Engineer

PHP

Langage de script exécuté sur le serveur web

Ce que c’est

PHP est un langage de script exécuté sur le serveur web : une requête arrive, le code PHP s’exécute et le navigateur reçoit le HTML produit, jamais le code lui-même. Les lettres signifient aujourd’hui « PHP: Hypertext Preprocessor », un acronyme récursif, même si elles désignaient à l’origine « Personal Home Page ». Un fichier PHP peut être du HTML ordinaire contenant du code entre les balises `<?php` et `?>`, ou uniquement du code par lequel un framework fait passer chaque requête. PHP exécute aussi des scripts en ligne de commande, mais sa conception privilégie la production de pages web.

Son histoire commence en 1994, lorsque Rasmus Lerdorf écrit des binaires CGI en C pour compter les visites de son CV en ligne et les nomme Personal Home Page Tools. Il publie les sources en juin 1995, et les outils deviennent PHP/FI. En 1997, Andi Gutmans et Zeev Suraski commencent à réécrire l’analyseur ; le résultat, PHP 3.0, est publié en juin 1998 sous le nom récursif. À son apogée, il est installé sur environ 10 % des serveurs web d’Internet. Leur moteur suivant, le Zend Engine, nommé d’après eux deux, fait tourner PHP 4.0 en mai 2000, puis PHP 5 suit en juillet 2004.

Sur un serveur actuel, PHP est couramment exécuté par PHP-FPM, le FastCGI Process Manager fourni avec lui. Un serveur web placé devant transmet chaque requête PHP à un pool de processus de traitement ; chaque pool peut s’exécuter sous son propre utilisateur et son propre groupe, avec son environnement et ses valeurs php.ini. Son gestionnaire de processus maintient un nombre fixe de processus, les lance à la demande ou fait varier leur nombre entre des limites. Le langage évolue par branches de versions, et les quatre dernières sont arrivées à un an d’intervalle. Chaque branche bénéficie de deux ans de maintenance active, avec des correctifs de bogues et de sécurité, puis de deux ans de correctifs de

sécurité critiques seulement, avant sa fin de vie. PHP 8.2, publié le 8 décembre 2022, reçoit des correctifs de sécurité jusqu'au 31 décembre 2026 ; PHP 8.5, publié le 20 novembre 2025, jusqu'au 31 décembre 2029.

PHP convertit les valeurs d'un type à l'autre quand une opération le demande, et l'opérateur de comparaison non stricte `==` le montre particulièrement. Si les deux côtés sont des chaînes numériques, ou si l'un est un nombre et l'autre une chaîne numérique, la comparaison est numérique. Avant PHP 8.0.0, toute chaîne comparée à un nombre était d'abord convertie en nombre : `0 == "a"` était donc vrai. L'opérateur strict `===` compare le type autant que la valeur, sans conversion, et le manuel conseille de le privilégier dans la plupart des cas.

```
<?php
var_dump(0 == "a");      // PHP 7: true, PHP 8: false
var_dump("1" == "01");   // true in both: two numeric strings
var_dump("10" == "1e1"); // true in both: 10 equals 1e1
var_dump(100 == "1e2");  // true in both
var_dump(0 === "a");     // false in both: === never converts
```

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que PHP ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. La partie invisible pour les utilisateurs : où résident les données, quelles règles s'appliquent et comment les autres systèmes peuvent y accéder.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 6 pipelines ETL/ELT conçus et exploités — 6 pipelines ETL sur trois moteurs de bases de données

6 pipelines ETL/ELT conçus et exploités Contactez-nous →

Pourquoi c'est important

L'importance de PHP tient surtout au code déjà écrit dans ce langage. WordPress et Joomla, les systèmes de gestion de contenu employés par de nombreuses organisations pour exploiter des sites qu'elles modifient elles-mêmes, sont des applications PHP : WordPress recommande PHP 8.3 ou supérieur, et Joomla 6 exige au moins PHP 8.3.0. WordPress fonctionne encore sur PHP 7.4, même si sa page de prérequis avertit que ces versions sont en fin de vie. Une entreprise peut dépendre de PHP pendant des années sans que personne en son sein n'en ait écrit une ligne, d'où l'oubli de la version sous-jacente. Un site continue de servir des pages lorsque sa branche atteint la fin de vie ; il cesse simplement de recevoir des correctifs, sans que rien à l'écran ne l'indique.

Les mises à niveau changent aussi le comportement discrètement. PHP 8.0 a modifié la comparaison des chaînes avec les nombres : une condition vraie sous PHP 7 peut donc être fausse sous PHP 8, sans erreur. PHP 8.4 a relevé de 10 à 12 le coût bcrypt par défaut de `password_hash()`, rendant chaque nouveau hachage de mot de passe plus coûteux à calculer sur le même matériel. Les deux changements sont documentés, et chacun n'arrive en production qu'au travers d'une mise à niveau que quelqu'un doit planifier, tester et prendre en charge.

