Technologie Web
PHP

Alexandre Pauchet

INSA Rouen - Département ASI

BO.B.RC.18, pauchet@insa-rouen.fr

Plan

Introduction (1/3)
Description

- PHP Hypertext Preprocessor
  - Langage de script intégré à HTML/XHTML, côté **serveur**
  - Le **serveur** parse les documents et interprète le code PHP
  - Le **client** reçoit uniquement le **résultat** du script (une page "générée")
  - Le code PHP est inclus dans des balises PHP : `<?php code-PHP ?>`
  - Voir : `http://www.php.net/`

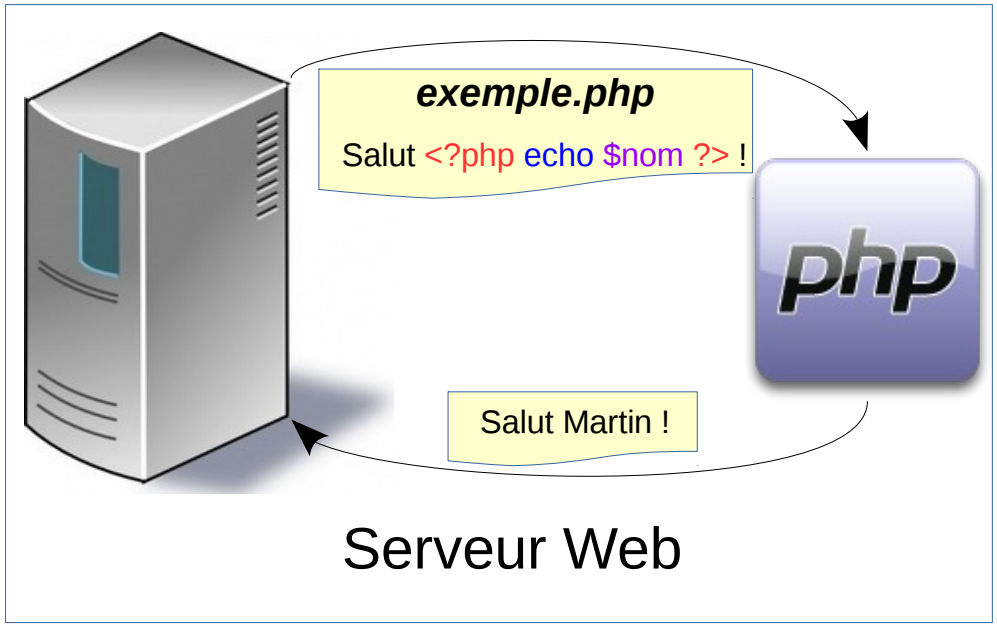
- Fonctionnalités diverses
  - Fonctionnalités équivalentes aux autres langages de scripts CGI
  - Support d'un important nombre de bases de données
  - Nombreuses librairies :
    - gestion des protocoles mail (imap, pop)
    - production de pdf, flash
    - gestion de XML
    - ...

Introduction (2/3)
Fonctionnement

Client Web
(navigateur)

exemple.php ?

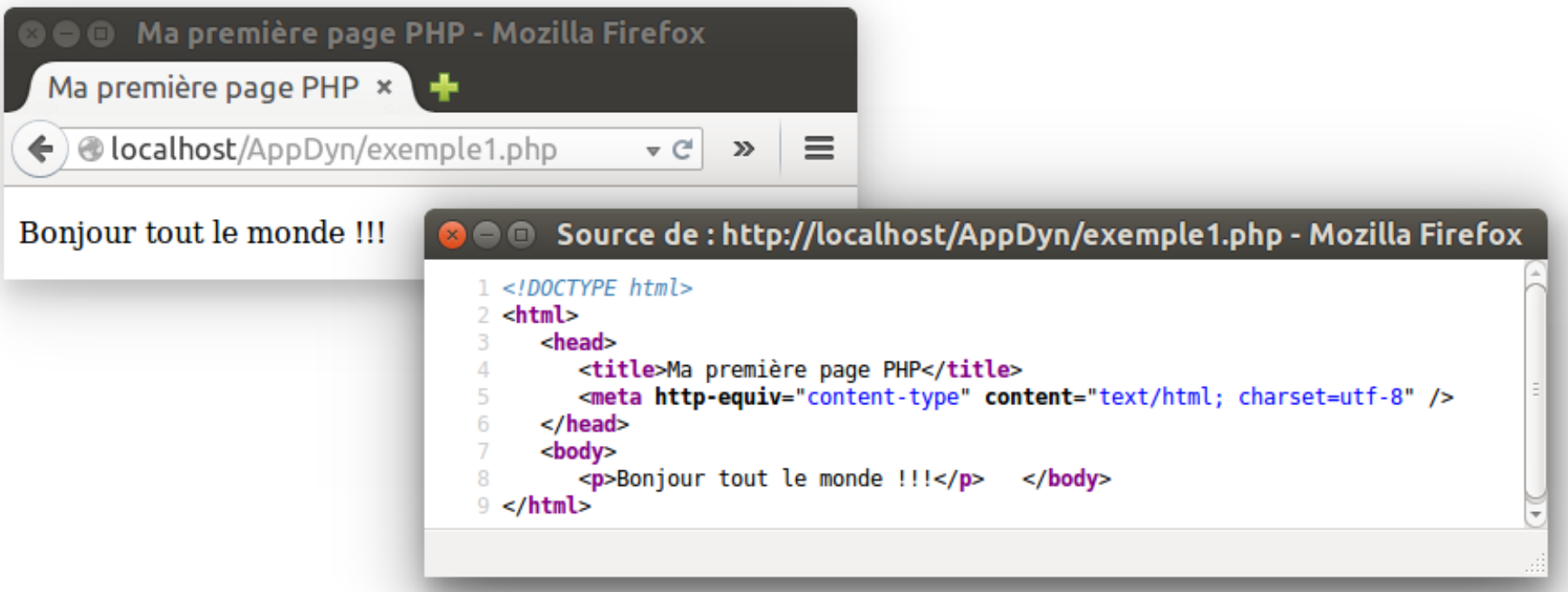
Salut Martin !



