Langage de Script PHP

Partie 2 : La persistance

Enseignants de l'EMSI RABAT

version éditée par: M. BELATAR

3éme Année - Ingénierie en Informatique et Réseaux

31 octobre 2023

Table of Contents

Table of Contents

PHP (Modèle)

```
$name = array();                          # Create
$name = array(value0, ..., valueN);
$name[index]                              # Get element value
$name[index] = value;                     # Set element value
$name[] = value;                          # Append PHP
```

PHP (Exemple)

```php
$a = array();                          # Empty array (length 0)
$a[0] = 23;                            # Stores 23 at index 0 (length 1)
$a2 = array("some", "strings", "in", "an", "array");
$a2[] = "Ooh!";                        # Add string to end (at index 5)
```

NB : Un tableau peut contenir des éléments de différents types.

PHP (Modèle)

```
$name = array();                          # Create
$name = array(key0 => value0, ..., keyN => valueN);
$name[key]                                # Get element value
$name[key] = value;                       # Set element value
$name[newKey] = value;                    # Append PHP
```

PHP (Exemple)

```
$p = array();                  # Empty array (length 0)
$p["firstmane"] = "Allan";     # Stores "Allan" at key "firstname"
$p["lastname"] = "Turing";     # Stores "Turing" at key "lastname"
```

Les indices sont des strings et les clefs sont **toujours** des chaînes de caractères.

PHP (Modèle)

```
foreach ($array as $key => $value) {
   statements ;
}
```

PHP (Exemple)

```
$dict = [ "oeuf" => "egg", "pomme" => "apple", "pain" => "bread" ];
foreach ($dict as $fr => $en) {
  echo "$fr se dit $en en anglais";
}
```

NB : Elle Permet d'itérer sur les couples (clef, valeur) d'un tableau associatif.

Table of Contents

PHP (Modèle)

include(filename) ;

PHP (Exemple)

```php
include("header.html");
include("shared-code.php");
```

- Insère le contenu du fichier paramètre dans le fichier courant.
- Permet une certaine modularité.
- Permet de partager des fonctions utiles dans plusieurs scripts.
- Voir aussi : include_once, require, et require_once.

Table of Contents

Nom	Categorie
file, file_get_contents, file_put_contents	Lecture/écriture d'un fichier en entier
basename, file_exists, filesize, fileperms, filemtime, is_dir, is_readable, is_writable, disk_free_space	Demande d'informations
copy, rename, unlink, chmod, chgrp, chown, mkdir, rmdir	Manipulation de fichiers et de répertoires
glob, scandir	Lecture de répertoires

contenu de foo.txt	file("foo.txt")	file_get_contents("foo.txt")
Hello how r u ? I'm fine	array("Hello\n", #0 "how r u ?\n", #1 "\n", #2 "I'm fine\n", #3)	"Hello\n how r u\n, # a single \n, # string I'm fine\n

- `file` retourne les lignes d'un fichier de texte sous la forme d'un tableau, chaque case du tableau contenant une ligne du fichier (une chaîne de caractères terminant par le caractère `'\n'`).
- `file_get_contents` retourne le contenu d'un fichier de texte sous la forme d'une unique chaîne de caractères.
- `file_put_contents` écrit une chaîne de caractères dans un fichier de texte.

PHP (Modèle)

```php
# affiche les lignes d'un fichier
$lines = file("todolist.txt");
foreach ($lines as $line) { # for ($i = 0; $i < count($lines); $i++)
    echo $line;
}
```

`file` retourne les lignes du fichier sous la forme d'un tableaux de chaînes de caractères qui se terminent par le caractère `'\n'`.

Pour le supprimer, il faut utiliser un second paramètre lors de la lecture du fichier :

```php
$lines = file("todolist.txt", FILE_IGNORE_NEW_LINES);
```

PHP (Modèle)

$array = explode(delimiter, string) ;
$string = implode(delimiter, array) ;

PHP (Exemple)

```php
$s = "3IIR EMSI 2019";
$a = explode(" ", $s);                    # ("3IIR", "EMSI", "2019")
$s2 = implode("...", $a);                 # ("3IIR...EMSI...2019")
```

Les fonctions **explode** et **implode** convertissent entre chaînes de caractères et tableaux.

