Krótka historia problemów z PHP

Pierwsze wydanie PHP miało miejsce w 1995 roku, a jego użycie szybko rosło. Regularne aktualizacje wersji zapewniające dodatkowe funkcje były wypychane aż do 2014 roku, kiedy to wydano PHP 5.6. Jednak w 2014 roku było już za późno. PHP od samego początku był bardzo łatwy w użyciu i nauce oraz był bardzo wyrozumiałym językiem – co było zarówno powodem wielkiego sukcesu, jak i kłopotów z drugiej strony. W latach 2010-2016 pojawiało się coraz więcej głosów ze strony społeczności programistów, że PHP nie podąża za nowoczesnymi praktykami języka programowania i nie jest wystarczająco rygorystyczny dla programistów, aby egzekwować dobry, poprawny kod. Spowodowało to również, że coraz więcej programistów backendowych przenosiło się do ASP.NET i Pythona do tworzenia stron internetowych. Dodatkowo, ze względu na wewnętrzne konflikty wokół PHP6 (który miał na celu zapewnienie obsługi Unicode), rozwój ZEND był opóźniony w stosunku do obecnych standardów. Jednak PHP wciąż było najczęściej używanym językiem sieciowym, a społeczność usłyszała te głosy i zareagowała. Ze względu na szerokie zastosowanie PHP, dość trudno było rozwijać język zgodnie z nowoczesnymi standardami przy jednoczesnym zachowaniu kompatybilności wstecznej, ale społeczność stawiła czoła tym problemom. Tak więc po długim czasie wydano PHP 7, a następnie PHP 8, dostarczając odpowiedzi na wiele problemów, które były bolączką programistów PHP.

Powody opóźnienia PHP6 i PHP7

Istniały dwa główne powody opóźnień w wydaniach nowych wersji PHP. Pierwszym z nich jest to, że istniała gałąź o nazwie PHP6, która miała na celu zapewnienie obsługi Unicode, aczkolwiek wokół tej wersji było wiele konfliktów, a wartość, jaką zapewniała, była nadal bardzo niska. Prawdą jest, że spowodowało to, że języki takie jak Python czy C# były lepsze w tym aspekcie, ale prawdą jest również to, że większość programistów i tak koduje w języku angielskim. Ostatecznie PHP6 zostało porzucone, choć kosztowało to społeczność wiele czasu i wysiłku, który w przeciwnym razie mógłby zostać lepiej wykorzystany.

Drugim powodem opóźnienia była kompatybilność wsteczna. Ponieważ 80% stron internetowych nadal używa PHP jako języka zaplecza, istniały zależności, które należało wziąć pod uwagę podczas opracowywania PHP7. Na szczęście deweloperzy wykonali świetną robotę i również tym razem pozwolili na pojawienie się innych rozwiązań, które pomogły w przejściu z PHP 5.6 do PHP 7 – takich jak PHP FPM

(pierwsze nieeksperymentalne wydanie w 2011 roku).

Główne problemy z PHP5

Głównym problemem PHP było to, że był zbyt łatwy do nauczenia – co było zarówno jego mocną, jak i słabą stroną. Podczas gdy mocna strona jest oczywista, słabą stroną było to, że programiści bez doświadczenia byli w stanie budować aplikacje, które działały, ale z powodu złego rozumowania i złego podejścia były albo niezabezpieczone, albo podatne na błędy. Nie wspominając już o ogromnym bałaganie w samych kodach.

Niektóre konkretne problemy, które spowodowały, że deweloperzy uznali PHP5 za przestarzałe i złe, to:

- Brak możliwości wymuszania typów parametrów funkcji
- Brak możliwości wymuszania zwrotów funkcji
- Zła obsługa wyjątków
- Brak nowoczesnych funkcji, takich jak operator statku kosmicznego, koalescencja zerowa, zdolność JIT
- Skomplikowane operacje na ciągach
- Brak natywnej obsługi JSON (tylko przez rozszerzenie)
- Ponieważ PHP5 nie wymuszało dobrego kodu z założenia, powstało wiele złego kodu

Warto zauważyć, że wszystkie powyższe i wiele więcej zostało rozwiązanych i naprawionych przez PHP 7 i 8.

Frameworki i PSR na ratunek

Niektóre z problemów, które pojawiły się w PHP5 zostały rozwiązane poprzez stworzenie standardów. Powstały następujące frameworki: Symphony, Laravel, CakePHP, Codeigniter; zostały one niemal natychmiast pokochane przez społeczność. Frameworki wymuszały pracę w określonych ramach – takich jak MVC lub MVVC – więc aplikacje tworzone przy ich użyciu były od razu lepiej zorganizowane.

