PHP

W PRZYKŁADACH

Jerzy Wawro

edukacja plus

VERBU 2018

PHP w przykładach

PHP

PHP w przykładach

Na podstawie publikacji open source przygotował Jerzy Wawro.

Opis środowiska programowania w dużej mierze pochodzi z publikacji "Using Zend Framework 3" udostępnianej na licencji Creative Common. Na analogicznej licencji udostępniane jest niniejsze opracowanie.

Uwaga!

Niniejsze opracowania jest w trakcie realizacji. Jest gotowe w około 90%.

Środowisko programowania w PHP

Do programowania w PHP zazwyczaj potrzebujesz:

- serwera WWW,
- silnika PHP (+ rozszerzenia)
- bazę danych.

Najczęściej będą to serwer Apache i baza danych Mysql, zainstalowane na serwere Linux (LAMP = Linux+Apache+Musql+PHP).

Wśród roszerzeń PHP warto zainstalować XDebug - do śledzenia i usuwania błędów (debugging).

Dodatkowo przyda się środowisko IDE - na przykład NetBeans lub nowoczesny edytor taki jak Atom lub Visual Studio Code.

Instalowanie Apache, PHP i MySQL w systemie Linux

Zaleca się używanie popularnej i dobrze wspieranej dystrybucji Linuksa, 32-bitowej (x86) lub 64-bitowej (amd64).

Istnieją dwie duże rodziny dystrybucji Linux: Debian i Red Hat. Debian to darmowy projekt o otwartym kodzie źródłowym. Ma on kilka odmian (dystrybucji), z których najpopularniejszą jest Linux Ubuntu. Red Hat to komercyjny rozproszony system operacyjny, który ma darmowe odmiany o nazwach Linux CentOS i Linux Fedora.

Red Hat Linux jest rozwijany przez Red Hat Inc. Red Hat Linux (lub jego darmowa modyfikacja CentOS) jest znany jako korporacyjny system operacyjny. Jego główną zaletą jest stabilność (niski wskaźnik awarii systemu). Ta stabilność jest jednak osiągana dzięki starannemu doborowi oprogramowania dostępnego "po wyjęciu z pudełka". Kiedy instalujesz taki system operacyjny w celu rozwoju aplikacji PHP, ta strategia może stać się problemem, ponieważ masz dostęp do starej (choć stabilnej) wersji PHP i innego oprogramowania. Nie znajdziesz nowego oprogramowania w repozytorium, więc jeśli chcesz go zainstalować, musisz pobrać go gdzieś, przeczytać instrukcję i ewentualnie (jeśli nie masz szczęścia) skompilować go samemu.

Z powyższych względów do programowania PHP lepiej wybrać Linux Ubuntu. Ubuntu jest rozwijany przez Canonical Ltd. Linux Ubuntu ma dwie edycje: Desktop Edition i Server Edition. Ubuntu Desktop to dystrybucja zawierająca środowisko graficzne, podczas gdy wersja Ubuntu Server ma tylko terminal konsolowy. W celu rozwoju aplikacji PHP zaleca się korzystanie z edycji Desktop.

Canonical zwykle wydaje nową wersję Linux Ubuntu co 6 miesięcy, w kwietniu i październiku, oraz wersję `LTS` co 2 lata. Na przykład w chwili pisania tego tekstu najnowszą wersją jest Ubuntu 18.04 Xenial Xerus LTS (wydany w kwietniu 2018 r.).

Wersje non-LTS mają krótki okres wsparcia (około 9 miesięcy), ale mają najnowsze wersje oprogramowania PHP. Z drugiej strony, wydania LTS mają dłuższy okres wsparcia(5 lat), ale nieco przestarzałe oprogramowanie PHP na starcie.