Pour les extractions complexes on peut utiliser **preg_split** avec des expressions régulières comme délimiteurs.

Contenu .txt (Exemple)

```
Steven Paul Jobs
Mark Elliot Zuckerberg
```

PHP (Exemple)

```php
foreach (file("names.txt") as $name) {
  $tokens = explode(" ", $name);
  echo "author: $tokens[2], $tokens[0]\n";
}
```

Résultat

author : Jobs, Steven
author : Zuckerberg, Mark

PHP (Modèle)

list($var1, ..., $varN) = array ;

Contenu .txt (Exemple)

```
Peter Parker
Spiderman
06 12 34 56 78
```

PHP (Exemple)

```php
list($name,$nickname,$phone) = file("personal.txt");
echo "$name (alias $nickname) : $phone";
```

Résultat : **Peter Parker (alias Spiderman) : 06 12 34 56 78**

- **glob** : retourne un tableau de tous les chemins de fichiers qui correspondent à un modèle donné.
  Les chemins sont de la forme `"foo/bar/myfile.txt"`
- **scandir** : retourne un tableau de tous les noms de fichiers contenus dans un répertoire donné.
  Les noms sont de la forme `"myfile.txt"`

On peut utliser le caractère spécial `*` (correspond à une chaîne de caractères quelconque)

PHP (Modèle)

```php
# réécrit le contenu de tous les fichiers du 'poetry' à l'envers
$poems = glob("poetry/*.txt");
foreach ($poems as $poemfile) {
$text = file_get_contents($poemfile);
file_put_contents($poemfile, strrev($text));
echo "I just reversed " . basename($poemfile) . "\n";}
```

glob filtre les chemins de fichiers avec le caracttère joker `'*'`

- `glob("foo/bar/*.doc")` retourne tous les chemins de fichiers `.doc` du répertoire `foo/bar`

basename retourne le nom relatif du fichier :

- `basename("foo/bar/baz.txt")` retourne `"baz.txt"`

PHP (Modèle)

```php
foreach (scandir("cours/PHP") as $filename ) {
   echo "{$filename}<br>" ;
}
```

Résultat

```
.
..
intro.pdf
exemple.php
```

- `scandir` retourne toujours, dans le tableau, le répertoire courant `(".")` et le répertoire parent `("..")`.
- Il est inutile d'utiliser la fonction **basename** puisque **scandir** ne retourne que les noms relatifs des fichiers (sans la partie répertoire).

PHP (Modèle)

```
/ inverse le contenu d'un fichier
$text = file_get_contents("poem.txt");
$text = strrev($text);
file_put_contents("poem.txt", $text);
```

file_get_contents retourne tout le contenu d'un fichier sous forme d'une unique chaîne de caractères.

- Si le fichier n'existe pas, retourne la chaîne vide.

file_put_contents copie une chaîne de caractères comme (nouveau) contenu d'un fichier, et remplace l'ancien contenu.

- Si le fichier n'existe pas, il est créé.

PHP (Modèle)

```php
# ajoute une nouvelle ligne au fichier "poem.txt"
$new_text = "inspired by Sir Edmund Spenser";
file_put_contents("poem.txt", $new_text, FILE_APPEND);
```

Tableau a faire

file_put_contents peut être invoquée avec comme troisième argument le mot-clef `FILE_APPEND` pour ajouter une chaîne de caractères à la suite du contenu existant.

Table of Contents

Une classe peut être vue comme une structure de données qui contient

- des attributs ou propriétés internes
- des méthodes : des fonctions définies à l'intérieur de la classe.

Chaque propriété/méthode se voit attribuer un mode permettant de définir l'acessibilité de la classe.

