

O'REILLY®



PHP

Nowe możliwości,
najlepsze praktyki

JĘZYK PHP POWRACA W WIELKIM STYLU!

Tytuł oryginału: Modern PHP

Tłumaczenie: Łukasz Piwko

ISBN: 978-83-283-1402-3

© 2015 Helion S.A.

Authorized Polish translation of the English edition of Modern PHP,
ISBN 9781491905012 © 2015 Josh Lockhart.

This translation is published and sold by permission of O'Reilly Media, Inc.,
which owns or controls all rights to publish and sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means,
electronic or mechanical, including photocopying, recording or by any information storage retrieval system,
without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej
publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną,
fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje
naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich
właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były
kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane
z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie
ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji
zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/phpnom>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<ftp://ftp.helion.pl/przyklady/phpnom.zip>

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Wstęp	13
-------------	----

Część I Składniki języka

1. Nowy PHP	19
Przeszłość	19
Teraźniejszość	20
Przyszłość	21
2. Funkcjonalność	23
Przestrzenie nazw	23
Do czego służą przestrzenie nazw	25
Deklaracja	25
Importowanie i aliasy	26
Porady	28
Interfejsy	30
Cechy	33
Do czego służą cechy	34
Jak utworzyć cechę	35
Sposób użycia cech	36
Generatory	37
Tworzenie generatora	37
Sposób użycia generatorów	38
Zamknięcia	40
Tworzenie zamknięć	40
Wiązanie stanu	41
Rozszerzenie Zend OPcache	43
Włączanie rozszerzenia Zend OPcache	44
Konfiguracja Zend OPcache	44
Korzystanie z Zend OPcache	45

Wbudowany serwer HTTP	46
Uruchamianie serwera	46
Konfiguracja serwera	46
Skrypty trasujące	47
Wykrywanie serwera wbudowanego	47
Wady	47
Co dalej	48

Część II Najlepsze praktyki

3. Standardy	51
PHP-FIG na ratunek	51
Współpraca między systemami szkieletowymi	52
Interfejsy	52
Automatyczne wczytywanie zależności	53
Styl	53
Co to jest PSR	53
PSR-1 — podstawowy styl kodowania	54
PSR-2 — restrykcyjny styl kodowania	55
PSR-3 — interfejs rejestratora dziennikowego	58
PSR-4 — automatyczne wczytywanie zależności	60
4. Komponenty	63
Dlaczego używać komponentów	63
Czym są komponenty	64
Komponenty a systemy szkieletowe	65
Nie wszystkie systemy szkieletowe są złe	65
Używaj odpowiednich narzędzi	66
Wyszukiwanie komponentów	67
Kupuj	67
Wybieraj	68
Pozwól poznać swoją opinię	69
Używanie komponentów PHP	69
Jak zainstalować Composera	70
Sposób użycia Composera	71
Przykładowy projekt	72
Composer i prywatne repozytoria	75
Tworzenie komponentów PHP	76
Nazwy dostawcy i pakietu	76
Przestrzenie nazw	77
Hierarchia plików	77

Plik composer.json	78
Plik README	80
Implementacja komponentu	80
Kontrola wersji	82
Wysyłanie komponentu do katalogu Packagist	82
Używanie komponentu	83
5. Najlepsze praktyki	85
Dezynfekcja, sprawdzanie i kontrola wyjścia	85
Dezynfekcja danych wejściowych	86
Sprawdzanie poprawności danych	88
Kontrolowanie danych wyjściowych	89
Hasła	90
Nie poznawaj haseł użytkowników	90
Nigdy nie ograniczaj haseł użytkowników	90
Nigdy nie wysyłaj haseł na adres e-mail	91
Mieszaj hasła użytkowników za pomocą algorytmu bcrypt	91
API mieszania haseł	92
Mieszanie haseł w wersjach PHP starszych od 5.5.0	96
Data, godzina i strefa czasowa	96
Ustawianie domyślnej strefy czasowej	96
Klasa DateTime	96
Klasa DateInterval	97
Klasa DateTimeZone	98
Klasa DatePeriod	99
Komponent nesbot/carbon	100
Bazy danych	100
Rozszerzenie PDO	101
Połączenia z bazą danych i nazwy źródeł danych	101
Instrukcje przygotowane	103
Wyniki zapytań	105
Transakcje	107
Łańcuchy wielobajtowe	110
Kodowanie znaków	111
Zwracanie danych w formacie UTF-8	111
Strumienie	112
Opakowania strumieni	112
Kontekst strumienia	115
Filtry strumieni	115
Tworzenie własnych filtrów strumieni	117

Błędy i wyjątki	119
Wyjątki	120
Procedury obsługi wyjątków	123
Błędy	123
Procedury obsługi błędów	125
Błędy i wyjątki w czasie pracy	126
Produkcja	128

Część III Wdrażanie, testowanie i dostrajanie

6. Hosting	133
Serwer współdzielony	133
Serwer wirtualny	134
Serwer dedykowany	134
PaaS	135
Wybór planu hostingowego	135
7. Wdrażanie serwera	137
Cel	137
Konfiguracja serwera	138
Pierwsze logowanie	138
Aktualizacja oprogramowania	139
Inny użytkownik niż root	139
Uwierzytelnianie poprzez SSH przy użyciu pary kluczy	140
Wyłączanie haseł i możliwości logowania się użytkownika root	141
PHP-FPM	142
Instalacja	142
Konfiguracja globalna	143
Konfiguracja puli	143
Serwer nginx	146
Instalacja	146
Host wirtualny	146
Automatyzacja wdrażania serwera	149
Zlecenie wdrażania serwerów	149
Dodatkowe źródła	149
Co dalej	149
8. Dostrajanie	151
Plik php.ini	151
Pamięć	152
Zend OPcache	153

Wysyłanie plików	154
Maksymalny czas wykonywania	155
Obsługa sesji	156
Buforowanie wyników	156
Bufor ścieżek	157
W następnym rozdziale	157
9. Wdrażanie aplikacji	159
Kontrola wersji	159
Automatyzacja wdrażania	159
Prostota	159
Przewidywalność	160
Odwracalność	160
Capistrano	160
Jak to działa	160
Instalacja	161
Konfiguracja	161
Uwierzytelnianie	162
Przygotowanie serwera	163
Haki Capistrano	163
Wdrażanie aplikacji	164
Cofanie aplikacji	164
Dodatkowe informacje	164
W następnym rozdziale	164
10. Testowanie	165
Jak testować	165
Kiedy testować	166
Przed	166
W trakcie	166
Po	166
Co testować	166
Jak testować	167
Testy jednostkowe	167
Programowanie przez testy	167
Programowanie behawioralne	167
PHPUnit	168
Struktura katalogów	169
Instalacja PHPUnit	170
Instalacja Xdebug	170
Konfiguracja PHPUnit	170