Serveur Web

Introduction (3/3)
Premier script PHP

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page PHP</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  </head>
  <body>
    <?php echo "<p>Bonjour le monde !!</p>\n"; ?>
  </body>
</html>
```



Syntaxe (1/14)
Syntaxe de base

Insertion d'une commande PHP :

```
<?php code PHP ?>
```

Séparateur d'instructions : le point virgule " ;"

```
<?php instruction1; instruction2; ... ?>
```

Commentaires : syntaxe à la C, C++ ou Shell

```
/* ... */
// ...
# ...
```

Syntaxe (2/14)
Les variables

Le typage des variables est dynamique

Syntaxe : `$NomDeVariable[=val];`

- règle de nommage : `$[a-zA-Z\_]([a-zA-Z0-9\_])*`
- sensibilité à la casse
- assignation par :
  - valeur : `$var1=$var2;`
  - référence : `$var1=&$var2;`
- Portée : locale à la fonction où elle est déclarée

Exemple :

```
<?php
   $var1=10;
   $var2=&$var1;
   $var1=20;
   echo "<p>".$var1." ".
        $var2."</p>";
?>
```

Affichera :

20 20

Syntaxe (3/14)
Variables globales

Déclaration de variable globale : `global $var;`

global.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page PHP</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  </head>
  <body>
    <?php
      function affiche() {
        global $var1;
        echo "<p>Hello ".$var1." !!!</p>\n";
      }
    ?>
    <?php
        $var1="John";
        affiche();
    ?>
  </body>
</html>
```

Syntaxe (4/14)
Variables superglobales

Les variables superglobales sont utilisables sans le mot clef `global`

Quelques tableaux superglobaux :

- `$_GLOBAL` contient des références sur les variables de l'environnement d'exécution global (clefs = noms des variables globales)
- `$_SERVER` variables fournies par le serveur web
- `$_GET` et `$_POST` variables transmises par les méthodes `GET` et `POST` du protocole HTTP
- `$_COOKIE`, `$_REQUEST`, `$_SESSION`, `$_FILES`, `$_ENV`

Syntaxe (5/14)
Les constantes

Syntaxe : `define("NOM_DE_LA_CONSTANTE", valeur)`

Les constantes :

- ne commencent pas par $
- sont définies et accessibles globalement dans tout le code
- ne peuvent pas être rédéfinies
- ne peuvent contenir que des booléens, des entiers, des flottants et des chaînes de caractères

Exemple

```
define("PHP","PHP Hypertext Preprocessor");
echo PHP;
```

Syntaxe (6/14)
Les types

4 types simples :

- entiers : `integer`
- réels : `float`, `double`
- booléens : `boolean` (`TRUE` ou `FALSE`)
- chaînes de caractères : `string`

2 types complexes :

- tableaux : `array`
- objets : `object`

2 types spéciaux :

- ressources : `resource` (ex : connexion BD)
- absence de valeur : `null`

Syntaxe (7/14)
Les tableaux

Principe : associations ordonnées de type *clef* $\Rightarrow$ *valeur*

Déclaration : `array( [clef =>] valeur, ...)`

- la clef est facultative, elle est de type entier ou chaîne de caractères ; *en cas d'omission, la valeur sera associée à la clef d'indice max+1*
- la valeur peut être de n'importe quel type

fruits.php

```php
$tab=array("fruit"=>"pomme",42,"légume"=>"salade",1.5e3);
foreach($tab as $cle => $valeur) {
  echo "<p>".$cle."=>".$valeur."</p>";
}
echo "<p>tab[1]=".$tab[1]."</p>";
$tab[]="peu importe";
echo "<p>tab[2]=".$tab[2]."</p>";
echo "<p>tab['fruit']=".$tab["fruit"]."</p>";
```

Syntaxe (8/14)
Opérations sur les tableaux

Attention, un tableau est toujours une référence !

- `count($array)` : nombre d'éléments
- `sort($array)` : trie le tableau
- `array_pop($array)` : récupère et supprime le dernier élément d'une liste (i.e. fonctionne comme une pile)
- `array_push($array, $elem1, ...)` : ajoute des éléments à la fin d'une liste (i.e. fonctionne comme une pile)
- `array_shift($array)` : récupère et supprime le premier élément d'une liste
- `array_unshift($array, $elem1, ...)` : ajout d'éléments en début de liste
- `array_merge($array1, $array2, ...)` : fusionne plusieurs tableaux
- `in_array($elem, $array)` : recherche d'un élément dans un tableau
- `array_key_exists($key, $array)` : recherche une clef dans un tableau
- `array_flip($array)` : inverse les clef et les valeurs d'un tableau

Syntaxe (9/14)
Détermination du type d'une variable

Type d'une variable : `string gettype($var);`

Test : `is_integer($var); is_double($var); is_array($var); ...`

Conversion dynamique : `$result = (type-désiré)$var;`

Instructions de vérification d'existence (formulaires) :

- `boolean isset($var);` retourne `FALSE` si `$var` n'est pas initialisée ou a la valeur `NULL`, `TRUE` sinon ;
- `boolean empty($var);` retourne `TRUE` si `$var` n'est pas initialisée, a la valeur `0`, `"0"`, ou `NULL`, `FALSE` sinon ;
- `boolean is_null($var);` retourne `TRUE` si `$var` n'est pas initialisée ou vaut `NULL`, `FALSE` sinon.

Syntaxe (10/14)
Exemple

val	gettype()	empty()	is_null()	isSet()	(bool)
$x = "";	string	true	false	true	false
$x = null;	NULL	true	true	false	false
var $x ; (not set)	NULL	true	true	false	false
$x = array();	Array	true	false	true	false
$x = false;	boolean	true	false	true	false
$x = 15;	integer	false	false	true	true
$x = 1;	integer	false	false	true	true
$x = 0;	integer	true	false	true	false
$x = -1;	integer	false	false	true	true !
$x = "15";	string	false	false	true	true
$x = "1";	string	false	false	true	true
$x = "0";	string	true	false	true	false !
$x = "-1";	string	false	false	true	true
$x = "foo";	string	false	false	true	true
$x = "true";	string	false	false	true	true
$x = "false";	string	false	false	true	true !

