Introduction au langage de programmation PHP

Les niveaux d'erreurs

Lorsqu'un comportement anormal survient, PHP peut déclencher une erreur. Une erreur contient plusieurs informations :

- un niveau d'importance
- un message explicatif
- le script en cause
- le numéro de la ligne en cause

PHP distingue plusieurs niveaux d'importance qui ont plus ou moins d'impact sur le déroulement du script. Voici le tableau allégé des constantes d'erreur de la documentation officielle :

Valeur	Constante	Description
1	E_ERROR (entier)	Les erreurs sont aussi affichées par défaut, et l'exécution du script est interrompue. Elles indiquent des erreurs qui ne peuvent pas être ignorées, comme des problèmes d'allocation de mémoire, par exemple.
2	E_WARNING (entier)	Les alertes sont affichées par défaut, mais n'interrompent pas l'exécution du script. Elles indiquent un problème qui doit être intercepté par le script durant l'exécution du script.
4	E_PARSE (entier)	Les erreurs d'analyse ne doivent être générées que par l'analyseur. Elles ne sont citées ici que dans le but d'être exhaustif.
8	E_NOTICE (entier)	Les remarques ne sont pas affichées par défaut, et indiquent que le script a rencontré quelque chose qui peut être une erreur, mais peut aussi être un événement normal dans la vie du script.
16	E_CORE_ERROR (entier)	Elles sont similaires aux erreurs E_ERROR, mais elles sont générées par le code source de PHP. Les fonctions ne doivent pas générer ce genre d'erreur.
32	E_CORE_WARNING (entier)	Elles sont similaires à E_WARNING, mais elles sont générées par le code source de PHP. Les fonctions ne doivent pas générer ce genre d'erreur.
64	E_COMPILE_ERROR (entier)	Elles sont similaires à E_ERROR, mais elles sont générées par le moteur Zend. Les fonctions ne doivent pas générer ce genre d'erreur.
128	E_COMPILE_WARNING (entier)	Elles sont similaires à E_WARNING, mais elles sont générées par le moteur Zend. Les fonctions ne doivent pas générer ce genre d'erreur.
256	E_USER_ERROR (entier)	Message d'erreur généré par l'utilisateur. Comparable à E_ERROR. Elle est générée par le programmeur en PHP par l'utilisation de la fonction trigger_error(). Les fonctions de PHP ne doivent pas générer ce genre d'erreur.
512	E_USER_WARNING (entier)	Message d'erreur généré par l'utilisateur. Comparable à E_WARNING. Elle est générée par le programmeur en PHP par l'utilisation de la fonction trigger_error(). Les fonctions de PHP ne doivent pas générer ce genre d'erreur.
1024	E_USER_NOTICE (entier)	Message d'erreur généré par l'utilisateur. Comparable à E_NOTICE. Elle est générée par le programmeur en PHP par l'utilisation de la fonction trigger_error(). Les fonctions de PHP ne doivent pas générer ce genre d'erreur.

2048	E_STRICT (entier)	Permet d'obtenir des suggestions de PHP pour modifier votre code, assurant ainsi une meilleure interopérabilité et compatibilité de celui-ci.
4096	E_RECOVERABLE_ERROR (entier)	Erreur fatale qui peut être captée. Ceci indique qu'une erreur probablement dangereuse s'est produite, mais n'a pas laissé le moteur Zend dans un état instable.
8192	E_DEPRECATED (entier)	Alertes d'exécution. Activer cette option pour recevoir des alertes sur les portions de votre code qui pourraient ne pas fonctionner avec les futures versions.
16384	E_USER_DEPRECATED (entier)	Message d'alerte généré par l'utilisateur. Fonctionne de la même façon que E_DEPRECATED, mise à part que le message est généré par votre code PHP en utilisant la fonction trigger_error().
32767	E_ALL (entier)	Toutes les erreurs et alertes supportées sauf le niveau E_STRICT avant la version 5.4.0 de PHP.