Klasa Whovian	171
Przypadek testowy WhovianTest	172
Uruchamianie testów	175
Pokrycie kodu	175
Ciągłe testowanie przy użyciu programu Travis CI	176
Konfiguracja	176
Wykonywanie testów	177
Dodatkowe źródła	178
W następnym rozdziale	178
11. Profilowanie	179
Kiedy używać profilerów	179
Typy profilerów	179
Xdebug	180
Konfiguracja	180
Uruchamianie	181
Analiza	181
XHProf	181
Instalacja	181
XHGUI	182
Konfiguracja	182
Uruchamianie	183
Profiler New Relic	183
Profiler Blackfire	183
Dodatkowe źródła	184
W następnym rozdziale	184
12. HHVM i Hack	185
HHVM	185
PHP w Facebooku	186
Zgodność HHVM i Zend Engine	187
Czy interpreter HHVM jest dla mnie	187
Instalacja	188
Konfiguracja	189
Rozszerzenia	189
Monitorowanie HHVM za pomocą narzędzia Supervisor	190
HHVM, FastCGI i nginx	191
Język Hack	192
Konwersja PHP na Hack	192
Co to jest typ	193
Typowanie statyczne	194

Typowanie dynamiczne	195
Hack jest uniwersalny	195
Sprawdzanie typów w języku Hack	196
Tryby Hack	196
Składnia języka Hack	197
Struktury danych języka Hack	199
HHVM i Hack a PHP	199
Dodatkowe źródła informacji	200
13. Społeczność	201
Lokalne grupy użytkowników PHP	201
Konferencje	201
Mentoring	202
Trzymaj rękę na pulsie	202
Strony internetowe	202
Listy mailingowe	202
Twitter	202
Podcasty	202
Humor	202

Dodatki

A Instalacja PHP	205
B Lokalne środowiska deweloperskie	221
Skorowidz	227

Wdrażanie serwera

Po wybraniu hostingu dla aplikacji należy skonfigurować serwer i przygotować go do działania. Szczerze mówiąc, wdrażanie serwera to bardziej sztuka niż nauka. Wszystko zależy od wymagań aplikacji, której chcesz na danym serwerze używać.



Jeśli korzystasz z usługi typu PaaS, to zarządzaniem serwerem zajmuje się dostawca. Twoja rola sprowadza się do wykonywania usługi dostawcy, aby przenieść aplikację na jego platformę.

Jeśli nie korzystasz z usługi typu PaaS, to musisz przygotować do działania swój serwer VPS lub dedykowany. „Wdrażanie serwera” brzmi strasznie, ale w rzeczywistości nie jest takie trudne (nie śmieję się), choć wymaga podstawowej umiejętności posługiwania się wierszem poleceń. Jeżeli konsola to dla Ciebie czarna magia, to lepiej skorzystaj z usług PaaS takich firm jak Engine Yard albo Heroku.

Nie uważam się za administratora systemów, ale znajomość podstaw administracji systemami jest niezwykle przydatna dla każdego programisty, ponieważ ułatwia mu pracę i stworzenie elastycznego środowiska pracy. W tym rozdziale podzielę się z Tobą swoją wiedzą na ten temat, abyś mógł bez problemu uruchomić terminal i za jego pomocą wdrożyć serwer dla swojej aplikacji. Na koniec podsunę kilka dodatkowych źródeł, w których można znaleźć więcej informacji.



W tym rozdziale zakładam, że umiesz edytować pliki tekstowe za pomocą działającego w wierszu poleceń edytora typu nano (<http://www.nano-editor.org>) lub vim (<http://www.vim.org>) (są dostępne w większości linuxów). Jeśli nie, musisz znaleźć inny sposób na otwieranie i edytowanie plików na swoim serwerze.

Cel

Najpierw musisz wydzierżawić serwer wirtualny lub dedykowany. Następnie trzeba zainstalować na nim serwer sieciowy, który będzie obsługiwał żądania HTTP. Na koniec należy skonfigurować grupę procesów PHP do obsługi żądań PHP i zarządzać nimi. Procesy te muszą współpracować z serwerem.

Jeszcze kilka lat temu standardowo instalowało się serwer Apache i jego moduł `mod_php`. Serwer ten tworzy proces potomny do obsługi *każdego* żądania HTTP. Moduł `mod_php` osadza interpreter PHP w każdym takim procesie — nawet w tych, które serwują tylko statyczne zasoby, takie jak JavaScript, obrazy czy arkusze stylów. Jest to nieefektywne zarządzanie zasobami. Dlatego serwera Apache używa coraz mniej programistów, którzy przerzucają się na bardziej wydajne rozwiązania.

Dziś używa się serwera `nginx` (<http://nginx.org/>), który rezyduje przed kolekcją procesów PHP-FPM (i przekazuje do niej żądania PHP). To właśnie rozwiązanie przedstawiam w tym rozdziale.

Konfiguracja serwera

Zacniemy od konfiguracji serwera VPS. Jestem absolutnym wielbicielem firmy Linode (<http://linode.com/>). Może nie jest to najtańszy dostawca usług, ale jeden z najbardziej niezawodnych. Wejdź na stronę Linode (albo swojego ulubionego dostawcy usług hostingowych) i zamów serwer VPS. Zostaniesz poproszony o wybranie dystrybucji Linuksa i zdefiniowanie hasła użytkownika `root` swojego nowego serwera.



Wielu dostawców serwerów VPS, np. Linode (<http://linode.com/>) i Digital Ocean (<https://www.digitalocean.com>), pobiera opłaty na godziny. To znaczy, że można uruchomić serwer VPS i pobawić się nim praktycznie za darmo.

Pierwsze logowanie

Gdy serwer jest już gotowy, należy się zalogować. Uruchom terminal w swoim lokalnym komputerze i połącz się z serwerem za pomocą polecenia `ssh`. Nie zapomnij wpisać adresu IP swojego VPS-a:

```
ssh root@123.456.78.90
```

Możesz zostać poproszony o potwierdzenie autentyczności swojego nowego serwera. Wpisz **yes** i naciśnij klawisz *Enter*:

```
The authenticity of host '123.456.78.90 (123.456.78.90)' can't be established.  
RSA key fingerprint is 21:eb:37:f3:a5:d3:c0:77:47:c4:15:3d:3c:dc:3c:d1.  
Are you sure you want to continue connecting (yes/no)?
```

Następnie zostaniesz poproszony o podanie hasła użytkownika `root`. Wpisz je i naciśnij klawisz *Enter*:

```
root@123.456.78.90's password:
```

Jesteś zalogowany w swoim nowym serwerze!

Aktualizacja oprogramowania

Kolejną czynnością powinno być zaktualizowanie systemu operacyjnego za pomocą poniższych poleceń:

```
# Ubuntu
apt-get update;
apt-get upgrade;

# CentOS
yum update
```

W trakcie ich wykonywania w konsoli będzie wyświetlanych wiele informacji dotyczących pobierania i instalowania aktualizacji różnych składników systemu. Jest to bardzo ważna pierwsza czynność, ponieważ zapewnia najnowsze aktualizacje i łatki zabezpieczeń serwera.

Inny użytkownik niż root

Nowy serwer nie jest zabezpieczony. Oto kilka porad, jak zwiększyć jego bezpieczeństwo.

Utwórz konto *innego użytkownika* niż *root* i w przyszłości do logowania się używaj tego nowego konta. Użytkownik *root* ma nieograniczone uprawnienia, jest bogiem. Może wykonać każde polecenie bez pytania o zgodę. *Dostęp do serwera przy użyciu konta tego użytkownika powinien być maksymalnie utrudniony.*

Ubuntu

Za pomocą polecenia przedstawionego na listingu 7.1 utwórz nowego użytkownika o nazwie `deploy`. Gdy zostaniesz o to poproszony, wpisz hasło i wykonaj pozostałe wyświetlane na ekranie instrukcje.