Ubuntu Desktop ma najnowszą wersję PHP i inne oprogramowanie dostępne z repozytorium. Wadą korzystania z takiej wersji jest to, że będziesz musiał uaktualnić ją do następnej wersji co 9 miesięcy (wraz z upłynięciem okresu wsparcia). Jeśli nie podoba Ci się perspektywa aktualizacji co 9 miesięcy, wybierz najnowszą wersję LTS. Ubuntu wspiera najnowszą obecnie **wersję PHP 7.x**

Instalowanie Apache i PHP

W nowoczesnych dystrybucjach Linuksa możesz łatwo pobrać i zainstalować oprogramowanie ze scentralizowanego *repozytorium* . Repozytorium zawiera tak zwane *pakiety* . Pakiet ma nazwę (na przykład `php` , `apache2`) i wersję.

Można zainstalować pakiet za pomocą jednego polecenia. Jednak to polecenie (jak i nazwa pakietu) mogą się różnić w zależności od używanej dystrybucji Linuksa. Na przykład, aby pobrać i zainstalować pakiety w dystrybucji Linuksa opartej na Debianie (np. Ubuntu Linux), należy użyć programu **apt** (apt-get). W dystrybucjach Red Hat (np. Fedora lub CentOS) używasz **yum** (menedżer pakietów RPM). Poniżej podano szczegółowe instrukcje instalacji dla tych systemów operacyjnych.

Debian lub Linux Ubuntu

Przede wszystkim zaleca się aktualizację systemu poprzez zainstalowanie najnowszych dostępnych aktualizacji. Aby to zrobić, w powłoce poleceń uruchom następujące polecenia:

```
sudo apt-get update

sudo apt-get upgrade
```

Powyższe polecenia uruchamiają narzędzie APT i instalują najnowsze aktualizacje pakietów systemowych. Polecenie `sudo` (oznacza `Super User DO`) pozwala uruchomić polecenia jako administrator systemu (root). W naszym przypadku uruchamiamyb tak `apt-get` . Zwykle używasz `sudo` , gdy chcesz podnieść swoje uprawnienia, aby zainstalować pakiet lub edytować jakiś plik konfiguracyjny.

> Komenda `sudo` może zażądać hasła. Po wyświetleniu monitu wprowadź hasło pod którym logujesz się do systemu i naciśnij Enter.

Następnie, z powłoki poleceń, uruchom następujące polecenia:

```
sudo apt-get install apache2

sudo apt-get install php

sudo apt-get install libapache2-mod-php
```

Powyższe polecenia pobierają z repozytorium i instalują najnowsze dostępne wersje serwera Apache HTTP Server, silnika PHP i modułu rozszerzającego PHP dla Apache.

> Powyższe polecenia mogą wymagać potwierdzenia podczas instalowania pakietu. Zaleca się odpowiedź Tak (naciśnij `y`, a następnie Enter).

Fedora, CentOS lub Red Hat Linux

Rozpoczynamy od aktualizacji systemu poprzez zainstalowanie najnowszych dostępnych aktualizacji. Aby to zrobić, w powłoce poleceń uruchom następujące polecenie:

```
sudo yum update
```

Powyższe polecenie uruchamia narzędzie YUM i instaluje najnowsze aktualizacje pakietu systemowego.

Następnie, z powłoki poleceń, uruchom następujące polecenia:

```
sudo yum install httpd

sudo yum install php
```

Powyższe polecenia są pobierane z repozytorium i instalowane są najnowsze dostępne wersje serwera Apache HTTP Server i PHP.

Następnie uruchom następujące polecenia, aby dodać serwer Apache HTTP Server do automatycznego uruchamiania systemu i uruchomić go:

```
sudo chkconfig --level 235 httpd on

sudo service httpd start
```

Sprawdzanie instalacji serwera www

Po skonfigurowaniu serwera WWW Apache HTTP sprawdź, czy jest on poprawnie zainstalowany i czy serwer widzi silnik PHP. Aby to zrobić, utwórz plik *phpinfo.php* w katalogu głównym dokumentu Apache.

Główny katalog to katalog, w którym można(domyślnie) przechowywać pliki sieciowe. Zazwyczaj katalog główny serwera Apache to /var/www/html/.

