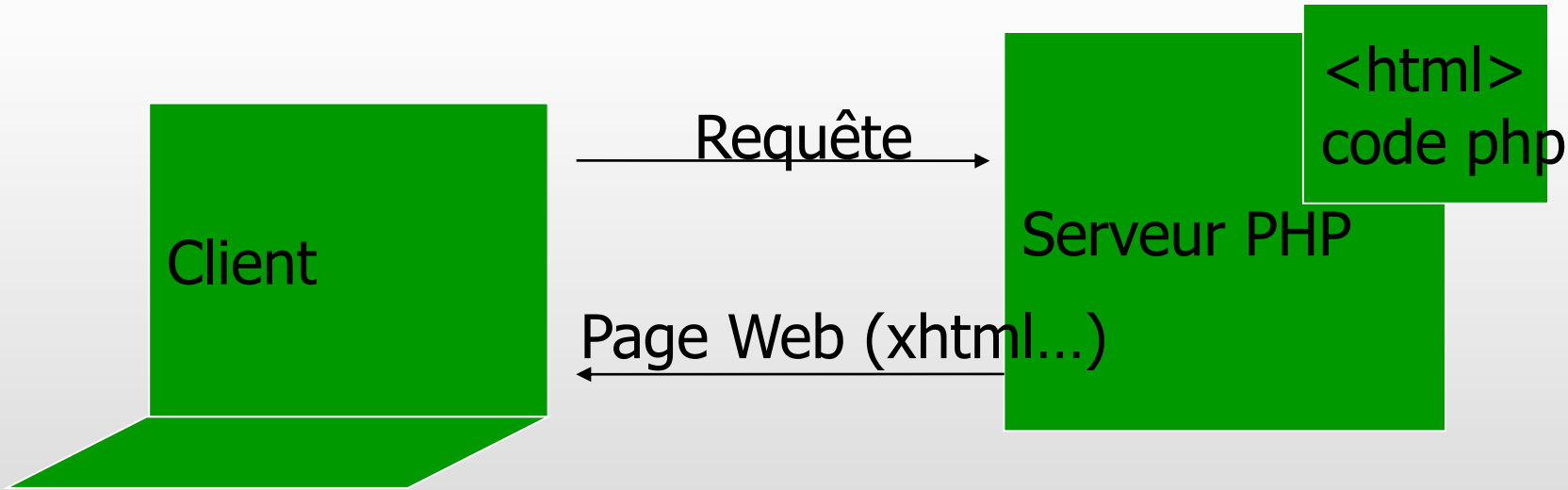
PHP

Remerciements

- Le matériel des diapositives est pris de différentes sources comprenant :

PHP

- PHP: Hypertext Preprocessor
- Code exécuté coté serveur
- Langage interprété [compilable]

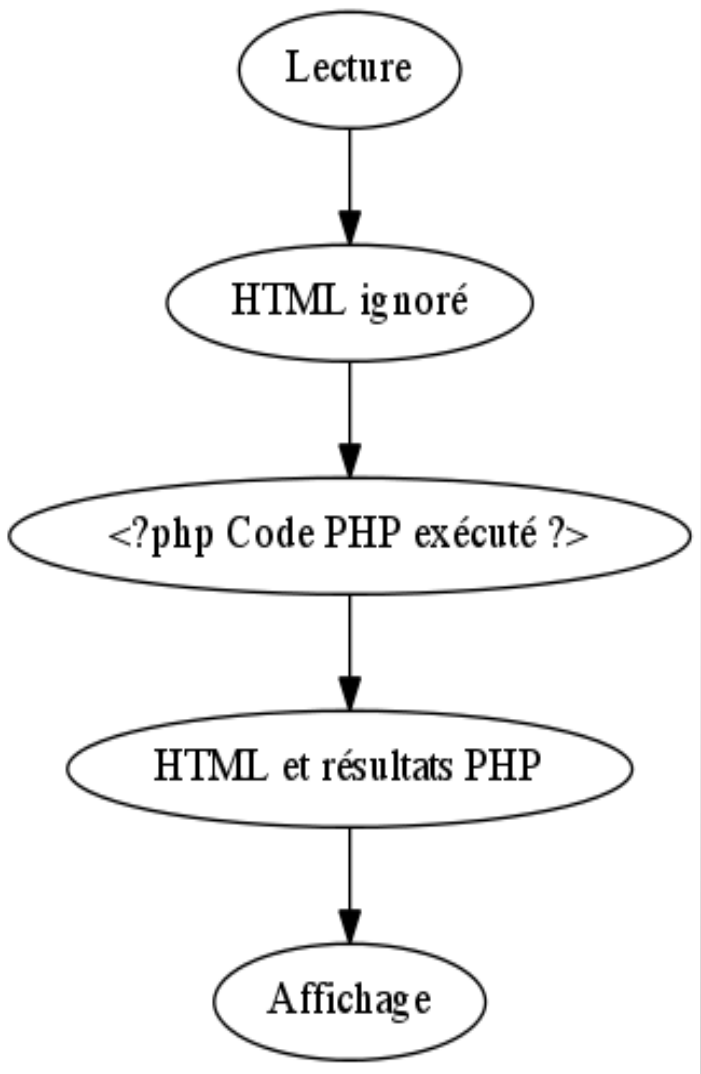


PHP

- Portabilité et distribution C/S, 3/3, P2P
- Accès aux bases de données facile
- Génération graphique sophistiquée
- Interopérabilité avec XML, parsing xml avec les standards SAX et DOM, Web services…
- Support des protocoles réseau (LDAP, IMAP, SNMP, POP3…)
- Traitement des chaînes de caractères (regex, perl)
- Nombreuses bibliothèques (PEAR, PECL, ...)
- Syntaxe proche du C
- Interpréteur souple
- Types de données classiques
- Typage à première affectation
- Conversion automatique de type
- Tableaux dynamiques à sélecteur
- Modèle objet intégré

Fonctionnement

- **L'interpréteur**
  - lit un fichier source .php puis génère un flux de sortie avec les règles suivantes :
    - toute ligne située à l'extérieur d'un bloc PHP (entre <?php et ?>) est recopiée inchangée dans le flux de sortie
    - le code PHP est interprété et génère éventuellement des résultats intégrés eux aussi au flux de sortie
    - les erreurs éventuelles donnent lieu à des messages d'erreurs qu'on retrouve également dans le flux de sortie (selon la configuration du serveur)
    - une page html pure sauvegardée avec l'extension .php sera donc non modifiée et renvoyée telle quelle . .



Types

- Quatre types de données simples :
  - Integer
    - Entier compris entre -2 147 483 647 et 2 147 483 647.
  - double
    - nombre à virgule flottante compris entre -1.78*10308 et +1.78*10308.
  - String
    - chaîne de caractères de longueur variable 'string' ou "$string".
  - Boolean
    - valeur booléenne true ou false.
- Les variables ne possèdent pas de types de données prédéfinis et changent de type selon leur contenu
  - $a = 1; $a = 1.3 ; $a = ‘ABC’ ;

Fonctions de test

- is_int()
- is_long()
- is_double()
- is_array()
- is_object()
- is_string()

Attention : N'oubliez pas comme en JavaScript la différence entre l'opérateur == et ===

Le premier vérifie l'égalité des contenus en ne tenant pas compte d'une éventuelle différence de typage (int ou string par exemple) tandis que le second vérifie une égalité stricte.

En d'autres termes : 5 == « 5 » est VRAI tandis que 5 === « 5 » est FAUX

PHP

- Entre guillemets simples ou doubles
  - 'Ma chaine $var' : pas interprétée
  - "Ma chaine $var " : interprétée
- Caractères de contrôle d'espace
- Caractères d'échappement \ permettant l'exécution du contrôle
  - \t tabulation
  - \n nouvelle ligne
  - \r retour à la ligne
  - \v tabulation verticale
- Possibilité d'imposer le suivi d'un format

PHP

- Tableaux classiques
  - // initialisation

```
$table=array(0,1,2,3,4,5);
```

  - // parcours simple

```
print 'table a ' . count($table) . " éléments\n";
for($i=0;$i<count($table);$i++)
    print "table[$i]=$table[$i]\n";
```

PHP

- Les tableaux associatifs
  - Aussi appelés dictionnaires
  - Tableaux associant une clé à une valeur
  - La clé peut être une chaîne de caractère ou un nombre
  - // Insertion d'une ligne
    $dico["cle"] = valeur;
    exemple : $dico['Pierre'] = 20 ;
  - // déclaration d'un tableau
    $dico=array('Pierre'=>20, 'Paul'=>22, 'Marie'=>18);
  - // Accès à un élément
    $variable = $dico["cle"]; exemple: $age = $dico['Pierre'] ;

Tableaux

- Principe
  - Création à l'aide de la fonction array()
  - Uniquement des tableaux à une dimension
    - Les éléments d'un tableau peuvent pointer vers d'autres tableaux
  - Les éléments d'un tableau peuvent appartenir à des types distincts
  - L'index d'un tableau en PHP commence de 0
  - Pas de limites supérieures pour les tableaux
  - La fonction count() pour avoir le nombre d'éléments d'un tableau