W pewnym momencie powstały standardy PSR – można o nich przeczytać tutaj: https://www.php-fig.org/psr/. Standardy te obejmowały niektóre elementy, które są

egzekwowane (np. jak tworzyć klasy) i niektóre, które nie były egzekwowane (np. jak komentować swój kod). Nawet jeśli niektóre z nich nie były egzekwowane przez sam język, wiele edytorów (na przykład Visual Studio Code) pozwalało na instalowanie wtyczek, które pomagały programistom niemal automatycznie przestrzegać tych standardów.

Teraz również sztuczna inteligencja może być wykorzystywana do ulepszania kodu. Choć nadal często tworzy ona bezużyteczne programy, może być bardzo pomocna przy tworzeniu testów, automatyzacji komentarzy lub wskazywaniu drobnych błędów. Z naszego doświadczenia mogę powiedzieć, że jeśli jest odpowiednio skonfigurowana, może poprawić kod i dostarczanie aplikacji napisanych w PHP i Reactjs.

Wszystko to razem zaowocowało znacznie lepszym kodem. Warto zauważyć, że w Sailing Byte używamy wszystkich czterech wymienionych aspektów (PSR, frameworki, edytory z odpowiednimi wtyczkami i AI), co w połączeniu z wewnętrznymi procedurami i CI/CD sprawia, że jest to więcej niż odpowiednie do tworzenia wysokiej jakości online enterprise SaaS services.

.

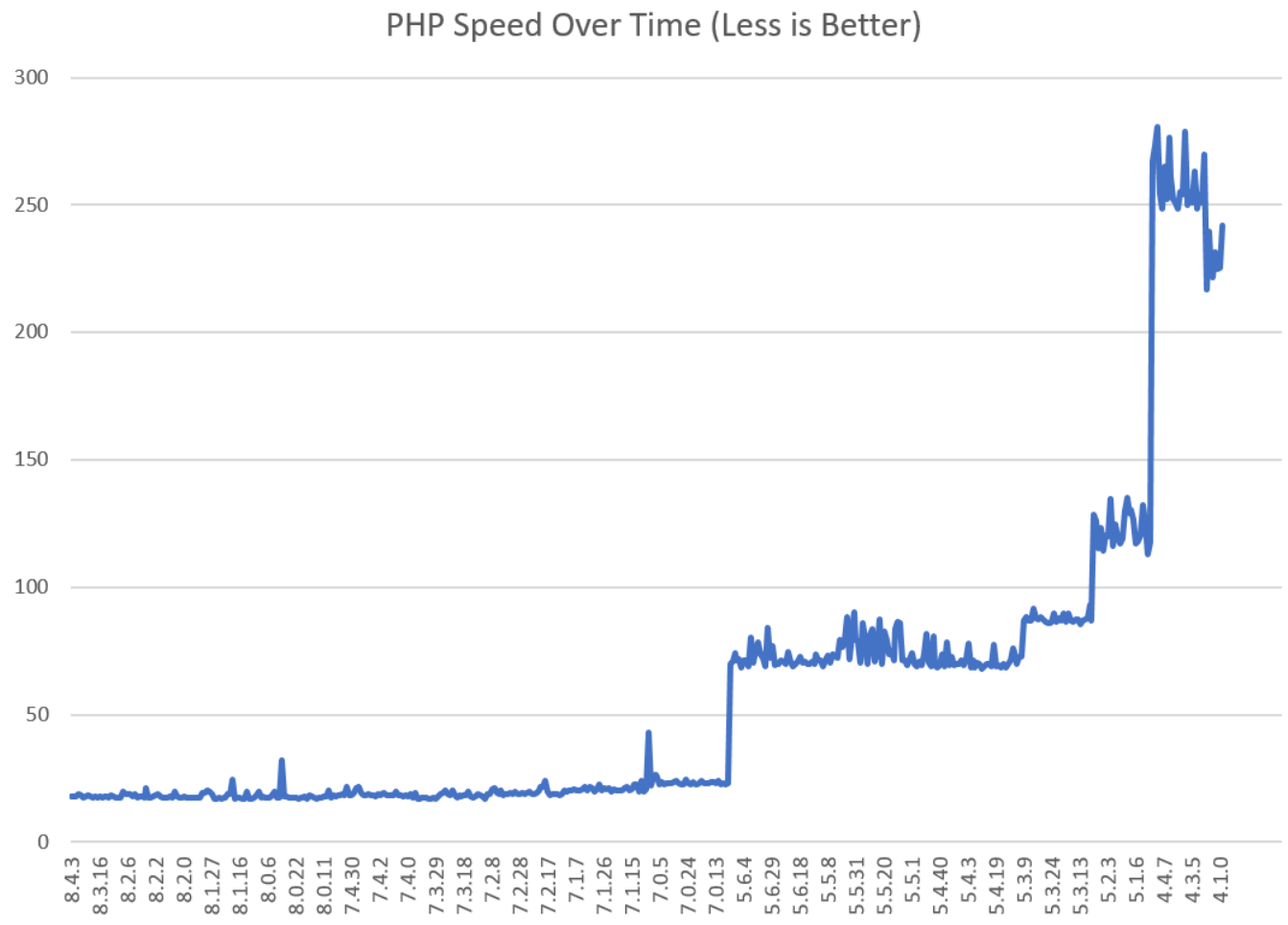
PHP 7 i PHP 8 – poprawa szybkości i pamięci