Syntaxe (11/14)
Opérateurs

Opérateurs identiques à ceux du C/C++/Java :

- opérateurs arithmétiques : `+ - * / %`
- in/décrémentation : `var++ var-- ++var --var`
- opérateurs logiques : `&& || !`
- comparaisons : `== != <= >= <>`
- concaténation de chaînes de caractères : `.`
- affectation : `= += -= *= ...`

Opérateurs spécifiques :

- `` `commande shell` `` (*ex* : `$a=`ls -ul``)
- `===` : teste la valeur et le type

Syntaxe (12/14)
Instructions de branchement

Proches du C/C++/Java :

Si-sinon-alors :

```
if(condition) {
  instructions
}
[elseif(condition) {
  instructions
}]
[else {
  instructions
}]
```

Switch-case :

```
switch(expression) {
  case 'valeur1':
    Instructions
    break;
  ...
  default:
    Instructions
    break;
}
```

Syntaxe (13/14)
Boucles

Proches du C/C++/Java :

Boucles for :

```
for($i=0; $i<N; $i++) {
  Instructions
}

foreach($tab as $val) {
  Instructions
}

foreach($tab as $cle=>$val) {
  Instructions
}
```

Boucles while :

```
while(condition) {
  Instructions
}

do {
  Instructions
} while(condition);
```

Syntaxe (14/14)
Répétition de code HTML

- Utilisation des boucles pour répéter du code HTML : Entrelacement code PHP / code HTML

repetitionHTML.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <title>Page PHP</title>
  </head>
  <body>
    <?php
      $tab=array("premier","second","troisième","...");
    ?>
    <ul>
      <?php foreach($tab as $elem) { ?>
        <li><?php echo $elem; ?></li>
      <?php } ?>
    </ul>
  </body>
</html>
```

Fonctions et modularité (1/7)
Les fonctions

Syntaxe

```
<?php
function nomDeFonction(arg1, arg2, ..., argN)[: type] {
        instructions;
        [return VALEUR;]
}
?>
```

- les noms de fonctions sont insensibles à la casse
- une fonction peut être utilisée avant sa définition
- la valeur en retour d'une fonction peut être fixée (PHP7) ; elle sera transtypée si nécessaire
- les arguments sont non typés et supportent une valeur par défaut
- la surcharge de fonctions n'est pas supportée
- passage d'arguments par valeur et référence supporté (&)
- les fonctions supportent un nombre variable d'arguments
- retour d'une unique valeur par la directive `return`

Fonctions et modularité (2/7)
Exemples de fonction

fonctions-math.php

```php
<?php
  function puissance($nombre, $exposant=1): float {
    $res = 1;
    for($i=abs($exposant)-1; $i>=0; $i--)
      $res = $res * $nombre;
    if($exposant > 0)
      $retour = $res;
    else
      $retour = 1/$res;
    return  $retour;
  }
  function incrementer(&$nombre, $increment=1) {
    $nombre += $increment;
  }
$val = 4;
incrementer($val, 2);
echo "val = ".$val."</p>";
echo "<p>puissance(2,-2) = ".puissance(2,-2)."</p>";
$fonction = 'puissance';
echo "<p>$fonction(2,4) = ".$fonction(2,4)."</p>";
 ?>
```

Fonctions et modularité (3/7)
Exemples de fonction

fonctions.php

```php
<?php
header("Content-Type: text/plain; charset=UTF-8");

function formatterUneListeDeMessages($listeDeMessages, $listeAuteurs) {
  $chaine = "";
  foreach($listeDeMessages as $id => $mess) {
    foreach($listeAuteurs as $nom => $ids) {
      if(in_array($id, $ids))
        $chaine = $chaine . $nom." : ".$mess."\n";
    }
  }
  return $chaine;
}

function ajouterMessages(&$listeDeMessages, &$listeAuteurs, $messages) {
  foreach($messages as $id => $message) {
    array_push($listeDeMessages, $message[1]);
    if(array_key_exists($message[0], $listeAuteurs)){
      array_push($listeAuteurs[$message[0]], count($listeDeMessages)-1);
    } else {
      $listeAuteurs[$message[0]] = array(count($listeDeMessages)-1);
    }
  }
}


...
```

Fonctions et modularité (4/7)
Exemples de fonction

fonctions.php

```
...
$messages = array();
$auteurs = array();
$messagesRecus = array(array("Bob", "Salut"),
                       array("Samantha", "Tiens ? Ca faisait longtemps !"),
                       array("Rencontres.com", "Salut Bob ! Rdv sur rencontres.com"),
                       array("Bob", "Oui. Quoi de neuf, Samantha ?"),
                       array("Samantha", "Rien de particulier..."));

ajouterMessages($messages, $auteurs, $messagesRecus);
echo "Liste de messages :\n" . formatterUneListeDeMessages($messages, $auteurs);

// Affiche "Liste de messages :
//          Bob : Salut
//          Samantha : Tiens ? Ca faisait longtemps !
//          Rencontres.com : Salut Bob ! Rdv sur rencontres.com
//          Bob : Oui. Quoi de neuf, Samantha ?
//          Samantha : Rien de particulier..."
?>
```

Fonctions et modularité (5/7)
Nombre variable d'arguments

fonction-nbArgsVariable.php

```php
<?php
header("Content-Type: text/plain; charset=UTF-8");

function f($req, $opt=null, ...$params) {
  echo "req: $req; opt: $opt; nb args: " . count($params) . "\n";
}

f(1);              // req: 1; opt: ; nb args: 0
f(1, 2);           // req: 1; opt: 2; nb args: 0
f(1, 2, 3);        // req: 1; opt: 2; nb args: 1
f(1, 2, 3, 4);     // req: 1; opt: 2; nb args: 2
f(1, 2, 3, 4, 5); // req: 1; opt: 2; nb args: 3

?>
```

Fonctions et modularité (6/7)

Inclusion de code externe

2 directives :

- `include_once("fichier");` remplacement par le contenu du fichier
- `require_once("fichier");` idem, mais arrêt du script en cas d'erreur (*ex* : absence de fichier)

inclusion.php

