Engineer

PHP

sproget, det meste af det gamle web kører på

Hvad det er

PHP er et scriptsprog, der afvikles på webserveren: en forespørgsel kommer ind, PHP-koden kører, og det, browseren modtager, er den HTML, koden har skrevet, aldrig selve koden. Bogstaverne står i dag for »PHP: Hypertext Preprocessor«, et rekursivt akronym, men de begyndte som »Personal Home Page«. En PHP-fil kan være almindelig HTML med kode placeret mellem `<?php` og `?>`, eller den kan bestå udelukkende af kode, som et framework sender alle forespørgsler igennem. Sproget kan også køre scripts fra kommandolinjen, men det er bygget til at lave websider.

Historien begynder i 1994, da Rasmus Lerdorf skrev en række CGI-programmer i C for at tælle besøg på sit online-CV og kaldte dem Personal Home Page Tools. Han frigav kildekoden i juni 1995, og værktøjerne voksede til PHP/FI. I 1997 gik Andi Gutmans og Zeev Suraski i gang med at omskrive parseren, og resultatet udkom som PHP 3.0 i juni 1998 under det rekursive navn; da udbredelsen toppede, var det installeret på omkring 10 % af internettets webservere. Deres næste motor, Zend Engine, opkaldt efter dem begge, bar PHP 4.0 i maj 2000, og PHP 5 fulgte i juli 2004.

På en server i dag køres PHP ofte af PHP-FPM, FastCGI Process Manager, som følger med sproget. En webserver foran sender hver PHP-forespørgsel videre til en pulje af arbejdsprocesser. Hver pulje kan køre som sin egen bruger og gruppe med sit eget miljø og sine egne php.ini-værdier, og procesmanageren holder et fast antal processer, starter dem efter behov eller lader antallet svinge mellem to grænser. Selve sproget udvikles i release-grene, og de seneste fire kom med et års mellemrum. Hver gren får to års aktiv support med fejl- og sikkerhedsrettelser, derefter to år med kun kritiske sikkerhedsrettelser, og så er den udløbet. PHP 8.2, udgivet 8. december 2022, får sikkerhedsrettelser til 31. december 2026; PHP 8.5, udgivet 20. november 2025, til 31. december 2029.

PHP konverterer mellem typer, når en operation kræver det, og den løse sammenligning `==` er der, hvor det mærkes mest. Er begge sider numeriske strenge, eller er den ene et tal og den anden en numerisk streng, sammenlignes de som tal. Før PHP 8.0.0 blev enhver streng, der blev sammenlignet med et tal, først lavet om til et tal, så `0 == "a"` var sandt. Den strenge operator `===` sammenligner typen såvel som værdien og konverterer intet, og manualen anbefaler selv at foretrække den i de fleste tilfælde.

```
<?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
```

Brug begrebet i en beslutning

Brug »Hvad er PHP?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Den del, brugerne aldrig ser: hvor data ligger, hvilke regler der gælder, og hvordan andre systemer må tale med den.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har designet og drevet 6 ETL/ELT-pipelines — 6 ETL-pipelines på tre databasemotorer

Har designet og drevet 6 ETL/ELT-pipelines Kontakt os →

Hvorfor det betyder noget

PHP betyder mest på grund af alt det, der allerede er skrevet i sproget. WordPress og Joomla, de indholdsstyringssystemer som rigtig mange organisationer bruger til websites, de selv redigerer, er PHP-applikationer: WordPress anbefaler PHP 8.3 eller nyere, og Joomla 6 kræver mindst PHP 8.3.0. WordPress kører stadig på PHP 7.4, men dets egen kravside advarer om, at de versioner har nået end of life. En virksomhed kan være afhængig af PHP i årevis, uden at nogen i huset har skrevet en linje i sproget, og derfor bliver versionen nedenunder glemt. Et site bliver ved med at levere sider, efter dets gren er udløbet; det holder bare op med at få rettelser, og intet på skærmen viser det.

Opgraderinger ændrer også adfærd, og det sker stille. PHP 8.0 ændrede, hvordan strenge sammenlignes med tal, så en betingelse, der var sand på PHP 7, kan være falsk på PHP 8, uden at der kommer nogen fejlmeddelelse. PHP 8.4 hævede standardomkostningen for bcrypt i `password_hash()` fra 10 til 12, så hver

ny adgangskode-hash koster mere at beregne på den samme hardware. Begge ændringer er dokumenteret, og begge når først produktion gennem en opgradering, som nogen skal planlægge, teste og eje.

Vores anbefaling er at behandle PHP-versionen som en afhængighed med en udløbsdato skrevet ved siden af. En eksisterende PHP-applikation, der virker, er værd at beholde: læg den på en understøttet gren, planlæg den næste opgradering i god tid før sikkerhedssupporten slutter, og giv den sin egen FPM-pulje og sin egen systembruger. Til et nyt projekt vælger man PHP, når opgaven er et site, kunden selv skal redigere i WordPress eller Joomla, eller når det team, der skal drive det efter overdragelsen, kender PHP godt; ellers vælger man efter teamets kompetencer. Sproget er sjældent risikoen; det er en version, ingen ejer.