Listing 7.1. Tworzenie użytkownika innego niż root w systemie Ubuntu

```
adduser deploy
```

Następnie przypisz użytkownika `deploy` do grupy `sudo` za pomocą poniższego polecenia:

```
usermod -a -G sudo deploy
```

W ten sposób nadasz temu użytkownikowi uprawnienia `sudo`, czyli umożliwisz mu wykonywanie zadań wymagających zwiększonych uprawnień po podaniu hasła.

CentOS

Utwórz nowego użytkownika o nazwie `deploy` za pomocą polecenia przedstawionego poniżej. Gdy zostaniesz o to poproszony, wpisz i potwierdź hasło:

```
adduser deploy
```

Następnie przypisz użytkownika `deploy` do grupy `wheel` za pomocą poniższego polecenia:

```
usermod -a -G wheel deploy
```

W ten sposób nadasz temu użytkownikowi uprawnienia `sudo`, czyli umożliwisz mu wykonywanie zadań wymagających zwiększonych uprawnień po podaniu hasła.

Uwierzytelnianie poprzez SSH przy użyciu pary kluczy

Aby zalogować się na serwerze jako użytkownik `deploy` ze swojego komputera, możesz skorzystać z poniższego polecenia:

```
ssh deploy@123.456.78.90
```

Zostaniesz poproszony o podanie hasła tego użytkownika i gdy spełnisz tę prośbę, zostaniesz zalogowany. Proces logowania można lepiej zabezpieczyć przez wyłączenie uwierzytelniania za pomocą hasła. Wadą haseł jest to, że są podatne na ataki typu brute force polegające na wielokrotnym zgadywaniu hasła w krótkich sesjach. Zamiast tego skorzystamy z **uwierzytelniania przez SSH przy użyciu pary kluczy** (ang. *SSH key-pair authentication*).

Uwierzytelnianie tą metodą to skomplikowany temat. Najprościej mówiąc, polega na utworzeniu pary „kluczy” na komputerze lokalnym. Jeden z nich jest prywatny (pozostaje na maszynie lokalnej), a drugi jest publiczny (zostaje wysłany na serwer). Nazwa **para kluczy** bierze się stąd, że wiadomości zaszyfrowane przy użyciu klucza publicznego można rozszyfrować tylko przy użyciu odpowiedniego klucza prywatnego.

Gdy zalogujesz się do serwera, korzystając z uwierzytelniania poprzez SSH parą kluczy, na serwerze zostanie utworzona losowa wiadomość, która zostaje zaszyfrowana przy użyciu klucza publicznego i wysłana do Twojego komputera lokalnego. W komputerze lokalnym następuje rozszyfrowanie wiadomości przy użyciu klucza prywatnego i zwrócenie jej do serwera. Wtedy serwer sprawdza tę rozszyfrowaną wiadomość i przydziela dostęp do serwera. Bardzo wszystko uprościłem, ale myślę, że wiesz, o co chodzi.

Jeśli logujesz się na serwerze z wielu różnych komputerów, to lepiej nie korzystaj z metody uwierzytelniania przy użyciu pary kluczy, ponieważ musiałbyś wygenerować osobną parę dla każdego z tych komputerów oraz wysłać na serwer klucz publiczny z każdej z tych par. W takich przypadkach lepszym rozwiązaniem jest pozostanie przy uwierzytelnianiu za pomocą hasła. Ale jeśli do logowania się na serwerze używasz tylko jednego komputera (jak robi to wielu programistów), to powinieneś wybrać metodę z parą kluczy. Klucze można wygenerować na komputerze lokalnym za pomocą poniższego polecenia:

```
ssh-keygen
```

Wykonaj pojawiające się na ekranie instrukcje i podaj potrzebne informacje, gdy zostaniesz o to poproszony. Polecenie to tworzy dwa pliki w komputerze lokalnym: `~/.ssh/id_rsa.pub` (klucz publiczny) i `~/.ssh/id_rsa` (klucz prywatny). Klucz prywatny powinien pozostać na komputerze lokalnym i nie należy go nikomu ujawniać. Klucz publiczny natomiast wysyła się na serwer. Można to zrobić za pomocą polecenia `scp` (ang. *secure copy* — bezpieczne kopiowanie):

```
scp ~/.ssh/id_rsa.pub deploy@123.456.78.90:
```

Nie zapomnij dodać dwukropka na końcu! Polecenie to wysyła klucz publiczny do katalogu głównego użytkownika `deploy` na serwerze. Następnie zaloguj się na serwerze jako użytkownik `deploy`. Po zalogowaniu sprawdź, czy istnieje katalog `~/.ssh`. Jeśli nie, utwórz go za pomocą poniższego polecenia:

```
mkdir ~/.ssh
```

W dalszej kolejności utwórz plik `~/.ssh/authorized_keys` za pomocą poniższego polecenia:

```
touch ~/.ssh/authorized_keys
```

W pliku tym będzie zapisana lista kluczy publicznych, które mogą być używane do logowania się na tym serwerze. Wykonaj poniższe polecenie, aby dodać swój przed chwilą wysłany na serwer klucz do pliku `~/.ssh/authorized_keys`:

```
cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
```

Na koniec należy jeszcze zmodyfikować uprawnienia dostępu do niektórych katalogów i plików, aby tylko użytkownik `deploy` miał dostęp do swojego katalogu `~/.ssh` oraz mógł odczytać zawartość swojego pliku `~/.ssh/authorized_keys`. Wykonaj poniższe polecenia:

```
chown -R deploy:deploy ~/.ssh;  
chmod 700 ~/.ssh;  
chmod 600 ~/.ssh/authorized_keys;
```

Zrobione! Od tej pory powinieneś móc połączyć się ze swojego komputera z serwerem bez podawania hasła.



Połączenie przez SSH z serwerem bez hasła jest możliwe tylko z komputera, na którym znajduje się klucz prywatny!

Wyłączanie haseł i możliwości logowania się użytkownika root

Serwer można jeszcze lepiej zabezpieczyć. W tym celu wyłączymy uwierzytelnianie przy użyciu hasła wszystkim użytkownikom oraz uniemożliwimy logowanie się użytkownika `root` — i koniec. Przypomnę, że użytkownik `root` może zrobić wszystko i dlatego staramy się maksymalnie utrudnić dostęp do serwera przy użyciu jego konta.

Zaloguj się na serwerze jako użytkownik `deploy` i otwórz plik `/etc/ssh/sshd_config` w dowolnym edytorze tekstu. Jest to plik konfiguracyjny serwera SSH. Znajdź w nim ustawienie `PasswordAuthentication` i zmień jego wartość na `no`. Jeśli jest wyłączone przez komentarz `;`, usuń go, aby je włączyć. Następnie znajdź ustawienie `PermitRootLogin` i jego wartość również zmień na `no`. Jeśli jest wyłączone przez komentarz `;`, usuń go, by je włączyć. Zapisz zmiany i aby je zastosować, ponownie uruchom serwer SSH za pomocą poniższego polecenia:

```
# Ubuntu  
sudo service ssh restart  
  
# CentOS  
sudo systemctl restart sshd.service
```

Gotowe. Zabezpieczyłeś serwer, możesz więc przejść do instalowania programów potrzebnych do działania aplikacji PHP. Wszystkie dalsze instrukcje należy wykonywać na serwerze jako użytkownik `deploy`.



