Engineer

PHP

språket som det mesta av den gamla webben körs på

Vad det är

PHP är ett skriptspråk som körs på webbservern: en begäran kommer in, PHP-koden körs, och det som når webbläsaren är den HTML som koden producerade, aldrig själva koden. Bokstäverna står i dag för ”PHP: Hypertext Preprocessor”, en rekursiv akronym, men de började som ”Personal Home Page”. En PHP-fil kan vara vanlig HTML med kod placerad mellan taggarna `<?php` och `?>`, eller ren kod som ett ramverk leder varje begäran genom. PHP kör också skript från kommandoraden, men språkets utformning lutar åt att producera webbsidor.

Historien börjar 1994, när Rasmus Lerdorf skrev en uppsättning CGI-program i C för att räkna besöken på sitt cv på nätet och kallade dem Personal Home Page Tools. Han släppte källkoden i juni 1995, och verktygen växte till PHP/FI. År 1997 började Andi Gutmans och Zeev Suraski skriva om parsern, och resultatet släpptes som PHP 3.0 i juni 1998 under det rekursiva namnet; när det var som störst fanns det installerat på ungefär 10 % av webbservrarna på internet. Deras nästa motor, Zend Engine, uppkallad efter de två, bar PHP 4.0 i maj 2000, och PHP 5 följde i juli 2004.

På en server i dag körs PHP ofta av PHP-FPM, FastCGI Process Manager, som följer med språket. En webbserver framför skickar varje PHP-begäran vidare till en pool av arbetsprocesser. Varje pool kan köras som en egen användare och grupp, med en egen miljö och egna värden i php.ini, och dess processhanterare håller ett fast antal arbetare, startar dem vid behov eller varierar antalet mellan två gränser. Språket utvecklas i versionsgrenar, och de fyra senaste kom med ett års mellanrum. Varje gren får två års aktiv support med rättningar av buggar och säkerhetsbrister, sedan två år till med enbart kritiska säkerhetsrättningar, och når därefter slutet på sin livslängd. PHP 8.2, som släpptes den 8 december 2022, får säkerhetsrättningar till den 31 december 2026, och PHP 8.5, som släpptes den 20 november 2025, får dem till den 31 december 2029.

PHP konverterar mellan typer när en operation kräver det, och den lösa jämförelseoperatorn `==` är där det märks mest. Om båda sidor är numeriska strängar, eller om den ena är ett tal och den andra en numerisk sträng, blir jämförelsen numerisk. Före PHP 8.0.0 omvandlades varje sträng som jämfördes med ett tal först till ett tal, så `0 == "a"` var sant. Den strikta operatorn `===` jämför typen lika väl som värdet och omvandlar ingenting, och manualen själv råder till att föredra den i de flesta fall.

```
<?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
```

Använd begreppet i ett beslut

Använd ”Vad är PHP?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Den del användarna aldrig ser: där data lagras, vilka reglerna är och hur andra system får kommunicera med den.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Utformade och drev 6 ETL/ELT-pipelines — 6 ETL-pipelines över tre databasmotorer

Utformade och drev 6 ETL/ELT-pipelines Hör av er →

Varför det spelar roll

PHP spelar roll mest på grund av det som redan är skrivet i det. WordPress och Joomla, de system för innehållshantering som väldigt många organisationer använder för att driva webbplatser som de själva redigerar, är PHP-applikationer: WordPress rekommenderar PHP 8.3 eller senare, och Joomla 6 kräver minst PHP 8.3.0. WordPress kör fortfarande på PHP 7.4, men sidan med dess krav varnar för att sådana versioner har nått slutet på sin livslängd. Ett företag kan vara beroende av PHP i åratal utan att någon där har skrivit en rad i språket, och det är därför versionen under glöms bort. En webbplats fortsätter att leverera sidor efter att dess gren har nått slutet på sin livslängd. Den slutar bara att få rättningar, och skärmen visar ingenting om det.

Uppgraderingar ändrar också beteende, i det tysta. PHP 8.0 ändrade hur strängar jämförs med tal, så ett villkor som var sant på PHP 7 kan vara falskt på PHP 8 utan att något fel visas. PHP 8.4 höjde standardkostnaden för bcrypt i `password_hash()`

från 10 till 12, vilket gör varje ny lösenordshash dyrare att beräkna på samma hårdvara. Båda ändringarna är dokumenterade, och var och en når produktion bara genom en uppgradering som någon måste planera, testa och äga.

Vår rekommendation är att behandla PHP-versionen som ett beroende med ett utgångsdatum skrivet bredvid. En befintlig PHP-applikation som fungerar är värd att behålla: lägg den på en gren som stöds, planera nästa uppgradering i god tid innan säkerhetssupporten tar slut, och ge den en egen FPM-pool och en egen systemanvändare. För ett nytt bygge väljer vi PHP när jobbet är en webbplats som kunden ska redigera i WordPress eller Joomla, eller när teamet som ska driva den efter överlämningen kan PHP väl, och annars väljer vi efter vad det teamet kan. Risken ligger sällan i språket, men ofta i en version som ingen äger.

