Le traitement des erreurs et les exceptions

Langage PHP avancé

El hadji Mamadou NGUER Enseignant chercheur en Informatique à l'UVS

Séquence 4 : Le traitement des erreurs et les exceptions

Objectifs spécifiques : A la suite de cette séquence, l'étudiant doit être capable de :

1. Traiter, avec des méthodes personnalisées, les erreurs produites dans une application web
2. Utiliser les exceptions pour modifier le flux normal de l'exécution d'un code si une condition d'erreur exceptionnelle se produit.

Séquence 4 : Le traitement des erreurs et les exceptions

Plan de la séquence :

1. Introduction
2. Le traitement des erreurs
3. Les exceptions
4. Conclusion

Introduction

- Le traitement des erreurs est une partie importante dans la création de scripts et d'applications Web,

- Si votre code manque de vérification d'erreur, votre programme peut être très professionnel mais sera ouvert aux risques de sécurité.

- Par défaut la gestion des erreurs dans PHP est très simple. Un message d'erreur avec le nom du fichier, un numéro de ligne et un message décrivant l'erreur sont envoyés au navigateur.

- On verra dans la suite différentes méthodes de traitement des erreurs:

  - La déclaration simple avec la fonction die()
  - La personnalisation des erreurs et les déclencheurs d'erreur
  - Le rapport d'erreur

Le traitement des erreurs

Gestion des erreurs avec die():

- Supposons qu'on veuille ouvrir le fichier inexistant test.txt :

```
<?php
    $file=fopen("test.txt","r");
?>
```

- On obtient l'erreur suivant :

Warning : fopen(test.txt) [function.fopen]: failed to open stream: No such file or directory in **C:\webfolder\test.php** on line **2**

- Pour éviter à l'utilisateur d'avoir un tel message d'erreur, on teste si le fichier existe avant d'essayer d'y accéder:

```
<?php
    if(!file_exists("test.txt")) {
        die("Fichier introuvable");
    } else {
        $file=fopen("test.txt","r");
    }
?>
```

Le traitement des erreurs

Gestionnaire d'erreurs personnalisée

- Pour créer un gestionnaire d'erreurs personnalisée on crée simplement une fonction spéciale qui peut être appelée lorsqu'une erreur se produit.
- Cette fonction doit être capable de gérer au moins deux paramètres (niveau d'erreur et message d'erreur) mais peut accepter jusqu'à 5 paramètres (facultativement: fichier, numéro de ligne et contexte d'erreur)

Syntaxe : fonctErreur (error_level, error_message, error_file, error_line, error_context)

Paramètre	Description
error_level	Obligatoire. Spécifie le niveau du rapport d'erreur pour l'erreur définie par l'utilisateur. Doit être un nombre. Voir le tableau ci-dessous pour connaître les niveaux de rapport d'erreur possibles
error_message	Obligatoire. Spécifie le message d'erreur pour l'erreur définie par l'utilisateur
error_file	Optionnel. Spécifie le nom du fichier dans lequel l'erreur s'est produite
error_line	Optionnel. Spécifie le numéro de ligne dans lequel l'erreur s'est produite
error_context	Optionnel. Spécifie un tableau contenant toutes les variables et leurs valeurs utilisées lors de l'erreur

Le traitement des erreurs

Niveau de rapport d'erreur

- Ces niveaux de rapport d'erreur sont les différents types d'erreur auxquels le gestionnaire d'erreur défini par l'utilisateur peut être utilisé :

Valeur	Constante	Description
2	E_WARNING	Erreurs d'exécution non fatales. L'exécution n'est pas arrêtée
8	E_NOTICE	Notice d'exécution. Le script a trouvé quelque chose qui pourrait être une erreur, mais pourrait également se produire lors de l'exécution normale d'un script.
256	E_USER_ERROR	Erreur fatale généré par l'utilisateur. C'est comme un E_ERROR défini par le programmeur en utilisant la fonction PHP trigger_error ()
512	E_USER_WARNING	Avertissement non fatal généré par l'utilisateur. C'est comme un E_WARNING mis par le programmeur en utilisant la fonction PHP trigger_error ()
1024	E_USER_NOTICE	Notice généré par l'utilisateur C'est comme un E_NOTICE mis par le programmeur en utilisant la fonction PHP trigger_error ()
4096	E_RECOVERABLE_ERROR	Erreur fatale capturable. C'est comme un E_ERROR mais peut être attrapé par un handle défini par l'utilisateur (voir aussi set_error_handler ())
8191	E_ALL	Toutes les erreurs et avertissements (E_STRICT est devenu une partie de E_ALL en PHP 5.4)

Le traitement des erreurs

Niveau de rapport d'erreur

- Exemple de fonction de gestion d’erreurs:

```
function fonctErreur($errno, $errstr) {
  echo "<b>Erreur:</b> [$errno] $errstr<br>";
  echo "Fin du script";
  die();
}
```

- Le code ci-dessus est une simple fonction de gestion d’erreurs. Lorsqu’elle est déclenchée, elle récupère et affiche le niveau d'erreur et un message d'erreur et ensuite met fin au script.

