Programmation orientée objet en PHP

La classe Exception

Nous avons vu précédemment que certaines fonctions pouvaient, en cas de problème, déclencher des erreurs de différents niveaux selon l'importance du dysfonctionnement. Il existe un mécanisme semblable en programmation objet : les exceptions. Lorsqu'elle rencontre un problème, une méthode peur créer un objet de type Exception et "jeter" cette exception. La classe Exception possède 4 attributs qui donnent des informations utiles sur le problème rencontré :

- message → le message de l'exception
- code → le code de l'exception
- file → le nom du fichier dans lequel l'exception a été créée
- line → la ligne où l'exception a été créée

La valeur de chaque attribut peut être récupéré via son getter. Par exemple, nous allons créer une classe Calculatrice avec une méthode diviser(). Cette méthode prendra en argument 2 entiers (op1 et op2), si op2 est égal à 0 alors la méthode créera une exception et la "jettera". Pour "jeter" une exception nous utiliserons le mot clé "throw". L'exécution de l'instruction throw met fin à la méthode : en cas d'exception, les instructions après le throw ne seront pas exécutées.

```php
<?php

// Calculatrice.php
class Calculatrice {

    public function diviser($op1, $op2) {
        if ($op2 === 0) {
            throw new Exception('Division par zéro impossible !');
        }
        return $op1 / $op2;
    }

}
```

```php
<?php

// lanceur.php
include './Calculatrice.php';
$calc = new Calculatrice();
echo $calc->diviser(12, 0);
```

Ce code déclenchera une erreur fatale due au déclenchement d'une exception.

Déclencher des exceptions c'est bien, gérer les problèmes c'est mieux. Une exception jetée par une méthode peut être propagée, ou captée et gérée dans un un bloc try/catch. Le bloc try/ctach signifie "essaie d'exécuter ce code, si une exception est jetée alors capte là et exécute le code contenu dans le bloc catch". Notez qu'ici, c'est celui qui appelle la méthode qui doit gérer le problème, le bloc try/catch sera donc dans la méthode, ou le script, qui appelle la méthode susceptible de déclencher une exception.
Nous allons gérer dans notre lanceur l'exception déclenchée par la méthode diviser() en affichant simplement un message ; mais nous aurions pu effectuer d'autres traitements (saisie d'une nouvelle valeur pour op2, initialisation par défaut du nombre op2, etc...). Ici la méthode qui déclenche l'exception est diviser(), celui qui appelle cette méthode c'est le lanceur, le bloc try/catch sera donc dans le lanceur.

```php
<?php
```

```php

// lanceur.php
include './Calculatrice.php';
$calc = new Calculatrice();
try {
    echo $calc->diviser(12, 0);
} catch (Exception $ex) {
    echo 'oups...';
}
```

Dans le bloc catch la variable $ex est une référence vers l'objet Exception qui a été projeté. Nous pouvons donc à partir de cette référence accéder aux attributs de Exception via des getters. Nous pourrions par exemple accéder au message de l'exception :

```php
<?php

// lanceur.php
include './Calculatrice.php';
$calc = new Calculatrice();
try {
    echo $calc->diviser(12, 0);
} catch (Exception $ex) {
    echo "message de l'exception : {$ex->getMessage()}";
}
```

Si une exception déclenchée par une méthode n'est pas traitée dans un bloc try/catch alors elle se propage au code appelant jusqu'à être gérée dans un bloc try/catch. Si finalement cette exception n'est pas gérée, PHP déclenche une erreur.
Par exemple si une méthode C appelle une méthode B qui appelle elle-même une méthode A. Si A jette une exception et que B ne la gère pas dans un bloc try/catch alors C devra la gérée pour ne pas déclencher d'erreur.

Il est tout à fait possible de créer des Exceptions personnalisées en héritant de la classe Exception.

Fin

Réglages de mon profil

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