Notre recommandation est de traiter la version de PHP comme une dépendance avec sa date d'expiration écrite à côté. Une application PHP existante qui fonctionne mérite d'être conservée : placez-la sur une branche maintenue, planifiez la prochaine mise à niveau bien avant la fin du support de sécurité et donnez-lui son propre pool FPM et son propre utilisateur système. Pour une nouvelle réalisation, choisissez PHP s'il s'agit d'un site que le client modifiera avec WordPress ou Joomla, ou si l'équipe qui l'exploitera après la remise du projet connaît bien PHP ; sinon, choisissez selon les compétences de cette équipe. Le risque est rarement le langage, mais une version dont personne n'est responsable.

À quoi cela ressemble en pratique

La couche web de notre pile technologique donne à PHP une place précise. Les pages pouvant être des fichiers sont construites statiquement, ce qui évite d'exécuter une application à chaque requête pour ces pages, et ce site est construit avec Hugo ; Django et PHP servent lorsqu'une application doit répondre aux requêtes. La même page dit comment choisir : selon le problème, ce que l'équipe du client pourra exploiter après notre départ, et la façon dont la solution échoue plutôt que son effet en démonstration. Cette règle concerne directement PHP : lorsque la bonne réponse est la pile que le client possède déjà, la page s'engage à le dire.

De 2015 à 2022, la fondatrice a travaillé comme responsable d'une équipe de développement de logiciels SIG, prenant en charge plus de trente projets SIG tout au long du développement, du déploiement et de l'assistance après leur mise en production. PHP était l'un des quatre langages de leur couche applicative, avec JavaScript, Python et SQL ; en dessous, toutes les données spatiales résidaient dans PostgreSQL avec PostGIS, et, à l'avant, se trouvaient des cartes Mapbox GL et Leaflet, dont certaines appelaient les API ArcGIS ou HERE. Quel que soit le langage, la responsabilité allait au-delà de la livraison et les problèmes n'étaient pas transmis à d'autres : vous concevez différemment quand vous savez que vous recevrez vous-même l'appel en cas de panne.

Où les problèmes surviennent

- **Une comparaison non stricte protège un secret.** Un jeton ou un condensat vérifié avec `==` est comparé selon les règles numériques de PHP dès que les deux côtés ressemblent à des nombres : deux chaînes différentes comme `"10"` et `"1e1"` sont donc jugées égales. Utilisez `password_verify()` pour les mots de passe et `hash_equals()` pour les autres secrets : son temps d'exécution ne révèle rien du contenu du secret et la chaîne de l'utilisateur doit être son second argument. Utilisez `===` partout ailleurs.
- **Le SQL est assemblé à partir des données de la requête.** Les instructions préparées de PDO prennent en charge l'échappement de leurs paramètres, et le manuel indique qu'une application les utilisant exclusivement est protégée contre l'injection SQL, avec une exception explicite : toute partie de la requête encore construite à partir d'une entrée non échappée. Un marqueur de paramètre représente une valeur complète ; tout ce qui n'est pas une valeur, comme la colonne de tri d'une liste, finit donc encore dans la chaîne. Vérifiez ces parties selon une liste fixe dans le code.
- **Le stockage des condensats de mots de passe est trop étroit.** `PASSWORD_DEFAULT` est actuellement bcrypt, et le manuel avertit qu'il est conçu pour changer lorsque des algorithmes plus robustes sont ajoutés. Il recommande une colonne pouvant dépasser 60 octets, avec 255 comme bon choix. Bcrypt tronque aussi le mot de passe à 72 octets. Stockez le condensat entier et appelez `password_needs_rehash()` après une connexion réussie afin de mettre à niveau les anciens condensats au fil des connexions.
- **La branche dépasse sa date de fin de vie.** L'application continue de fonctionner, donc personne ne planifie la mise à niveau, et le saut finit par couvrir plusieurs branches à la fois. Notez la date de fin du support de sécurité de la branche utilisée, mettez à niveau une branche à la fois et exécutez la suite de tests sur la branche suivante avant l'expiration de l'actuelle.
- **Le pool FPM est partagé et trop permissif.** Un même pool servant plusieurs applications sous un même utilisateur permet à une faille de l'une d'atteindre les fichiers de toutes. `security.limit_extensions` vaut par défaut `.php .phar`, et le manuel conseille de le limiter à `.php` pour qu'un serveur web mal configuré ne puisse pas transmettre d'autres fichiers à PHP. Donnez à chaque application son propre pool et son propre utilisateur, et réglez `pm.max_children` selon ce que la mémoire du serveur peut supporter.

Pour aller plus loin

- Manuel PHP : qu'est-ce que PHP ? — la courte définition du projet, avec le premier exemple de PHP intégré au HTML.
- PHP : versions maintenues — politique de maintenance et dates de maintenance active et de sécurité seule pour chaque branche actuelle.

- Manuel PHP : opérateurs de comparaison — comparaisons non stricte et stricte, résultats de PHP 7 et PHP 8 côte à côte, et tableau des comparaisons par type.
- Manuel PHP : password_hash — algorithmes, coût, largeur de colonne, limite de 72 octets et changement du coût par défaut en 8.4.
- Manuel PHP : instructions préparées et procédures stockées — liaison des paramètres par PDO et cas où l'injection SQL reste possible.
- Manuel PHP : configuration de FPM — directives des pools, modes du gestionnaire de processus, `clear_env` et `security.limit_extensions`.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que PHP ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/php/