Gestion des Exceptions en PHP

La gestion des exceptions en PHP permet de **détecter, intercepter et gérer les erreurs** de manière propre et contrôlée. Elle améliore la **fiabilité** et la **sécurité** des applications.

🔹 1. Introduction aux Exceptions en PHP

Une **exception** est une erreur qui **interrompt** l'exécution normale d'un script. Plutôt que de laisser le programme planter, PHP permet de capturer ces erreurs grâce à **try...catch...finally**.

### **Syntaxe de base : Try - Catch - Finally**

***try {***

***// Code qui peut générer une exception***

***} catch (Exception $e) {***

***// Code qui gère l'exception***

***} finally {***

***// Code qui s'exécute toujours (facultatif)***

***}***

🔹 2. Exemple de gestion d'exception en PHP

### **Exemple 1 : Division par zéro**

***function division($a, $b) {***

***if ($b == 0) {***

***throw new Exception("Erreur : Division par zéro !");***

***}***

***return $a / $b;***

***}***

***try {***

***echo division(10, 0);***

***} catch (Exception $e) {***

***echo "Exception capturée : " . $e->getMessage();***

***} finally {***

***echo "<br>Fin du programme.";***

***}***

**Sortie :**

Exception capturée : Erreur : Division par zéro !

Fin du programme.

🔹 3. Les Types d'Exceptions en PHP

PHP dispose de plusieurs classes d'exceptions :

| **Classe d'Exception** | **Description** |
| --- | --- |
| Exception | Classe de base pour toutes les exceptions. |
| ErrorException | Convertit les erreurs PHP en exceptions. |
| PDOException | Exceptions liées à la base de données PDO. |
| InvalidArgumentException | Levée lorsqu'un argument n'est pas valide. |
| RuntimeException | Erreurs d'exécution. |

Exemple 2 : Exception spécifique (InvalidArgumentException)

***function setAge($age) {***

***if ($age < 0) {***

***throw new InvalidArgumentException("L'âge ne peut pas être négatif !");***

***}***

***return "Âge valide : " . $age;***

***}***

***try {***

***echo setAge(-5);***

***} catch (InvalidArgumentException $e) {***

***echo "Erreur : " . $e->getMessage();***

***}***

**Sortie :**

Erreur : L'âge ne peut pas être négatif !

**PHP et SGBD (Système de Gestion de Bases de Données)**

PHP peut interagir avec plusieurs **Systèmes de Gestion de Bases de Données (SGBD)** pour stocker, récupérer et manipuler des données. Les plus populaires sont :

* **MySQL / MariaDB** ✅ (le plus utilisé avec PHP)
* **PostgreSQL** 🔥 (puissant et fiable)
* **SQLite** 💾 (léger et embarqué)
* **SQL Server** (Microsoft)
* **Oracle Database** (gros systèmes d’entreprise)

Dans ce cours, nous allons voir **comment connecter PHP à différents SGBD** et effectuer des opérations **CRUD (Create, Read, Update, Delete)**.

**🔹 1. Connexion à un SGBD avec PHP**

**1.1 Connexion à MySQL avec MySQLi**

***$conn = new mysqli("localhost", "root", "", "database\_name");***

***if ($conn->connect\_error) {***

***die("Échec de la connexion : " . $conn->connect\_error);***

***}***

***echo "Connexion réussie !";***

**1.2 Connexion à MySQL avec PDO**

***try {***

***$pdo = new PDO("mysql:host=localhost;dbname=database\_name", "root", "");***

***$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);***

***echo "Connexion réussie !";***

***} catch (PDOException $e) {***

***die("Échec de la connexion : " . $e->getMessage());***

***}***

**1.3 Connexion à PostgreSQL avec PDO**

***try {***

***$pdo = new PDO("pgsql:host=localhost;dbname=database\_name", "user", "password");***

***echo "Connexion à PostgreSQL réussie !";***

***} catch (PDOException $e) {***

***die("Erreur de connexion : " . $e->getMessage());***

***}***

**1.4 Connexion à SQLite avec PDO**

***try {***

***$pdo = new PDO("sqlite:database.db");***

***echo "Connexion SQLite réussie !";***

***} catch (PDOException $e) {***

***die("Erreur de connexion : " . $e->getMessage());***

***}***

**🔹 2. CRUD (Create, Read, Update, Delete) avec PHP et MySQL**

**2.1 Créer une table MySQL**

***CREATE TABLE users (***

***id INT AUTO\_INCREMENT PRIMARY KEY,***

***name VARCHAR(100),***

***email VARCHAR(100) UNIQUE***

***);***

**2.2 Insérer des données en MySQLi**

***$stmt = $conn->prepare("INSERT INTO users (name, email) VALUES (?, ?)");***

***$stmt->bind\_param("ss", $name, $email);***

***$name = "Alice";***

***$email = "alice@example.com";***

***$stmt->execute();***

***echo "Utilisateur ajouté !";***

**2.3 Insérer des données en PDO**

$***stmt = $pdo->prepare("INSERT INTO users (name, email) VALUES (?, ?)");***

***$stmt->execute(["Alice", "alice@example.com"]);***

***echo "Utilisateur ajouté !";***

**2.4 Lire des données en MySQLi**

***$result = $conn->query("SELECT \* FROM users");***

***while ($row = $result->fetch\_assoc()) {***

***echo $row['name'] . " - " . $row['email'] . "<br>";***

***}***

**2.5 Lire des données en PDO**

***$stmt = $pdo->query("SELECT \* FROM users");***

***while ($row = $stmt->fetch(PDO::FETCH\_ASSOC)) {***

***echo $row['name'] . " - " . $row['email'] . "<br>";***

***}***

**2.6 Modifier des données (UPDATE)**

**✅ MySQLi**

***$stmt = $conn->prepare("UPDATE users SET name = ? WHERE email = ?");***

***$stmt->bind\_param("ss", $new\_name, $email);***

***$new\_name = "Alice Updated";***

***$email = "alice@example.com";***

***$stmt->execute();***

***echo "Utilisateur mis à jour !";***

**✅ PDO**

***$stmt = $pdo->prepare("UPDATE users SET name = ? WHERE email = ?");***

***$stmt->execute(["Alice Updated", "alice@example.com"]);***

***echo "Utilisateur mis à jour !";***

**2.7 Supprimer des données (DELETE)**

**✅ MySQLi**

***$stmt = $conn->prepare("DELETE FROM users WHERE email = ?");***

***$stmt->bind\_param("s", $email);***

***$email = "alice@example.com";***

***$stmt->execute();***

***echo "Utilisateur supprimé !";***

**✅ PDO**

***$stmt = $pdo->prepare("DELETE FROM users WHERE email = ?");***

***$stmt->execute(["alice@example.com"]);***

***echo "Utilisateur supprimé !";***

**🔹 3. Sécuriser PHP et MySQL contre les attaques**

1. **Utiliser des requêtes préparées** ($stmt->prepare()) pour éviter les injections SQL.
2. **Valider et filtrer les entrées utilisateur** (filter\_var(), htmlspecialchars()).
3. **Éviter d'afficher les erreurs SQL en production** (PDO::ERRMODE\_EXCEPTION).
4. **Gérer les sessions de manière sécurisée** (session\_start(), session\_regenerate\_id()).

**📌 Conclusion**

* **PHP peut interagir avec plusieurs SGBD** (MySQL, PostgreSQL, SQLite, etc.).
* **PDO est plus flexible que MySQLi** car il supporte plusieurs bases de données.
* **Les requêtes préparées sont essentielles** pour la sécurité contre les injections SQL.