```
1  <!DOCTYPE html>
2  <html>
3     <head>
4      <title>Page PHP</title>
5      <meta http-equiv="content-type" content="text/html;charset=utf-8" />
6      <meta http-equiv="Content-Style-Type" content="text/css" />
7    </head>
8    <body>
9      <?php include_once("Entete.inc.php"); ?>
10     <p>contenu normal de la page</p>
11     <p align="right">
12       <i><?php include_once("Pieddepage.inc.php"); ?></i>
13     </p>
14   </body>
15 </html>
```

Fonctions et modularité (7/7)
Conventions de nommage des fichiers

Attention

Les fichiers dont l'extension n'est pas `.php` ne sont pas parsés, et donc directement lisible par une requête HTTP

- Bonne pratique : .inc.php
- Bonne pratique : .conf.php
- Bonne pratique : .class.php

- Mauvaise pratique (par exemple) : `.inc`

Chaînes de caractères (1/9)
Déclaration et fonctionnement

- Les chaînes peuvent être déclarées avec :
  - Simples quotes : `$t='texte';`
  - Doubles quotes : `$t="texte";`
- Fonctionnement différent : entre doubles quotes, les variables et les caractères échappatoires sont interprétés

Exemples, pour `$t="Mot";`

Double cotes	Résultat	Simple cote	Résultat
`"Texte"`	Texte	`'Texte'`	Texte
`"Texte $t"`	Texte Mot	`'Texte $t'`	Texte $t
`"Texte \n Suite"`	Texte Suite	`'Texte \n Suite'`	Texte \n Suite
`"Texte \"guillemets\""`	Texte "guillemets"	`'Texte \"guillemets"\'`	Texte \"guillemets"\
`"L'heure"`	L'heure	`'L\'heure'`	L'heure

Chaînes de caractères (2/9)
Opérations sur les chaînes de caractères

- Longueur : `int strlen(string $ch)`
- Répétition : `string str_repeat(string $cr, int nb)`
- Minuscules : `string strtolower(string $ch)`
- Majuscules : `string strtoupper(string $ch)`
- Initiales en majuscules : `string ucwords(string $ch)`
- 1^{ere} lettre en majuscule : `string ucfirst(string $ch)`
- Suppression de caractères en début de chaîne :
  `string ltrim(string $ch, string liste)`
- Suppression de caractères en fin de chaîne :
  `string rtrim(string $ch, string liste)`
- Suppression de caractères en début et fin de chaîne :
  `string trim(string $ch, string liste)`

Chaînes de caractères (3/9)
Exemple

traitementString.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <title>Page PHP</title>
  </head>
  <body>
    <?php
      $prenom = "...JEan__";
      $nom = "  BONNEAu";
      $adresse = "10 rue abraham lincoln";
      $email = "jean-BONNEAU@asi.insa-rouen.fr";

      $complet = ucfirst(strtolower(trim($prenom,'._')));
      $complet .= " ".strtoupper(ltrim($nom,' '));
      $espaces = strlen($complet)+3;
      echo $complet." : ".ucwords($adresse)."<br />";
      echo str_repeat(".", $espaces).strtolower($email);
    ?>
  </body>
</html>
```

Chaînes de caractères (4/9)

Sous-chaînes de caractères

- Recherche sensible à la casse (retourne tous les caractères de `$ch` depuis la 1^{ere} occurence de `$ch2` jusqu'à la fin) :
  `string strstr(string $ch, string $ch2)`
- Recherche insensible à la casse :
  `string stristr(string $ch, string $ch2)`
- Extraction de chaînes de caractères :
  `string substr(string $chr, int indice, int N)`
- Décompte du nombre d'occurences d'une sous-chaîne :
  `int substr_count(string $ch, string $ssch)`
- Remplacement :
  `string str_replace(string $oldssch, string $newssch, string $ch)`
- Position : `int strpos(string $ch, string $ssch)`

Chaînes de caractères (5/9)
Exemples

traitementString2.php

```
<?php
  $ch = "Un pot de lait et un pot de miel";

  echo strstr($ch, "pot")."<br />";
        // affiche "pot de lait et un pot de miel"

  echo substr($ch, 18, 6)."<br />";
        // affiche "un pot"

  echo substr_count($ch, "pot")."<br />";
        // affiche "2"

  echo str_replace("pot", "broc", $ch)."<br />";
        // affiche "Un broc de lait et un broc de miel"

  echo strpos($ch, "un pot")."<br />";
        // affiche "18"
?>
</body>
</html>
```

Chaînes de caractères (6/9)

Les expressions rationnelles

Une **expression rationnelle** (RegEx) permet de définir un *motif* de caractères, représentatif d'un ensemble de chaînes de caractères.

- Caractère(s) : `""` ou `' '` (ex : `"a"`, `"ab"`)
- Caractères spéciaux : `\.`, `\$`, `\^` , `\?`, `\\`, `\[`, `\]`, `\(`, `\)`, `\+` et `\*`
- Classe de caractères : `[]` (ex : `[xyz]`, `[a-z]`)
- Classes de caractères prédéfinies :
  - `[[:alnum:]]` : caractères alphanumériques
  - `[[:alpha:]]` : caractères alphabétiques
  - `[[:ctrl:]]` : caractères de contrôle
  - `[[:digit:]]` : chiffres
  - `[[:punct:]]` : caractères de punctuation
  - `[[:space:]]` : caractères d'espaces
  - `[[:upper:]]` : majuscules

Chaînes de caractères (7/9)

Modèles généraux

- N'importe quel caractère : `.`
- 0 ou 1 fois : `?` (ex : `"https?"`)
- Au moins une fois : `+`
- 0, 1 ou plusieurs fois : `*` (ex : `"mat.*"`)
- Exactement n fois : `"{n}"`
- Au moins n fois : `"{n,}"`
- Entre n et m fois : `"{n,m}"`
- Groupements : `()` (ex : `"(ma)*"`)
- Alternative : `|` (ex : `"(\.net)|(\.com)"`)

Exemples

```
"[[:digit:]]{2}/[[:digit:]]{2}/[[:digit:]]{4}"
"[[:alnum:]]*\.[[:alnum:]]*@asi\.insa-rouen\.fr
```

Chaînes de caractères (8/9)
Fonctions de recherche et de remplacement

- Fonctions de recherche

```
int preg_match ( string $modeleregex , string $chaine
    [, array &$matches ] )
```

- Fonctions de remplacement

```
mixed preg_replace ( mixed $modeleregex , mixed
    $replacement , mixed $chaine )
```

Remarque

Le modèle d'une expression rationnelle est déclarée entre / /, à l'intérieur d'une chaîne de caractères.

Ex : `"/ftp:\/\/.*:.*/"`

Chaînes de caractères (9/9)
Exemple

expressionsRationelles.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page PHP</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  </head>
  <body>
    <?php
      $chaine = "La dernière version est :\nPHP 4 (4ème version)\nVive le PHP 4
          ";
      $chaine = preg_replace("/4/", "5", $chaine);
      $chaine = preg_replace("/\n/", "<br />", $chaine);
      echo $chaine;
    ?>
  </body>
</html>
```

Les fichiers (1/3)
Ouverture des fichiers

- **Ouverture :** `$fichier=fopen("NOM", "MODE");` avec `MODE` valant :
  - `r`, `r+` : lecture et lecture/écriture, pointeur au début
  - `w`, `w+` : écriture et lecture/écriture, avec création ou effacement, pointeur au début
  - `a`, `a+` : écriture et lecture/écriture, pointeur à la fin, avec création
  - `x`, `x+` : création en écriture et lecture/écriture, pointeur au début, erreur en cas d'existence du fichier
  - `c`, `c+` : création en écriture et lecture/écriture, pointeur au début, sans erreur
- **Verrouillage d'un fichier :**

