Développement PHP
Partie 2 : Bases 2 de PHP

DENIS LOEUILLET – IFA - 2017

Partie 2 : Bases 2 de PHP

- Les erreurs en PHP
- Gestion des erreurs
- Débugage
- Inclure des portions de pages
- Coder proprement

Partie 2 : Bases 2 de PHP
Les erreurs en PHP

http://php.net/manual/fr/book.errorfunc.php

- On ne réussit jamais un script du premier coup
- PHP nous aide à déterminer l'emplacement des erreurs
  - PHP peut afficher les erreurs si on le lui demande
  - Une fois le site en ligne mieux vaut les désactiver pour des raisons de sécurité et d'esthétique

Partie 2 : Bases 2 de PHP
Les erreurs en PHP

1. Les erreurs les plus courantes
2. Empêcher tout affichage

Partie 2 : Bases 2 de PHP
Les erreurs en PHP : les erreurs les plus courantes

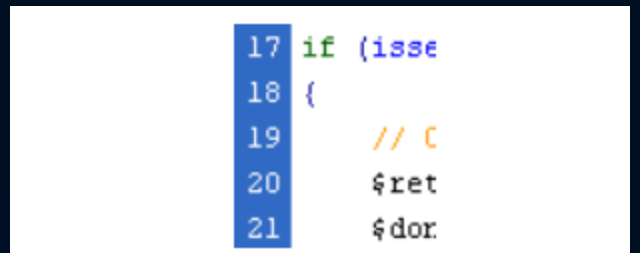
- Nous allons passer en revue les erreurs suivantes :
  - « *Parse error* »
  - « *Undefined function* »
  - « *Wrong parameter count* »

Partie 2 : Bases 2 de PHP
Les erreurs en PHP : les erreurs les plus courantes

- « Parse error »
  - On peut considérer cette erreur comme l'erreur de base
  - Impossible de programmer en PHP sans y avoir droit un jour
  - Le message d'erreur obtenu ressemble à celui-ci :

```
1  Parse error: parse error in fichier.php on line 15
```

    - Ce message vous indique une erreur :
      - dans fichier.php
      - à la ligne 15.
    - Généralement, cela veut dire que votre problème se situe à la ligne 15, mais ce n'est pas toujours le cas
    - Parfois c'est la ligne précédente qui a un problème : pensez donc à regarder autour de la ligne indiquée.
    - Les éditeurs spécialisés indiquent les numéro des lignes sur la gauche du script :



Numérotation des lignes dans Notepad++

Partie 2 : Bases 2 de PHP
Les erreurs en PHP : les erreurs les plus courantes

- « Parse error »
  - En fait, une « parse error » est une instruction PHP mal formée
  - Il peut y avoir plusieurs causes :
    - Vous avez oublié le point-virgule à la fin de l'instruction. Comme toutes les instructions doivent se terminer par un point-virgule, si vous oubliez d'en mettre un, ça provoquera une « parse error »
    - Vous avez oublié de fermer un guillemet (ou une apostrophe, ou une parenthèse)
    - Vous vous êtes trompés dans **la concaténation,** vous avez peut-être oublié un point
    - Il peut aussi s'agir d'une accolade mal fermée (pour un « if », par exemple). Vérifiez que vous avez correctement fermé toutes vos accolades. Si vous oubliez d'en fermer une, il est probable que la « parse error » vous indique que l'erreur se trouve à la dernière ligne du fichier (c'est-à-dire à la ligne 115 si votre fichier comporte 115 lignes).

Donc, si on vous indique une erreur à la dernière ligne, il va probablement falloir relire tout le fichier PHP à la recherche d'une accolade mal fermée !

Si on vous dit que l'erreur est à la ligne 15 et que vous ne voyez vraiment pas d'erreur à cette ligne, n'hésitez pas à chercher l'erreur à la ligne juste au-dessus, elle s'y trouve peut-être !

Partie 2 : Bases 2 de PHP
Les erreurs en PHP : les erreurs les plus courantes

- « Undefined function »
  - Une autre erreur assez classique : la fonction inconnue. Vous obtenez ce message d'erreur :

```
1 Fatal Error: Call to undefined function: fonction_inconnue() in fichier.php on line 27
```

    - Ça signifie que vous avez utilisé une fonction qui n'existe pas ou qui n'a pas encore été définie
      - soit la fonction n'existe vraiment pas
      - soit la fonction existe vraiment, mais PHP ne la reconnaît pas.
        - C'est parce que cette fonction se trouve dans une extension de PHP que vous n'avez pas activée.
          Par exemple, si vous essayez d'utiliser la fonction « imagepng » alors que vous n'avez pas activé la bibliothèque GD pour les images en PHP, on vous dira que la fonction n'existe pas. Activez la bibliothèque qui utilise la fonction et tout sera réglé.