Tableaux

- Les tableaux indicés et les tableaux associatifs
  - Tableau indicé
    - Accéder aux éléments par l'intermédiaire de numéros

```
$tableau[indice] = valeur;
$jour[3] = "Mercredi";
$note[0] = 20;
$tableau = array(valeur0, valeur1,..., valeurN);
$jour = array("Dimanche", "Lundi", "Mardi",
  "Mercredi", "Jeudi", "Vendredi", "Samedi");
$note = array(20, 15, 12.6, 17, 10, 20, 11, 18,
  19);
$variable = $tableau[indice];
$JJ = $jour[6]; // affecte "Samedi" à $JJ
echo $note[1] + $note[5];
```

Tableaux

- Les tableaux indicés et les tableaux associatifs
  - Tableau associatif (ou table de hachage)
    - Les éléments sont référencés par des chaînes de caractères associatives en guise de nom: la clé d'index

```
$tableau["indice"] = valeur;
$jour["Dimanche"] = 7
$jour["Mercredi"] = "Le jour des enfants"

$tableau = array(ind0 => val0, ind1 => val1,..., indN =>
  valN);
$jour = array("Dimanche" => 1, "Lundi" => 2, "Mardi" =>
  3, "Mercredi" => 4, "Jeudi" => 5, "Vendredi" => 6,
  "Samedi" => 7);

$variable = $tableau["indice"];
$JJ = $jour["Vendredi"]; //affecte 6 à $JJ
echo $jour["Lundi"]; //retourne la valeur 2
```

Tableaux

- Tableaux multidimensionnels
  - Pas d'outils pour créer directement des tableaux multidimensionnels
  - L'imbrication des tableaux est possible

```
$tab1 = array(Val0, Val1,..., ValN);
$tab2 = array(Val0, Val1,..., ValN);
// Création d'un tableau à deux dimensions
 $tableau = array($tab1, $tab2);
$mois  =  array("Janvier",  "Février",  "Mars",  "Avril",  "Mai",
   "Juin", "Juillet", "Août", "Septembre", "Octobre", "Novembre",
   "Décembre");
$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi",
   "Vendredi", "Samedi");
&element_date = array(&mois, &jour);

$variable = $tableau[indice][indice];
$MM = $element_date[0][0]; //affecte "Janvier" à $MM
echo $element_date[1][5] . " 7 " . $element_date[0][2] . "2002";
   // retourne "Jeudi 7 Mars 2002"
```

Lecture des éléments d'un tableau

- Avec une boucle for

```
for ($i=0; $i<count($tab) ; $i++){
if ($tab[$i]== "a") {echo $tab[$i], "<br  />"; }}
```

- Avec une boucle while

```
$i=0;
while ($tab[$i]){
if ($tab[$i][0] =="a" ) {echo $tab[$i], "<br /> "; }}
```

- Avec La boucle foreach

```
$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi",
   "Vendredi", "Samedi");
$i = 0;
foreach($jour as $JJ) { echo "La cellule n° ". $i . " : " . $JJ .
   "<br>"; $i++; }
```

Tableaux

Lecture des éléments d'un tableau

- Parcours d'un tableau associatif
  - Réalisable en ajoutant avant la variable $valeur, la clé associée

```
$tableau = array(clé1 => val1, clé2 => val2, ..., cléN =>
  valN);
foreach($tableau as $clé => $valeur) {
echo "Valeur ($clé): $valeur"; }

$jour = array("Dimanche" => 7, "Lundi" => 1, "Mardi" => 2,
  "Mercredi" => 3, "Jeudi" => 4, "Vendredi" => 5, "Samedi" =>
  6);
foreach($jour as $sJJ => $nJJ) {
echo "Le jour de la semaine n° ". $nJJ . " : " . $sJJ .
  "<br>"; }
```

Tableaux

- Fonctions de tri
  - Tri selon les valeurs
    - La fonction sort() effectue un tri sur les valeurs des éléments d'un tableau selon un critère alphanumérique :selon les codes ASCII :
      - « a » est après « Z » et « 10 » est avant « 9 »)
      - Le tableau initial est modifié et non récupérables dans son ordre original
      - Pour les tableaux associatifs les clés seront perdues et remplacées par un indice créé après le tri et commencant à 0
    - La fonction rsort() effectue la même action mais en ordre inverse des codes ASCII.
    - La fonction asort() trie également les valeurs selon le critère des codes ASCII, mais en préservant les clés pour les tableaux associatifs
    - La fonction arsort() la même action mais en ordre inverse des codes ASCII
    - la fonction natcasesort() effectue un tri dans l'ordre alphabétique non ASCII (« a » est avant « z » et « 10 » est après « 9 »)

Tableaux

- Fonctions de tri
  - Tri sur les clés
  - La fonction ksort() trie les clés du tableau selon le critère des codes ASCII, et préserve les associations clé /valeur
    - La fonction krsort() effectue la même action mais en ordre inverse des codes ASCII

```
<?php
$tab2                                =                                array
   ("1622"=>"Molière","1802"=>"Hugo","1920"=>"Vian") ;
ksort ($tab2);
echo "<h3 >  Tri sur les clés de \$tab2 </h3>" ;
foreach ($tab2 as $cle=>$valeur) {
echo "<b> l'élément a pour clé : $clé; et pour valeur : $ valeur
   </b> <br />";
}
?>
```

Tableaux

- Les fonctions de tableaux

$tableau = array_count_values($variable);

retourne un tableau comptant le nombre d'occurrences des valeurs d'un tableau.

$tableau = array_diff($var_1, $var_2, ..., $var_N);

retourne dans un tableau contenant les valeurs différentes entre deux ou plusieurs tableaux.

$tableau = array_intersect($var_1, $var_2, ..., $var_N);

retourne un tableau contenant les enregistrements communs aux tableaux entrés en argument.

$tableau = array_flip($variable);

intervertit les paires clé/valeur dans un tableau.

$tableau = array_keys($variable [, valeur]);

retourne toutes les clés d'un tableau ou les emplacements d'une valeur dans un tableau.

Tableaux

$tableau = array_filter($variable, "fonction")

retourne un tableau contenant les enregistrements filtrés d'un tableau à partir d'une fonction.

```
<?php
function impair($var)
{return ($var % 2 == 1);}
function pair($var)
{return ($var % 2 == 0);}
$array1 = array ("a"=>1, "b"=>2, "c"=>3, "d"=>4,
  "e"=>5);
$array2 = array (6, 7, 8, 9, 10, 11, 12);
echo "Impairs :\n";
print_r(array_filter($array1, "impair"));
echo "Pairs :\n";
print_r(array_filter($array2, "pair"));
?>
```

Tableaux

- Les fonctions de tableaux

$tableau = array_map($var_1 [, $var_2, ..., $var_N], 'fonction');

applique une fonction à un ou plusieurs tableaux.

$tableau = array_merge($var_1, $var_2, ..., $var_N);

enchaîne des tableaux entrés en argument afin d'en retourner un unique.

$tableau = array_merge_recursive($var_1, $var_2, ..., $var_N);

enchaîne des tableaux en conservant l'ordre des éléments dans le tableau résultant. Dans le cas de clés communes, les valeurs sont placées dans un tableau.

true | false = array_multisort($var, critère1, critère2 [, ..., $var_N, critère1, critère2])

trie un ou plusieurs tableaux selon un ordre croissant ou décroissant (SORT_ASC ou SORT_DESC) et selon une comparaison alphabétique, numérique ou de chaîne de caractères (SORT_REGULAR, SORT_NUMERIC ou SORT_STRING).

$tableau = array_pad($variable, taille, valeur);

recopie tout un tableau en ajustant sa taille à l'argument correspondant et en bourrant d'une valeur spécifiée les éléments vides.

PHP/Logique*Arithmétique

+	Addition	5 + 2
-	Soustraction	5 - 2
*	Multiplication	5 * 2
/	Division	5 / 2
%	Modulo	5%2

==	Egalité	$v == 5
<, <=	Infériorité	$v < 5
>, >=	Supériorité	$v >= 7
!=	Différent	$v != 7
%	Modulo	5%2

PHP/Opérateurs logiques et sur bits

&& and	et	$c1 && $c2 $c1 and $c2
|| or	ou	$c1 || $c2 $c1 or $c2
xor	ou exclusif	$c1 xor $c2
!	négation	!$c1
? :	Affectation conditionnelle (opérateur ternaire)	$c1 == 0 ? $c2 : $c3

&	et	$c1 & $c2
|	ou	$c1 | $c2
^	ou exclusif	$c1 ^ $c2
~	négation	~$c1
>> <<	Décalage à droite Décalage à gauche	$c1 <<2 $c2>>3

PHP Structures conditionnelles

- **if** (cond) { instructions 1; }
  elseif (cond) {instructions 2; }
  …
  else(cond) {instructions N; }
- **switch** (expression) {
  case val1: instruction; break;
  case val2: instruction; break;
  …
  default: instruction;
  }
- **for**(init, cond, evolution) {
  instructions;
  }
- **while** (condition) {
  instructions;
  [break;]
  }
- **do** {instructions}
  while (condition) ;

Inclusion de code et sortie

- Require ("fichier")
  - Recopie un fichier de code à sa place
  - Exemple : require ("opendb.php")
- Include ("fichier")
  - Idem require mais évalue et exécute le code à chaque rencontre de include – possibilité include_once()
- Exit
  - Permet de quitter une page PHP
- Die ('message')
  - Die écrit le message et termine le programme