1. public : accessible à partir de n'importe quel objet de la classe, et même dans les classes et objets dérivés
2. private : accessible uniquement à l'intérieur de la classe. Aucun objet ne peut y accéder
3. protected : accessible dans la classe et dans les classes dérivées

NB : Pour chaque propriété, ou fonction il faut préciser son mode. Toute propriété ou méthode dans la classe est accédée à l'intérieur de la classe à l'aide du mot clé $this

Modèle

```
class NomClasse{
        mode $var ;
        mode function nommethode( $var1, $var2, $var3){
                #Code de la fonction
        } }
```

Exemple

```
class Livre{
       public $isbn;
       private $titre;
       public function afficher( ){
              echo $this->isbn."<br/>";
      }
}
```

Constructeur/Destructeur de classe

Les constructeurs sont des fonctions spécifiques qui permettent d'affecter des valeurs aux attributs lors de la création d'un objet.
Le destructeur est prévu pour libérer l'espace mémoire alloué à l'objet.
Ils ont toujours des modes public.

Modèle

```
class NomClasse{
        public function _construct( $var1, $var2, $var3){
                $v1=$var1 ;
        } }
```

Exemple

```
class Livre{
        public function _construct( $i, $t){
                $titre=$t ;
                $isbn=$i ;
        } }
```

Objets : propriétés et méthodes

Une fois la classe définie, nous pouvons créer un objet et accéder à ses propriétés et méthodes (publiques).

Modèle

```
#Creation de l'objet
$obj1 = new NomClasse();
#Accès aux proriétés
$obj1->prop;
#Accès aux méthodes
$obj1->nomfunction(listevar);
```

Exemple

```
$l = new Livre("12345", "Apprendre PHP");
$code = $l->isbn;
$l->affichage();
```

POO : Héritage

L'héritage permet à des classes (mères) de transférer des propriétés et méthodes à d'autres classes (filles).
Par exemple, un chat possède les attributs et fonctions générales d'un animal.
L'héritage se fait à l'aide du mot-clé extends. Il faut que la classe mère soit d'abord définie.

Modèle

```
class B extends A {
        public function _construct ($varfille , $ varparent) {
                parent : :_construct($ varparent)
        }
}
```

Table of Contents

Accès aux BDD avec PDO

- PDO : PHP Data Objects
- Extension PHP fournissant une interface pour accéder à une base de données
- Fournit une interface d'abstraction pour l'accès aux données
- Ne fournit PAS une abstraction de base de données
  - SQL spécifique au moteur du SGBD
  - Fonctionnalités présentes / absentes
- Interface orientée objet
- Compatible avec plusieurs SGBD et dispose de différents drivers, dont : **MySQL, Oracle, PostgreSQL, SQLite, ODBC et DB2, MS SQL Server, Informix, Firebird, IBM ...**

Connexion PDO - MySQL

PHP (Modèle)

```php
$db = new PDO($dsn, $username=null, $password=null, $options=null);
```

PHP (Exemple MySQL)

```php
$db = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
```

Les informations de la basse de données (host + nom de BDD + utilisateur + mot de passe) ainsi que la commande de connexion ci-dessus sont généralement isolées dans un fichier à part pour être incluses dans tous les scripts où on va avoir besoin de faire des opération sur la BDD.

Gestion des erreurs - Connexion

PHP (Exemple MySQL)

```
try{
  $db = new PDO("mysql:host=localhost;dbname=test", $user,
$pass);
} catch(PDOException $e){
  die( "Erreur: ".$e->getMessage()."<br>");
}
```

La fonction die(...) permet d'envoyer un texte au client (comme 'echo') et d'arrêter l'interprétation du script.

Gestion des erreurs - Hormis connexion

PHP (Exemple)

```php
$db->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
```

- PDO::ERRMODE_SILENT (par défaut)
  - Mode silencieux, mise en place d'un code d'erreur
  - $db->errorCode() et $db->errorInfo()
- PDO::ERRMODE_WARNING
  - Émission d'une erreur de type E_WARNING
- PDO::ERRMODE_EXCEPTION
  - Déclanchement d'une exception de type PDOException
  - La technique la plus propre pour gérer les erreurs

Opérations de base : query(...)