Przejście na PHP8 pozwala programistom na korzystanie ze znacznie większej liczby funkcji, które przypominają inne języki. Niektóre elementy są bardzo podobne do C# lub C++, a nawet języków skryptowych, takich jak Python. Pełną listę usprawnień można znaleźć na Wikipedii, o części z nich wspomniałem już w tym artykule, ale jest jeszcze jeden aspekt.

Pomiędzy PHP5.6 i PHP8.1 nastąpiła znaczna poprawa szybkości i wykorzystania pamięci. Zmierzono, że PHP8.1 jest 3 do 4 razy szybsze niż PHP5.6! Dodatkowo zużycie pamięci spadło około 3-krotnie. A PHP 8.4 jest jeszcze szybsze, jak widać na poniższym wykresie.

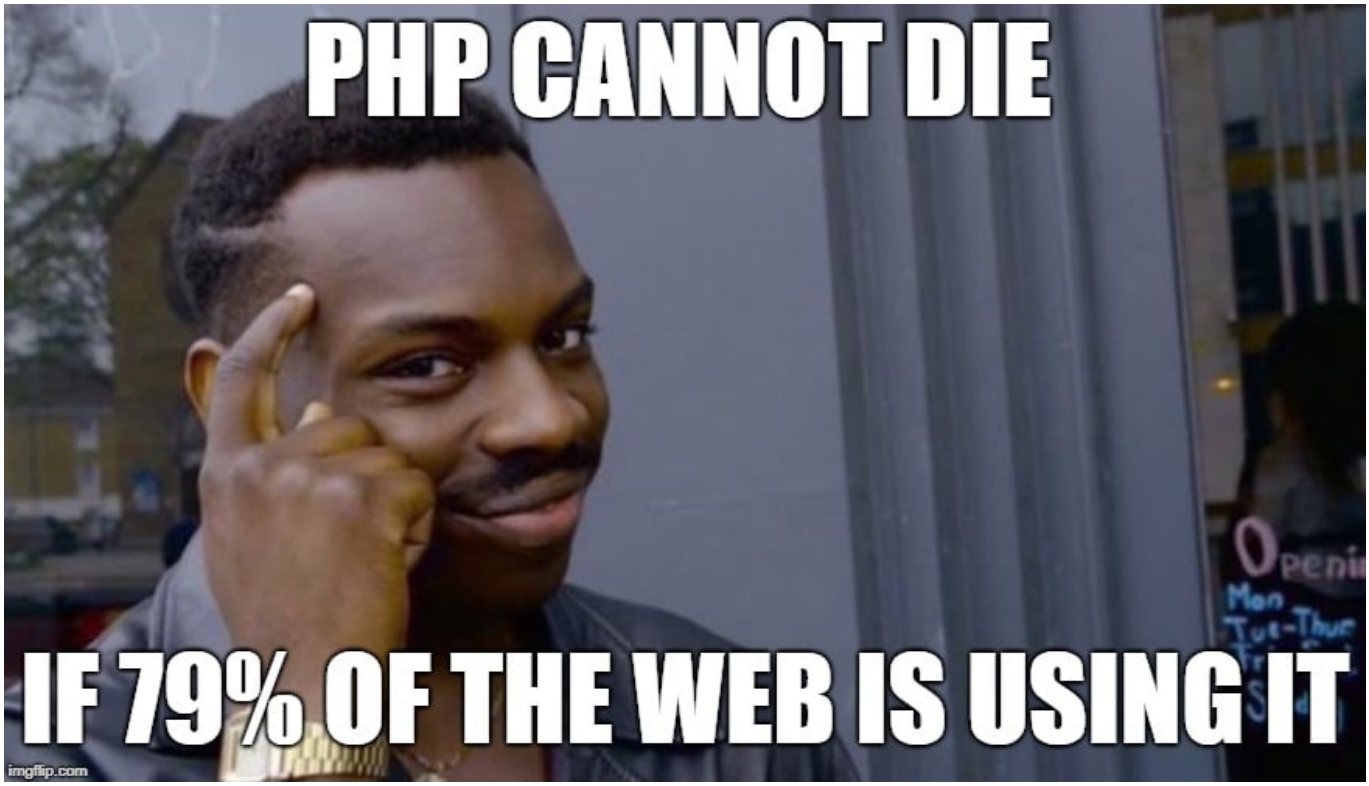
Jest to wynikiem przepisania ZEND – silnika interpretera PHP, napisanego w języku C. Zarówno optymalizacja kodu, jak i lepsza obsługa pamięci są obecnie znacznie ważniejszym czynnikiem niż w przeszłości, ponieważ użytkownicy końcowi oczekują, że strony internetowe będą otwierać się niemal natychmiast. Wyniki badań pokazują, że jeśli strona ładuje się zbyt wolno, użytkownicy chętniej ją zamykają,