Remarque :
Il se peut aussi que vous essayiez d'utiliser une fonction qui n'est pas disponible dans la version de PHP que vous avez.
Vérifiez dans le manuel dans quelles versions de PHP cette fonction est disponible.

Partie 2 : Bases 2 de PHP
Les erreurs en PHP : les erreurs les plus courantes

- « Wrong parameter count »
  - Si vous utilisez mal une fonction, vous aurez cette erreur :

```
1  Warning: Wrong parameter count for fonction() in fichier.php on line 112
```

    - Cela signifie que vous avez oublié des paramètres pour la fonction, ou même que vous en avez trop mis.
    - Consultez le mode d'emploi de la fonction pour savoir combien de paramètres elle prend et quels sont ceux qui sont facultatifs.
    - Dans les versions actuelles de PHP, le message d'erreur vous donne même le nombre de paramètres que vous avez oubliés !

```php
1  <?php
2  $search_array = array('premier' => 1, 'second' => 4);
3  if (array_key_exists('premier')) {
4      echo "L'élément 'premier' existe dans le tableau";
5  }
```

Warning: array_key_exists() expects exactly 2 parameters, 1 given in **/homepages/31/d218891204/htdocs/clickandbuilds/LMIR/test.php** on line **4**

Partie 2 : Bases 2 de PHP
Les erreurs en PHP : empêcher tout affichage

- Il existe une méthode très simple, mais très efficace pour éviter l'affichage des erreurs, qui est d'appeler la fonction « error_reporting() » en lui passant o en argument, argument servant à indiquer le niveau de rapport d'erreurs.

```php
<?php
error_reporting(0);

// On affiche une variable non initialisée
// ce qui devrait provoquer une notice.
echo $var;

// On divise par 0,
// ce qui devrait provoquer un avertissement.
$var2 = 5/0;

// On inclut un fichier qui n'existe pas,
// ce qui devrait provoquer un avertissement et une erreur fatale.
require('non-existing');
```

Partie 2 : Bases 2 de PHP
Les erreurs en PHP : empêcher tout affichage

- Vous pouvez aussi directement changer la valeur de error_reporting dans le *php.ini*, si vous y avez accès et si vous voulez supprimer l'affichage des erreurs par défaut.
- Si vous voulez empêcher l'affichage d'une erreur pour une fonction en particulier, vous pouvez vous contenter de la faire précéder d'un « @ ».

```php
<?php
error_reporting(E_ALL);

// On affiche une variable non initialisée
// ce qui devrait provoquer une notice.
echo $var;

// On divise par 0,
// ce qui devrait provoquer un avertissement.
$var2 = 5/0;

// On inclut un fichier qui n'existe pas,
// ce qui devrait provoquer un avertissement et une erreur fatale.
if(!@include '/non-existing') {
    echo 'Le fichier n\'existe pas !';
}
```

Partie 2 : Bases 2 de PHP
Gestion des erreurs

Les solutions précédentes sont est efficaces, mais le visiteur ne comprendra pas nécessairement pourquoi l'action n'a pas été effectuée correctement, et c'est là que la gestion des erreurs intervient,

Partie 2 : Bases 2 de PHP
Gestion des erreurs

1. Votre propre fonction de gestion des erreurs
2. Déclencher vos erreurs
3. Pour la route

Partie 2 : Bases 2 de PHP
Gestion des erreurs

1. Votre propre fonction de gestion des erreurs

- Pour créer notre propre fonction de gestion des erreurs nous allons utiliser :
  - La fonction « set_error_handler() » qui va :
    http://php.net/manual/fr/function.set-error-handler.php
    - ignorer le rapport d'erreurs standard de PHP
    - et définir une fonction de callback qui gérera les erreurs ;
    - en particulier, il vous sera possible d'en déclencher grâce à « trigger_error() » (que nous verrons plus loin).
- Une fonction de callback est définie de la manière suivante sur Wikipédia :

  « En informatique, une fonction de rappel (callback en anglais) est une fonction qui est passée en argument à une autre fonction. Cette dernière peut alors faire usage de cette fonction de rappel comme de n'importe quelle autre fonction, alors qu'elle ne la connaît pas par avance. »

Partie 2 : Bases 2 de PHP
Gestion des erreurs : votre propre fonction

- En PHP, il existe plusieurs sortes de fonctions et pour chacune, la syntaxe de son appel en tant que fonction de callback est différente
- Nous allons en étudier une sorte : les fonctions simples
  - La syntaxe pour appeler en tant que fonction de callback une fonction simple est la suivante :
    - on passe le nom de la fonction comme une chaîne de caractères

```php
<?php
set_error_handler('ma_fonction');
?>