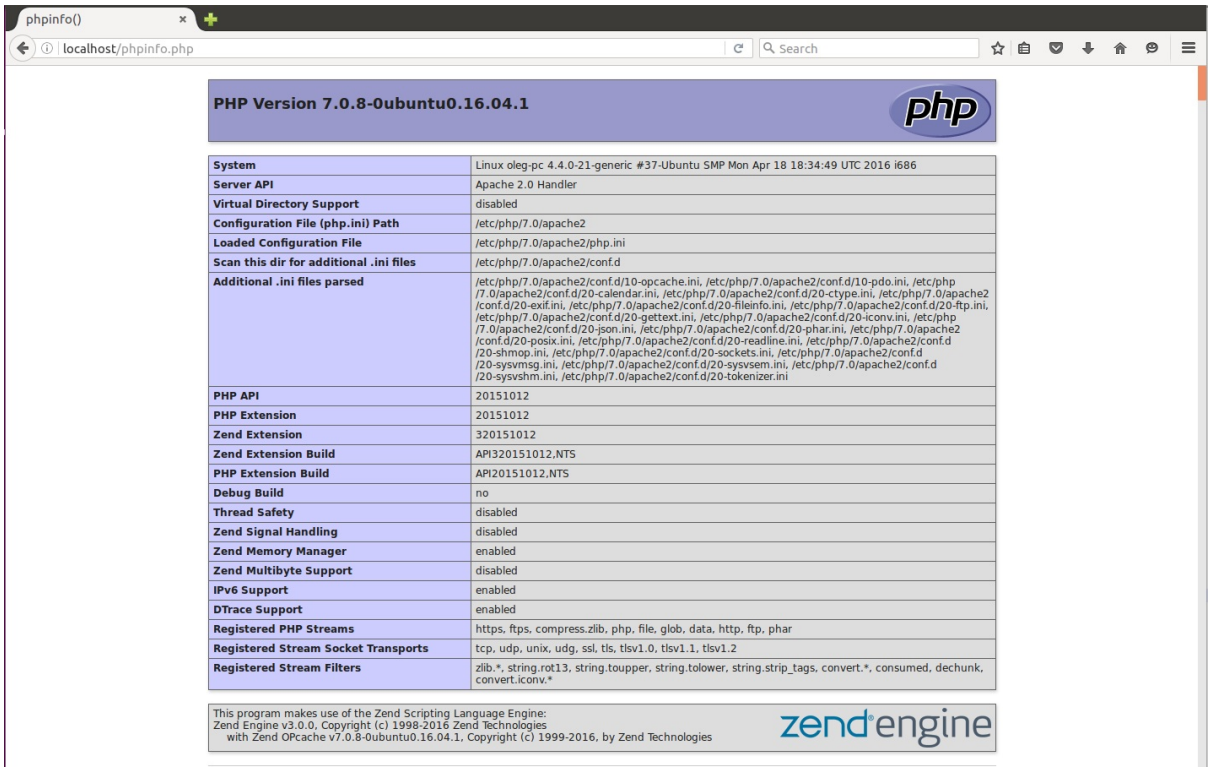
> Dla ułatwienia nawigacji po strukturze katalogów i edytowanie plików, warto zaleca zainstalować Midnight Commandera (wygodny menedżer plików i edytor tekstu).
> Aby zainstalować Midnight Commander w systemie Debian lub Linux Ubuntu, wpisz:

> `sudo apt-get install mc`
>
> Poniższe polecenie instaluje Midnight Commandera w Fedorze, CentOS lub Red Hat Linux:
>
> `sudo yum install mc`
>
> Po instalacji możesz uruchomić menedżera plików za pomocą komendy `mc` i edytować plik a
> plik tekstowy za pomocą następującej komendy (uruchamianej też klawiszem F4):
>
> `mcedit/path/to/file`
>
> Jeśli potrzebujesz uprawnień administracyjnych, aby edytować plik, dodaj komendę `sudo` do powyższego polecenia.