Les fonctions

- Bloc d'instructions avec paramètres optionnels créé par l'intermédiaire de l'instruction **function**

```
function nom_fonction([$param_1 [= val_déf], ..., $param_N [= val_déf]])
{
        Bloc d'instructions...
        [return valeurs;] }
```

- Appel de fonction de la forme :

```
$resultat = nom_fonction([arg_1, ..., arg_N]);
```

Portée des variables

```
function SurfaceTriangle($largeur, $hauteur) {
    $resultat = ($largeur * $hauteur) / 2;
    return $resultat; }

echo SurfaceTriangle(20, 10) .
" cm<sup>2</sup>"; // retourne 100 cm²

echo "($largeur * $hauteur) / 2 = " . $resultat; // instruction
erronée : variables indéfinies
```

Portée des variables

- Par défaut, toutes les variables sont locales
  - Leur portée se réduit à la fonction ou au bloc de leur déclaration
  - Pour déclarer une variable globale, on peut utiliser le tableau $_GLOBALS[ ]

```
<?php $_GLOBALS['MaVar']="Bonjour"; ?>
```

- Constantes

```
<?php
        define("USER","TOTO");
        echo USER; // Notez l'absence de $ ici
?>
```

Exemples : fonctions sur variables

- gettype($var)
  - retourne le type de la variable argument
- settype($var)
  - change le type d'une variable, ce qui peut impliquer une conversion
- isset($var) / empty($var)
  - permet de tester si une valeur a été affectée à la variable / ou ne l'a pas été
- unset($var)
  - détruit une variable
- is_int/is_double/is_string
  - Permettent de tester le type d'une variable

Exemple

```
<?php
    // ceci est un commentaire   // variable utilisée sans avoir été déclarée
    $nom="dupont";
    print "nom=$nom\n";     // un affichage écran
    // un tableau avec des éléments de type différent
    $tableau=array("un","deux",3,4);
    $n=count($tableau);      // son nombre d'éléments
    for($i=0;$i<$n;$i++)
          print "tableau[$i]=$tableau[$i]\n";
    // initialisation de 2 variables avec le contenu d'un tableau
    list($chaine1,$chaine2)=array("chaine1","chaine2");
    // concaténation des 2 chaînes
    $chaine3=$chaine1.$chaine2;
    // affichage résultat
    print "[$chaine1,$chaine2,$chaine3]\n";
```

Exemple

```
affiche($chaine1); // utilisation fonction
afficheType($n); // le type d'une variable peut être connu
afficheType($chaine1);
afficheType($tableau);
$n="a changé"; // le type d'une variable peut changer en cours d'exécution
afficheType($n);
$res1=f1(4); // une fonction peut rendre un résultat
print "res1=$res1\n";
// une fonction peut rendre un tableau de valeurs
list($res1,$res2,$res3)=f2();
print "(res1,res2,res3)=[$res1,$res2,$res3]\n";
$t=f2(); // on aurait pu récupérer ces valeurs dans un tableau
for($i=0;$i<count($t);$i++)
    print "t[$i]=$t[$i]\n";
```

Exemple

```
for($i=0;$i<count($t);$i++)     // des tests
    if (getType($t[$i])=="string")    // n'affiche que les chaînes
            print "t[$i]=$t[$i]\n";
for($i=0;$i<count($t);$i++) // d'autres tests
     if (getType($t[$i])=="integer" and $t[$i]>10) // n'affiche que les entiers >10
            print "t[$i]=$t[$i]\n";
$t=array(8,5,0,-2,3,4); // une boucle while
$i=0;
$somme=0;
while($i<count($t) and $t[$i]>0){
            print "t[$i]=$t[$i]\n";
            $somme+=$t[$i]; //$somme=$somme+$t[$i]
            $i++; //$i=$i+1
}//while
print "somme=$somme\n";
exit; // fin programme
?>
```

Les chaînes en PHP

- Guillemets ou Cotes :

```
<?php
    $var="Hello PHP";
    $machaine="le contenu de \$var est $var<br>";
    echo $machaine;
   //ou avec des '  ':
   $mystring='le contenu de $var est '.$var;
   echo $mystring;
?>
```

- La concaténation : A l'aide de .
- La longueur d'une chaine : <?php int lg=strlen($chaine); ?>

Chaines en PHP

- Accéder au caractère i d'une chaine :

  <?php **echo** $chaine[i]; ?>

  La chaine est traitée comme un tableau indexé par un entier

  La plupart des tableaux de PHP sont indexés par des chaines...

- Mettre en majuscules/minuscules :
  - avec strtoupper() pour obtenir des majuscules
  - avec strtolower() pour mettre en minuscules
  - avec ucfirst() pour mettre en majuscule la première lettre d'une chaine
  - avec ucwords() pour mettre en majuscule la première lettre de chaque mot dans une chaine
- Cherche de sous-chaines ou de motifs dans une chaine
  - avec strstr()
  - avec stristr()
  - avec ereg() ou eregi()

TD

- TD

POO

- En PHP 5, il y a un tout nouveau model objet. La gestion des objets en PHP a été complètement réécrite, permettant de meilleurs performances ainsi que plus de fonctionnalités.
- Chaque définition de classe commence par le mot-clé **class** , suivi par le nom de la classe, qui peut être quelconque à condition que ce ne soit pas un mot réservé en PHP. Suivent une paire d'accolade contenant la définition des membres et des méthodes. Une pseudo-variable **$this** est disponible lorsqu'une méthode est appelée depuis un contexte objet.

POO

- Définition simple d'une classe :

```
<?php
class SimpleClass
{
   // déclaration d'un membre
   public $var = 'une valeur par défaut';

   // déclaration de la méthode
   public function displayVar() {
     echo $this->var;
   }
}
?>
```

POO

- **Le mot clé new :**
  - Pour créer une instance d'un objet, un nouvel objet doit être créé et assigné à une variable. Un objet doit toujours être assigné lors de la création d'un nouvel objet à moins qu'un l'objet ait un constructeur défini qui lance une exception en cas d'erreur. Création d'une instance :

```
<?php
   $instance = new SimpleClass();
?>
```

POO

- **Le mot clé extends**
  - Une classe peut hériter des méthodes et des membres d'une autre classe en utilisant le mot clé extends dans la déclaration. Il n'est pas possible d'étendre de multiples classes, une classe peut uniquement hériter d'une seule classe de base.
  - Les méthodes et membres hérités peuvent être surchargés, à moins que la classe parent ait défini une méthode comme final. Pour surcharger, il suffit de redéclarer la méthode avec le même nom que celui défini dans la classe parent. Il est possible d'accéder à une méthode ou un membre surchargé avec l'opérateur **parent ::**

POO

- Le mot clé extends

```
<?php
    class SimpleClass
      {
       // déclaration d'un membre
       public $var = 'une valeur par défaut';
       // déclaration de la méthode
       public function displayVar() {
           echo $this->var;
                  }
      }
```

```
// extension de la classe
class ExtendClass extends SimpleClass
{
    // Redéfinition de la méthode parent
    function displayVar()
     {
        echo "Classe étendue\n";
       parent::displayVar();
     }
}
$extended = new ExtendClass();
$extended->displayVar();
?>
```

POO

- **Constructeurs** : void __construct (mixed args , ...)
  - Les classes qui possèdent une méthode constructeur appellent cette méthode à chaque création d'une nouvelle instance de l'objet.
  - Note : Les constructeurs parents ne sont pas appelés implicitement si la classe enfant définie un constructeur. Si vous voulez utiliser un constructeur parent, il sera nécessaire de faire appel à parent::__construct().
- **Exemple** :

```
<?php
class BaseClass {
   function __construct() {
      print "In BaseClass constructor\n";
                                  }
               }
```

```
class SubClass extends BaseClass {
   function __construct() {
      parent::__construct();
      print "In SubClass constructor\n";
   }
}
$obj = new BaseClass();
$obj = new SubClass();
?>
```

POO

- Destructeurs : void __destruct ( )
  - PHP 5 introduit un concept de destructeur similaire aux autres langages orientés objet, comme le C++ . La méthode destructeur doit être appelée aussitôt que toutes les références à un objet particulier sont effacées ou lorsque l'objet est explicitement détruit. Exemple avec un Destructeur

```
<?php
class MyDestructableClass {
       function __construct() {
            print "In constructor\n";
            $this->name = "MyDestructableClass";
                                        }
         function __destruct() {
                        print "Destruction de " . $this->name . "\n";
                                    }
                                   }
$obj = new MyDestructableClass(); ?>
```

POO

- Visibilité
  - La visibilité d'une propriété ou d'une méthode peut être définie en préfixant la déclaration avec un mot-clé : **public** , **protected** ou **private** . Les éléments déclarés publics (public) peuvent être utilisés par n'importe quelle partie du programme. L'accès aux éléments protégés (protected) est limité aux classes et parents hérités (et à la classe qui a défini l'élément). L'accès aux éléments privés (private) est uniquement réservé à la classe qui les a définis.