  ```
  bool flock($fichier, int $operation)
  ```
- **Fermeture :** `fclose($fichier);`

Les fichiers (2/3)
Gestion des fichiers

- **Lecture d'une ligne :**
  `string fgets($fichier [, integer nbOctets])`
- **Lecture d'un caractère :**
  `string fgetc($fichier)`
- **Ecriture d'une ligne :**
  `integer fputs($fichier, string)`
- **Test de fin de fichier :**
  `boolean feof($fichier)`
- **Positionnement :** `fseek($fichier, int $position);`

Les fichiers (3/3)
Exemple

fichier.php

```
<!DOCTYPE html>
<html>
  <head>
        <title>Lecture/Ecriture dans un fichier</title>
        <meta http-equiv="Content-type" content="text/html; charset=utf-8"/>
  </head>
     <?php
        $fichier=fopen("fichier.txt", "a");
        fputs($fichier, "Une phrase\n");
        fclose($fichier);

        $fichier=fopen("fichier.txt", "r");
        echo "<p>Dans le fichier fichier.txt :</p>";
        while(!feof($fichier)){
          echo fgetc($fichier);
        }
        fclose($fichier);
     ?>
</html>
```

Les formulaires (1/3)
GET/POST

Les champs d'un formulaire sont disponibles à travers les variables superglobales `$_GET` et `$_POST`

Remarques :

- Si le submit est une image, les coordonnées du click sont transmises via `$sub_x` et `$sub_y`
- Il est possible d'utiliser des tableaux à une dimension pour des formulaires ayant par exemple des listes à choix multiples
- Les caractères spéciaux (&, ", ', <, >) doivent être transformés en caractères interprétables par le navigateur :

```
string htmlentities(string [,int $flags])
string html_entity_decode(string [,int $flags])
```

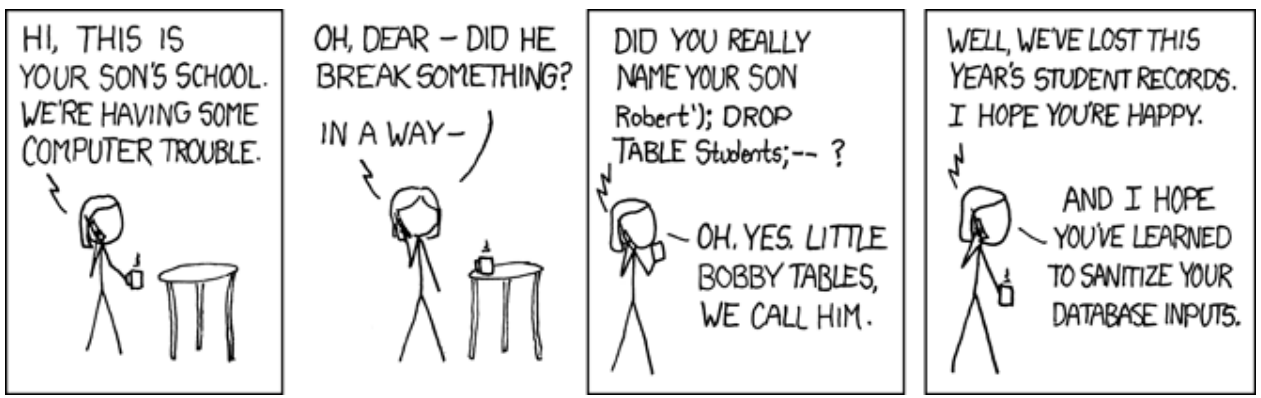
- La balise HTML `<pre>...</pre>` peut être utilisée pour interpréter les espaces, tabulations et sauts de ligne.

Les formulaires (2/3)

Utilité de Htmlentities : contrer les attaques de type XSS

htmlentities.php

```php
<?php
  $phrase = "J'aime le <strong>gras</strong> !";
  $a = htmlentities($phrase);
  $b = html_entity_decode($a);
  echo $phrase . "<br/>"; // J'aime le gras ! (avec "gras" en gras)
  echo $a . "<br/>";      // J'aime le <strong>gras</strong> !
  echo $b;                // J'aime le gras ! (avec "gras" en gras)
?>
```



Tiré de `http://xkcd.com/327/`.

Les formulaires (3/3)
Exemple de traitement de formulaire

formulaire.php

```
<form action="formulaire.php" method="POST">
  <label>Prenom</label>     <input type="text" name="prenom" size="10" value="Alex"/>
  <label>Nom</label>        <input type="text" name="nom" size="20" value="P"/><br/>
  <label>OS : Unix</label>  <input type="radio" name="os" value="unix"/>
  <label>OS/2</label>       <input type="radio" name="os" value="os/2"/>
  <label>Windows</label>    <input type="radio" name="os" value="windows"/><br/>
  <input type="text" name="ligne" value="<script type='text/javascript'>alert('Pub
       !');</script>" size="20"/><br/>
  <input type="submit" name="action" value="Envoyer"/>
  <input type="reset" value="Effacer"/>
</form>
<p>
  <?php
    if(isset($_POST['action']) && !empty($_POST['prenom']) && !empty($_POST['nom'])
         && !empty($_POST['os']) && !empty($_POST['ligne'])) {
      echo $_POST['prenom']." ".$_POST['nom']." utilise un système d'exploitation ".
           $_POST['os']."<br/>";
      echo "<p>htmlentities(Ligne)-> ".htmlentities($_POST['ligne'])."</p>";
      echo "<p>Ligne-> ".$_POST['ligne']."</p>";
    }
    else
      echo "<p>Tous les champs doivent être renseignés</p>";
  ?>
</p>
```

Les classes (1/18)
Déclaration

Syntaxe

```
class NomClasse {
  public/protected/private $attribut1[ = constante1];
  public/protected/private $attribut2[ = constante2];
  ...
  public/protected/private function __construct(...) { // constructeur
  }

  public/protected/private function methode1(...) { // méthode
  }

  public/protected/private function methode2(...) { // méthode
  }
  ...
}
```

- Seules les initialisations par constante sont autorisées
- Encapsulation/Accessibilité :
  - `public` : accès universel
  - `protected` : accès réservé à la classe et aux classes dérivées
  - `private` : accès réservé à la classe

Les classes (2/18)
Attributs et méthodes

- Instanciation : `$objet = new NomClasse(...)`
- Accès aux attributs et méthodes par l'opérateur "->"

  `$objet->attribut` / `$objet->methode()`
- L'accès aux attributs dans les méthodes se fait par `$this->attribut`
- La surcharge des méthodes dans une même classe n'est pas possible, mais la redéfinition dans une classe fille l'est

Exemple (objet.php)

