PHP OBJET

Niveau 1

L'OBJET

Les interfaces

Les interfaces sont semblables aux classes abstraites. Elles ne peuvent pas être instanciées, mais vous ne pouvez pas implémenter de méthode dans les classes abstraites. Ces classes servent donc uniquement à obliger les classes héritant de l'interface à redéfinir les méthodes déclarées dans l'interface. Ces méthodes sont forcément publiques.
Une interface se déclare avec le mot clé interface.

Voici le code de l'interface action :

```
<?php
interface action {
function courir();
function manger();
}
?>
```

Pour signaler qu'une classe implémente une ou plusieurs interfaces, vous devez ajouter le motclé implements suivi
du nom des interfaces.
Voici un exemple signalant que la classe Chat implemente l'interface action :

Les interfaces

```
<?php
class Chat extends Animal implements action
{
        ...
        //méthodes de l'interface
        function courir() {
            echo "Le chat court.<br />";
        }
        function manger() {
            echo "Le chat mange.<br />";
        }
}
?>
```

La page utilisation.php devient :

Les interfaces

```
<?php
//chargement des classes
include('Animal.class.php');
include('action.class.php');
include('Chat.class.php');
//instanciation de la classe Chat qui appelle le constructeur
de la classe Animal
$chat = new Chat("blanc",4);

//appelle des méthodes de l'interface action
echo $chat->courir();
echo $chat->manger();
?>
```

Il est possible d'implémenter plusieurs interfaces en les séparant par une virgule.

Les classes anonymes

Cette fonctionnalité, arrivée avec PHP 7, permet d'utiliser une classe sans avoir à définir entièrement une classe. Cette classe n'a pas de nom et est utilisée une seule fois. Cela se fait grâce à l'instruction new class.

```
//création de la classe anonyme
$classe_anonyme = new Class {
        public $message = "Bonjour";
        public function getMessage() {
            return "Salut";
        }
};
echo $classe_anonyme->getMessage();
```

Affiche :

Salut.

Les traits

Cette fonctionnalité, arrivée avec PHP 5.4, permet de réutiliser du code dans deux classes indépendantes. Sa syntaxe est :

```
trait nom_du_trait {
//définition des propriétés et méthodes.
}
```

Prenez l'exemple de deux classes indépendantes Facture et Indemnite ayant une méthode Calcul_taux_tva.

```
<?php
//déclaration du trait
trait MonTrait
{
        public function Calcul_ttc($montant)
        {
             return $montant*1.2; //retourne le montant TTC
        }
}
class Facture
{
use MonTrait;
}
```

Les traits

```
class Indemnite
{
use MonTrait;
}

$facture = new Facture;
//affichage du montant TTC de la facture
echo $facture->Calcul_ttc(10)."<br />";
$indemnite = new Indemnite;
//affichage du montant TTC de l'indemnité
echo $indemnite->Calcul_ttc(20);
?>
```

Cela permet aux deux classes **Facture** et **Indemnite** d'utiliser la même méthode **calcul_ttc** sans recourir à l'héritage.

Uniform Variable Syntax

Cette syntaxe de variable uniforme vous permet de résoudre certaines incohérences sur l'évaluation des expressions.
Depuis PHP 5.6, il est possible d'accéder à une propriété avec cette syntaxe :
$obj->{$properties['name']};

Par exemple :

```
<?php
//Déclaration de la classe Objet
class Objet { public $couleur ="rouge"; }
//Instanciation de la classe Objet
$obj = new Objet();
$properties['name'] = "couleur";
echo $obj->{$properties['name']};
?>
```

En effet, $obj->{$properties['name']} revient à écrire $obj->couleur.

Uniform Variable Syntax

Depuis PHP 7, il est possible d'attacher des appels statiques.
Par exemple :

```
<?php
//Déclaration de la classe 1
class classe1 { static $nom1 = 'classe2'; }
//Déclaration de la classe 2
class classe2 { static $nom2 = 'Bonjour'; }
//Déclaration de la méthode permettant d'afficher Bonjour
classe2::$nom2 = function () { echo "Bonjour !"; };
$classe1 = 'classe1';
//Appel statique
($classe1::$nom1::$nom2)();
?>
```

Affiche :

Bonjour !

Autoload

La fonction spl_autoload_register() permet d'éviter d'inclure manuellement toutes les classes en début de chaque page PHP. Cette fonction prend en paramètre votre fonction permettant d'inclure les classes nécessaires.
Par exemple :
Autoload

```
<?php
//Affectation de la fonction gérant l'autoload
spl_autoload_register('monAutoload');
function monAutoload($className)
{
        //Les classes sont dans le dossier class
        $path = '/class/';
        //Le nom des fichiers doit correspondre au nom de la classe
        include $path.$className.'.php';
}
//Instanciation de la classe
$myClass = new MyClass();
?>
```

©