Fiche de révision

Les exceptions en PHP

1. Qu'est-ce que c'est ?

L'utilisation des exceptions en PHP est une méthode puissante et structurée pour gérer les erreurs et les conditions exceptionnelles dans les applications.

Une exception est un objet qui représente une erreur ou une condition anormale dans un programme. En PHP, les exceptions sont des instances de la classe Exception ou de ses sous-classes.

Voici les caractéristiques principales des exceptions en PHP :

- **Message d'Erreur** : Chaque exception peut transporter un message d'erreur qui peut être défini lors de la création de l'exception. Ce message est destiné à fournir une description claire de l'erreur et peut être récupéré en utilisant la méthode getMessage().
- **Code d'Erreur** : Les exceptions peuvent également inclure un code d'erreur, un numéro entier qui peut servir à catégoriser les types d'erreurs. Le code peut être passé en deuxième argument au constructeur de l'exception et récupéré avec la méthode getCode().
- **Fichier et Ligne** : Les exceptions en PHP stockent automatiquement le nom du fichier et le numéro de ligne où l'exception a été lancée. Ces informations peuvent être extraites à l'aide des méthodes getFile() et getLine().
- **Trace de Pile** : PHP fournit une trace de pile (stack trace) au moment où l'exception est lancée, qui peut être récupérée en utilisant la méthode getTrace(). Cette trace de pile peut être très utile pour déboguer en montrant l'ordre des appels de fonctions qui ont mené à l'exception.

Les exceptions offrent un moyen structuré de gérer les erreurs. En utilisant les blocs **try**, **catch**, et **finally**, les développeurs peuvent écrire du code qui réagit de manière appropriée aux erreurs, tout en maintenant le reste du programme fonctionnel.

Au début du bloc « try », l'état de l'exécution du code est sauvegardé. Ensuite le code est exécuté. En cas d'exception levée, le code revient à l'état avant exécution du bloc « try » et ensuite exécuter le bloc cach puis finally.

Si une exception n'est pas attrapée dans un bloc catch, elle se propage vers le haut dans la pile d'appels jusqu'à ce qu'elle soit attrapée. Si elle n'est jamais attrapée, le script s'arrêtera et PHP affichera un message d'erreur. C'est la cause des « FATAL ERROR » et « ERREUR 500 ».

> ***« Ne commentez pas un mauvais code, réécrivez-le » Brian Kernighan, Co-écrivain du premier livre sur le langage C***

2. Comment faire ?

2.1. Lancer une exception

Les exceptions peuvent être déclenchées automatiquement (quand on fait une bétise dans le code !). Vous pouvez aussi "lancer" (`throw`) une exception.

Lorsqu'une exception est lancée volontairement ou non, le flux normal du programme est interrompu.

Ici, nous avons décidé de détecter un paramètre incorrect et de lancer une exception nous-même. En effet, on ne peut pas diviser par zéro !

2.2. Attraper une exception

Pour gérer une exception, vous devez **entourer** le code susceptible de lancer une exception dans un bloc try, suivi d'un ou de plusieurs blocs catch qui définissent comment réagir à différentes exceptions.

```
try {
    echo diviser(0);
} catch (Exception $e) {
    echo "Erreur capturée : " . $e->getMessage();
}
```

2.3. Que faire dans tous les cas ?

Le bloc finally est exécuté après les blocs try et catch, quel que soit le résultat. Il est souvent utilisé pour effectuer un nettoyage, comme fermer des fichiers ou des connexions à des bases de données, indépendamment de la survenue d'une exception.

```
try {
    echo diviser(2);
} catch (Exception $e) {
    echo "Erreur capturée : " . $e->getMessage();
} finally {
    echo "Bloc finally exécuté.";
}
```

2.4. Aller plus loin

Vous pouvez créer vos propres classes d'exception pour représenter des types d'erreurs spécifiques en étendant la classe Exception ou toute autre classe d'exception intégrée.

```
class DivisionParZeroException extends Exception {

…

}
```

3. Bonnes Pratiques

Voici quelques bonnes pratiques avec les exceptions.

- **Evitez les exceptions** : Préférez les tests de garde plutôt que les exceptions, elles sont la pour gérer les imprévus.
- **Utilisez des exceptions pour les conditions exceptionnelles** : Les exceptions ne doivent pas être utilisées pour contrôler le flux normal d'un programme.
- **Soyez spécifique avec les blocs catch** : Attrapez uniquement les exceptions que vous pouvez gérer de manière significative.
- **Propagez les exceptions si nécessaire** : Si votre méthode ne peut pas gérer une exception, laissez-la se propager.

C’est très utile pour gérer l’imprévu : par exemple, le serveur de base de données ne répond pas.