```
class Acces
{
  public $varPublique = "variable publique";
  protected $varProtegee = "variable protégée";
  private $varPrivee = "variable privée";
  public function lecturePublique()
    { echo "<p>Fonction publique</p>"; }
  protected function lectureProtegee()
    { echo "<p>Fonction protégée</p>"; }
  private function lecturePrivee()
    { echo "<p>Fonction privée</p>"; }
}
$acces = new Acces();
echo "varPublic : $acces->varPublique";
$acces->lecturePublique();
```

Les classes (3/18)
Constructeurs et destructeurs

- PHP5 permet les constructeurs unifiés et les destructeurs :

```
void __construct([arguments]) {...}
void __destruct() {...}
```

- En cas d'héritage, appel explicite du constructeur/destructeur de la classe mère dans le corps du constructeur de la classe fille :

```
parent::__construct([arguments]);
parent::__destruct();
```

Les classes (4/18)
Héritage

Syntaxe

```
class ClassDerivée extends ClasseHéritée {
  ...
}
```

- si une classe dérivée n'a pas de constructeur, celui de la classe mère est appelé
- la propagation d'appel des constructeurs n'est pas automatique
- l'opérateur de résolution portée est "::"
- `parent` est un mot clef permettant l'accès à la classe mère

Les classes (5/18)
Exemple d'héritage

Personne.class.php

```php
<?php
  class Personne {
    protected $nom;

    public function __construct($nom) {
      $this->nom = $nom;
    }

    public function info() {
      return $this->nom;
    }
  }
?>
```

Les classes (6/18)
Exemple d'héritage

Etudiant.class.php

```php
<?php
  class Etudiant extends Personne {
    public $numero;

    function __construct($num, $nom) {
      parent::__construct($nom);
      $this->numero = $num;
    }

    public function info() {
      return $this->nom."(".$this->numero.")";
    }
  }
?>
```

Les classes (7/18)
Exemple d'héritage

heritage.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <title>Page PHP</title>
    <?php require("Personne.class.php"); ?>
    <?php require("Etudiant.class.php"); ?>
  </head>
  <body>
    <?php
      $personne = new Personne("John");
      echo "<p>Personne : ".$personne->info()."</p>";

      $etudiant = new Etudiant(1203, "Samantha");
      echo "<p>Etudiant : ".$etudiant->info()."</p>";
    ?>
  </body>
</html>
```

Les classes (8/18)
Les classes abstraites

- Classes abstraites en PHP5
  - PHP5 permet la création de **classes abstraites**, ne permettant pas l'instanciation d'objets mais servant de classe de base pour la création de classes dérivées.
  - `abstract` sert à déclarer les méthodes et classes abstraites.

Les classes (9/18)
Exemple de classe abstraite

compte.php

```
<?php
  abstract class Compte {
    protected $nom;
    protected $solde;
    public function __construct($nom, $solde) {
      $this->nom = $nom;
      $this->solde = $solde;
    }
    abstract protected function getInfo();
  }

  class CompteCheque extends Compte{
    // NB : inutile quand il y a le même nombre d'arguments
    public function __construct($nom, $solde) {
      parent::__construct($nom, $solde);
    }
    public function getInfo() {
      return "Compte cheque de ".$this->nom." : ".$this->solde;
    }
  }
?>
</head>
<body>
  <?php
    $ccq = new CompteCheque("John", 1000);
    echo "<p>".$ccq->getInfo()."</p>";
  ?>
```

Les classes (10/18)
Les interfaces

- Interfaces en PHP5
  - déclarées par le mot clef `interface`
  - ne contenant aucune déclaration d'attribut
  - ne contenant aucune implémentation de méthode
  - dont les déclarations de méthodes sont `public`
  - implémentées par une classe par `implements` ; une classe peut implémenter plusieurs interfaces.

Les classes (11/18)
Exemple d'interface

interface.php

```
    <?php
      interface Fonction {
        public function calculer();
      }

      class Addition implements Fonction {
        private $var1;
        private $var2;
        public function __construct($var1, $var2) {
          $this->var1 = $var1;
          $this->var2 = $var2;
        }
        public function calculer() {
          return $this->var1+$this->var2;
        }
      }
    ?>
  </head>
  <body>
    <?php
      $addition = new Addition(10, 20);
      echo "<p>addition(10,20) = ".$addition->calculer()."</p>";
    ?>
  </body>
</html>
```

Les classes (12/18)
Méthodes et classes finales

En PHP5 il est possible d'empêcher toute redéfinition de méthode ou de classe, par le mot clef `final`.

```
class UneClasse {
  final function methode()
    { ... }
}
```

empêche toute redéfinition de `methode()` dans les classes dérivées.

```
final class UneClasse {
  ...
}
```

empêche toute dérivation de la classe `uneClasse`.

Les classes (13/18)
Clonage d'objet

- Le clonage d'objet est possible en PHP5
  - Les opérateurs d'affectation = et de référence & permettent de copier un objet, mais les modifications sur la copie sont répercutées sur l'original.
  - Pour éviter cela, il faut **cloner** les objets par :

```
$objetclone = clone $objet;
```

  - Pour modifier les propriétés de l'objet cloné, il faut définir la méthode prédéfinie `__clone()`.

Les classes (14/18)
Exemple de clonage

Clonage.inc.php

```php
<?php
  class Personne {
    protected $nom;
    public function __construct($nom) {
      $this->nom = $nom;
    }
    public function info() {
      return $this->nom;
    }
    public function __clone() {
      $this->nom = "Clone de ".$this->nom;
    }
  }
?>
```

Les classes (15/18)
Exemple de clonage

Etudiant.class.php

```php
<?php
  class Etudiant extends Personne {
    public $numero;

    function __construct($num, $nom) {
      parent::__construct($nom);
      $this->numero = $num;
    }

    public function info() {
      return $this->nom."(".$this->numero.")";
    }
  }
?>
```

Les classes (16/18)
Exemple de clonage

clonage.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page PHP : clonage</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <?php require("Clonage.inc.php"); ?>
    <?php require("Etudiant.class.php"); ?>
  </head>
  <body>
    <?php
      $personne = new Personne("John");
      $clone = clone $personne;
      echo "<p>personne : ".$personne->info()."</p>";
      echo "<p>clone : ".$clone->info()."</p>";

      $etudiant = new Etudiant(1235, "Samantha");
      $etudiant2 = $etudiant;
      $clone = clone $etudiant;
      $etudiant->numero = 55555;
      echo "<p>etudiant : ".$etudiant->info()."</p>";
      echo "<p>etudiant2 : ".$etudiant2->info()."</p>";
      echo "<p>clone : ".$clone->info()."</p>";
    ?>
  </body>
</html>
```

Les classes (17/18)
Attributs et méthodes statiques (PHP5)

- Déclaration par le mot clef `static`
- Accès par '`nomclasse::`'

statique.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <title>Page PHP</title>
    <?php
      class Statique
      {
        public static $varStatique = "variable statique";
        public static function fonctionStatique()
          { echo "<p>Fonction statique : ".Statique::$varStatique."</p>"; }
      }
    ?>
  </head>
  <body>
    <?php
      Statique::fonctionStatique();
      Statique::$varStatique = "var. stat.";
      Statique::fonctionStatique();
    ?>
  </body>
</html>
```

Les classes (18/18)
Divers

- Méthode `_toString():string` : permet de formater l'affichage d'une instance de la classe.