Bezpieczeństwo serwerów to dynamiczna dziedzina, w której postępy cały czas należy śledzić. Oprócz zastosowania wcześniej opisanych zabiegów zalecam też zainstalowanie zapory ogniowej. Użytkownicy systemu Ubuntu mogą skorzystać z UFW (<https://help.ubuntu.com/community/UFW>), a właściciele CentOS-a mogą używać iptables (<http://wiki.centos.org/HowTos/Network/IPTables>).

PHP-FPM

PHP-FPM (<http://php.net/manual/en/install.fpm.php>) (ang. *PHP FastCGI Process Manager*) to oprogramowanie zarządzające pulą powiązanych procesów PHP, która odbiera i obsługuje żądania od serwera sieciowego, takiego jak np. nginx. PHP-FPM tworzy jeden nadrzędny proces (zazwyczaj wykonywany przez użytkownika *root* systemu operacyjnego), który steruje przekazywaniem żądań HTTP do procesów potomnych. Ponadto proces nadrzędny sprawuje kontrolę nad tworzeniem procesów potomnych PHP (do obsługi dodatkowego ruchu sieciowego w aplikacji) i ich kasowaniem (jeśli są zbyt stare i niepotrzebne). Każdy proces puli PHP-FPM żyje dłużej niż pojedyncze żądanie HTTP i może obsługiwać 10, 50, 100, 500, a nawet więcej żądań HTTP.

Instalacja

PHP-FPM najłatwiej zainstalować przy użyciu systemowego menedżera pakietów, za pomocą poleceń pokazanych poniżej.



Szczegółowy poradnik na temat instalacji PHP-FPM znajduje się w dodatku A.

Ubuntu

```
sudo apt-get install python-software-properties;
sudo add-apt-repository ppa:ondrej/php5-5.6;
sudo apt-get update;
sudo apt-get install php5-fpm php5-cli php5-curl \
    php5-gd php5-json php5-mcrypt php5-mysqlnd;
```

CentOS

```
sudo rpm -Uvh \
    http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-5.noarch.rpm;
sudo rpm -Uvh \
    http://rpms.famillecollet.com/enterprise/remi-release-7.rpm;
sudo yum -y --enablerepo=epel,remi,remi-php56 install php-fpm php-cli php-gd \
    php-mbstring php-mcrypt php-mysqlnd php-opcache php-pdo php-devel;
```



Jeśli instalacja za pomocą polecenia EPEL rpm nie uda się, uruchom przeglądarkę internetową i wejdź na stronę http://dl.fedoraproject.org/pub/epel/7/x86_64/e/. Poszukaj na niej aktualnej wersji pakietu EPEL i użyj jej.

Konfiguracja globalna

W Ubuntu główny plik konfiguracji PHP-FPM to `/etc/php5/fpm/php-fpm.conf`. W systemie CentOS jest to z kolei plik `/etc/php-fpm.conf`. Otwórz go w wybranym edytorze tekstu.



Pliki konfiguracyjne PHP-FPM są w formacie INI. Więcej informacji na jego temat można znaleźć w Wikipedii (<https://pl.wikipedia.org/wiki/INI>).

Poniżej opisuję dwa najważniejsze ustawienia, które zalecam zmienić. Domyślnie mogą one być wyłączone za pomocą komentarza, więc w razie potrzeby włącz je, usuwając ten komentarz. Ustawienia te nakazują głównemu procesowi PHP-FPM ponownie rozpocząć działanie, jeśli określona liczba jego procesów potomnych ulegnie awarii w zdefiniowanym czasie. Jest to podstawowe zabezpieczenie procesów PHP-FPM, które może pomóc rozwiązać proste problemy. Nie pomoże jednak w przypadku takich usterek jak źle napisany kod PHP.

```
emergency_restart_threshold = 10
```

Maksymalna liczba procesów potomnych PHP-FPM, które mogą ulec awarii w określonym czasie, zanim główny proces PHP-FPM dokona ponownego uruchomienia.

```
emergency_restart_interval = 1m
```

Ilość czasu dotycząca ustawienia `emergency_restart_threshold`.



Więcej informacji na temat globalnych ustawień konfiguracyjnych PHP-FPM można znaleźć na stronie <http://php.net/manual/en/install.fpm.configuration.php>.

Konfiguracja puli

W innym miejscu pliku konfiguracyjnego PHP-FPM znajduje się sekcja o nazwie `Pool Definitions` (definicje pul). W tym podrozdziale opisuję ustawienia dla każdej z nich. Pula PHP-FPM to zbiór powiązanych ze sobą procesów potomnych PHP. Zazwyczaj każda aplikacja ma własną osobną pulę.

W Ubuntu sekcja `Pool Definitions` zawiera następujący tekst:

```
include=/etc/php5/fpm/pool.d/*.conf
```

W systemie CentOS pliki definicji pul są dołączane na początku głównego pliku konfiguracyjnego za pomocą poniższego kodu:

```
include=/etc/php-fpm.d/*.conf
```

Kod ten nakazuje PHP-FPM załadowanie poszczególnych plików z definicjami pul z katalogu `/etc/php5/fpm/pool.d/` (Ubuntu) lub `/etc/php-fpm.d/ directory` (CentOS). Przejdź do tego katalogu, w którym powinien znajdować się jeden plik o nazwie `www.conf`. Jest to plik konfiguracyjny dla domyślnej puli PHP-FPM o nazwie `www`. Otwórz go w wybranym edytorze tekstu.



Każda konfiguracja puli PHP-FPM zaczyna się od znaku [, po którym wpisuje się nazwę puli i znak]. Na przykład domyślna konfiguracja PHP-FPM zaczyna się od napisu [www].

Każda pula PHP-FPM działa jako użytkownik systemu operacyjnego i określona grupa. Osobiście zazwyczaj uruchamiam pule jako osobnych użytkowników bez uprawnień użytkownika *root*. Dzięki temu mam ułatwione zadanie, gdy chcę znaleźć procesy wybranej aplikacji w wierszu poleceń za pomocą poleceń `top` i `ps aux`. Jest to też po prostu dobry nawyk, ponieważ procesy każdej puli PHP-FPM są ograniczone uprawnieniami swojego użytkownika systemowego i grupy.

Teraz pokażę, jak skonfigurować domyślną pulę `www`, aby działała jako użytkownik i grupa `deploy`. Otwórz zatem plik konfiguracyjny puli `www` w dowolnym edytorze tekstu. Poniżej opisuję ustawienia, których domyślne wartości zalecam zmienić:

`user = deploy`

Użytkownik systemowy będący właścicielem procesów potomnych tej puli PHP-FPM. Należy wpisać nazwę użytkownika systemowego niemającego uprawnień użytkownika *root*.

`group = deploy`

Grupa systemowa będąca właścicielem procesów potomnych tej puli PHP-FPM. Należy wpisać nazwę użytkownika systemowego niemającego uprawnień użytkownika *root*.

