Développer en PHP objet.

Depuis la version 5 de php il est possible de programmer en POO (Programmation Orienté Objet ). La POO n’est pas un nouveau langage c’est juste une façon optimisée d’organiser son code. Aujourd’hui les logiciels demande tellement de ligne de code qu’il n’est plus possible de coder son programme de façon linéaire. Tout le monde utiliser la POO.

Pourquoi parle t-on d’objet ? en fait on part du principe que "tout" peut être caractérisé par un objet. Une cafetière est un objet, une table est un objet, un humain est un objet, le soleil est un objet, l’univers est un objet…

Comment caractérise-t-on un objet ? Comme tout est objet il vient naturellement le principe de caractériser un objet. Les 2 principes fondamentaux pour caractériser un objet sont : Ses Propriétés est Ses fonctionnalités. ( en langage informatique les fonctionnalités d’un objet sont appelées ses Méthodes.

Pourquoi caractériser des Objet plutôt que de développer en linéaire.

Le fait de créer des caractérisations d’objet dans notre code à 2 avantages très efficaces : Tout d’abords il est très facile d’identifier la partie de code qui concerne notre objet car il ne sera qu’ à un endroit du programme. Deuxièmement il est tout à fait possible de réutiliser notre objet dans un autre programme. Il existe encore d’autre avantage que vous découvrirez par la suite.

## Mon premier Objet PHP.

Lorsque l’on programme on a besoin de caractériser les objets que nous avons besoin dans notre programme. Rappeler vous toujours que "Tout" est objet. Je vais prendre le cas très simple en php d’une page de connexion. La page de connexion php sert à gérer l’authentification des utilisateurs de notre application web. On pourrait donc naturellement se dire pourquoi ne pas utiliser un objet Utilisateur ;)

Vocabulaire : Le fait de caractériser un objet est appelé "une class" d’objet. Pour utiliser l’objet dans notre code on dit qu’on instancie la class de notre objet.

Ma première class PHP.

Avant de programmer il faut faire un peu d’analyse est respecter certain standard pour que chaque développeurs se comprennent.

Si je prends l’exemple d’un utilisateur de mon site :

Se le caractérise par les propriétés suivante : Login, MdP

Il possède les fonctionnalités suivante : SeConnecter() , SeDeconnecter()

Utilisateur

Voici sa représentation analytique

-Nom

-Login

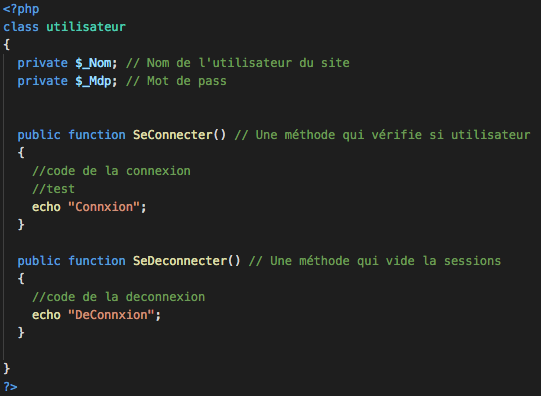
-----------------

SeConnecter()

SeDeconnecter()

Déclarer une class en PHP.

Pour caractériser un objet en php on utilise le mot clés "class".



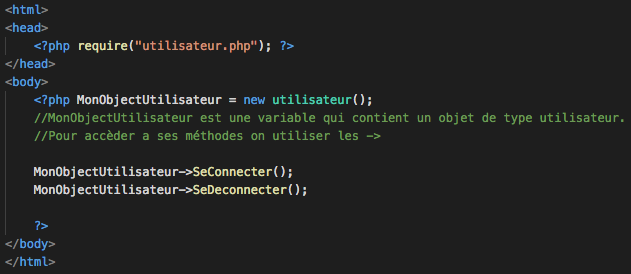
Voilà notre classe est créer. Elle ne fait rien de spéciale pour le moment mais elle existe. Il faudra enregistrer cette class en Utilisateur.php le fichier porte le nom de la class c’est plus logique

Les propriétés sont des variables précéder de mot clés Private ( pour les bon développeur ) car seul les méthodes déclarer dans la class peuvent les modifiers.