- Une fois qu’une fonction de gestion des erreurs est créée, il faut décider quand elle doit être déclenchée.

Le traitement des erreurs

Définir le gestionnaire d'erreurs

- Le gestionnaire d'erreur par défaut est le gestionnaire d'erreur intégré.
- Il est possible de le modifier pour ne présenter que certaines erreurs, de sorte que le script peut gérer différentes erreurs de différentes manières.
- Pour utiliser notre gestionnaire d'erreurs personnalisé pour toutes les erreurs, on le met de la manière suivante :

```
set_error_handler("customError");
```

- Comme nous voulons que notre fonction personnalisée gère toutes les erreurs, le set_error_handler () n'a besoin que d'un paramètre, un second paramètre pourrait être ajouté pour spécifier un niveau d'erreur.

Le traitement des erreurs

Définir le gestionnaire d'erreurs

Exemple : Test du gestionnaire d'erreur en essayant de générer une variable inexistante:

```
<?php

  //Fonction de gestion d’erreur
  function customError($errno, $errstr) {
    echo "<b>Erreur:</b> [$errno] $errstr";
  }

  //Définir le gestionnaire d’erreur
  set_error_handler("customError");

  //Erreur de déclenchement.
  echo ($test);

  ?>
```

Résultat généré : **Erreur:** [8] Undefined variable: test

Le traitement des erreurs

Déclenchement d'une erreur

Dans un script où les utilisateurs peuvent entrer des données, il est utile de déclencher des erreurs lorsqu'une entrée illégale se produit. En PHP, cela se fait par la fonction **trigger_error ()**.

Exemple : Dans cet exemple, une erreur se produit si la variable "test" est supérieure à "1":

```
<?php
$test=2;
if ($test>=1) {
  trigger_error("La valeur doit être égale ou inférieure à 1");
}
?>
```

La sortie du code ci-dessus devrait être comme ceci:

Notice : La valeur doit être égale ou inférieure à 1
in **C:\webfolder\test.php** on line **6**

Le traitement des erreurs

Déclenchement d'une erreur

- Une erreur peut être déclenchée n'importe où dans un script, et en ajoutant un deuxième paramètre, on peut spécifier le niveau d'erreur.

Types d'erreurs possibles:

E_USER_ERROR - Erreur d'exécution fatale générée par l'utilisateur. Erreurs qui ne peuvent être récupérées. L'exécution du script est interrompue

E_USER_WARNING - Avertissement de temps d'exécution non fatal généré par l'utilisateur. L'exécution du script n'est pas interrompue

E_USER_NOTICE - Par défaut. Avis d'exécution généré par l'utilisateur.
Le script a trouvé quelque chose qui pourrait être une erreur, mais pourrait également se produire lors de l'exécution d'un script normalement

Le traitement des erreurs

Déclenchement d'une erreur

Exemple:

```
<?php  //Fonction de gestion d’erreur
function customError($errno, $errstr) {
  echo "<b>Erreur:</b> [$errno] $errstr<br>";
  echo "Fin de script";
  die();
}

// Définir le gestionnaire d’erreur
set_error_handler("customError",E_USER_WARNING);

// Erreur de déclenchement.
$test=2;
if ($test>=1) {
  trigger_error("La valeur doit être <= 1",E_USER_WARNING);
}
?>
```

Le traitement des erreurs

Enregistrement des erreurs

- Par défaut, PHP envoie un journal d'erreur au système de journalisation du serveur ou à un fichier, selon la configuration de error_log est définie dans le fichier php.ini.

- En utilisant la fonction error_log (), on peut envoyer des journaux d'erreur à un fichier spécifié ou à une destination distante.

- L'envoi de messages d'erreur par courrier électronique peut être une bonne manière d'être informé des erreurs spécifiques.