nawet nie przeglądając jej zawartości. Właśnie dlatego poprawa szybkości jest tak ważna.



Utworzyłem ten wykres przy użyciu danych z [onlinephp.io](), gdzie przeprowadzają wiele testów na każdej wersji PHP.

Rynek potrzebuje aktualizacji PHP i Ty też powinieneś ją zaktualizować!



Prawie 79% stron internetowych korzysta z PHP. To bardzo dużo i obecnie nie sposób wyobrazić sobie sieci bez PHP. Ze względu na ogromny sukces tego języka, jeszcze przez wiele lat będziemy potrzebować programistów PHP – zarówno tych pracujących nad frameworkami (Laravel, Symfony), jak i tych pracujących nad stronami CMS (takimi jak Joomla czy WordPress).

Potrafi wiele zdziałać.

Co może być lepsze w nowych wersjach PHP?

Oczywiście, PHP nie jest idealnym językiem, wciąż wymaga pewnych ulepszeń. Nawet jeśli niektóre ulepszenia są już zaimplementowane, ludzie nie wiedzą o ich istnieniu (np. możliwość obsługi procesów – dzięki czemu możliwe jest wykonywanie asynchroniczne). To, co zdecydowanie można poprawić, to uczenie maszynowe dla sztucznej inteligencji – jest to dziedzina, która jest obecnie najlepiej obsługiwana przez Pythona. Innym obszarem, w którym PHP można ulepszyć, jest dodanie natywnych interfejsów dla urządzeń IoT (ponieważ jest ich coraz więcej), a także przygotowanie do obsługi Web3.0 (inteligentne kontrakty). Co faktycznie zostanie wdrożone? Przyszłość pokaże.

Jak aktualizować stronę bez jej niszczenia?

Proces ten musi być starannie zaplanowany. PHP rzadko żyje obecnie jako samodzielny skrypt: najprawdopodobniej używasz jakiegoś frameworka (takiego jak Laravel) lub CMS (takiego jak WordPress) i pakietów kompozytorów – aby przyspieszyć rozwój, być bezpieczniejszym i tak dalej. Jeśli chcesz, aby wszystko działało, PHP i zależności powinny być aktualizowane w określonej kolejności w zależności od tego, czego używasz. Oprócz tego należy pamiętać o komponentach, które żyją wokół tego wszystkiego, takich jak Apache/Nginx, MySQL/MariaDB, rozszerzenia PHP i tak dalej. Im starszy jest twój skrypt, tym bardziej skomplikowany może być proces – nawet do tego stopnia, że bardziej efektywne może być przepisanie wszystkiego od zera niż jego aktualizacja.

Gdzie jesteśmy teraz z PHP 8.4?

Jeśli nadal nie używasz PHP 8.4 lub przynajmniej PHP 8.3, to prawdopodobnie powinieneś

przeczytać ten artykuł o krytycznych kwestiach biznesowych związanych z używaniem przestarzałego oprogramowania. Korzystanie z przestarzałego oprogramowania stanowi ogromne ryzyko dla firmy. Znajdziesz tam również pomoc dotyczącą aktualizacji Laravel i WordPress. Jeśli jeszcze nie dokonałeś migracji do PHP 8.4, to być może niektóre z tych ulepszeń pomogą Ci podjąć decyzję:

- **Haki właściwości Jednym z najważniejszych dodatków są haki właściwości, które pozwalają właściwościom definiować niestandardową logikę operacji odczytu i zapisu bez tradycyjnych metod getter/setter.**
- **Asymetryczna widoczność właściwości PHP 8.4 wprowadza asymetryczną widoczność, pozwalając właściwościom mieć różne poziomy widoczności dla operacji odczytu i zapisu.**
- **Uproszczona instancja klasy Nowa funkcja bez nawiasów eliminuje potrzebę zawijania nowych wyrażeń w nawiasy podczas łączenia metod lub uzyskiwania dostępu do właściwości.**
- **Ulepszone funkcje tablicowe PHP 8.4 dodaje**

cztery nowe funkcje tablicowe, które upraszczają typowe zadania przetwarzania danych:

- **array_find(): Zwraca pierwszy element spełniający warunek**

  .

- **array_find_key(): Zwraca klucz pierwszego pasującego elementu**
- **array_any(): Sprawdza, czy co najmniej jeden element spełnia warunek**
- **array_all(): Sprawdza, czy wszystkie elementy spełniają warunek**

- **Deprecation Attribute Nowy atrybut #[Deprecated] zapewnia znormalizowany sposób oznaczania funkcji, metod i stałych klas jako przestarzałych**

  .

- **Nowoczesne API DOM PHP 8.4 wprowadza nowe API DOM z obsługą HTML5 poprzez przestrzeń nazw Dom. Nowy interfejs API zapewnia parsowanie zgodne ze specyfikacją i nowoczesne metody wygody.**

- **Obiektowy interfejs BcMath Nowa klasa BcMathNumber umożliwia obiektową arytmetykę dowolnej precyzji ze standardowymi operatorami matematycznymi.**
- **Dodatkowe ulepszenia PHP 8.4 zawiera wiele innych ulepszeń:**
  - **Ulepszone PDO: Podklasy specyficzne dla sterownika (PdoSqlite, PdoMySql itp.) z dostosowanymi metodami**
  - **Wielobajtowe funkcje łańcuchowe: mb_trim(), mb_ltrim(), mb_rtrim(), mb_ucfirst() i mb_lcfirst()**
  - **Nowe funkcje BCMath: bcceil(), bcdivmod(), bcfloor() i bcround()**
  - **Ulepszenia DateTime: Nowe metody tworzenia znaczników czasu i obsługi mikrosekund**
  - **Ulepszone zaokrąglanie: Nowe wyliczenie RoundingMode z dodatkowymi trybami zaokrąglania**
  - **Parsowanie żądań: Nowa funkcja request_parse_body() do parsowania żądań HTTP innych niż POST**
- **Wydajność i bezpieczeństwo Wydanie zawiera ulepszenia wydajności, zaktualizowaną bazę**

znaków Unicode do wersji 16, zwiększony domyślny koszt Bcrypt z 10 do 12 w celu zwiększenia bezpieczeństwa oraz różne poprawki błędów i ogólne porządki

.

**Więc cóż... na co czekasz? **

A co z PHP 9?

Nie ma obecnie daty premiery i być może będziemy musieli trochę poczekać. Prawdopodobnie zanim usłyszymy o PHP 9, zobaczymy PHP8.6 i 8.6. Możemy jednak przewidzieć pewne ulepszenia, których można się spodziewać w najbliższej przyszłości. Prawdopodobnie będą to:

- **Bardziej precyzyjna obsługa różnych typów zmiennych (takich jak traktowanie boolean i null jako liczb)**
- **Ograniczenie nieparzystych przyrostów ciągów**
- **Interpretowanie pustych ciągów jako liczb**
- **Lepsza obsługa serializacji**

- **Podpisy funkcji**
- **Surowsze reguły tworzenia tablic**
- **Uproszczona interpolacja ciągów**
- **Niektóre ostrzeżenia mogą być teraz błędami**
- **Oczywiście usunięcie przestarzałych funkcji**

Jaki jest więc wniosek?

Język PHP żyje i jest w dobrej formie – lepszej niż kiedykolwiek. Wiele stron internetowych wciąż z niego korzysta i nie wydaje się, by miało się to w jakikolwiek sposób zmienić. PHP jest znacznie bardziej nowoczesnym językiem niż lata temu, kiedy miał problemy – problemy, które zostały rozwiązane. Z tego powodu wciąż jest dużo pracy dla programistów PHP i jest pewne, że nie zmieni się to w przyszłości. Nowe wersje są regularnie wydawane, a każda z nich przynosi więcej ulepszeń do obecnego stanu, więc wygląda na to, że możemy spodziewać się jeszcze więcej ulepszeń czegoś, co już jest świetne.