Les méthodes sont des fonctions précèder du mot clés "public" pour que les autres objets du programme puissent y accéder.

Créer son premier objet en PHP.

Pour créer un objet il faut avoir définit avant sa class c’est à dire l’avoir caractérisé dans un fichier par ses propriétés et ses méthodes. Puis importer ce fichier à l’endroit ou l’on veut utiliser notre class gràce au mot clé "include" ou "require".



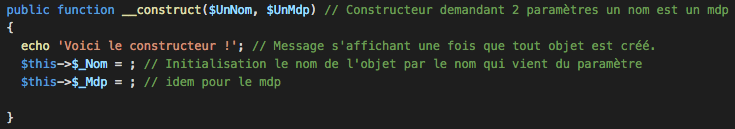
Avec une seul "class" je peux instancier une infinité d’objet. Ici je pourrais avoir 10 objects MonOjectUtilisateur1 , MonOjectUtilisateur 2 … MonOjectUtilisateur 10 .

Différencier les instances d’une class.

Si je fais une dizaine de new Utilisateur() , mes objets seront tous identiques il est possible en POO de passer par un "Constructeur" c’est à dire une méthode particulière qui sera lancer uniquement à la création de l’objet . Par analogie il existe aussi un destructeur.

## Le constructeurs en php.

Dans certain langage comme en C++ le constructeur est une méthode qui porte le nom de la class. En php elle n’a pas de nom mais il faudra rajouter le mot clé " \_\_construct" avec 2 underscores. Un constructeur peut prendre 0 ou N paramètres. Ici dans l’example j’ai créé un constructeur avec 2 paramètres : unNom et UnMdp.



pour créer un objet Utilisateur maintenant je fais :

$User1 = new Utilisateur("Julien","1234") ; //je passe 2 valeurs car mon constructeur attends 2 paramètres.

Voilà dans mon code j’ai créer mon premier objet qui à pour propriété un nom = Julien et un mdp = 1234.

Tout au long de ma programmation je pourrais utiliser la variable $User1 pour accéder à cet utilisateur.

Attention si vous créez un constructeur qui attend un ou plusieurs paramètre vous serez obligé de créer tous vos objets en passant ses paramètres lors de l’utilisation de new.

Les Autres concepts objets.

En POO il existe plein de concept pratique qu’il est possible de mettre en pratique en POO PHP.

## This.

$this est un objet particulier en POO il permet de représenté l’objet lui même à l’intérieur d’une Classe.

Si je fait $this->\_Nom j’accède à la propriété $\_Nom de ma classe utilisateur.

PERSONNAGE

-Nom

-Nom

-----------------

Attaquer(int)

## L’héritage.

L’héritage est un concept fortement utilisé en POO. Il permet à une classe d’hériter des méthodes et propriété ( protected et public ) d’un autre classe. On dit qu’une Classe hérite d’une autre classe parente.

GUERRIER

-Force

-----------------

Fraper()

Exemple Un classe Guerrier peut hériter d’une classe personnage. Ainsi lors de la création d’un objet Guerrier on pourra avoir accès à toutes les méthodes de l’objet personnage.

Class Personnage{

Public function Attaquer($degat){

….

}

public function \_\_construct(){

…

}

}

Class Guerrier Extends Personnage{

$Force = 100 ;

public function \_\_construc(){

parent ::\_\_construct() ;

}

….

}

G1 = new Guerrier() ;

//ici on peu appeler la méthode Attaquer sur un Objet Guerrier alors que cette méthode //fait partie de la classe Personnage.

G1-> Attaquer (10) ;

G1-> Fraper () ;

Chaque fois qu’un classe Possède des méthodes identiques à d’autre, alors elles peuvent hériter d’une même classe Parentes qui possèderais ces méthodes.

## Les logs

## En PHP

Enfin en PHP, le code ressemble quelque peu au code en C

|  |  |
| --- | --- |
| 1  2  3  4 | <?php  openlog("MonTag", LOG\_PID, LOG\_USER);  syslog(LOG\_DEBUG, "mon message 4 depuis un script PHP");  closelog(); |

Résultat sur votre debian :

|  |  |
| --- | --- |
| 1  2 | $ tail -n 1 /var/log/syslog  Jan 15 11:27:40 ikaros MonTag[1449]: mon message 4 depuis un script PHP |