

PHP try catch



```
try {  
    //code goes here that could lead to an  
    exception  
}  
catch (Exception $e) {  
    //exception handling code goes here  
}
```



```
}  
//exception handling code goes here  
catch (Exception $e) {
```

BTS SIO :: SLAM :: BLOC 2

LES EXCEPTIONS

**UNE EXCEPTION EST GENEREE
LORSQU'UN TRAITEMENT NE
SE DÉROULE PAS BIEN !**

**CE "PAS BIEN" N'EST PAS
PRÉVISIBLE LORS DE LA
PROGRAMMATION ...**

**EXEMPLE UNE RESSOURCE
INACCESSIBLE CAR LE
SERVEUR EST EN PANNE !**

**UNE URL IMPOSSIBLE À
JOINDRE A CAUSE D'UNE
COUPURE DE WIFI**

UNE DIVISION PAR ZÉRO !

EXCEPTION

- ▶ Dans ce cas une exception est « lancée » (throw) et il faut essayer de la « capturer » (catch)
- ▶ L'objet lancé doit être de la classe Exception ou d'une classe qui en hérite.
- ▶ Souvent c'est le système ou PHP qui lance cette exception. Mais nous verrons que le développeur a cette possibilité.

SYNTAXE

- ▶ Exemple avec PDO. Celle-ci génère des exceptions de la classe PDOException qui hérite de Exception

```
try
{
    $db = new PDO("mysql:host=$host;dbname=$db;charset=utf8",$user,$pw);
    // Pour afficher les erreurs
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
catch (Exception $e)
{
    die('Erreur : ' . $e->getMessage());
}
```

SYNTAXE

- ▶ On peut avoir plusieurs blocs `catch` si on veut capturer différentes classes d'exception.
- ▶ Lorsqu'une exception est lancée dans le bloc **`try`** par une instruction, les instructions suivantes ne sont pas exécutées.
- ▶ Si aucune exception n'est levée alors les instructions du bloc **`catch`** ne sont pas exécutées.

SYNTAXE

```
// connection à la base de données avec test si il y a une erreur
try
{
    $db = new PDO("mysql:host=$host;dbname=$db;charset=utf8",$user,$pw);
    // Si il y a une erreur lors de la création de l'objet PDO
    // Les lignes suivantes ne sont pas exécutées
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
catch (Exception $e)
{
    // Cette instruction n'est exécutée que si le bloc try génère une exception de type Exception.
    die('Erreur : ' . $e->getMessage());
}
```

EXCEPTION

Fatal error: Uncaught Error: Call to a member function fetch() on bool in /Applications/

- ▶ Si une exception est lancée et non capturée, une erreur fatale est émise.

```
// Liste de pilotes
public function getList()
{
    // Retourne tous les pilotes
    $pilotes = [];
    $q = $this->_db->query('SELECT NumP, NameP, Address, Salary FROM PILOTS ORDER BY NameP');
    while ($donnees = $q->fetch(PDO::FETCH_ASSOC))
    {
        $pilotes[] = new Pilote($donnees);
    }
    return $pilotes; // On retourne un tableau d'objets Pilote
}
```

COMMENT CRÉER UNE EXCEPTION

La liste est vide !

Gestion des pilotes

- ▶ la syntaxe est simple
throw new Exception('Erreur de requête.');
- ▶ Dans le cas précédent on pourrait par exemple avoir ceci:

```
// Liste de pilotes
public function getList()
{
    // Retourne tous les pilotes
    $pilotes = [];
    try
    {
        $q = $this->_db->query('SELECT NumP, NameP, Address, Salary FROM PILOTS ORDER BY NameP');
        if (!$q)
        {
            throw new Exception("La liste est vide !",12);
        }
        while ($donnees = $q->fetch(PDO::FETCH_ASSOC))
        {
            $pilotes[] = new Pilote($donnees);
        }
        return $pilotes; // On retourne un tableau d'objets Pilote
    }
    catch(Exception $e)
    {
        echo $e->getMessage();
    }
}
```