POO (exemple de visiblité d'une variable)

```
<?php
    /**
     * Définition de MyClass
     */
   class MyClass
      {
          public $publicc = 'Public';
          protected $protectedd = 'Protected';
          private $privatee = 'Private';

          function printHello()
                    {
                    echo $this->privatee;
                    echo $this->protectedd;
                    echo $this->privatee;
                    }
      }
```

L'opérateur de résolution de portée (::) :

- Il fournit un moyen d'accéder aux **membres statiques ou constants** ainsi qu'aux éléments **redéfinis par la classe**.
- Lorsque vous référencez ces éléments en dehors de la définition de la classe, **utilisez le nom de la classe**.

```
<?php
        class MyClass {
                const CONST_VALUE = 'Une valeur constante';
        }
                echo MyClass::CONST_VALUE;
?>
```

Self et parent

- Deux mots-clé spéciaux, **self** et **parent** , sont utilisés pour accéder aux membres ou aux méthodes depuis la définition de la classe.

```php
<?php
class MyClass {
    const CONST_VALUE = 'Une valeur constante';
}
echo MyClass::CONST_VALUE;
?>
```

```php
:: depuis la définition de la classe
<?php
class OtherClass extends MyClass
{
    public static $my_static = 'variable
                statique';

    public static function doubleColon() {
        echo parent::CONST_VALUE . "\n";
        echo self::$my_static . "\n";
    }
}
OtherClass::doubleColon();
?>
```

Statique

- Le fait de déclarer des membres ou des méthodes comme statiques vous permet d'**y accéder sans avoir besoin d'instancier la classe**. Un membre déclaré comme statique ne peut être accédé avec l'objet instancié d'une classe (bien qu'une méthode statique le peut).
- La déclaration static doit être faite après la déclaration de visibilité. Pour des raisons de compatibilité avec PHP 4, si aucune déclaration de visibilité n'est utilisée, alors le membre ou la méthode sera traité comme s'il avait été déclaré comme public .
- Comme les méthodes statiques sont appelables sans instance d'objet créée, la pseudo variable **$this n'est pas disponible dans la méthode déclarée en tant que statique**.

PHP

- **Statique :**
  - En fait, les appels de méthodes statiques sont résolus au moment de la compilation. Lorsque l'on utilise un nom de classe explicite, la méthode est déjà identifiée complètement et aucune notion d'héritage n'est appliquée. Si l'appel est effectuée par le mot clé self , alors self est traduit en la classe courante, qui est la classe appartenant au code. Ici aussi, aucune notion d'héritage n'est appliquée.
  - **On ne peut pas accéder à des propriétés statiques à travers l'objet en utilisant l'opérateur ->** .

Exemple avec un membre statique

```
<?php
class Foo
{
    public static $my_static = 'foo';
    public function staticValue() {
        return self::$my_static;
                                    }
}
class Bar extends Foo
{
    public function fooStatic() {
        return parent::$my_static;
                                    }
}

print Foo::$my_static . "\n";
$foo = new Foo();
print $foo->staticValue() . "\n";
print $foo->my_static . "\n";
// propriété my_static non définie
// $foo::my_static n'est pas possible
print Bar::$my_static . "\n";
$bar = new Bar();
print $bar->fooStatic() . "\n";
?>
```

Constantes de classe

- Il est possible de définir des valeurs constantes à l'intérieur d'une classe, qui ne seront pas modifiables. Les constantes diffèrent des variables normales du fait qu'on n'utilise pas le symbole $ pour les déclarer ou les utiliser. Tout comme pour les membres statiques , on ne peut pas accéder aux valeurs constantes depuis une instance de l'objet (en utilisant $object::constant).

```php
<?php
class MyClass
{
      const constant = 'valeur constante';
      function showConstant() {
      echo  self::constant . "\n";
                                    }
}
echo MyClass::constant . "\n";
$class = new MyClass();
$class->showConstant();
// echo $class::constant;  n'est pas autorisé
?>
```

Abstraction de classes

- PHP 5 introduit les classes et les méthodes abstraites. Il n'est pas autorisé de créer une instance d'une classe définie comme abstraite. Toutes les classes **contenant au moins une méthode abstraite doivent également être abstraites**. Pour définir une méthode abstraite, il faut simplement déclarer la signature de la méthode et ne fournir aucune implémentation.
- Lors de l'héritage depuis une classe abstraite, toutes les méthodes marquées comme abstraites dans la déclaration de la classe parent doivent être définies par l'enfant ; de plus, ces méthodes doivent être définies avec la même (ou plus faible) visibilité . Par exemple, si la méthode abstraite est définie comme protégée, l'implémentation de la fonction doit être définie en tant que protégée ou publique.

```
<?php
        abstract class AbstractClass
        …
```

- Interfaces
  - Les interfaces objet vous permettent de créer du code qui spécifie quelles méthodes et variables une classe peut implémenter, sans avoir à définir comment ces méthodes seront gérées.
  - Les interfaces sont définies en utilisant le mot clé interface , de la même façon qu'une classe standard mais sans aucun contenu de méthode.
  - Toutes les méthodes déclarées dans une interface doivent être publiques.
- implements
  - Pour implémenter une interface, l'opérateur implements est utilisé. Toutes les méthodes de l'interface doivent être implémentées dans une classe ; si ce n'est pas le cas, une erreur fatale sera émise. Les classes peuvent implémenter plus d'une interface en séparant chaque interface par une virgule.

Interfaces

```php
<?php
// Declaration de l'interface 'iTemplate'
interface iTemplate
{
    public function setVariable($name, $var);
    public function getHtml($template);
}
```

```php
// Implémentation de l'interface
// Ceci va fonctionner
class Template implements iTemplate
{
    private $vars = array();
    public function setVariable($name, $var)
    {
        $this->vars[$name] = $var;
    }
    public function getHtml($template)
    {
        foreach($this->vars as $name => $value) {
            $template = str_replace('{' . $name . '}', $value,
$template);
        }
        return $template;
    }
}
```

Surcharge

- Les appels de méthodes et l'accès aux membres peuvent être surchargés via les méthodes **__call , __get et __set** . Ces méthodes ne seront déclenchées que si votre objet, hérité ou non, ne contient pas le membre ou la méthode auquel vous tentez d'accéder. Toutes les méthodes surchargées doivent être définies en tant que **public** .
- Depuis PHP 5.1.0, il est également possible de surcharger les fonctions isset et unset via, respectivement, les méthodes __isset et __unset.
- Mot clé ‘final’
  - PHP 5 introduit le mot-clé " final " qui empêche les classes filles de surcharger une méthode en en préfixant la définition par le mot-clé "final". Si la classe elle-même est définie comme finale, elle ne pourra pas être étendue.

Parcours d'objets

- PHP 5 fournit une façon de définir les objets de manière à ce qu'on puisse **parcourir une liste de membres** avec une structure foreach . Par défaut, toutes les propriétés visibles seront utilisées pour le parcours.

```
$class = new MyClass();
foreach($class as $key => $value) {
    print "$key => $value\n";
}
```

- Comme nous le montre l'affichage, l'itération foreach affiche toutes les variables visibles disponibles. Pour aller plus loin, vous pouvez implémenter l'interface interne de PHP 5 nommée **Iterator** . Ceci permet de déterminer comment l'objet doit être parcouru.

```
<?php
    class MyIterator implements Iterator
```

Masques

- Les masques sont un moyen de décrire les meilleures pratiques et les bonnes conceptions. Ils proposent une solution flexible aux problèmes habituels de programmation.
  - Usine
  - Singleton
- Méthodes magiques
  - Les noms de fonction __construct , __destruct, __call , __get , __set , __isset , __unset, __sleep , __wakeup , __toString , __set_state , __clone et __autoload sont magiques dans les classes PHP. Vous ne pouvez pas utiliser ces noms de fonction dans aucune de vos classes sauf si vous voulez modifier le comportement associé à ces fonctions magiques.
  - Le but de __sleep est de clore toutes les connexions aux bases de données que l'objet peut avoir, valider les données en attente ou effectuer des tâches de nettoyage.
  - Le but de __wakeup est de rétablir toute connexion base de données qui aurait été perdue durant la linéarisation et d'effectuer des tâches de réinitialisation.
  - La méthode __toString détermine comment la classe doit réagir lorsqu'elle est convertie en chaîne de caractères

Auto-chargement de classes

- De nombreux développeurs qui créent des applications orientées objet, créent un fichier source par définition de classe. L'inconvénient majeur de cette méthode est d'avoir à écrire une longue liste d'inclusions de fichier classes au début de chaque script : une inclusion par classe.

- En PHP 5, ce n'est plus nécessaire. Vous pouvez définir la fonction **__autoload** qui va automatiquement être appelée si une classe n'est pas encore définie au moment de son utilisation. Grâce à elle, vous avez une dernière chance pour inclure une définition de classe, avant que PHP ne déclare une erreur.

- Note : Les exceptions lancées depuis la fonction __autoload ne peuvent être interceptées par un bloc catch : elles provoqueront une erreur fatale.