Il y a deux choses importantes à noter :

1. les niveaux d'errreurs ERROR (E_ERROR, E_USER_ERROR, etc.) mettent fin au script
2. les niveaux d'erreurs E_USER_XXX sont des erreurs générées par l'utilisateur

L'utilisateur, lorsqu'il implémente une fonction peut très bien déclencher une erreur s'il le pense nécessaire. Par exemple, si j'implémente une fonction additionner() qui prend en paramètre deux entiers, je peux très bien décider que le passage d'un argument d'un autre type déclenchera une erreur, et que cette erreur sera d'un niveau très élevé : ERROR. Je vais donc, au sein de ma fonction, déclencher une erreur de type E_USER_ERROR en appelant la fonction trigger_error().

```php
<?php

// erreurs_1.php
function additionner($nb1, $nb2) {
    if (is_integer($nb1) && is_integer($nb2)) {
        return $nb1 + $nb2;
    } else {
        trigger_error('argument invalide', E_USER_ERROR);
    }
}

echo additionner(10, 20) . PHP_EOL; // affiche 30 puis un saut de ligne
echo additionner('azerty', 45); // erreur et fin du script !
echo 'après'; // ne sera pas exécutée
```

Il est possible, mais relativement déconseillé, de bloquer l'affichage de l'erreur en préfixant le nom de la fonction incriminée par un @ :

```php
<?php

// erreurs_2.php
function additionner($nb1, $nb2) {
    if (is_integer($nb1) && is_integer($nb2)) {
        return $nb1 + $nb2;
    } else {
        trigger_error('argument invalide', E_USER_ERROR);
    }
}

echo additionner(10, 20) . PHP_EOL; // affiche 30 puis un saut de ligne
echo @additionner('azerty', 45); // erreur et fin du script !
echo 'après'; // ne sera pas exécutée
```

Ou encore d'appeler une autre fonction en cas d'erreur (le niveau d'erreur est abaissé à NOTICE pour ne pas interrompre le script) :

```php
<?php

// erreurs_3.php
function additionner($nb1, $nb2) {
    if (is_integer($nb1) && is_integer($nb2)) {
        return $nb1 + $nb2;
    } else {
        trigger_error('argument invalide', E_USER_NOTICE);
    }
}

echo additionner(10, 20) . PHP_EOL; // affiche 30 puis un saut de ligne
echo additionner('azerty', 45) or print('oups');
echo 'après'; // cette fois sera exécutée
```

Nous allons créer une fonction qui assurera le traitement de mon erreur, puis je vais enregistrer cette fonction auprès du gestionnaire d'erreur PHP. Lorsqu'une erreur se présentera, le gestionnaire d'erreurs regardera quelle est la fonction à appeler selon le niveau d'erreur et appellera la fonction adaptée. Je peux créer une fonction de traitement d'erreur pour un niveau ou pour tous les niveaux d'erreur. La fonction de traitement de l'erreur va être passée en argument à une autre fonction et va être appelée au bon moment par cette dernière. La fonction de traitement de l'erreur est ici utilisée comme un callback, ou une fonction de rappel.

```php
<?php

// erreurs_4.php
// fonction de traitement de l'erreur
function my_handler($errno, $errstr) {
    echo "un problème est survenu : $errstr";
}

// enregistrement de la fonction pour les E_USER_NOTICE dans le gestionnaire
set_error_handler('my_handler', E_USER_NOTICE);

function additionner($nb1, $nb2) {
    if (is_integer($nb1) && is_integer($nb2)) {
        return $nb1 + $nb2;
    } else {
        trigger_error('argument invalide', E_USER_NOTICE);
    }
}

echo additionner('azerty', 45);
```

La gestion des erreurs peut être configurée dans le fichier php.ini via le directives :

- display_errors : activation des erreurs
- error_reporting : niveau des erreurs signalées
- log_errors : active le log des erreurs
- error_log : chemin vers le fichier de log

Fin

ADMINISTRATION

Administration du cours

Réglages de mon profil

Connecté sous le nom « Arnaud Lemais » (Déconnexion)
Intro PHP