



## Php : la gestion des exceptions

Formation

**Développeur Web et Web Mobile**

Module : 04

**Développer la partie back-end d'une application web**

Séquence : 03

**Développer une interface utilisateur web dynamique**

Séance : 01


**Développer des scripts serveurs**

Libellé réduit:	PhpExcep
Type de document:	Ressource
Version:	1
Date de mise à jour:	19/12/2022

# Sommaire

## Sommaire

I	Introduction.....	1
II	la classe de base.....	1
II.1	Générer une exception .....	2
II.2	Lancer ou lever une exception.....	2
II.3	Attraper (intercepter) une exception .....	3
III	Créer ses propres classes d'exceptions .....	5
IV	Centraliser le traitement des exceptions.....	6

	Auteur	Centre de Créteil	Formation	Date Mise à jour	Page 1
	Didier Bonneau	GRN 164	DWWM	20/11/2020	cours php - les exceptions.docx

## I INTRODUCTION

PHP met à disposition du programmeur un mécanisme de gestion des exceptions. Il a été introduit à partir de PHP 5 en complément du modèle orienté objet.

Au même titre qu'en Java, C++ et même Javascript, les exceptions permettent de personnaliser et d'organiser la gestion des « erreurs » dans un programme informatique.

Ici le mot « erreurs » ne signifie pas « bug », qui est un comportement anormal de l'application développée, mais plutôt « cas exceptionnel » à traiter différemment dans le déroulement du programme.

## II LA CLASSE DE BASE

Depuis sa version 5 PHP dispose en natif d'une classe « **Exception** » capable de prendre en charge le traitement des cas exceptionnels susceptibles d'apparaître pendant l'exécution d'un programme en générant des objets de type « Exception ».

En fin de compte, une exception n'est rien de plus qu'une instance de cette classe que l'on va pouvoir construire et propager dans l'application.

*A titre indicatif :*

*Code source de la classe native Exception (extrait de la documentation officielle de PHP)*

*Exception*


```
{
    /* Propriétés */
    protected string $message; // message de l'exception
    protected int $code; // code de l'exception défini par
                           l'utilisateur
    protected string $file; // nom du fichier source de
                           l'exception
    protected int $line; // numéro de la ligne de la
                           source de l'exception

    /* Méthodes */
    public __construct([ string $message="" [, int $code=0
                           [, Exception $previous=NULL ]]]);

    final public string getMessage(void); // message de
                                           l'exception
    final public mixed getCode(void); // code de l'exception
    final public string getFile(void); // nom fichier source
    final public int getLine(void); // ligne du fichier source
    final public array getTrace(void); // tableau de trace
    final public string getTraceAsString(void); // chaîne
                                           formatée de trace

    final public Exception getPrevious ( void )
    final private void __clone(void) ;

    public string __toString(void); // chaîne pour affichage
}
```

	Auteur	Centre de Créteil	Formation	Date Mise à jour	Page 1
	Didier Bonneau	GRN 164	DWWM	20/11/2020	cours php - les exceptions.docx

## II.1 GENERER UNE EXCEPTION

La création d'une exception est réalisée par l'appel au constructeur de la classe « Exception ». Voir le code ci-contre.

## II.2 LANCER OU LEVER UNE EXCEPTION

Une exception n'est utile que si elle est créée lorsqu'un évènement exceptionnel se déroule pendant l'exécution du programme.

Par exemple : une requête SQL qui échoue, un fichier impossible à ouvrir, une valeur d'un formulaire inattendue pour un champ...

Lorsqu'un tel évènement se produit, c'est que quelque chose d'inhabituel s'est passé. Par conséquent, la poursuite de l'exécution du programme doit être interrompue et géré par du code particulier.

C'est là qu'interviennent réellement le mécanisme des exceptions.

Pour réaliser cette opération, le programme doit automatiquement « lancer » une exception qui va se propager à travers le programme et qui devra être « attrapée » afin de réaliser le traitement adéquate.

Le lancement d'une exception provoque immédiatement l'interruption du déroulement normal du programme.

Le lancement d'une exception à travers le programme est réalisée grâce au mot-clé « **throw** ».

### Exemple de création d'une instance d'Exception

```
<?php

// Création de l'objet Exception
$except = new Exception('Une erreur s\'est produite');

// Affiche le message d'erreur
echo $except->getMessage();

?>
```

### Exemple de lancement d'une Exception à travers le programme

```
<?php


$psw = $_POST['psw'];

if (!password_verify($psw, $pswBdd)) {
    throw new Exception('Votre mot de passe est incorrect !');
}

// L'instruction « echo » ci-dessous ne sera jamais exécutée
// si une exception est lancée.
// L'exécution normale du programme est interrompue

echo 'Bienvenu';

?>
```

	Auteur	Centre de Créteil	Formation	Date Mise à jour	Page 2
	Didier Bonneau	GRN 164	DWWM	20/11/2020	cours php - les exceptions.docx

## II.3 ATTRAPER (INTERCEPTER) UNE EXCEPTION

Les exceptions doivent permettre de traiter différemment les cas exceptionnels survenant au cours d'un programme. Pour cela, il est nécessaire de pouvoir « intercepter / attraper » l'exception générée pour appliquer le traitement adéquat.

C'est là qu'intervient le bloc « **try / catch** ».

