+ POO

Syntaxe & Astuces

Balises
<?php ... ?> ou <? ... ?>
<% ... %>
<script language="php">
... </script>

Commentaires
// ou # : sur 1 ligne

/* commentaire sur
plusieurs lignes */

Types
RAPPEL : langage faiblement typé, donc type en fonction du contexte d'utilisation.

-> Simple
bool
int, integer
double, float, real
string

-> Composé
array
object (instances)

-> Spéciaux/Systèmes
ressource
null
mixed (plusieurs types)

Manipulations de variables
-> accolades
```
$var1 = "valeur";
$var2 = "var1";
echo "$var1"; //out: "valeur"
echo "${var2}"; // "var1"
echo "{$var2}"; // "{var1}"
echo "${$var2}"; // "valeur"
```
-> copies
alias : $b = &$a;
clone : $b = clone $a;
classique : $b = $a;

Opérations arithmétiques
-> Arithmétiques
+ - * / %
+=1 => ++
-=1 => --

-> Comparaison
== != < <= >= >

-> Logique
&& and || or ! not xor

-> Concaténation : .

-> Comparaison de types
=== !==
instanceof (objet/classe)

-> Système
@ : contrôle d'erreur
`...` : commande shell

Définitions & Déclarations

```
class MaClasse {
public $variable1;
private $variable2;

public static $varClasse = 0;

public function maFonction($v1, &£v2) {
// instructions ;
/* $v1 ne subit pas de modification en
dehors de la fonction, $v2 si!! */
}

public static function varGl() {
return self::$varClasse;
}
}
```

Instanciation

Instance
$objet = new MaClasse();

Utilisation d'une variable
$objet -> variable1 = 10;

Appel d'une fonction
$objet -> maFonction(1, 'ab');
$this : à l'intérieur d'une fonction

Niveaux de visibilité

public : visible de tous
protected : accès private + ss-classes
private : accès interne uniquement

Classes Abstraites & Interfaces

Classe abstraite
```
abstract class MaSuperClasse {
//fonction abstraite, définie dans les classes héritées,
abstract public function superFonction();
}
```

Interface
- Déclaration
```
interface MonInterface  [ extends MaSuperInterface] {
public function maFonction();
}
```

- Implémentation
```
class MaClasse implements MonInterface {
public function maFonction()
{...}
}
```

Architectures applicatives : dénominations

- index.php : fichier à la racine, exécuteur/controller principal
- layout.php : fichier à la racine, vue principale
- style : dossier de feuilles de styles et des images de design
- modules/actions : dossiers d'exécution pour les CU
  (chaque module contient plusieurs dossier d'actions)
- public/admin/install : dossiers d'espaces d'utilisations (facultatif)
- noyau : dossier d'espace protégé
- plugins/templates : dossiers d'addons de l'appli (si prévu)
- scripts : dossier de scripts javascripts
- libs/librarys : dossier des librairies utilisées
- images_data : images du contenu
- [class_name].class.php : fichiers de classes
- [module].inc.php : bibliothèque de fonctions
- comportement.js : fichier pour la gestion de l'aspect dynamique et de l'ergonomie de la page
- interactions.js + traitement.php : fichiers pour la gestion d'AJAX

Méthodes

Architecture globale
```
require()
require_once()
include()
include_once()
define(nom, "val")  //ctes
```

Variables
set(); unset(); isset();
get();
settype(); //conversion

POO
-> existence (bool)
```
class_exists("...");
interface_exists("...");
method_exists("...");
property_exists("...");
```
-> liste de ... (array)
```
get_class_methods("...");
get_class_vars("...");
get_declared_classes("...");
get_declared_interfaces("...");
```
-> instances
```
$obj instanceof MaClasse;
    //retourne un bool
get_class($obj);
    //retourne un string
is_subclass_of($obj, 'classe parent');
    /* retourne un bool,
       true si c'est l'instance d'une
       ss-classe */
```
-> constantes
Nom des classes et méthodes exécutées au moment de l'appel
__CLASS__
__METHOD__

Surcharges de méthodes
```
__construct();
__destruct();
__clone();
__get();
__set();
__unset();
__isset();
__call();
__callStatic();
__sleep(); //serialize()
__wakeup(); //unserialize()
__toString();
```

Extensions de classe
```
//MaClasse peut être abstraite ou non
classe MaClassebis extends MaClasse { ... }
```

Méthodes héritées
```
parent ::maFonction();
--> appel de la fonction
    de la classe implémentée
self::maFonction();
-->fonction de la classe actuelle
```