PHP (Exemple)

```php
$sql="SELECT id,nom,prenom,email FROM client";
$rs=$db->query($sql);
echo "<table>";
while ($rec=$rs->fetch(PDO::FETCH_ASSOC)) {
  echo "<tr data-id=\"".$rec["id"]."\"><td>".$rec["nom"]. " "
.$rec["prenom"]."</td><td>".$rec["email"]."</td></tr>";
}
echo "</table>";
```

Nous avons séléctionné tous les enregistrements dans la table "client" et nous les avons affichés sous forme de tableau HTML.

- La méthde fetch permet de récupérer un enregistrement et pointer sur le suivant.

Opérations de base : fetch(...)

- On peut récupérer l'enregistrement courant dans différents formats :
  - Tableau associatif : PDO::FETCH_ASSOC
  - Tableau indexé : PDO::FETCH_NUM
  - Les deux à la fois : PDO::FETCH_BOTH (par defaut)
  - Objet : PDO::FETCH_OBJ
  - ...
- La méthde fetchAll() permet de récupérer tous les enregistrements d'un coups.

PHP (Exemple)

```
$allRecs = $rs->fetchAll();
foreach ( $allRecs as $rec ) {
  echo "<tr data-id=\"".$rec["id"]."\"><td>".$rec["nom]. " "
.$rec["prenom"]."</td><td>".$rec["email"]."</td></tr>";
}
...
```

Opérations de base : prepare(...)

Quand on veut lancer plusieurs fois la même requete en changeant seulement quelques paramètres, il est préférable de la "préparer".

- C'est plus "propre"
- C'est plus performant
- Inclut une protection contre les injections SQL

Pour lancer une requête "préparée" il faut :

1. La préparer avec
   `prepare("requete sql contenant des marqueurs '?' ou ':marqueur'")`
2. L'executer avec la méthode `execute([tableau, de, paramètres])`

Opérations de base : prepare(...)

PHP (Exemple)

```php
$sql="SELECT nom,calories FROM fruit WHERE calories < ? AND
couleur = ?";
$rs=$db->prepare($sql);
$rs->execute([100,"rouge"]);
$allRed = $rs->fetchAll();
$rs->execute([120,"jaune"]);
$allYellow = $rs->fetchAll();
$rs->execute([90,"vert"]);
$allGreen = $rs->fetchAll();
```

Les requêtes "executées" après avoir été "préparées" sont plus rapides que celles lancées par un simple "query".

Opérations de base : exec(...)

La méthode PDO `exec("requête SQL")` permet d'executer une requête SQL en écriture et retourner le nombre d'enregistrements affectés.

PHP (prototype)

```
public PDO::exec(string $statement): int|false
```

PHP (Exemple)

```
$nb = $db->exec("DELETE FROM fruit WHERE calories > 200");
$nb = $db->exec("UPDATE fruit SET calories='160' WHERE
nom='avocat'");
$nb = $db->exec("INSERT INTO client (nom, prenom, email) VALUES
('DOE','John','john@doe.com')");
$clientId = $db->lastInsertId();
```

Opérations de base

- La méthode PDO `lastInsertId()` donne l'ID du dernier enregistrement inséré. Elle est très utile quand la table dispose d'une clé primaire numérique de type AUTO_INCREMENT.
- Les requêtes d'écriture (INSERT, UPDATE, DELETE ...) peuvent elles aussi être "préparées" et "executées"

PHP (Exemple)

```php
$sql = "DELETE FROM fruit WHERE calories > ? AND couleur <> ?";
$rs = $db->prepare($sql);
$rs->execute([100,"rouge"]);
```

Les transactions

- Une transaction consiste à confirmer plusieurs requêtes SQL à la fois. On utilise donc la methode PDO `commit()`
- Si on a un problème au cours du traitement de nos requêtes, on peut annuler toutes les requêtes précédantes de la transaction en cours avec `rollBack()`
- Une transaction commance toujours par `beginTransaction()`
- Les transactions sont importantes dans les cas où l'on ne doit rien faire si une requete parmi plusieurs retourne un résultat non désiré
- Par exemple: dans une BDD MySQL de type MyISAM, quand je supprime enregistrement, je dois supprimer toutes ses relations dans les autres tables (clès étrangères pointant vers cet enregistrement). Si l'une de ces requêtes n'aboutit pas, on doit annuler toutes les requêtes de suppressions précédantes.