- Class MaClasse [extends SuperClass]

```
function MaClasse(parametre1, parametre2) {
    this.attributPublic = parametre1;
    var attributPrive = parametre2;
    this.methodePublique = function() { }
    var methodePrivee = function() { }
}
```

- Instanciation
  - var objetDeMaClasse = new MaClasse("valeur1", "valeur2");
- Appel des méthodes
  - $obj = new Classe(); //Instanciation de classe
  - $obj->methodeInstance(); //Application de méthode


```
class Livre extends Produit {
    private $auteur;
    private $editeur;
    private $nb_pages;
    function _construct($auteur, $editeur, $nb_pages) {
        $this->auteur = $auteur;
        $this->editeur = $editeur;
        $this->nb_pages = $nb_pages; }
    function changer_nb_pages($nb){…}
    }
  $livre = new Livre(…);
  $livre->changer_nb_pages(150) ;
```

Exercice

- Réalisez une classe Personne (class.Personne.php) qui permet de décrire un utilisateur à partir de son nom, prénom, login, mail, mot de passe, date de naissance, date d'inscrition, statut, etc.
- Après le constructeur, il faut coder les méthodes de :
  - mise à jour des informations updateData($info)
  - sauvegarde des informations saveData()
  - destruction du compte delDate()
  - Méthodes __get et __set pour les attributs privés

PHP et les bases de données

- Supporte de nombreuses bases de données
  - DB2, Dbase, Oracle, SqlServer, ODBC, PostgreSQL, Unix DBM…
- Interface spécifique à chaque base
- Couche d'abstraction de bases de données
  - Exemple : PEAR:MDB2
- Principe d'interface similaire
  - Exemple : MySQL

Plusieurs moyens de se connecter à une DB MySQL

- L’extension mysql_ : ce sont des fonctions qui permettent d'accéder à une base de données MySQL et donc de communiquer avec MySQL. Toutefois, ces fonctions sont vieilles et on recommande de ne plus les utiliser aujourd'hui.
- L’extension mysqli_: ce sont des fonctions améliorées d'accès à MySQL. Elles proposent plus de fonctionnalités et sont plus à jour.
- L'extension PDO : c'est un outil complet qui permet d'accéder à n'importe quel type de base de données. On peut donc l'utiliser pour se connecter aussi bien à MySQL que PostgreSQL ou Oracle.

MySQL

- Ouvrir une connexion au serveur :

  mysql_connect
- Sélectionner une base de données de travail :

  mysql_select_db
- Effectuer une requête :

  mysql_query
- Lire le résultat d'une requête :

  mysql_result, mysql_fetch_array, ...
- Fermer la connexion :

  mysql_close

phpMyAdmin

- phpMyAdmin can manage a whole MySQL server as well as a single database over the World Wide Web.
- Official Site: http://www.phpmyadmin.net/
- Documentation: http://www.phpmyadmin.net/documentation/
- Features
  - Browser-based, Supporting PHP5.3+, MySQL 5.0+, Open Source
- There are five authentication modes offered:
  - http:
  - cookie
  - signon
  - config(the less secure one, not recommanded).
  - Swekey authentication mode:


```php
<?php
// Connexion et sélection de la base
$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')
    or die('Impossible de se connecter : ' . mysql_error());
echo 'Connected successfully';
mysql_select_db('my_database') or die('Impossible de sélectionner la base de
   données');
// Exécution des requêtes SQL
$query = 'SELECT * FROM my_table';
$result = mysql_query($query) or die('Échec de la requête : ' .
   mysql_error());
// Affichage des résultats en HTML
echo "<table>\n";
while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
    echo "\t<tr>\n";
    foreach ($line as $col_value) {
        echo "\t\t<td>$col_value</td>\n"; // ...
?>
```

SGBD : Abstraction

- Exemple de bibliothèque d'abstraction : PEAR::MDB2

```
<?php
require_once('MDB2.php');

$mdb2 =& MDB2::connect('pgsql://usr:pw@localhost/dbnam');
if (PEAR::isError($mdb2)) { die($mdb2->getMessage()); }
// La requête...
$res =& $mdb2->query('SELECT * FROM clients');
if (PEAR::isError($res)) { die($res->getMessage()); }
while (($row = $res->fetchRow())) {
   foreach($row as $col) { echo $col; }
}
$mdb2->disconnect();
?>
```

PDO : Interface d'accès aux BDD

- PDO (PHP Data Objects) : interface parue avec PHP 5 qui vise à utiliser des bases de données sans avoir à s'occuper du SGBD utilisé derrière. Ainsi, il est tout à fait possible de faire un code qui marchera avec MySQL mais aussi avec Oracle, ODBC, etc.
- Avantages de PDO :
  - interface pour SGBD : plus besoin de s'occuper de savoir quelle SGBD est derrière (en théorie) ;
  - orienté objet : les objets PDO et PDOStatement peuvent être étendus, il est donc tout à fait possible de personnaliser et remodeler une partie du comportement initial ;
  - exception : les objets de l'interface PDO utilisent des exceptions, il est donc tout à fait possible d'intégrer facilement un système de gestion des erreurs.

PDO : Interface d'accès aux BDD

- **Activation** :
  - Avec PHP 5, cette extension n'est pas activée par défaut. Afin de remédier au problème, ouvrez le 'php.ini' que vous utilisez pour PHP 5 puis rendez-vous dans la partie où vous avez la liste des extensions. Faites une recherche sur '; Windows Extensions', vous devriez y arriver directement. Bien : maintenant, vérifiez que vous avez la ligne suivante :
    - ;extension=php_pdo.dll
  - Si c'est le cas, décommentez la ligne afin d'avoir :
    - extension=php_pdo.dll
  - Sinon, rajoutez la ligne à la main.
  - Vous venez donc d'activer PDO mais cela n'est pas suffisant : maintenant, il vous faut activer le driver correspondant au type de BDD que vous souhaitez utiliser.
  - Pour cela rien de plus simple : imaginons que vous vouliez activer MySQL, rajoutez pour cela la ligne suivante (ou activez-la si elle est présente) :
    - extension=php_pdo_mysql.dll
  - Bien, enregistrez, relancez le serveur et vous voilà prêts à travailler avec l'extension PDO qui utilise MySQL

PDO : Interface d'accès aux BDD

- Création d'une connexion :

```
<?php
$PARAM_hote='localhost'; // le chemin vers le serveur
$PARAM_port='3306';
$PARAM_nom_bd='TIC'; // le nom de votre base de données
$='root'; // nom d'utilisateur pour se connecter
$PARAM_mot_pPARAM_utilisateurasse=''; // mot de passe de l'utilisateur
    pour se connecter
$connexion=new
    PDO('mysql:host='.$PARAM_hote.';port='.$PARAM_port.';dbname='.$PA
    RAM_nom_bd, $PARAM_utilisateur, $PARAM_mot_passe);
?>
```

PDO : Interface d'accès aux BDD

- Création d'une connexion avec gestion des erreurs :

```
<?php
try
{
        $connexion=new
    PDO('mysql:host='.$PARAM_hote.';dbname='.$PARAM_nom_bd,
    $PARAM_utilisateur, $PARAM_mot_passe);
}
catch(Exception $e)
{
        echo 'Erreur : '.$e->getMessage().'<br />';
        echo 'N° : '.$e->getCode();
}
?>
```

PDO : Interface d'accès aux BDD

- **Requêtes** :
  - Comme vous le savez, il est possible sur une BDD de récupérer des informations, mais aussi d'effectuer des changements (qui peuvent prendre la forme d'ajout, suppression ou modification).
  - Si vous souhaitez récupérer une information (SELECT), vous devrez alors utiliser 'query' ; si c'est pour un apporter des changements (INSERT, UPDATE, DELETE) à la BDD, vous devrez utiliser 'exec'.

PDO : Interface d'accès aux BDD

- **Requêtes** :
  - Utilisation de la méthode : exec
  - Supposons que nous souhaitons modifier le mot de passe des membres :

```
<?php
   $connexion=new
   PDO("mysql:host=$PARAM_hote;dbname=$PARAM_nom_bd",
   $PARAM_utilisateur, $PARAM_mot_passe); // connexion à la BDD


   $connexion->exec("UPDATE membres SET mot_pass='toto'"); // on
   //modifie le mot de passe de tous les utilisateurs (oui, ça n'a aucun intérêt
   //mais c'est pour l'exemple)
?>
```

PDO : Interface d'accès aux BDD

- **Requêtes** :
  - Utilisation de la méthode : query
  - Imaginons que nous souhaitons connaître tous les membres :

```
<?php
$connexion=new PDO("mysql:host=$PARAM_hote;dbname=$PARAM_nom_bd", $PARAM_utilisateur, $PARAM_mot_passe); // connexion à la BDD
$resultats=$connexion->query("SELECT membre FROM membres ORDER BY membre ASC"); // on va chercher tous les membres de la table qu'on trie par ordre croissant
$resultats->setFetchMode(PDO::FETCH_OBJ); // on dit qu'on veut que le résultat soit récupérable sous forme d'objet
while( $ligne = $resultats->fetch() ) // on récupère la liste des membres
{
      echo 'Utilisateur : '.$ligne->membre.'<br />'; // on affiche les membres
}
$resultats->closeCursor(); // on ferme le curseur des résultats
?>
```

- + d'info :
  - http://www.siteduzero.com/tutoriel-3-34790-pdo-interface-d-acces-aux-bdd.html