- Sérialisation d'objets : Pour faciliter l'enregistrement d'objets dans des fichiers il est conseillé d'utiliser les méthodes `serialize` et `unserialize` pour en enregistrer une représentation linéaire.

Les exceptions (1/4)

La classe `Exception`

- PHP5 introduit la classe prédéfinie `Exception`
  - 2 attributs :
    - `message` : message d'erreur (`string`)
    - `code` : code d'erreur facultatif (`int`)
  - Méthodes :
    - `getMessage()` : accesseur sur le message de l'objet
    - `getCode()` : accesseur sur le code d'erreur de l'objet
    - `getFile()` : retourne le fichier contenant l'erreur
    - `getLine()` : retourne la ligne renvoyant l'exception
    - `__toString()` : retourne une chaîne descriptive de l'exception

Les exceptions (2/4)
Récupération des exceptions

Code type

```php
try {
  // Code à surveiller
  // Ce code peut renvoyer des exceptions non de votre
      fait
  if(erreur prevue) {
    throw new Exception();
  }
  else {
    // Résultat;
  }
}
catch(Exception $except) {
  // Gestion des erreurs
}
```

Les exceptions (3/4)
Personnalisation

- Héritage d'exception : le mécanisme de l'*héritage* permet d'étendre la classe `Exception`

```
class MonException extends Exception
{
  public function alerte($mess) {
    echo "<script type='text/javascript'> alert('
        Erreur n°".$this->getCode()."\n".$this->
        getMessage()."\n".$mess."') </script>";
  }
}
```

Les exceptions (4/4)
Exemple

exception.php

```
<head>
  <title>Page PHP</title>
  <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  <?php
    class MonException extends Exception
    {
      public function alerte() {
        echo "<script type=\"text/javascript\"> alert('Erreur ".$this->
            getCode()."\\n".$this->getMessage()."')</script>";
      }
    }
  ?>
</head>
<body>
  <?php
    $num = 100; $denom = 0;
    try {
      if($denom==0) { throw new MonException("Division par 0"); }
      else { echo "$num/$denom = ",$num/$denom; }
    }
    catch(MonException $except) {
      $except->alerte();
    }
  ?>
</body>
```

Sessions et authentification (1/5)
Session

- **La gestion de session permet de stocker des données entre les différentes pages visitées**
  - `session_start()` : crée ou restaure une session
    - À mettre obligatoirement avant tout envoi d'en-tête
    - À mettre dans toute page participant à une session
  - la variable superglobale `$_SESSION` permet l'enregistrement de variables dans la session :
    - `$_SESSION["`*`nomdelavariable`*`"]=`x ajoute ou modifie une variable à/de la session en cours
    - `$_SESSION["`*`nomdelavariable`*`"]` accède à la valeur d'une variable de la session en cours
    - `unset($_SESSION["`*`nomdelavariable`*`"])` retire la variable
  - `session_destroy()` détruit la session en cours

Remarque

L'utilisation d'une session ne nécessite pas forcément d'authentification.

Sessions et authentification (2/5)
Authentification

- 2 méthodes d'authentification
  - par HTTP : en utilisant le système d'identification, et la capacité des navigateurs d'ouvrir une fenêtre de connexion

```php
<?php
  if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="connexion"');
    header('HTTP/1.1 401 Unauthorized');
    echo "Authentification requise";
    exit;
  } else {
    // vérification du pass
    echo "<p>login: {$_SERVER['PHP_AUTH_USER']}</p>";
    echo "<p>password: {$_SERVER['PHP_AUTH_PW']}</p>";
  }
?>
```

  - par un formulaire pour saisir pseudo et mot de passe

- Une fois le login et mot de passe récupérés, il faut vérifier si l'utilisateur a le droit de se connecter. Les infos peuvent être stockées dans un fichier simple, un fichier du type `.htaccess`, ou dans une base de données.

Sessions et authentification (3/5)

Exemple d'authentification et de session 1/3

authentification-form.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>authentification-form</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  </head>
  <body>

    <form action="authentification-page1.php" method="POST">
      <label>Username : </label> <input type="text" name="login" size="20"/>
      <label>Password : </label>    <input type="password" name="password" size
          ="20"/>
      <input type="submit" name="submit" value="Login"/>
    </form>
  </body>
</html>
```

Sessions et authentification (4/5)

Exemple d'authentification et de session 2/3

authentification-page1.php

```php
<?php require("login.inc.php") ?>
<!DOCTYPE html>
<html>
  <head>
    <title>authentification-page1</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <meta http-equiv="Content-Style-Type" content="text/css" />
  </head>
  <body>
    <?php
      $login = $_POST['login']; $id = $_SESSION['userId'];
      echo "$login - $id";
    ?>
    <h1> page 1 </h1>
    <a href="authentification-page2.php">page2</a>
    <a href="logout.php">logout</a>
  </body>
</html>
```

authentification-page2.php

idem (changement des "1" en "2" et inversement)

Sessions et authentification (5/5)

Exemple d'authentification et de session 3/3

login.inc.php

```php
<?php
session_start();
if ( isset($_POST["submit"]) ) {
  // recherche de l'utilisateur : fichier plat, htaccess ou bd
  $loginAutorise="alex";
  $passwordAutorise="andre";
  $userIdAutorise="1";
  if ( $_POST['login'] == $loginAutorise && $_POST['password'] ==
       $passwordAutorise )
    $_SESSION["userId"] = $userIdAutorise;
  else header("Location: ./authentification-form.php");
}
  elseif(!isset($_SESSION["userId"])) header("Location: ./authentification-form
       .php");
?>
```

logout.php