Så ser det ut i praktiken

Webbskiktet i vår teknikstack ger PHP en bestämd plats. Sidor som kan vara filer byggs statiskt, så att ingen applikationskod behöver köras när dessa sidor efterfrågas, och den här webbplatsen byggs av Hugo. Django och PHP används där det finns en applikation att leverera. Samma sida anger hur valet görs: efter problemet, efter vad kundens team kan driva när vi har gått och efter hur saken fallerar snarare än hur den ser ut i en demo. Den regeln gäller PHP direkt, för när rätt svar är stacken som kunden redan har, lovar sidan att säga det.

Från 2015 till 2022 arbetade grundaren som teamledare för utveckling av GIS-mjukvara och tog mer än trettio GIS-projekt genom utveckling, driftsättning och support när de väl var i drift. Geodatan låg i PostgreSQL med PostGIS, och webbkartorna byggdes på Mapbox GL och Leaflet, ibland mot API:erna från ArcGIS eller HERE, med applikationslagret i JavaScript, Python, PHP och SQL. PHP var ett av fyra språk i det lagret, och sidans huvudpoäng gäller vart och ett av dem: ansvaret sträckte sig längre än till leveransen, problemen lämnades aldrig över till någon annan, och man designar annorlunda när man vet att det är en själv som blir uppringd om något går sönder.

Var det går fel

- **En lös jämförelse vaktar en hemlighet.** En token eller ett hashvärde som kontrolleras med `==` jämförs enligt de numeriska reglerna i PHP så fort båda sidor ser ut som tal, så två olika strängar som `"10"` och `"1e1"` räknas som lika. Använd `password_verify()` för lösenord och `hash_equals()` för andra hemligheter: dess körtid avslöjar ingenting om innehållet i hemligheten, och den förväntar sig användarens sträng som andra argument. Använd `===` överallt annars.
- **SQL sätts ihop av data från begäran.** Förberedda satser i PDO citerar sina parametrar, och manualen säger att en applikation som enbart använder dem är skyddad mot SQL-injektion, med ett uttalat undantag: varje del av frågan som fortfarande byggs av indata som inte har escapats. En platshållare står för ett helt värde, så allt som inte är ett värde, till exempel kolumnen som en lista sorteras efter, hamnar ändå i strängen. Kontrollera de delarna mot en fast lista i koden.
- **Lösenordshashar lagras för snävt.** `PASSWORD_DEFAULT` är bcrypt i dag, och manualen varnar för att den är utformad att ändras när starkare algoritmer läggs till. Den rekommenderar en kolumn som kan växa utöver 60 byte, med 255 som ett bra val. Bcrypt kortar dessutom av lösenordet vid 72 byte. Lagra hela hashvärdet, och anropa `password_needs_rehash()` efter en lyckad inloggning så att äldre hashvärden uppgraderas allteftersom användarna loggar in.
- **Grenen når slutet på sin livslängd.** Applikationen fortsätter att fungera, så uppgraderingen hamnar aldrig i någons kalender, och det hopp som till slut måste tas spänner över flera grenar på en gång. Skriv ned datumet då säkerhetssupporten för den använda grenen tar slut, uppgradera en gren i taget och kör testsviten mot nästa gren innan den nuvarande löper ut.
- **FPM-poolen är delad och slapp.** En pool som betjänar flera applikationer som en enda användare låter en brist i en av dem nå filerna i alla. `security.limit_extensions` har `.php .phar` som standard, och manualen råder till att begränsa den till `.php` så att en felkonfigurerad webbserver inte kan lämna över andra filer till PHP. Ge varje applikation en egen pool och användare, och sätt `pm.max_children` till vad serverns minne rymmer.

Läs mer

- PHP Manual: What is PHP? — projektets egen korta definition, med det första exemplet på PHP inbäddat i HTML.
- PHP: Supported Versions — supportpolicyn och datumen för aktiv support och enbart säkerhetsrättningar för varje aktuell gren.
- PHP Manual: Comparison Operators — lös och strikt jämförelse, utdata från PHP 7 och PHP 8 sida vid sida, och jämförelsetabellen per typ.

- PHP Manual: password_hash — algoritmer, kostnad, kolumnbredd, gränsen på 72 byte och den ändrade standardkostnaden i 8.4.
- PHP Manual: Prepared statements and stored procedures — hur PDO binder parametrar och var SQL-injektion fortfarande är möjlig.
- PHP Manual: FPM configuration — pooldirektiven, lägena för processhanteraren, `clear_env` och `security.limit_extensions`.

Ordlista

Välj utifrån era faktiska förutsättningar

Beskriv arbetsflödet, befintliga gränssnitt och användarna som behöver förändringen. Vi kan definiera en testbar leverans och en överlämning som gör att teamet kan underhålla lösningen.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är PHP?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

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

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/php/