TD

cURL

- La librairie cURL permet de facilement communiquer avec de nombreux types de serveurs applicatifs en parlant le même langage que ceux-ci.
  - Ce langage est définit par ce qu'on appelle un protocole dont les plus connus sont sans aucun doute HTTP et FTP. L'extension cURL permet d'interagir en PHP avec tous ces protocoles que nous employons de manière quotidienne sans avoir à gérer la connexion ou encore sans se soucier de la manière dont il faut écrire la requête ou en recevoir la réponse.
  - Connection en HTTPS, FTP, SSL, Passage par un proxi
- Bibliothèque portable
- Communication entre serveur applicatifs
- Interaction transparente via les protocoles courants

cURL : protocoles

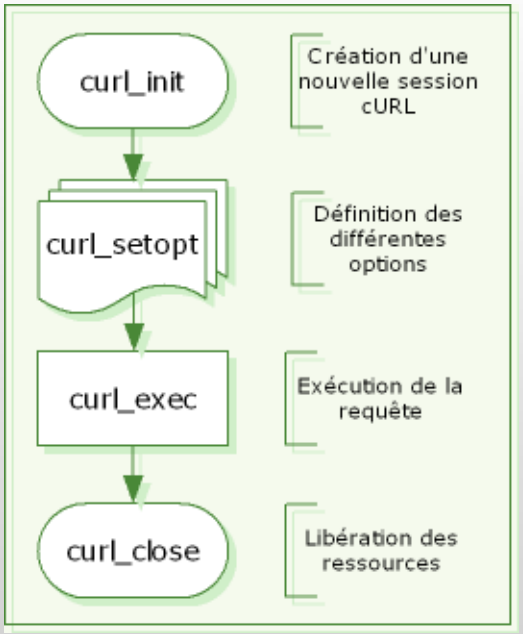
- Support de nombreux protocoles
  - HTTP: authentification, Get, Post, gestion des cookies et des sessions
  - FTP : authentification, liste, envoi et réception de fichiers
  - LDAP : authentification, lecture
  - Autres : Telnet, Gopher, Dict, File
- Chiffrage et certificat SSL
- Connexions via un proxy

cURL fonctionnement

```
<?php
/** Ce code récupère le flux ATOM de Google News et le
   retourne directement au navigateur **/
# Initialisation
$ch = curl_init("http://news.google.fr/"
   ."nwshp?hl=fr&tab=wn&output=atom");
# Exécution
curl_exec($ch);

# Fermeture de la session cURL
curl_close($ch);

# C'est tout !
?>
```



cURL

- cURL est utilisable dans d'autres langages
- cURL fait bien plus !
- Exemple :
  - récupération d'un flux ATOM dans un fichier temporaire et upload sur un serveur FTP distant


```
<?php
define('TMP_FILE', "tmp_atom.txt");
# Initialisation ... vide !
$ch = curl_init();
# Configuration de l'URL et du retour
curl_setopt($ch, CURLOPT_URL,
   "http://news.google.fr/nwshp?hl=fr&tab=wn&output=atom");
# Création du fichier de destination et ouverture rw
$fp = fopen(TMP_FILE, "w+");
# Lien : mettre le contenu ici
curl_setopt($ch, CURLOPT_FILE, $fp);
# Exécution
curl_exec($ch);
if(curl_errno($ch) != 0)  { echo "Erreur lors du téléchargement\n";
   die(); }
# Fermeture de la session cURL
curl_close($ch);
```


```
# Seek au début du fichier de destination
fseek($fp, 0);
# Et c'est reparti !
$ch_ftp = curl_init();
# L'emplacement FTP de destination
curl_setopt($ch_ftp, CURLOPT_URL,
   "ftp://ftp_login:password@ftp.exemple.fr/leFluxAtom");
# Indique à cURL qu'on uploade
curl_setopt($ch_ftp, CURLOPT_UPLOAD, 1);
# Transfert en ASCII
curl_setopt($ch_ftp, CURLOPT_TRANSFERTEXT, 1);
# Passage du pointeur
curl_setopt($ch_ftp, CURLOPT_INFILE, $fp);
# .. et la taille du fichier
curl_setopt($ch_ftp, CURLOPT_INFILESIZE, filesize(TMP_FILE));
# ET HOP !
curl_exec ($ch_ftp);
if(curl_errno($ch_ftp) != 0)  { echo "Erreur lors de l'upload\n";
   die(); }
curl_close ($ch_ftp);
echo "Terminé avec succès !\n"
?>
```

Terminologies

- Classe: type de données défini par le programmeur.
- Objet: instance de la classe. On défini la classe une seule fois et on l'utilise plusieurs fois pour créer des objets.
- Donnée membre: aussi appelée attribut ou propriété et représente une valeur faisant partie des données de la classe.
- Fonction membre: aussi appelée méthode et représente une fonction permettant d'agir sur les données de la classe.
- Classe parent: c'est la classe utilisée pour dériver une nouvelle classe. On l'appelle aussi classe de base.
- Classe enfant: c'est la nouvelle classe dérivée à partir d'une classe parent.

Encapsulation

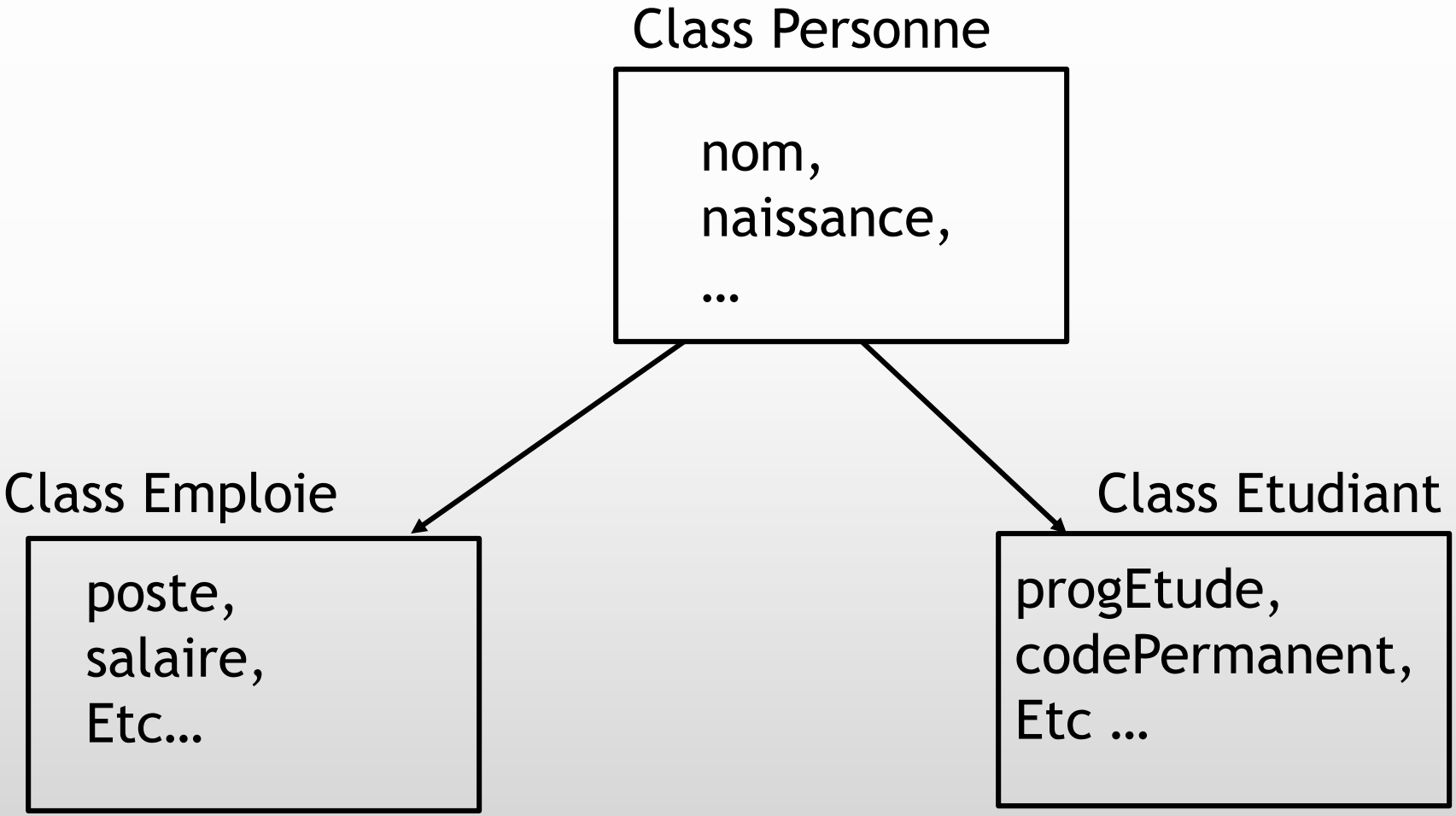
- L'encapsulation consiste à cacher les détails internes utiles au fonctionnement et à l'implémentation du type de données.
- Au niveau programmation on pourra décider de rendre disponible seulement certaines portions du type de données au monde extérieur et cacher le reste (les détails d'implémentation). De cette manière le concepteur qui décide de modifier la portion cachée peut le faire sans risque d'affecter les programmeurs l'utilisant car il est certain qu'ils ne voient pas cette portion.
- Il est également possible de forcer la modification des données en passant par un mutateur (fonction de modification) qui permettra de valider le changement avant qu'il soit effectuer.
- De la même manière il est possible de forcer la lecture en passant par un accesseur (fonction de lecture).