```

    - C'est donc la fonction « ma_fonction () » qui sera utiliser pour gérer les erreurs

Partie 2 : Bases 2 de PHP
Gestion des erreurs : votre propre fonction

- La fonction de gestion des erreurs doit avoir au moins 2 arguments :
  - le type d'erreur (il existe des constantes que nous verrons plus loin)
  - le message de l'erreur.
- Il existe 3 arguments optionnels :
  - le nom du fichier dans laquelle l'erreur a été rencontrée ;
  - la ligne de l'erreur dans le fichier ;
  - le contexte de l'erreur ;

Remarque :
Je ne me servirai pas du dernier, mais libre à vous d'aller voir la doc pour plus d'informations sur ce paramètre.

Partie 2 : Bases 2 de PHP
Gestion des erreurs : votre propre fonction

- Exemple :

```php
<?php
function my_error_handler($no, $str, $file, $line){
    echo '<p>Erreur ['.$no.'] : '.$str.'<br/>';
    echo 'Survenue dans le fichier : "'.$file.'" à la ligne '.$line.'.</p>';
}
?>
```

- $no contient en fait le type de l'erreur sous forme numérique, chacune de ces valeurs correspond à une constante ; une liste exhaustive de ces constantes est disponible dans la doc : http://php.net/manual/fr/errorfunc.constants.php.
- Dans cette liste, vous pouvez voir apparaître 3 constantes particulières : E_USER_ERROR, E_USER_WARNING et E_USER_NOTICE : générées par le programmeur
- Ces constantes sont celles que nous manipulerons et que nous enverrons à la fonction de gestion d'erreurs à l'aide de « trigger_error() ». Ainsi, nous pouvons définir des comportements différents de la fonction suivant le type d'erreur.

```php
<?php
function my_error_handler($no, $str, $file, $line){
    switch($no){
        // Erreur fatale
        case E_USER_ERROR:
            echo '<p><strong>Erreur fatale</strong> : '.$str.'</p>';
            exit;//on arrete le script
            break;

        // Avertissement
        case E_USER_WARNING:
            echo '<p><strong>Avertissement</strong> : '.$str.'</p>';
            break;

        // Note
        case E_USER_NOTICE:
            echo '<p><strong>Note</strong> : '.$str.'</p>';
            break;

        // Erreur générée par PHP
        default:
            echo '<p><strong>Erreur inconnue</strong> ['.$no.'] : '.$str.'<br/>';
            echo 'Dans le fichier : "'.$file.'", à la ligne '.$line.'.</p>';
            break;
    }
}
```

Partie 2 : Bases 2 de PHP
Gestion des erreurs

2. Déclencher vos erreurs

- La fonction qui déclenche les erreurs est « trigger_error() »
  http://php.net/manual/fr/function.trigger-error.php
  (qui se traduit d'ailleurs par « déclencher erreur ») ; celle-ci est très simple d'utilisation
- trigger_error() nécessite 2 arguments :
  - le message d'erreur ;
  - le type d'erreur (optionnel, par défaut E_USER_NOTICE).
- Le nom du fichier et le numéro de la ligne sont transmis automatiquement.
- Vous ne pouvez déclencher que des erreurs de la famille E_USER, ou vous aurez droit à un warning

```php
set_error_handler('my_error_handler');

if(empty($_GET['empty'])){
    trigger_error('Vous devez préciser la valeur de "empty" ou une valeur nulle lui sera imposée.', E_USER_NOTICE);
    $empty = 0;
}

if(empty($_GET['div']))
    trigger_error('Vous ne pouvez pas diviser par 0', E_USER_WARNING);
else
    echo (5/$_GET['div']);

if(!is_file('non-existing'))
    trigger_error('Un fichier indispensable à l\'exécution du script est manquant.', E_USER_ERROR);
?>
```

Partie 2 : Bases 2 de PHP
Gestion des erreurs : pour la route

3. Pour la route
   - Quelques fonctions également très utiles :
     - « error_log() »
       http://php.net/manual/fr/function.error-log.php
       Vous permet d'enregistrer un message dans l'historique de PHP, dans un fichier, ou de vous l'envoyer par mail ;
     - « restore_error_handler() »
       http://php.net/manual/fr/function.restore-error-handler.php
       Permet de réutiliser l'ancienne version de gestion des erreurs (qui peut être la fonction PHP par défaut, ou une autre fonction utilisateur) ;
     - « user_error »
       http://php.net/manual/fr/function.user-error.php
       est un alias de trigger_error
     - « error_get_last() »
       http://php.net/manual/fr/function.error-get-last.php
       (PHP >= 5.2.0) renvoie un array contenant les informations à propos de la dernière erreur survenue.