Le traitement des erreurs

Envoyer un message d'erreur par e-mail

- L'exemple ci-dessous envoie un courrier électronique avec un message d'erreur et finit le script si une erreur spécifique se produit:

- <?php **//Fonction de gestion d'erreur**
function customError($errno, $errstr) {
  echo "Erreur: [$errno] $errstr
";
  echo "Le Webmaster a été notifié";
  error_log("Erreur: [$errno] $errstr",1,
  "alaxam@exemple.com","From: webmaster@exemple.com");
}

// Définir le gestionnaire d'erreur
set_error_handler("customError",E_USER_WARNING);

// Erreur de déclenchement.
$test=2;
if ($test>=1) {
  trigger_error("La valeur doit être égale ou inférieure à 1",E_USER_WARNING);
}
?>

Le traitement des erreurs

Envoyer un message d'erreur par e-mail

- La sortie du code ci-dessus devrait être comme ceci:
  Erreur: [512] La valeur doit être égale ou inférieure à 1
  Le Webmaster a été notifié

- Et le courrier reçu ressemble à ceci:
  Erreur: [512] La valeur doit être égale ou inférieure à 1

- Cela ne doit pas être utilisé avec toutes les erreurs. Des erreurs régulières doivent être enregistrées sur le serveur en utilisant le système de journalisation PHP par défaut.

Les exceptions

Qu'est-ce qu'une exception?

- Les exceptions sont utilisées pour modifier le flux normal de l'exécution d'un code si une condition d'erreur (exceptionnelle) spécifiée se produit. Cette condition s'appelle une exception.
- Lorsqu'une exception est déclenchée:
  - L'état actuel du code est enregistré
  - L'exécution du code passera à une fonction de gestion des exceptions prédéfinie (personnalisée)
  - Selon la situation, le gestionnaire peut alors reprendre l'exécution à partir de l'état du code enregistré, terminer l'exécution du script ou continuer le script à partir d'un emplacement différent dans le code
- **On verra dans la suite :**
  - L'utilisation basique des exceptions
  - La création d'un gestionnaire d'exception personnalisé
  - Les exceptions multiples
  - La relance d'une exception
  - La définition d'un gestionnaire d'exception de niveau supérieur

Les exceptions

Utilisation basique des exceptions

- Lorsqu'une exception est lancée, le code qui suit ne sera pas exécuté, et PHP tentera de trouver le bloc "catch" correspondant.
- Si une exception n'est pas détectée, une erreur fatale sera émise avec un message "Exception non détectée".
- **Exemple :**

```
<?php  //Fonction avec une exception
 function checkNum($number) {
   if($number>1) {
     throw new Exception("La valeur doit être <= 1");
   }
   return true;
 }

//Déclenchement de l'exception
 checkNum(2);
 ?>
```

Résultat obtenu :

Fatal error : Uncaught exception 'Exception' with message 'La valeur doit être <= 1' in C:\webfolder\test.php:6
Stack trace: #0 C:\webfolder\test.php(12): checkNum(28) #1 {main} thrown in **C:\webfolder\test.php** on line **6**

Les exceptions

Utilisation de try, throw et catch

Un code d'exception approprié devrait inclure les clauses :

- **try** - Une fonction utilisant une exception devrait être dans un bloc "try". Si l'exception n’est pas déclenchée, le code se poursuivra normalement. Sinon une exception est lancée avec "throw".

- **throw** - C'est la clause qui déclenche une exception. Chaque clause "throw" doit avoir au moins une clause "catch".

- **catch** - Un bloc "catch" récupère une exception et crée un objet contenant les informations d'exception.

Les exceptions

Utilisation de try, throw et catch

- **Exemple :** Le code ci-dessous lance une exception et la capture:

```
<?php
  function checkNum($number) {//Fonction avec une exception
    if($number>1) {
      throw new Exception("Le nombre doit être  <= 1");
    }
    return true;
  }

  try {//Declencher une exception dans un bloc "try"
    checkNum(2);
    // Si l'exception est lancée, ce texte ne sera pas affiché
    echo 'Si vous voyez ceci, le nombre est inférieur ou égal à 1';
  }catch(Exception $e) {//Capturer une exception
    echo 'Message: ' .$e->getMessage();
  }
  ?>
```

Les exceptions

Utilisation de try, throw et catch

- **Explication de l’exemple:**

1. La fonction checkNum () est créée. Elle vérifie si un nombre est supérieur à 1. Si c'est le cas, une exception est lancée
2. La fonction checkNum () est appelée dans un bloc "try"
3. L'exception dans la fonction checkNum () est lancée
4. Le bloc "catch" récupère l'exception et crée un objet ($e) contenant les informations d'exception
5. Le message d'erreur de l'exception est affiché avec echo en appelant $ e-> getMessage () à partir de l'objet d'exception

Cependant, une façon de contourner la règle «chaque lancée doit avoir une capture» consiste à définir un gestionnaire d'exception de haut niveau pour gérer les erreurs qui s’y glissent.

Conclusion

- Dans cette séquence, nous avons vu :
  1. Le traitement des erreurs
  2. Les exceptions

Il vous reste maintenant effectuer les tests de connaissance et les travaux pratiques de la séquence pour consolider vos acquis.