Héritage

- Le concept d'héritage est le plus important de la POO. Il permet la réutilisation de type de base défini par l'utilisateur tout en permettant de spécialiser le type.
- C'est la base des notions de réutilisation de composants logiciels.
- L'idée est de pouvoir définir (dériver) une nouvelle classe en se servant d'une classe existante (base).
- La classe dérivée hérite des membres de la classe de base tout en lui ajoutant de nouveaux.
- Il s'agit d'écrire de nouvelles classes plus spécifiques en se servant des définitions de base.
- Par exemple, nous pouvons dériver une nouvelle classe Employe en se servant de la classe de base Personne.
- Permet de définir la relation "est un". Par exemple, un employé est une personne ou encore un étudiant est une personne. Un cercle est une forme géo…


PHP

Héritage


Class Personne
nom,
naissance,
…
Class Emploie
poste,
salaire,
Etc…
Class Etudiant
progEtude,
codePermanent,
Etc …

Constructeurs/destructeurs

- Un constructeur servira à initialiser le contenu d'un objet et même dans certains cas à allouer l'espace nécessaire aux données membres de l'objet.
- Le constructeur est appelé automatiquement lors de la création de l'objet, il est alors impossible de l'oublier.
- Un destructeur permet de libérer les ressources utilisées par un objet.

Constructeurs/destructeurs

- 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 propriétés (variables) et des méthodes (fonctions). On dit propriétés et méthodes membres

```
class NomDeLaClasse{
        var $propriete1;
        var $propriete2;
              function__construct() {
        //constructeur
                  }
        functionfonction() {
          // méthode   } }
```

Instanciation

- Afin d'instancier un objet, il faut utiliser le mot clé new $maVar = new NomDeLaClasse();
- Afin de faire référence à une propriété de l'objet, il faut utiliser le symbole ->

```
$maVar->prop = 23;
```

La classe Employe

```
class Employe {
        var $nom;
        var $salaire;
        function __construct($n, $s) {
                $this->nom = $n;
                $this->salaire = $s;
        }
        function toHTML() {
                return "<strong>Le nom:</strong><em>$this->nom</em>".
                "<strong>sal:</strong><em>$this->salaire</em>";
        }
    }
    $bob = new Employe( "Bob", 45000 );
    echo $bob->toHTML();
```

Utilisation de $this

- Afin de faire référence à une propriété à l'intérieur de la classe, il faut utiliser :

  $this$this->age = 23;

- Le constructeur est automatiquement appelé lors de l'instanciation d'un objet de la classe.
- Le constructeur est une méthode dont le nom est __construct( ) (notez le double souligné).
- En PHP4 le constructeur portait le même nom que la classe.
- Seulement le dernier constructeur défini sera utilisé.
- Le desctructeur est automatiquement appelé lorsqu'on détruit un objet. Le nom du destructeur est __destruct( ). Pour détruire un objet la fonction unset(objet) sera utilisée.

Le mot clé private

- Dans l'exemple précédent avec $bob utilisant la classe Employe les propriétés $nom et $salaire sont publiques (par défaut). Les propriétés sont accessibles aux utilisateurs de la classe.
- Il est alors possible de faire: $bob->salaire = -12000;
- En POO ce comportement n'est pas désirable, car les propriétés ne devraient être accessibles que par des méthodes de la classe. Cela permet un minimum de cohérence des données ainsi la méthode qui sert à modifier le salaire peut valider la nouvelle valeur avant de faire ces modifications.

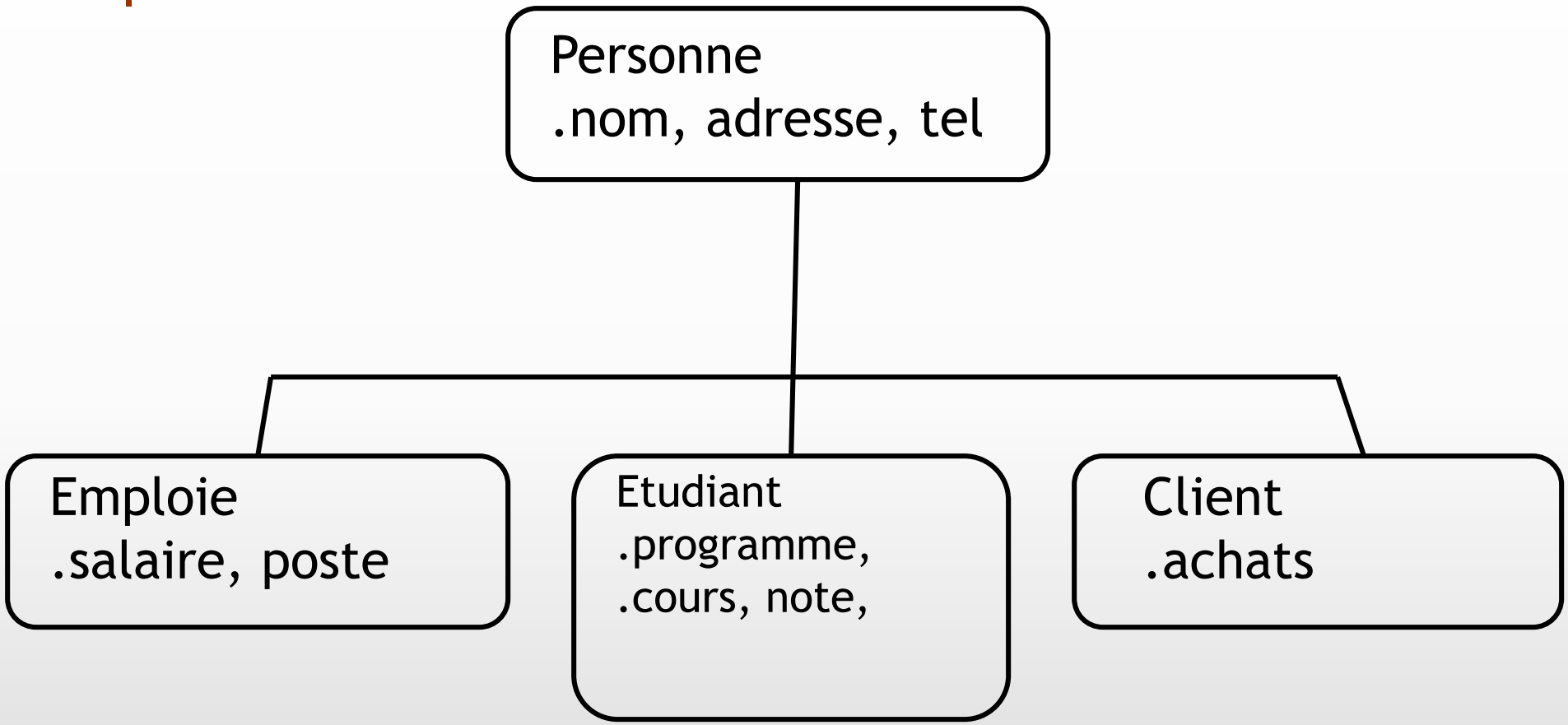
```
private $nom; private $salaire;
```

- Il est aussi possible de rendre privé des méthodes, qui ne pourront être utilisées qu'à l'intérieur de la classe.

L’héritage plus spécifiquement

- L’héritage consiste à réutiliser une classe de base (plus générale) pour en faire une version plus spécialisée.
- Par exemple, nous pourrions faire une classe Personne qui contient les propriétés nom et adresse pour ensuite dériver une nouvelle classe Employe à partir de Personne. Cette nouvelle classe permettrait d’ajouter la propriété salaire et les méthodes qui s’y rattache. Dans la même ordre d’idée nous pourrions dériver une autre classe Etudiant à partir de la classe Personne pour lui ajouter son dossier étudiant et les méthodes permettant de le manipuler.

L'héritage



Les classes dérivées héritent des propriétés nom, adresse et tel en plus de pouvoir utiliser les méthodes définies dans la classe de bas

La classe Personne

```
class Personne {
        private $nom;
        private $adresse;
        private $tel
        function __construct($n, $a, $t) {
                $this->nom = $n;
                $this->adresse = $a;
                $this->tel = $t;
                }
        function toHTML() {
                // …
                }
        }
        $toto = new Personne( "toto", "123 rue Laval", "514…");
        echo $toto->toHTML();
```

La classe dérivée Employe

```
class Employe extends Personne {
        private $salaire;
        private $poste;
        function __construct($n, $a, $t, $s, $p){
                parent::__construct($n, $a, $t);
                $this->salaire = $s;$this->poste = $p;}
        function toHTML() {// …}}
        $bob = new
        Employe("toto","123…","514…",45000,"prog");
        echo $bob->toHTML();
```

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();
```

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:
  - Accesseurs: pour lire un attribut getAtt()par exemple: echo $bob->getNom();
  - Mutateurs: pour modifier la valeur setAtt($var)par exemple: $bob->setNom("Larue");

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.

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()).
- L'utilisation la plus simple de toHTML() est donc:

```
echo $monObj->toHTML();
```

- Cette façon de faire simplifie l'utilisation de librairies de gestion de page qui deviennent de plus en plus la norme pour des projets d'envergure.

Concrètement: $_GETet $_POST

- Il est généralement avantageux de ne pas accéder directement $_GETet $_POST à l'intérieur d'objets si ceux-ci ne font pas partie de la couche de présentation («interface HTML»).
- Il vaut mieux ajouter des paramètres aux méthodes et passer $_GET ou $_POST comme paramètres aux fonctions

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»).

Polymorphisme et héritage multiple

- Le polymorphisme est supporté partiellement en PHP 5. La redéfinition d'une méthode dans une classe enfant permettra de surcharger ce nom et la méthode correspondante sera appelée en fonction du type de l'objet utilisé.
- L'héritage multiple n'est pas supporté en PHP 5.

Notions avancées de POO

- Membres protected
- Interfaces
- Constantes
- Classes abstraites
- Utilisation des méthodes de la classe de base
- Surcharge des méthodes

Autres Fonctions PHP: Les dates et les heures

- **Les fonctions de date et d'heure**

true | false = checkdate(mois, jour, année);

vérifie la validité d'une date.

$chaine = date(format [, nombre]);

retourne une chaîne de caractères date/heure selon le format spécifié et représentant la date courante par défaut.

$tableau = getdate([nombre]);

retourne les éléments de date et d'heure dans un tableau associatif.

$tableau = gettimeofday();

retourne l'heure courante dans un tableau associatif.

$chaine = gmdate(format [, nombre]);

retourne une chaîne de caractères date/heure GMT/CUT selon le format spécifié et représentant la date courante par défaut.

$nombre = gmmktime(heure, minute, seconde, mois, jour, année [, 1/0]);

retourne l'instant UNIX d'une date GMT spécifiée et avec éventuellement une heure d'hiver

Autres Fonctions PHP(2)

- **Les fonctions de date et d'heure**

  $chaine = gmstrftime(format [, nombre]);

  formate une date/heure GMT/CUT en fonction des paramétrages locaux définis par setlocale.

  $tableau = localtime([nombre][, tab_associatif]);

  retourne l'heure locale dans un tableau indicé par défaut ou associatif (1)

  $chaine = microtime();

  retourne l'instant UNIX courant en secondes et microsecondes (1 janvier 1970 à 0H00)

  $nombre = mktime(heure, minute, seconde, mois, jour, année [, 1/0]);

  retourne l'instant UNIX d'une date spécifiée et avec éventuellement une heure d'hiver (1)

  $chaine = strftime(format [, instant]);

  formate une date/heure locale avec les options locales

  $nombre = time();

  retourne l'instant UNIX courant

Autres Fonctions PHP(3)

- **Les formats de date et d'heure**

  a représente am (matin) ou pm (après-midi).
  A représente AM (matin) ou PM (après-midi).
  B représente une heure Internet Swatch.
  d représente le jour du mois sur deux chiffres allant de 01 à 31.
  D représente le jour de la semaine en trois lettres et en anglais (Sun, ..., Sat).
  F représente le mois complet en anglais (January, ..., December).
  g représente une heure au format 12 heures allant de 1 à 12.
  G représente une heure au format 24 heures allant de 1 à 24.
  h représente une heure au format 12 heures avec un zéro de complément allant de 00 à 11.
  H représente une heure au format 24 heures allant de 00 à 23.
  i représente les minutes allant de 00 à 59.
  I est égal à 1 si l'heure d'été est activée ou 0 pour l'heure d'hiver.
  j représente le jour du mois allant de 1 à 31.
  l représente le jour de la semaine complet et en anglais (Sunday, ..., Saturday).
  L est égal à 1 si l'année est bissextile, sinon 0.
  m représente un mois allant de 01 à 12.
  M représente un mois en trois lettres et en anglais (Jan, ..., Dec).

Autres Fonctions PHP(4)

- **Les formats de date et d'heur**

n	représente un mois allant de 1 à 12.
O	représente la différence d'heures avec l'heure de Greenwich (+0100).
r	représente un format de date conforme au RFC 822 (Mon, 25 Mar 2002 05:08:26 +0100).
s	représente les secondes allant de 00 à 59.
S	représente le suffixe ordinal d'un nombre en anglais et sur deux lettres th ou nd.
t	représente le nombre de jours dans le mois (28, 29, 30 ou 31).
T	représente le fuseau horaire.
U	représente les secondes depuis une époque.
w	représente le jour de la semaine allant de 0 (Dimanche) à 6 (Samedi).
Y	représente une année sur quatre chiffres (2002).
y	représente une année sur 2 chiffres (02).
z	représente le jour de l'année allant de 0 à 365.
Z	représente le décalage horaire en secondes.


PHP

XML

- eXtensible Markup Language : langage de balisage extensible
- Assurer l'intéropabilité entre les SIs

```
<article>
   <title> Extensible Markup Language
</title>   <para>
        <acronym> XML </acronym>
        « langage de balisage
extensible »
   </para>
</article>
```

- Parsing d’un document XML
- Séparation présentation / contenu
- Analyse syntaxique
- Librairie *expat*

XML

- Deux types d'approche :
  - L'approche hiérarchique : DOM
  - L'approche événementielle : SAX

```
<debut> Bienvenue </debut>
```

Start Element : debut
Start CDATA section, value : Bienvenue
Close Element : debut

XML

- Créer l'analyseur : *xml_create_parser()*
- Association aux 7 gestionnaires :
  - *xml_set_element_handler()*
  - *xml_set_character_data_handler()*
  - *xml_set_external_entity_ref_handler()*
  - *xml_set_unparsed_entity_decl_handler()*
  - *xml_set_processing_instruction_handler()*
  - *xml_set_notation_decl_handler()*
  - *xml_set_default_handler()*

XML

- *xml_set_element_handler()* :
  - `function ouverture ($parser, $name, $attrs)`
  - `function fermeture ($parser, $name)`
  - `xml_set_element_handler($xml_parseur, "ouverture", "fermeture");`
- *xml_set_character_data_handler()* :
  - `function texte ($parser, $data_text)`
  - `xml_set_character_data_handler($xml_parseur, "texte");`
- *xml_set_default_handler()* :
  - `function defaut ()`
  - `xml_set_default_handler($xml_parseur,"defaut");`


```
<?php
/** Ce code récupère le flux ATOM de Google News et récupère
   les titres des articles **/
# Fonctions de retour (callback)
include('lib.sax.php');
$ch =
   curl_init("http://news.google.fr/nwshp?hl=fr&tab=wn&output
   =atom");
# Nous voulons récupérer le contenu dans une variable
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
# Exécution, contenu dans la variable $atomik
$atomik=curl_exec($ch);
if(curl_errno($ch) != 0)  { echo "Erreur lors de la
   récupération du flux\n"; die(); }
curl_close($ch);
```


```
# Création du parser
$sax=xml_parser_create();
/** Définition des fonctions de retour (callback) **/
# 1. Les tags
xml_set_element_handler($sax, 'openTag', 'closeTag');
# 2. Le contenu
xml_set_default_handler($sax, 'parseContent');

# Go go parsing
xml_parse($sax, $atomik, true); // premier et dernier morceau
echo xml_error_string(xml_get_error_code($sax));
?>
```


```php
<?php
/** Fonctions de callback **/
$printContent=false; // est ce qu'on doit afficher le contenu des balises ?
function openTag($parser, $name, $attribs)
{
    global $printContent;
    // traitement en fonction du tag ouvert
    switch(strtolower($name))
    {
        // ceci étant un exemple, on va écrire directement
        // et ne traiter qu'un échantillon de tags
        case 'title':
            echo '<h2>'; $printContent=true; break;
        case 'subtitle':
            // ... résultat : http://chipo.oxileo.net/~francois/sax.php
    }
}
```


```
function closeTag($parser, $name)
{
    global $printContent;
    switch(strtolower($name))
    {
        case 'title':
            echo '</h2>'; $printContent=false; break;
        case 'subtitle':
            // ...
    }
}

function parseContent($parser, $data)
{
    global $printContent;
    if($printContent) echo $data;
}
?>
```

Programmation objet

- TD (*partie 4*) :
  - Ecrire un formulaire qui récupère les informations de la base de données pour les injecter dans un fichier xml.
  - Générer le fichier xml personnes.xml

AJAX

- **AJAX** est un acronyme signifiant Asynchronous JavaScript and XML
- Permet d'appeler des données depuis un client Web sur un serveur en asynchrone
- AJAX nécessite une architecture client/serveur Web
- Composants conformes aux standards du W3C.

AJAX

- AJAX n'est pas une technologie mais l'utilisation conjointe d'un ensemble de technologies
  - **HTML** (ou XHTML) pour la structure sémantique des informations ;
  - **CSS** ou **XSL** pour la présentation des informations ;
  - **DOM** et **Javascript** pour afficher et interagir dynamiquement avec l'information présentée ;
  - l'objet **XMLHttpRequest** pour échanger et manipuler les données de manière asynchrone avec le serveur Web.
  - **XML** pour le transfert de données

AJAX

- Récupérer uniquement les données nécessaires via HTTP **XMLHttpRequest**
- Requêtes peuvent être « asynchrones »
- Applications plus réactives, la quantité de données échangées entre le navigateur et le serveur HTTP étant fortement réduite
- Chargement de la première page peut être pénalisé si l'application utilise une bibliothèque AJAX volumineuse

CONCLUSION ET REF

- Langage pour développer des pages Web dynamiques
- Très complet, nombreuses bibliothèques, facile …
- Possibilités de développements multi-serveurs, web services, pair à pair, etc.
- Références et compléments :
  - http://www.laltruiste.com/
  - http://php.developpez.com/cours/
  - http://www.expreg.com/ereg.php
  - http://www.manuelphp.com/
  - http://g-rossolini.developpez.com/tutoriels/php/cours/
  - http://www.php-mysql-tutorial.com/