`listen = 127.0.0.1:9000`

Adres IP i numer portu, na którym pula PHP-FPM nasłuchuje i przyjmuje żądania od serwera `nginx`. Wartość `127.0.0.1:9000` oznacza, że pula nasłuchuje połączeń na lokalnym porcie 9000. Ja używam portu o tym numerze, ale może to być dowolny inny nieuprzywilejowany port (każdy o numerze większym od 1024), który nie jest jeszcze używany przez inny proces systemowy. Do tego ustawienia wracam jeszcze przy opisie konfiguracji hosta wirtualnego na serwerze `nginx`.

`listen.allowed_clients = 127.0.0.1`

Adresy IP, które mogą wysyłać żądania do tej puli PHP-FPM. Ze względów bezpieczeństwa ustawiam tę opcję na `127.0.0.1`, co oznacza, że tylko bieżący komputer może przekazywać żądania do tej puli. Domyślnie ustawienie to może być wyłączone za pomocą komentarza, konieczne może być więc jego włączenie.

`pm.max_children = 51`

Wartość określająca maksymalną liczbę istniejących jednocześnie procesów puli PHP-FPM. Nie da się powiedzieć, jaka wartość jest najlepsza w tym ustawieniu. Trzeba przetestować aplikację, sprawdzić, ile pamięci wykorzystuje każdy proces, i ustawić taką liczbę procesów, jaką komputer jest w stanie obsłużyć. Większość małych i średnich aplikacji PHP wykorzystuje od 5 do 15 MB pamięci na proces (choć to tylko przybliżone dane). Jeśli dysponujesz 512 MB pamięci dla puli PHP-FPM, to możesz tę opcję ustawić na wartość 512 pamięci / 10 MB na proces, czyli 51 procesów.

`pm.start_servers = 3`

Liczba procesów puli PHP-FPM, które są dostępne natychmiast po uruchomieniu PHP-FPM. W tym przypadku również nie da się podać jedynej poprawnej wartości. Dla większości małych i średnich aplikacji zalecam wartość 2 lub 3. Dzięki temu początkowe żądania aplikacji nie muszą czekać na inicjację procesów PHP-FPM, ponieważ dwa lub trzy procesy już działają i czekają.

`pm.min_spare_servers = 2`

Najmniejsza liczba procesów puli PHP-FPM, jaka istnieje, gdy aplikacja PHP jest nieaktywna. Wartość ta powinna mieścić się w podobnych granicach co wartość ustawienia `pm.start_servers`. Ustawienie to gwarantuje, że nowe żądania HTTP nie będą musiały czekać na inicjację nowych procesów do obsługi żądań.

`pm.max_spare_servers = 4`

Największa liczba procesów puli PHP-FPM, jaka istnieje, gdy aplikacja PHP jest nieaktywna. Typowo wartość tego ustawienia powinna być nieco większa niż ustawienia `pm.start_servers`. Ustawienie to gwarantuje, że nowe żądania HTTP nie będą musiały czekać na inicjację nowych procesów do obsługi żądań.

`pm.max_requests = 1000`

Maksymalna liczba żądań HTTP, jaką każdy proces z puli PHP-FPM obsługuje, zanim zostanie poddany recyklingowi. Ustawienie to pomaga uniknąć akumulacji wycieków pamięci powodowanych przez źle napisane rozszerzenia i biblioteki PHP. Zalecam wartość 1000, ale lepiej dostosować ją do konkretnych potrzeb aplikacji.

`slowlog = /ścieżka/do/slowlog.log`

Bezwzględna ścieżka systemowa do pliku dziennika, gdzie rejestrowane są informacje o żądaniach HTTP, których przetwarzanie trwało dłużej niż `{n}` sekund. Informacje te są przydatne w znajdowaniu wąskich gardeł w aplikacjach PHP. Pamiętaj, że użytkownik i grupa puli muszą mieć uprawnienia zapisu w tym pliku. Wartość `/ścieżka/do/slowlog.log` to tylko przykład. Zamień ją na własną ścieżkę.

`request_slowlog_timeout = 5s`

Ilość czasu, po jakiej następuje zrzut zawartości stosu bieżącego żądania HTTP do pliku dziennika określonego w ustawieniu `slowlog`. Wybór wartości zależy od tego, co uważamy za powolne żądanie. Na początek rozsądnym ustawieniem wydaje się 5s.

Po wprowadzeniu i zapisaniu zmian w pliku konfiguracyjnym PHP-FPM ponownie uruchom proces nadrzędny za pomocą poniższego polecenia:

```
# Ubuntu
```

```
sudo service php5-fpm restart
```

```
# CentOS
```

```
sudo systemctl restart php-fpm.service
```



Więcej o konfiguracji pul PHP-FPM można przeczytać na stronie <http://php.net/manual/install.fpm.configuration.php>.

Serwer nginx

Serwer nginx (wym. in gen ex) to serwer sieciowy podobny do Apache, ale znacznie łatwiejszy w konfiguracji i często wykorzystujący mniej pamięci systemowej. Nie ma miejsca na szczegółowe opisywanie tego programu, ale przynajmniej pokażę, jak go zainstalować na swojej maszynie i przekazać odpowiednie żądania do puli PHP-FPM.

Instalacja

Najprostszą metodą instalacji serwera nginx jest użycie menedżera pakietów systemu operacyjnego.

Ubuntu

W Ubuntu serwer nginx można zainstalować przy użyciu PPA, czyli gotowego archiwum serwisowanego przez społeczność użytkowników:

```
sudo add-apt-repository ppa:nginx/stable;  
sudo apt-get update;  
sudo apt-get install nginx;
```

CentOS

W systemie CentOS serwer nginx instaluje się przy użyciu repozytorium oprogramowania EPEL, które dodaliśmy wcześniej. W domyślnych repozytoriach oprogramowania tego systemu może nie być najnowszej wersji serwera nginx:

```
sudo yum install nginx;  
sudo systemctl enable nginx.service;  
sudo systemctl start nginx.service;
```

Host wirtualny

Teraz pokażę, jak zdefiniować **wirtualnego hosta** na serwerze nginx dla aplikacji PHP. Host wirtualny to grupa ustawień określających nazwę domeny, położenie plików oraz sposób przekazywania żądań HTTP do puli PHP-FPM aplikacji.

Najpierw musimy wybrać miejsce do przechowywania plików aplikacji. Musi to być katalog, do którego odczytu i zapisu ma uprawnienia niebędący *rootem* użytkownik *deploy*. W tym przykładzie wybrałem katalog `/home/deploy/apps/example.com/current`. Dodatkowo potrzebny jest katalog na dzienniki — ja wybrałem `/home/deploy/apps/logs`. Poniższe polecenia tworzą te katalogi i nadają im odpowiednie zezwolenia:

```
mkdir -p /home/deploy/apps/example.com/current/public;  
mkdir -p /home/deploy/apps/logs;  
chmod -R +rx /home/deploy;
```

Swoją aplikację PHP umieść w katalogu `/home/deploy/apps/example.com/current`. W konfiguracji hostów wirtualnych serwera nginx przyjęto, że aplikacja zawiera katalog `public/`, pełniący rolę katalogu głównego dokumentów wirtualnego hosta.