```php
<?php
require("login.inc.php");
session_destroy();
header("Location: ./authentification-form.php");
?>
```

PHP et XML (1/6)
Lecture d'un fichier XML

- L'extension SimpleXML (PHP5) fournit les fonctions permettant
  - d'accéder au contenu d'un fichier XML :
    `simplexml_element simplexml_load_file(string $fileName)`
  - de lire des éléments :
    `simplexml_element->element`
    `simplexml_element->elementMultiple[x]`
  - de lire les attributs des éléments
    - Si le nom de l'attribut est connu :
      `$xml->element["nomAttribut"]`
    - Si le nom de l'attribut est inconnu :
      `simplexml_element xmlElement->attributes()`

PHP et XML (2/6)
Fichier XML

bib.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<biblio>
   <livre editeur="Bob" prix="5.00">
    <auteur>John Aipu</auteur>
    <titre>Il était une fois</titre>
  </livre>
  <livre version="3.05" prix="2.50">
    <auteur>Yann Napu</auteur>
    <titre>La suite</titre>
  </livre>
</biblio>
```

PHP et XML (3/6)

Exemple de lecture des éléments

XML1.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <title>Page PHP</title>
  </head>
  <body>
    <?php
      $xml = simplexml_load_file("./data/bib.xml");
      $livre = $xml->livre[1]->titre." ";
      $livre.="(".$xml->livre[1]->auteur.")";
      echo "2nd livre : ".$livre."<br />";
      echo "liste des titres :<br />";
      $i = 1;
      foreach($xml->livre as $cle=>$val) {
        echo $cle." ".$i." : ".$val->titre."<br />";
        $i++;
      }
    ?>
  </body>
</html>
```

PHP et XML (4/6)
Exemple de lecture d'attributs

XML2.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <title>Page PHP</title>
  </head>
  <body>
    <?php
      $xml = simplexml_load_file("data/bib.xml");
      foreach($xml->livre as $livre) {
        $i = 0;
        echo "Livre ".$i." : ";
        foreach($livre->attributes() as $att=>$val){
          echo "$att=$val ";
        }
        echo "<br />";
        $i++;
      }
    ?>
  </body>
</html>
```

PHP et XML (5/6)

Modification et enregistrement des données

- Modification d'un fichier XML
  - Les données (attributs et éléments) peuvent être modifiées :
    ```
    $xmlelement->element="Nouvelle valeur d'élément"
    $xmlelement->element[i]="Nouvelle valeur d'élément"
    $xmlelement->element["att"]="Nouvelle valeur
    d'attribut"
    ```
  - Ajout d'attributs et éléments :
    ```
    $xmlelement->element->addChild("name"[, "valeur"]);
    $xmlelement->element->addAttribute("type", "valeur")
    ```
  - Enregistrement des modifications :
    ```
    boolean $xmlelement->asxml("Fichier.xml")
    ```

PHP et XML (6/6)

Modification et enregistrement des données

XML3.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page PHP</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <meta http-equiv="Content-Style-Type" content="text/css" />
  </head>
  <body>
    <?php
      $xml = simplexml_load_file("data/bib.xml");
      $xml->livre[1]->titre = "Nouveau titre";
      $xml->addchild("livre");
      $xml->livre[2]->addChild("titre", "Un 3ème");
      $xml->livre[2]->addChild("auteur", "Aude Haibu");
      $xml->livre[2]->addAttribute("prix", "7.50");
      if($xml->asxml("data/bibMODIFIED.xml"))
        echo "Enregistrement réalisé";
    ?>
  </body>
</html>
```

Bases de données (1/5)
Description de SQLite

- PHP supporte un grand nombre de bases de données
  Oracle, Sybase, Ingres II, MySQL, PostgreSQL, **SQLite (inclus dans PHP5)**, ...

- Caractéristiques de SQLite
  - SGBD embarqué dans la distribution de PHP5
    $\Rightarrow$ pas de processus indépendant
  - léger et rapide (pas d'architecture client-serveur)
  - SQLite implémente la norme SQL 92
  - classes facilitant son interaction avec PHP
  - BD en un et un seul fichier
  - pas d'insertions concurrentes (base verrouillée)
  - accès concurrents en lecture seule

- Documentation officielle
  - `http://fr.php.net/manual/fr/ref.sqlite.php`
  - `http://sqlite.org/`

Bases de données (2/5)
Ouverture et fermeture

- Accès à la base
  - Ouverture : `$db = new PDO("sqlite:path/filename");`
  - Fermeture : `unset($db);`
  - Script type d'accès à la base :

```php
<?php
  error_reporting(E_ALL);
  try {
    /* creation de la BD */
    $db = new PDO("sqlite:./data/DATABASE/personnes.sqlite");
    /* errors -> exceptions */
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    ...
    $db->exec("INSERT ...); // Requête SQL (insertion)
    $result = $db->query("SELECT ..."); // Requête SQL (sélection)
    $db->beginTransaction();
    ... // Série de requêtes SQL
    $db->commit();
    unset($db);
  }
  catch(PDOException $e) {
    // Traitement des Exceptions
  }
?>
```

Bases de données (3/5)
Exemple - Création d'une table

SQLite-Creation.php

```
    error_reporting(E_ALL);
    try
    {
      $requete = "CREATE TABLE IF NOT EXISTS personnes (
      id INTEGER PRIMARY KEY,
      nom TEXT NOT NULL
      )";
      /* creation de la BD */
      $db = new PDO("sqlite:./data/DATABASE/personnes.sqlite");
      /* errors -> exceptions */
      $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
      /* requete de creation */
      $db->query($requete);
      unset($db);
      echo 'Table créée';
    }
    catch(PDOException $e)
    {
      echo $e->getMessage();
    }
  ?>
</body>
```

Bases de données (4/5)
Exemple - Insertion de données

SQLite-Insertion.php

```
    error_reporting(E_ALL);
    try
    {
      /* creation de la BD */
      $db = new PDO("sqlite:./data/DATABASE/personnes.sqlite");
      /* errors -> exceptions */
      $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
      /* debut de transaction */
      $db->beginTransaction();
      $db->exec("INSERT INTO personnes
          ('nom')
          VALUES ('John');");
      $db->exec("INSERT INTO personnes
          ('nom')
          VALUES ('Samantha');");
      /* commit des insertions */
      $db->commit();
      unset($db);
      echo 'Insertions réalisées';
    }
    catch(Exception $e)
    { echo $e->getMessage(); }
  ?>
</body>
```

Bases de données (5/5)
Exemple - Sélection dans une table

SQLite-Selection.php

```
    error_reporting(E_ALL);

    try
    {
      /* creation de la BD */
      $db = new PDO("sqlite:./data/DATABASE/personnes.sqlite");
      /* errors -> exceptions */
      $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
      /* requete de selection */
      $requete = "SELECT * FROM personnes";
      $result = $db->query($requete);
      unset($db);
      foreach($result as $row)
      {
        echo '<p>'.$row['id'].': '.$row['nom'].'</p>';
      }
    }
    catch(Exception $e)
    {
      echo $e->getMessage();
    }
  ?>
</body>
```