W pliku *phpinfo.php* wprowadź metodę PHP `phpinfo()` w następujący sposób:

```
<?php
  phpinfo();
?>
```

Otwórz plik w swojej przeglądarce. Powinna wyświetlić się standardowa strona z informacjami o PHP (na przykład rysunek A.1).



Tworzenie wirtualnego hosta Apache

Zend Framework 3 wymaga utworzenia *wirtualnego hosta* dla twojej strony internetowej. Termin hosta wirtualnego oznacza, że możesz uruchomić kilka stron internetowych na tym samym komputerze.

Witryny wirtualne są różnicowane według nazwy domeny (np. *site.mydomain.com* i *site2.mydomain.com*). Każdy wirtualny host ma swój własny katalog główny dokumentu, umożliwiający umieszczanie plików internetowych w dowolnym miejscu w systemie (nie tylko w katalogu */var/www/html*).

W systemie Debian lub Ubuntu Linux

Masz przykład domyślnego hosta wirtualnego w */etc/apache2/sites-available/000-default.conf* (patrz poniżej).

```
<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header t
    # match this virtual host. For the default virtual host(this file) th
    # value is not decisive as it is used as a last resort host regardles
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example th
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Wszystko, co musisz zrobić, to po prostu edytować ten plik wirtualnego hosta w razie potrzeby i ponownie uruchomić Apache, aby zastosować zmiany.

Możesz również skopiować ten plik i utworzyć inny wirtualny host, gdy potrzebujesz kilku stron internetowych do działania na tym samym komputerze. Na przykład, aby utworzyć kolejny wirtualny plik hosta o nazwie *001-vhost2.conf* , wpisz następujące polecenie z powłoki poleceń:

```
cd/etc/apache2/sites-available
sudo cp 000-default.conf 001-vhost2.conf
sudo a2ensite 001-vhost2.conf
```

> Nazwa hosta wirtualnego zaczyna się od prefiksu(np. *000* , *010* itd.), Który definiuje priorytet. Serwer WWW Apache próbuje kierować żądanie HTTP do każdego wirtualnego hosta po kolei (najpierw do *000-default* , następnie do *001-vhost2*), a jeśli jakiś wirtualny host nie może obsłużyć żądania, następny jest wypróbowany i tak dalej.

W Fedorze, CentOS lub Red Hat Linux

Istnieje przykład wirtualnego hosta w pliku */etc/httpd/conf/httpd.conf* . Przewiń yen plik do samego dołu - do sekcji o nazwie `Virtual Hosts` . Możesz edytować tę sekcję w razie potrzeby i ponownie uruchomić Apache, aby zastosować zmiany.

Instalowanie serwera bazy danych MySQL

MySQL to bezpłatny system zarządzania relacyjnymi bazami danych (projekt zarządzany przez Oracle). MySQL to najpopularniejszy system baz danych używany w PHP.

Debian lub Linux Ubuntu

Aby zainstalować bazę danych MySQL, wpisz:

```
sudo apt-get install mysql-server
sudo apt-get install mysql-client
sudo apt-get install php-mysql
```

Powyższe polecenia instalują odpowiednio komponent serwera MySQL, składnik klienta MySQL i moduł rozszerzający MySQL dla PHP.

Konfigurowanie serwera bazy danych MySQL

Podczas instalacji serwera MySQL zostaje utworzony *użytkownik* root. Domyślnie użytkownik root nie ma hasła, więc musisz ustawić je ręcznie. Będziesz potrzebował tego hasła do tworzenia innych użytkowników baz danych MySQL.

Aby połączyć się z serwerem MySQL, wprowadź następujące polecenie:

```
mysql -u root
```

Pojawi się wiersz polecenia MySQL. W wierszu polecenia wprowadź następujące polecenie i naciśnij klawisz Enter (w poleceniu poniżej, zastąp `<your_password>` swoim hasłem):