Każdy host wirtualny nginx ma własny plik konfiguracyjny. Jeśli używasz systemu Ubuntu, utwórz plik konfiguracyjny `/etc/nginx/sites-available/example.conf`. Użytkownicy systemu CentOS powinni natomiast utworzyć plik `/etc/nginx/conf.d/example.conf`. Otwórz plik `example.conf` w swoim ulubionym edytorze tekstu.

Ustawienia wirtualnego hosta nginx wpisuje się w bloku `server {}`. Poniżej znajduje się kompletna zawartość pliku konfiguracyjnego wirtualnego hosta:

```
server {
    listen 80;
    server_name example.com;
    index index.php;
    client_max_body_size 50M;
    error_log /home/deploy/apps/logs/example.error.log;
    access_log /home/deploy/apps/logs/example.access.log;
    root /home/deploy/apps/example.com/current/public;

    location / {
        try_files $uri $uri/ /index.php$is_args$args;
    }

    location ~ /\.php {
        try_files $uri =404;
        fastcgi_split_path_info ^(.+\.(php|\.))$;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param SCRIPT_NAME $fastcgi_script_name;
        fastcgi_index index.php;
        fastcgi_pass 127.0.0.1:9000;
    }
}
```

Skopiuj ten kod do pliku konfiguracyjnego `example.conf`. Nie zapomnij zmienić ustawienia `server_name` i ścieżek `error_log`, `access_log` i `root`. Oto zwięzłe objaśnienie każdego z zastosowanych ustawień:

`listen`

Numer portu, na którym serwer nginx nasłuchuje przychodzących żądań HTTP. W większości przypadków dla ruchu HTTP jest to port 80, a dla HTTPS — 443.

`server_name`

Nazwa domeny identyfikującej danego hosta wirtualnego. Należy wpisać własną domenę oraz ustawić ją na adres IP swojego serwera. Serwer nginx wysyła żądania HTTP do tego wirtualnego hosta, jeśli treść nagłówka `Host:` pasuje do wartości `server_name`.

`index`

Pliki serwowane domyślnie, jeśli żaden nie zostanie określony w identyfikatorze URI żądania HTTP.

`client_max_body_size`

Maksymalny rozmiar treści żądania HTTP akceptowany przez nginx dla tego hosta wirtualnego. Jeśli treść żądania będzie większa niż dozwolona wartość, nginx zwróci odpowiedź HTTP z rodziny `4xx`.

`error_log`

Ścieżka do pliku dziennika błędów hosta wirtualnego.

`access_log`

Ścieżka do dziennika dostępu hosta wirtualnego.

`root`

Główny katalog dokumentów.

Dodatkowo w przedstawionej konfiguracji znajdują się jeszcze dwa bloki `location`. Zawierają one informacje na temat sposobu obsługi żądań HTTP pasujących do adresów URL zbudowanych wg określonych wzorców. Pierwszy blok `location / {}` zawiera dyrektywę `try_files`, która szuka prawdziwych plików pasujących do URI z żądania. Jeśli katalog nie zostanie znaleziony, przepisuje URI z żądania HTTP na `/index.php` i dołącza łańcuch zapytania, jeśli jest dostępny. Przepisany adres URL lub każde żądanie kończące się przyrostkiem `.php` są obsługiwane przez blok `location ~ \.php {}`.

Blok `location ~ \.php {}` przekazuje żądania HTTP do puli PHP-FPM. Pamiętaj, że w jej konfiguracji został ustawiony port nasłuchu żądań o numerze 9000? Blok ten przekazuje właśnie żądania PHP do portu 9000, po czym pula PHP-FPM przejmuje kontrolę.



W bloku `location ~ \.php {}` znajduje się trochę więcej kodu. Uniemożliwia on przeprowadzanie ataków polegających na zdalnym wykonywaniu kodu (<http://bit.ly/remote-ex>).

W systemie Ubuntu konieczne jest utworzenie łącza symbolicznego między plikiem konfiguracyjnym hosta wirtualnego a katalogiem `/etc/nginx/sites-enabled/` za pomocą poniższego polecenia:

```
sudo ln -s /etc/nginx/sites-available/example.conf \
/etc/nginx/sites-enabled/example.conf;
```

Następnie należy ponownie uruchomić serwer nginx, korzystając z poniższego polecenia:

```
# Ubuntu
sudo service nginx restart

# CentOS
sudo systemctl restart nginx.service
```

Aplikacja działa i jest gotowa do użytku! Serwer nginx można skonfigurować na wiele sposobów. W tym rozdziale opisałem tylko najważniejsze ustawienia, ponieważ jest to książka o języku PHP, a nie o serwerze nginx. Więcej informacji znajdziesz w źródłach wymienionych poniżej:

- <http://nginx.org/>,
- <https://github.com/h5bp/server-configs-nginx>,
- <https://serversforhackers.com/editions/2014/03/25/nginx/>.

Automatyzacja wdrażania serwera

Wdrażanie serwera to czasochłonny i niezbyt przyjemny proces, zwłaszcza gdy trzeba ręcznie wdrożyć dużą liczbę serwerów. Na szczęście istnieją narzędzia pomagające w automatyzacji tych czynności. Oto niektóre z najpopularniejszych:

- Puppet (<http://puppetlabs.com/>),
- Chef (<https://www.getchef.com/chef/>),
- Ansible (<http://www.ansible.com/home>),
- SaltStack (<http://www.saltstack.com/>).

Każde narzędzie jest inne, ale wszystkie służą do tego samego celu — automatycznie wdrażają nowe serwery zgodnie z określoną specyfikacją. Jeśli więc zarządzasz wieloma serwerami, warto zainteresować się tymi programami, bo dzięki nim można oszczędzić mnóstwo czasu.

Zlecenie wdrażania serwerów

Istnieją też internetowe usługi wdrażania serwerów na życzenie klienta. Jedną z nich jest Forge (<https://forge.laravel.com/>) Taylora Otwell. Byłem beta testerem tej aplikacji i muszę powiedzieć, że jest naprawdę pomocna. W ofercie znajdują się wdrożenia serwerów u takich dostawców jak Linode, Digital Ocean i wielu innych popularnych firm oferujących serwery VPS.

Każdy serwer wdrażany przez Forge jest automatycznie zabezpieczony przy użyciu technik, które opisałem w tym rozdziale, oraz na każdym instalowane jest oprogramowanie nginx z PHP-FPM. Ponadto Forge ułatwia wdrażanie aplikacji PHP, instalację certyfikatów SSL, tworzenie zadań CRON oraz wykonywanie wielu innych nudnych lub trudnych zadań administracyjnych. Jeśli administracja systemami nie jest Twoją mocną stroną, to gorąco polecam Ci usługę Forge.

Dodatkowe źródła

Administracja systemami to dla mnie bardzo ciekawy temat. Nie chciałbym zajmować się tym na co dzień, ale lubię od czasu do czasu pogrzebać w wierszu poleceń. Moim zdaniem najlepszym źródłem wiedzy dla administratorów systemów jest książka *Servers for Hackers* (<https://book.serversforhackers.com/>) Chrisa Fida.

Co dalej

W tym rozdziale opisałem metody wdrażania serwerów dla aplikacji PHP. W następnym po-każę Ci, jak wycisnąć z serwera siódme poty dla zwiększenia wydajności Twojej aplikacji.