COMMENT CRÉER UNE EXCEPTION

- L'objet de type Exception à beaucoup de méthodes qui peuvent être précieuses.

```
try
{
    $q = $this->_db->query('SELECT NumP, NameP, Address, Salary FROM PILOTS ORDER BY NameP');
    if (!$q)
    {
        throw new Exception("La liste est vide !",12);
    }
    while ($donnees = $q->fetch(PDO::FETCH_ASSOC))
    {
        $pilotes[] = new Pilote($donnees);
    }
    return $pilotes; // On retourne un tableau d'objets Pilote
}
catch(Exception $e)
{
    echo $e->getMessage()."<br>";
    echo "Erreur numéro ".$e->getCode()."<br>";
    echo "Fichier ".$e->getFile()."<br>";
    echo "Ligne".$e->getLine()."<br>";
}
```

COMMENT CRÉER UNE EXCEPTION

- ▶ On peut trouver un bloc finally toujours après le ou les catch. Dans ce cas celui ce bloc est toujours exécuté. Qu'il y est eu ou non une Exception de levée.
- ▶ Ceci sert par exemple pour fermer la connexion à une base de données.

```
// Autres méthodes outils

// Liste de pilotes
public function getList()
{
    // Retourne tous les pilotes
    $pilotes = [];
    try
    {
        $q = $this->_db->query('SELECT NumP, NameP, Address, Salari');
        if (!$q)
        {
            throw new Exception("La liste est vide !",12);
        }
        while ($donnees = $q->fetch(PDO::FETCH_ASSOC))
        {
            $pilotes[] = new Pilote($donnees);
        }
        return $pilotes; // On retourne un tableau d'objets Pilote
    }
    catch(Exception $e)
    {
        echo $e->getMessage()."<br>";
        echo "Erreur numéro ".$e->getCode()."<br>";
        echo "Fichier ".$e->getFile()."<br>";
        echo "Ligne".$e->getLine()."<br>";
    }
    finally
    {
        // ici on pourrait par exemple fermer la connexion
        echo "Ici c'est toujours exécuté !";
    }
}
```

CREER SA PROPRE CLASSE EXCEPTION

```
<?php
/**
 * Définition d'une classe d'exception personnalisée
 */
class MyException extends Exception
{
    // Redéfinissez l'exception ainsi le message n'est pas facultatif
    public function __construct($message, $code = 0, Throwable $previous = null) {

        // traitement personnalisé que vous voulez réaliser ...

        // assurez-vous que tout a été assigné proprement
        parent::__construct($message, $code, $previous);
    }

    // chaîne personnalisée représentant l'objet
    public function __toString() {
        return __CLASS__ . ": [{$this->code}]: {$this->message}\n";
    }

    public function customFunction() {
        echo "Une fonction personnalisée pour ce type d'exception\n";
    }
}
```

EXCEPTIONS PREDEFINIES

- ▶ RuntimeException : erreur d'exécution
- ▶ InvalidArgumentException
- ▶ LengthException : taille invalide

```
/**
 * Set the author.
 *
 * @param string $value Author value to be set.
 */
public function setAuthor(string $value) {
    // Check if length exceeds maximum.
    if (strlen($value) > self::AUTHOR_MAX_LENGTH) {
        // Create local variables for string interpolation.
        $length = strlen($value);
        $max = self::AUTHOR_MAX_LENGTH;
        $diff = $length - $max;
        throw new LengthException("Cannot set Author containing $length bytes, which exceeds th
e maximum of $max by $diff bytes.");
    }
    $this->author = $value;
}
```

TAF

Réinvestir ces nouvelles connaissances dans votre code pour la base PILOT.

Essayer de penser aux cas qui pourraient lever une exception et implémenter le code afin que le programme ne s'arrête pas !