

1. Principe sur le traitement standard des erreurs dans un projet Web

Le traitement standard des erreurs repose sur la classe technique `Erreur` et le répertoire `erreur` de l'application qui permet de personnaliser le traitement des erreurs 401, 403, 404 et des erreurs détectées par programmation.

2. L'erreur 401

L'erreur 401 est un code d'état HTTP qui signifie "Unauthorized". Elle est renvoyée lorsque le serveur nécessite une authentification pour accéder à la ressource demandée par le client, mais que le client n'a pas fourni de jeton d'authentification ou des informations d'identification dans l'en-tête `Authorization` de sa requête (par exemple, sous forme de "Basic" ou "Bearer token").

Si les informations d'authentification sont manquantes, incorrectes ou expirées, le serveur répond avec le code HTTP 401 Unauthorized pour indiquer que l'accès est refusé en raison d'une absence ou d'une erreur dans les informations d'authentification

3. L'erreur 403

L'erreur 403 signifie "Forbidden". Elle est renvoyée lorsque le serveur comprend la requête du client, mais refuse de l'autoriser. Cela peut se produire lorsque le client n'a pas les autorisations nécessaires pour accéder à la ressource demandée ou lorsque la ressource est interdite d'accès pour des raisons de sécurité.

4. L'erreur 404

L'erreur 404 signifie "Not Found". Elle est renvoyée lorsque le serveur ne peut pas trouver la ressource demandée par le client. Cela peut se produire lorsque le client demande une page Web qui n'existe pas ou qui a été déplacée vers un nouvel emplacement.

5. Traitement des erreurs 401, 403 et 404

Lorsque Apache détecte une erreur HTTP (comme 401, 403 ou 404), il renvoie généralement une page d'erreur par défaut au client. Ces pages peuvent varier légèrement en fonction de la configuration du serveur, mais par défaut, Apache génère des pages simples pour ces erreurs.

Par exemple pour une erreur 404 :

Not Found

The requested URL was not found on this server.

Apache/2.4.62 (Win64) OpenSSL/3.1.7 PHP/8.3.14 mod_fcgid/2.3.10-dev Server at gestion-correction Port 80

Ce n'est pas très jolie. Il est facile de prendre en charge ces erreurs afin de renvoyer une page personnalisée plus adaptée à la charte graphique du développement.

Pour cela il faut définir une redirection dans le fichier `.htaccess` de l'application

La gestion des erreurs

Fichier .htaccess placé à la racine

```
# redirection des erreurs 401, 403 et 404 vers des pages personnalisées
ErrorDocument 404 /erreur/404.php
ErrorDocument 403 /erreur/403.php
ErrorDocument 401 /erreur/401.php
```

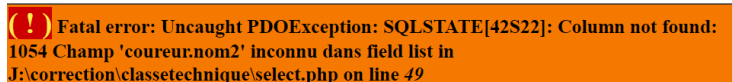
Les scripts 404.php, 403.php et 401.php du répertoire erreur contiennent une interface sobre mais qui s'adapte à tous les projets n'ayant aucune dépendance avec des composants externes.

Exemple sur une erreur 404



6. Les erreurs de programmation

Sur le serveur Apache de développement, les erreurs de programmation sont automatiquement prises en charge par Xdebug. Les erreurs sont affichées sur la page ou dans la console lorsque l'erreur survient lors d'un appel Ajax.



7. Les erreurs détectées par l'application

La classe Erreur est une classe statique utilitaire qui centralise la gestion des erreurs serveur dans une application PHP moderne, qu'elles soient rencontrées lors d'un appel AJAX ou dans un accès direct via une URL. Elle uniformise les traitements associés aux erreurs (journalisation, envoi de réponse JSON, redirection, blocage de visiteur, etc.) et s'interface avec une dépendance implicite : la classe Journal (journalisation).

La classe Erreur regroupe les méthodes permettant :

- d'envoyer une réponse JSON formatée en cas d'erreur côté client (envoyerReponse),
- d'afficher une page dédiée à l'erreur côté serveur (afficherReponse),
- de choisir dynamiquement le bon canal de retour selon le contexte d'appel (traiterErreur),
- de bloquer une IP considérée comme malveillante (bloquerVisiteur),
- et de restituer un message humainement lisible associé à un code HTTP (getErreurHttp).

Elle repose sur un mécanisme de type d'erreur ('global', 'system' ou 'nomChamp') pour :

- afficher un message utilisateur explicite (global),
- remplacer le message par un message générique et consigner l'erreur en base de données (system).
- Afficher un message utilisateur en dessous du champ mentionné (appel ajax seulement)

Les erreurs déclenchées par un trigger retourne un message commençant par un # ce qui permet de distinguer dynamiquement si le message transmis est un message final (#Mon message) ou un message technique à logger.

Cette classe possède principalement 3 méthodes statiques :

Erreur ::**envoyerReponse**(string \$message, ?string \$type = null): void

Cette méthode s'utilise dans les scripts PHP **toujours appelés en Ajax**.

La réponse est envoyée dans le format JSON et le script en cours est arrêté (exit)

Erreur ::**afficherReponse**(string \$message, ?string \$type = null): void

Cette méthode s'utilise dans les scripts PHP **toujours appelés directement**.

Elle redirige l'appel vers le script /erreur/index.php en transmettant le message d'erreur dans une variable de session.

Erreur ::**traiterErreur**(string \$message, ?string \$type = null): void

Cette méthode s'utilise pour tous les **scripts PHP pouvant être à la fois appelés directement ou par un appel Ajax**.

C'est le cas des scripts des classes métiers, des classes techniques et des scripts 'autoload'.

Elle essaye de détecter le type d'appel en détectant la présence de l'entête HTTP_X_REQUESTED_WITH qui doit contenir la valeur XMLHttpRequest

La classe Erreur est donc un parfait contrôleur d'erreurs centralisé pensé pour un cadre **MVC AJAXisé**. Elle distingue le canal de l'erreur (AJAX ou non), le niveau de gravité (global ou system) et l'origine (utilisateur ou technique)

Dans le cadre d'un appel Ajax, la fonction de rappel doit se charger d'afficher l'erreur sur l'interface, soit dans une zone de l'interface prévue à cet effet, soit sous le champ concerné, soit dans une boîte de dialogue. Les erreurs sont contenu dans la propriété 'error' de la réponse JSON renvoyé par le serveur. La fonction appelAjax du composant /composant/fonction/ajax centralise ce traitement ce qui permet au développeur de ne pas avoir à se charger du traitement des erreurs.

Exemple : opération d'ajout dans un table

```
import {appelAjax} from "/composant/fonction/ajax.js";

function ajouter() {
    appelAjax({
        url: '/ajax/ajouter.php',
        data: {
            table: 'annonce',
            nom: nom.value,
            description: description.value,
            date: date.value
        },
        success: () => {
            retournerVersApresConfirmation("Annonce enregistrée", "/consultation/annonce/");
        }
    });
}
```