A

- adnotacje
 - argumentów, 198
 - typu zwrotnego, 198
 - własności, 198
- aktualizowanie systemu operacyjnego, 139
- aliasy, 26
- API mieszania haseł, 92
- ASCII, 110
- autoloader, 60
- autoloader PSR-4, 62
- automatyczne
 - ładowanie, 30
 - ładowanie zależności, 61
 - wczytywanie komponentów, 73
 - wczytywanie zależności, 53, 60
- automatyzacja
 - wdrażania, 159
 - wdrażania serwera, 149

B

- baza danych MySQL, 101
- bezpieczeństwo serwera, 139
- biblioteka SPL, 40
- błąd, error, 119, 123
- błędy w czasie pracy, 126
- bufor
 - kodów operacyjnych, 20, 153
 - kodu bajtowego, 43
 - ścieżek, 157
 - Zend OPcache, 45

- buforowanie, 153
- buforowanie wyników, 156

C

- Capistrano, 160
 - cofanie aplikacji, 164
 - haki, 163
 - host wirtualny, 163
 - instalacja, 161
 - konfiguracja, 161
 - uwierzytelnianie, 162
 - wdrażanie aplikacji, 164
- cecha Geocodable, 35
- cechy, traits, 33, 34
- CentOS 7, 207
- ciągle testowanie, 176
- Code Sniffer, 58
- cofanie aplikacji, 164
- Composer, 70, 75
- czas wykonywania, 155

D

- dane wyjściowe, 89
- data, 96
- definicja
 - cechy, 35
 - klasy, 32
- deklaracja, 25
- dezynfekcja danych wejściowych, 86
- domieszka, 33
- dostęp do katalogów, 212
- dostrajanie, 151

F

- FastCGI, 191
- filtry strumieni, 115
- format
 - CacheGrind, 181
 - UTF-8, 111
- FTP, 20
- funkcja
 - error_log(), 128
 - htmlentities(), 86
 - stream_filter_append(), 115, 116
- funkcje anonimowe, 40
- funkcjonalność, 23

G

- generator, 37
- Git, 20, 159
- GitHub, 24, 82
- godzina, 96

H

- haki Capistrano, 163
- hasła, 90
- HHVM, 185
 - instalacja, 188
 - komunikacja z serwerem, 191
 - konfiguracja, 189
 - monitorowanie, 190
 - rozszerzenia, 189
- hierarchia plików, 77
- Homebrew, 211

host wirtualny, 146, 163
hosting, 133
HTML, 86

I

implementacja komponentu, 80
importowanie, 26
importowanie wielu elementów,
28
informacje o profilu
użytkownika, 88
instalacja
Capistrano, 161
Composer, 70
HHVM, 188
komponentów, 71
PHP, 209
Linux, 205
OS X, 208
Windows, 219
PHP-FPM, 142
PHPUnit, 170
rozszerzeń PHP, 214
serwera nginx, 146
Xdebug, 170
instalacja XHProf, 181
instrukcja
return, 38
use func, 28
instrukcje przygotowane, 103
interfejs, 30, 52
interfejs rejestratora
dziennikowego, 58
interpreter, 21
interpreter HHVM, 187

J

język Hack, 21, 192
konwersja PHP, 192
składnia, 197
sprawdzanie typów, 196
struktury danych, 199
tryby, 196
typ, 193
typowanie dynamiczne, 195
typowanie statyczne, 194

języki interpretowane, 43
JIT, just in time, 21

K

catalog
Packagist, 82, 83
src/, 215
klasa
\$widget, 198
DateInterval, 97
DatePeriod, 99
DateTime, 96
DateTimeZone, 98
Exception, 29
PHPUnit_Framework_
TestCase, 173
Response, 24
StreamDocument, 32
Whovian, 171, 173, 174
Widget, 198

kodowanie

ASCII, 110
UTF-8, 111

kompilacja

PHP, 219
ze źródła, 214

kompilator

HPHPc, 187
JIT, 21

kompilowana wersja PHP, 214

komponent, 63, 64

nesbot/carbon, 100
Whoops, 127

konferencje, 201

konfiguracja

Capistrano, 161
HHVM, 189
PHP, 215
PHPUnit, 170

puli, 143

serwera, 46

aktualizacja

oprogramowania, 139

bezpieczeństwo, 139

pierwsze logowanie, 138

uwierzytelnianie, 140

wyłączanie haseł, 141

Travis CI, 176
Xdebug, 180
XHGUI, 182
Zend OPcache, 44

kontekst strumienia, 115

kontrola

typów, 21
wersji, 82, 159

kontroler frontu, 47

kontrolowanie danych
wyściowych, 89

konwersja PHP na Hack, 192

L

Laravel Homestead, 225

logowanie, 94, 138

lokalne

grupy użytkowników, 201
środowiska deweloperskie,
221

Ł

łańcuchy wielobajtowe, 110

M

maksymalny czas wykonywania,
155

MAMP, 208

mechanizm

HHVM, 21
obsługi PHP, 21
Zend Engine, 21

menedżer

pakietów, 205
zależności, 20

mentoring, 202

metoda

__construct(), 173, 198
addRoute(), 43
alert(), 52
bindTo(), 42
critical(), 52
debug(), 52
dispatch(), 43
emergency(), 52
error(), 52

- info(), 52
- getContent(), 31
- getId(), 31
- getWidget(), 198
- notice(), 52
- respondTo(), 173
- say(), 174
- warning(), 52

mieszanie haseł, 91, 96

monitorowanie HHVM, 190

N

najlepsze praktyki, 85

narzędzia wiersza poleceń

- XCode, 211

narzędzie

- Chef, 224
- PuPHPet, 226
- Puppet, 224
- Supervisord, 190
- Vaprobash, 226

nazwa

- dostawcy, 76
- komponentu, 71
- pakietu, 76
- źródła danych, 101

O

obsługa

- błędów, 125
- sesji, 156
- wyjątków, 123

opakowanie strumienia, 112–114

P

PaaS, platform as a service, 135

pakiet testów, 168

pamięć, 152

pamięć podręczna realpath, 157

para kluczy, 140

parametr związany, 104

PHP

- instalacja, 205
- w Facebooku, 186

PHP-FIG, 51, 53

PHP-FPM, 142

instalacja, 142

konfiguracja globalna, 143

konfiguracja puli, 143

PHPUnit, 168

- instalacja, 170
- konfiguracja, 170

pierwsze logowanie, 138

plan hostingowy, 135

plik

- composer.json, 78
- composer.lock, 73
- deploy.rb, 162
- php.ini, 151, 219
- production.rb, 162
- README, 80
- scan.php, 74
- Vagrantfile, 223