La structure conditionnelle d'un bloc « try / catch » est la suivante :

```
try {
    Liste d'actions à appeler
    Ces actions peuvent potentiellement « lancer » des
    exceptions à travers le programme
}
catch (Exception $e)
{
    Bloc des actions spéciales à effectuer lorsqu'une
    exception de type Exception (précisé dans le
    catch) est levée
    Ici, l'objet Exception sera récupérée dans la variable
    « $e » donnée derrière le type de l'exception
}
```

Lorsqu'une exception est levée, elle remonte dans le programme et est interceptée par le premier bloc « try / catch » englobant qu'elle rencontre. L'exécution du bloc « catch » est alors réalisée puis les instructions situées après ce bloc sont exécutées normalement.

Si l'exception n'est pas levée, le programme continu normalement après le bloc « catch » et le bloc « catch » sera ignoré.

### Exemple de création d'une instance d'Exception

```
<?php
```

```
$psw = $_POST['psw'];
```

```
try {
```


```
    if ( ! password_verify($psw, $pswBdd)) {
        throw new Exception('Erreur de mot de passe !');
    }
```

```
echo 'Bienvenu';
```

```
}
```

```
catch (Exception $e) {
    echo 'L\'erreur suivante a été générée : ' . "\n";
    echo $e->getMessage();
}
```

```
?>
```

	Auteur	Centre de Créteil	Formation	Date Mise à jour	Page 3
	Didier Bonneau	GRN 164	DWWM	20/11/2020	cours php - les exceptions.docx

Il est possible d'imbriquer les blocs « try / catch » comme les instructions « if »

L'exemple ci-contre donne un exemple de vérification d'un login.

Récupération du champ de formulaire « psw » dans une variable psw.

- Si la variable psw est vide, une exception « Veuillez entrer le mot de passe » est levée et est interceptée par le bloc « catch » le plus bas (celui qui suit le premier bloc « try »).
- Si le psw n'est pas vide, le programme continu et tombe sur le deuxième bloc « try / catch » qui teste la valeur de psw. Si elle n'est pas celle attendue, une exception « Erreur de mot de passe » est levée et interceptée par le bloc catch qui suit. L'instruction « echo 'Bienvenu' » ne sera pas exécutée.
- Si il n'y a pas d'exceptions levées, 'Bienvenu' est affiché et le programme se termine normalement (les deux blocs « catch » sont ignorés).

### Exemple d'utilisation de bloc try { } catch() { } imbriqués


```
<?php
```

```
$psw = $_POST['psw'];
```

```
try {
    if (empty($psw)) {
        throw new Exception('Veuillez entrer le mot de passe');
    }

    try {
        if (password_verify($psw, $pswBdd)) {
            throw new Exception('Erreur de mot de passe !');
        }
        catch (Exception $e) {
            echo 'L\'erreur suivante a été générée : ' . "\n";
            echo $e->getMessage();
        }
    }
    catch (Exception $e) {
        echo 'L\'erreur suivante a été générée : ' . "\n";
        echo $e->getMessage();
    }
}

?>
```

	Auteur	Centre de Créteil	Formation	Date Mise à jour	Page 4
	Didier Bonneau	GRN 164	DWWM	20/11/2020	cours php - les exceptions.docx

### III CREER SES PROPRES CLASSES D'EXCEPTIONS

Quel sont les avantages à créer des classes dérivées ?

Avoir plusieurs types d'exceptions permet de les regrouper par « entité » présentent dans l'application. Les utilisateurs, les produits, les bases de données ...

De plus, la dérivation de la classe Exception permet d'ajouter des propriétés et méthodes supplémentaires aux objets de base « Exception ».

Les classes prédéfinies dans PHP possèdent leur propre classe d'exception.

A titre d'exemple vous pouvez regarder la classe « PDOException » de l'objet « PDO » qui étend la classe « RuntimeException » qui elle-même étend la classe de base « Exception ».

Le code ci-contre présente la manière la plus simple de développer une nouvelle classe d'exception personnalisée.

- Une propriété \$date contenant le « TimeStamp » courant sera valorisé lors de l'instanciation de l'objet.
- La méthode « heureDeException » permettra à partir de l'exception levée de récupérer l'heure à laquelle cette dernière s'est produite.

#### Principe de création d'une classe dérivée de Exception


```
<?php
```

```
/**
 * Fichier monException.class.php
 */
class MonException extends Exception
{
    /* nouvelles propriétés */
    private $date;

    public function __construct($message=NULL, $code=0)
    {
        // appel du constructeur de la classe mère
        parent::__construct($message, $code);
        // valorisation de la propriété date
        $this->date = getDate() ;
    }

    public function heureDeException() {
        return date("H:i:s", $this->date);
    }
}

?>
```

	Auteur	Centre de Créteil	Formation	Date Mise à jour	Page 5
	Didier Bonneau	GRN 164	DWWM	20/11/2020	cours php - les exceptions.docx

## IV CENTRALISER LE TRAITEMENT DES EXCEPTIONS

Si l'application traite plusieurs classes d'exceptions, il est possible de centraliser le traitement des différentes exceptions en créant un seul bloc « try » suivi de plusieurs blocs « catch ».

C'est au niveau de chaque bloc « catch » que l'on précisera le type d'exception qui doit intercepter.

Attention toutefois intercepter les exceptions plus spécifiques avant celles plus générales.

### Principe de centralisation des Exceptions

```
<?php
```

```
try {
```

```
    Taitements pouvant lever des exceptions de type différents
}
```

```
catch (MonException $e) {
```

```
    echo 'une exception de type monException à été levée';
```

```
    echo $e->getMessage();
```

```
}
```


```
catch (Exception $e) {
```

```
    echo 'une exception générale à été levée';
```

```
    echo $e->getMessage();
```

```
}
```

```
?>
```

	Auteur	Centre de Créteil	Formation	Date Mise à jour	Page 6
	Didier Bonneau	GRN 164	DWWM	20/11/2020	cours php - les exceptions.docx