Sådan ser det ud i praksis

Weblaget i vores teknologivalg giver PHP en bestemt plads. Sider, der kan være filer, bygges statisk, så de ikke kræver applikationskørsel ved hver forespørgsel, og dette site bygges af Hugo; Django og PHP bruges, hvor der er en applikation at levere. Samme side forklarer, hvordan valget træffes: efter problemet, efter hvad kundens team kan drive, når vi er gået, og efter hvordan tingen fejler snarere end hvordan den tager sig ud i en demo. Den regel gælder også PHP: når det rigtige svar er den stak, kunden allerede har, lover siden at sige det.

Fra 2015 til 2022 arbejdede grundlæggeren som teamlead for GIS-softwareudvikling og førte mere end tredive GIS-projekter gennem udvikling, udrulning og support, efter de var gået i drift. PHP var et af fire sprog i applikationslaget, sammen med JavaScript, Python og SQL. Det lag sad mellem de rumlige data i PostgreSQL med PostGIS og webkortene, der var lavet med Mapbox GL og Leaflet og af og til kaldte ArcGIS- eller HERE-API'erne. Sidens hovedpointe gælder alle fire sprog: ansvaret rakte forbi leveringen, problemer blev ikke sendt videre, og man designer anderledes, når man ved, at det er en selv, der får opkaldet, hvis noget går i stykker.

Hvor det går galt

- **En løs sammenligning vogter en hemmelighed.** Et token eller en hash, der tjekkes med `==`, sammenlignes efter PHP's talregler, når begge sider ligner tal, så to forskellige strenge som `"10"` og `"1e1"` regnes for ens. Brug `password_verify()` til adgangskoder og `hash_equals()` til andre hemmeligheder: dens køretid afslører intet om hemmelighedens indhold, og den forventer brugerens streng som andet argument. Brug `===` alle andre steder.
- **SQL bygges af data fra forespørgslen.** PDO's prepared statements citerer selv deres parametre, og manualen siger, at en applikation, der udelukkende bruger dem, er beskyttet mod SQL-injection, med én nævnt undtagelse: de dele af forespørgslen, der stadig bygges af input, som ikke er escapet. En pladsholder står for en hel værdi, så alt, der ikke er en værdi, for eksempel den kolonne en liste sorteres efter, ender stadig i strengen. Tjek de dele mod en fast liste i koden.
- **Adgangskode-hashes gemmes for snævert.** `PASSWORD_DEFAULT` er bcrypt i dag, og manualen advarer om, at den er designet til at skifte, efterhånden som stærkere algoritmer kommer til. Den anbefaler en kolonne, der kan vokse ud over 60 bytes, og nævner 255 som et godt valg. Bcrypt afkorter desuden adgangskoden ved 72 bytes. Gem hele hashen, og kald `password_needs_rehash()` efter et vellykket login, så ældre hashes opgraderes, efterhånden som folk logger ind.
- **Grenen passerer sin udløbsdato.** Applikationen virker stadig, så ingen planlægger opgraderingen, og det spring, der til sidst kommer, dækker flere grene på én gang. Notér sikkerhedssupportens slutdato for den gren, der er i brug, opgradér én gren ad gangen, og kør testsuiten mod den næste gren, før den nuværende udløber.
- **FPM-puljen er delt og løst sat op.** Én pulje, der kører flere applikationer som én bruger, lader en fejl i den ene nå filerne i dem alle. `security.limit_extensions` står som standard til `.php .phar`, og manualen råder til at begrænse den til `.php`, så en fejlkonfigureret webserver aldrig sender andre filer til PHP. Giv hver applikation sin egen pulje og bruger, og sæt `pm.max_children` til det, serverens hukommelse kan bære.

Læs mere

- PHP Manual: What is PHP? — projektets egen korte definition med det første eksempel på PHP indlejret i HTML.
- PHP: Supported Versions — supportpolitikken og datoerne for aktiv support og sikkerhedssupport for hver aktuel gren.

- PHP Manual: Comparison Operators — løs og streng sammenligning, uddata fra PHP 7 og PHP 8 side om side og sammenligningstabellen efter type.
- PHP Manual: password_hash — algoritmer, omkostning, kolonnebredde, grænsen på 72 bytes og den nye standardomkostning i 8.4.
- PHP Manual: Prepared statements and stored procedures — hvordan PDO binder parametre, og hvor SQL-injection stadig er mulig.
- PHP Manual: FPM configuration — puljedirektiver, procesmanagerens tilstande, `clear_env` og `security.limit_extensions`.

Ordliste

Vælg ud fra jeres faktiske vilkår

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er PHP?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/php/