pliki binarne, 220

podłączanie repozytoriów

- wzorów, 213

podstawowy styl kodowania, 54

pokrycie kodu, 175

polecenie

- ./configure, 217
- phpunit, 169
- vagrant, 222

połączenia z bazą danych, 101

ponawianie mieszania hasła, 95

poprawność danych, 88

portal

- GitHub, 82
- Packagist, 68

procedury obsługi

- błędów, 125
- wyjątków, 123

produkcja, 128

profiler, 179

- Blackfire, 183
- New Relic, 183
- Xdebug, 180
- XHGUI, 182
- XHPProf, 181

profilowanie, 179

program, *Patrz także* narzędzie

- Ansible, 20
- Capistrano, 160
- Chef, 20
- Git, 20
- PHP-FPM, 142

Pupper, 20

Travis CI, 176

Vagrant, 20

programowanie

- behawioralne, 167
- przez testy, 167

prywatne repozytoria, 75

przechwytywanie wyjątków, 121

przestrzeń nazw, 23, 77

- globalna, 29
- ModernPHP, 26

przypadek testowy, 168

przypadek testowy WhovianTest, 172

PSR, PHP standards

- recommendation, 53

PSR-1, 54

PSR-2, 55

PSR-3, 58

PSR-4, 60

puddło Vagrant, 223

R

rejestracja użytkowników, 92

repozytorium EPEL, 207

restrykcyjny styl kodowania, 55

root, 141

rozszerzenie

- PDO, 101
- PHP Xdebug, 170
- Zend OPcache, 43, 44

S

serwer

- dedykowany, 134
- nginx

 - host wirtualny, 146
 - instalacja, 146

VPS, 138

wbudowany HTTP, 46

- konfiguracja, 46
- skrypty trasujące, 47
- uruchamianie, 46
- wady, 47
- wykrywanie, 47

wirtualny, 134

współdzielony, 133

- serwis GitHub, 24, 82
- sesja, 156
- składnia języka Hack, 197
- skrypty trasujące, 47
- słowo kluczowe
 - use, 28
 - yield, 37
- specyfikacja, 20
- SPL, Standard PHP Library, 40
- społeczność, 201
- sprawdzanie
 - poprawności danych, 88
 - typów, 196
- SQL, 87
- SSH, 140
- standard, 51
- standard PSR, 20
- strefa czasowa, 96
- struktura katalogu, 72, 169
- struktury danych języka Hack, 199
- strumień, 112
 - cel, 113
 - file://, 113
 - filtry, 115
 - kontekst, 115
 - opakowanie, 113
 - php://, 114
 - schemat, 113
- styl, 53
- synchronizacja folderów, 224
- system Git, 159
- systemy
 - kontroli wersji, 20, 159
 - szkieletowe, 51, 65
- zsyfrowanie haseł, 91

Ś

- ścieżka, 157

T

- testowanie, 165
- testy
 - funkcjonalne, 166
 - jednostkowe, 167
- transakcje, 107
- tworzenie
 - aliasów klas, 27
 - cechy, 35

- filtrów strumieni, 117
- generatora, 37
- interfejsów, 27
- komponentów PHP, 76
- opakowań strumieni, 114
- pliku php.ini, 219
- przestrzeni nazw, 27
- zamknąć, 40
- typ, 193
- typowanie
 - dynamiczne, 195
 - statyczne, 194
- typy profilerów, 179

U

- uruchamianie
 - serwera, 46
 - testów, 175
- usługa typu PaaS, 135, 137
- ustawianie domyślnej strefy czasowej, 96
- utajnianie danych użytkownika, 102
- UTF-8, 111
- uwierzytelnianie, 140, 162
- użytkownik root, 141
- używanie
 - cech, 36
 - Composera, 71
 - generatorów, 38
 - komponentów PHP, 63, 69, 83
 - narzędzi, 66
 - pary kluczy, 140
 - profilera, 179
 - rejestratora PSR-3, 59
 - Zend OPcache, 45

V

- Vagrant, 222
- VirtualBox, 221
- VPS, virtual private server, 134

W

- WAMP, 220
- wbudowany serwer HTTP, 46

- wczytywanie zależności, 53
- wdrażanie
 - aplikacji, 159, 164
 - maszyny wirtualnej, 224
 - serwera, 137, 149
- weryfikacja hasła, 95
- wiązanie stanu, 41
- wirtualny host, 146
- włączanie rozszerzenia Zend OPcache, 44
- współczynnik pracy, 91
- wybór planu hostingowego, 135
- wyjątek, exception, 119, 120
- wyjątki w czasie pracy, 126
- wykonywanie testów, 168, 177
- wykrywanie serwera
 - wbudowanego, 47
- wyłączanie haseł, 141
- wyniki zapytań, 105
- wysyłanie plików, 154
- wyszukiwanie komponentów, 67

X

- Xdebug, 180
 - konfiguracja, 180
 - uruchamianie, 181
- XHGUI, 182
 - konfiguracja, 182
 - uruchamianie, 183
- XHProf, 181
 - instalacja, 181

Z

- zamknięcia, 40
- zamknięcie łańcucha, 42
- zapytania SQL, 87, 105
- zarządzanie pulą powiązanych procesów, 142
- Zend Engine, 187
- Zend OPcache, 43, 153
- Zend Server, 220
- zgłaszanie wyjątków, 120
- zlecanie wdrażania serwerów, 149
- zmienna środowiskowa PATH, 213
- znak wielobajtowy, 110
- zwracanie danych, 111

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**

Przekonaj się, jakie nowości kryje PHP!

PHP to dziś jeden z najbardziej niedocenianych języków programowania. W sieci można znaleźć mnóstwo artykułów wykluczających jego zastosowanie w zaawansowanych projektach. A przecież w obecnej wersji język ten jest pełnoprawnym kandydatem do zadań specjalnych! Przekonaj się, jak używać go we właściwy sposób!

Ta książka powinna obalić wszystkie mity o niedojrzałości języka PHP. Sięgnij po nią i przekonaj się, jakie fantastyczne możliwości daje Ci jego najnowsza wersja. Wśród nich znajdziesz domknięcia, wbudowany serwer HTTP, przestrzenie nazw oraz generatory. Ponadto poznasz standardy kodowania PSR oraz przekonasz się, jak korzystać z gotowych komponentów. Na kolejnych stronach znajdziesz informacje na temat najlepszych praktyk i dowiesz się, jak skonfigurować środowisko uruchomieniowe tak, żeby osiągnąć jego najwyższą wydajność. Ta książka jest obowiązkową lekturą dla wszystkich programistów języka PHP. Język PHP właśnie się odradza – lepszy i silniejszy!

Josh Lockhart – twórca popularnego szkieletu Slim Framework. Pomysłodawca inicjatywy PHP The Right Way, promującej najlepsze praktyki kodowania w języku PHP. Programista w firmie New Media Campaigns, z siedzibą w Karolinie Północnej.

Dzięki tej książce:

- poznasz nowości w języku PHP
- wykorzystasz wbudowany serwer HTTP do szybkiego uruchamiania projektów
- zaznajomisz się z najlepszymi praktykami tworzenia kodu
- dostosujesz środowisko uruchomieniowe tak, żeby osiągnąć najwyższą wydajność
- przetestujesz swoje rozwiązanie
- odkryjesz język PHP na nowo

Helion 

36528 numer katalogowy

księgarnia internetowa

<http://helion.pl>

zamówienia telefoniczne



0 801 339900



0 601 339900

Sprawdź najnowsze promocje:

• <http://helion.pl/promocje>

Książki najchętniej czytane:

• <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

• <http://helion.pl/novosci>

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel.: 32 230 98 63

e-mail: helion@helion.pl

<http://helion.pl>

sięgnij po WIĘCEJ



KOD KORZYŚCI

ISBN 978-83-283-1402-3



9 788328 314023