```
SET PASSWORD FOR 'root'@'localhost' = '<your_password>';
```

Jeśli polecenie zostanie wykonane pomyślnie, wyświetlony zostanie następujący komunikat:

```
Query OK, 0 rows affected (0.16 sec)
```

Teraz musimy utworzyć nową bazę danych, która będzie przechowywać tabele. Aby to zrobić, wpisz:

```
CREATE DATABASE test_db;
```

Powyższe polecenie tworzy pusty schemat, który będziemy później wypełniać.

Następnie chcemy stworzyć kolejnego użytkownika bazy danych o nazwie `test_user`, który będzie używany do łączenia się z bazą danych. Aby utworzyć użytkownika, wpisz następujące polecenie (w poleceniu poniżej zamień `<your_password>` na hasło użytkownika):

```
GRANT ALL PRIVILEGES ON test_db.* TO 'test_user'@'localhost'
 IDENTIFIED BY '<your_password>';
```

Powyższe polecenie tworzy użytkownika o nazwie `test_user` i przyznaje użytkownikowi wszystkie uprawnienia do bazy danych `test_db`.

Na koniec wpisz `quit`, aby opuścić monit MySQL.

Instalowanie Apache, PHP i MySQL w systemie Windows

Zdecydowanie zaleca się używanie Linuksa do rozwoju programów w PHP. Większość systemów serwerowych ma zainstalowany system Linux. Jeśli używasz Windowsa do codziennych zadań, nadal możesz zainstalować Linuksa na maszynie wirtualnej (na przykład na VirtualBox) i uruchomić Apache, PHP i MySQL na tym wirtualnym komputerze. Jeśli w tym samym czasie chciałbyś używać NetBeans w Windows, możesz to zrobić - po prostu skonfiguruj katalog współdzielony (na przykład skonfiguruj serwer Samby na maszynie wirtualnej).

Jeśli zdecydujesz się zainstalować Apache, PHP i MySQL w Windows (co nie jest zalecane), pamiętaj jednak, że konfiguracja w Windows może być trudniejsza niż w Linuksie.

Istnieje dwie najpopularniejsze dystrybucje Apache + MySQL + PHP:

- WampServer
- XAMPP

Wybierz jedną i zainstaluj ją na swoim komputerze z serwerem Windows.

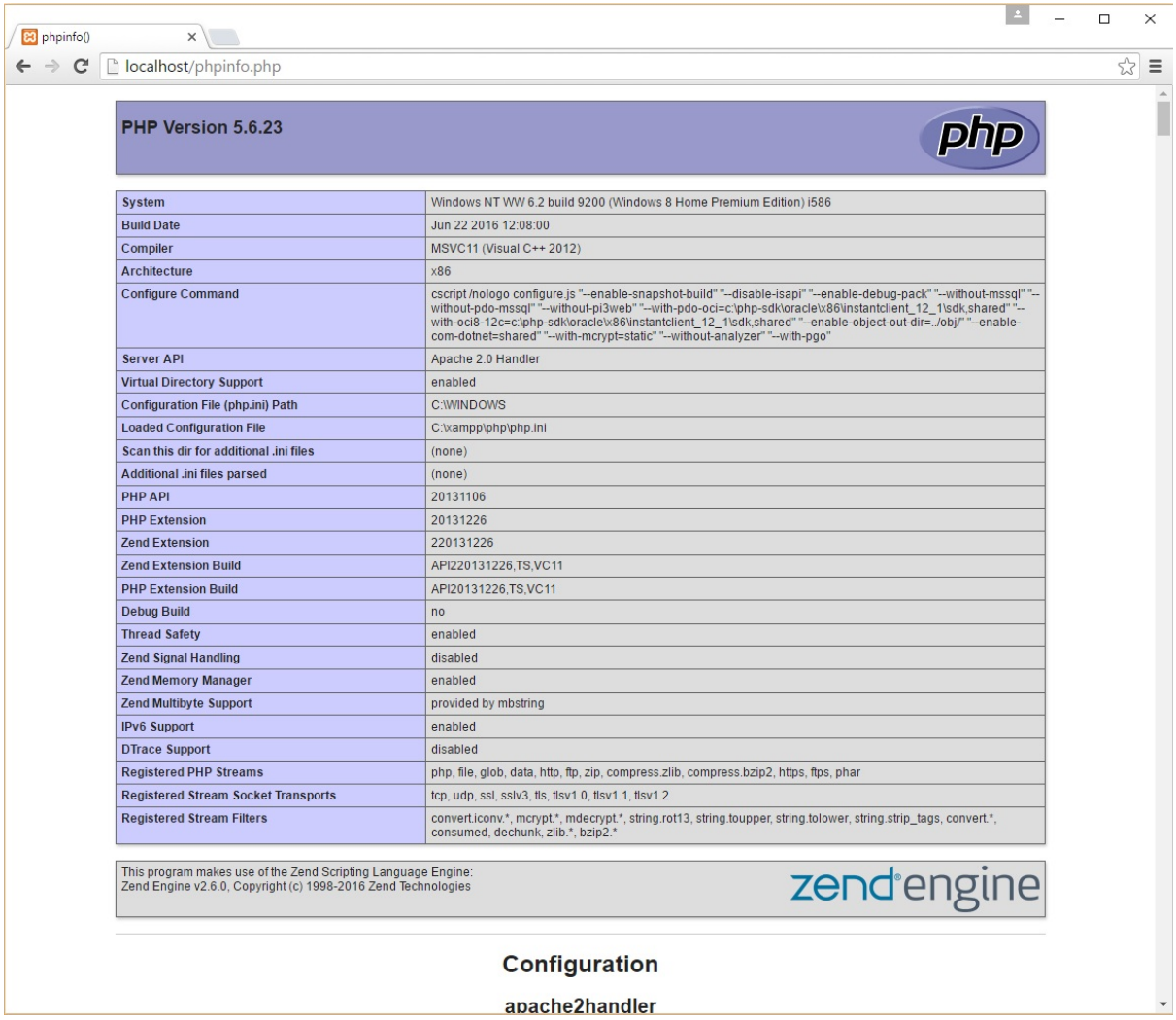
Sprawdzanie instalacji serwera sieciowego

Po skonfigurowaniu serwera sieciowego sprawdź, czy jest on poprawnie zainstalowany i czy serwer Apache rozpoznaje silnik PHP. Aby sprawdzić, czy Apache i PHP są poprawnie zainstalowane, utwórz plik *phpinfo.php* w katalogu głównym dokumentu Apache.

W pliku *phpinfo.php* wprowadź metodę PHP `phpinfo()` w następujący sposób:

```
<?php
    phpinfo();
?>
```

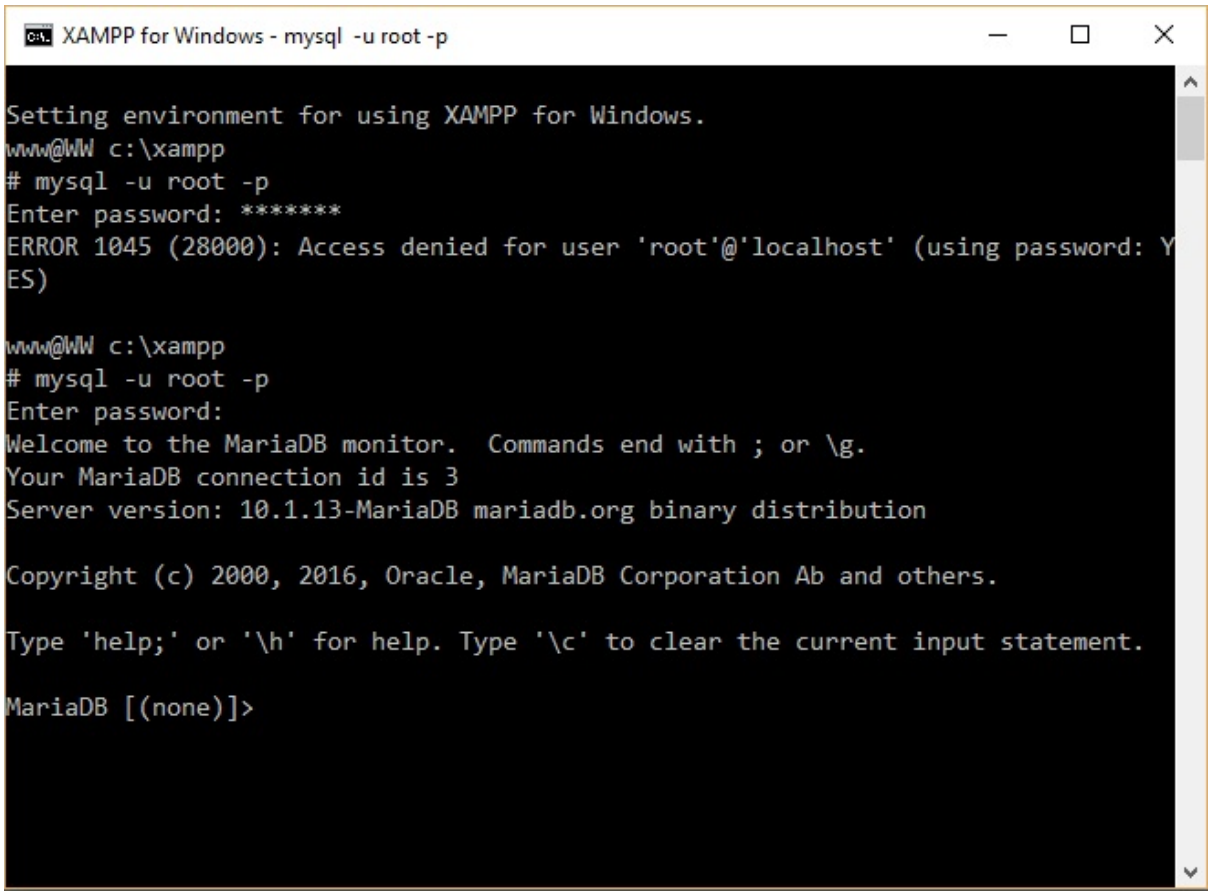
Otwórz plik w przeglądarce. Powinna wyświetlić się standardowa strona z informacjami o PHP(rysunek A.3).



Konfigurowanie serwera bazy danych MySQL w systemie Windows

Teraz chcemy stworzyć schemat bazy danych i użytkownika bazy danych. Będziemy używać klienta wiersza poleceń MySQL. Zapoznaj się z dokumentacją WAMP lub XAMPP, aby dowiedzieć się, jak to zrobić.

Konsola klienta linii poleceń MySQL wygląda następująco (patrz rysunek A.5):



Teraz musimy utworzyć nową bazę danych, która będzie przechowywać tabele. Aby to zrobić, wpisz następujące polecenie w oknie klienta MySQL:

```
CREATE DATABASE test_db;
```

Powyższe polecenie tworzy pustą bazę danych, którą zapełnimy później. Jeśli polecenie zostanie wykonane pomyślnie, wyświetlony zostanie następujący komunikat:

```
Query OK, 0 rows affected (0.16 sec)
```

Następnie chcemy stworzyć kolejnego użytkownika bazy danych o nazwie `test_user`, który będzie używany przez stronę internetową do łączenia się z bazą danych. Aby utworzyć użytkownika, wpisz następujące polecenie (w poleceniu poniżej wymień `<your_password>` na odpowiednie hasło):

```
GRANT ALL PRIVILEGES ON test_db.* TO 'test_user'@'localhost'
 IDENTIFIED BY '<your_password>';
```

Powyższe polecenie tworzy użytkownika o nazwie `test_user` i nadaje użytkownikowi wszystkie uprawnienia do schematu bazy danych `test_db`.

Konfiguracja PHP

Aby skonfigurować PHP dla twojego środowiska programistycznego, musisz edytować plik konfiguracyjny PHP (*php.ini*) i dostosować niektóre parametry.

> W różnych dystrybucjach systemu Linux plik konfiguracyjny PHP może znajdować się w różnych ścieżkach. Aby edytować plik konfiguracyjny PHP w systemie Debian lub Linux Ubuntu, wpisz:
>
> `sudo mcedit/etc/php/7.0/apache2/php.ini`
>
> Wpisz następujące polecenie, aby edytować plik *php.ini* w Fedorze, CentOS lub Red Hat Linux:
>
> `sudo mcedit/etc/php.ini`

W środowisku programistycznym zaleca się ustawienie następujących parametrów obsługi błędów i rejestrowania, jak poniżej. Zmusi to PHP do wyświetlania błędów na stronach PHP na ekranie.

```
error_reporting = E_ALL
display_errors = On
display_startup_errors = On
```

> Aby wygodnie wyszukiwać w pliku, naciśnij `F7` w oknie edytora Midnight Commandera i
> wprowadź szukany ciąg(nazwę szukanego parametru).

Ustaw swoje ustawienia strefy czasowej (zastąp symbol `<your_time_zone>` twoją strefą czasową, na przykład `UTC` lub `Poland/Warsaw`):

`date.timezone = <your_time_zone>`

Ustaw parametry `max_execution_time` , `upload_max_filesize` oraz `post_max_size` , aby umożliwić przesyłanie dużych plików przez POST. Na przykład ustawienie `upload_max_filesize` na `128M` pozwala przesłać pliki o rozmiarze do 128 megabajtów. Ustawienie `max_execution_time` na zero pozwala na wykonanie skryptu PHP na czas nieokreślony.

```
max_execution_time = 0
post_max_size = 128M
upload_max_filesize = 128M
```

Gdy wszystko będzie gotowe, zapisz zmiany naciskając klawisz *F2* , a następnie naciśnij *F10* , aby wyjść z edytora Midnight Commander.

Restartowanie serwera WWW Apache

Po edycji plików konfiguracyjnych, zazwyczaj musisz ponownie uruchomić serwer Apache HTTP Server, aby zastosować zmiany. Robisz to za pomocą następującego polecenia (w systemie Debian lub Linux Ubuntu):

```
sudo service apache2 restart
```

lub następujące (w Fedorze, CentOS lub Red Hat):

```
sudo service httpd restart
```

Włączanie modułu mod_rewrite Apache

Wiele programów (na przykład Zend Framework) wymaga włączenia modułu Apache *mod_rewrite* . Moduł *mod_rewrite* służy do przepisywania żądanych adresów URL na podstawie pewnych reguł, przekierowując użytkowników serwisu na inny adres URL.

W systemie Debian lub Ubuntu Linux

Aby włączyć moduł `mod_rewrite` Apache'a, wpisz następujące polecenie:

```
a2enmod mod_rewrite
```

Na koniec zrestartuj serwer WWW Apache, aby zastosować zmiany.

W Fedorze, CentOS lub Red Hat Linux

W tych dystrybucjach Linux `mod_rewrite` jest domyślnie włączony, więc nie musisz nic robić.

Środowisko deweloperskie

Instalowanie rozszerzenia PHP XDebug

Aby móc debugować witryny, zaleca się zainstalowanie rozszerzenia *XDebug* . Rozszerzenie XDebug pozwala zajrzeć do działającego programu, zobaczyć zmienne przekazywane przez klienta, chodzić po stosie wywołań i profilować kod PHP. XDebug zapewnia również możliwości analizy zasięgu kodu, które są przydatne podczas pisania testów jednostkowych dla twojego kodu.

W systemie Debian lub Ubuntu Linux

Aby zainstalować XDebug, wystarczy wpisać następujące polecenie:

```
sudo apt-get install php-xdebug
```

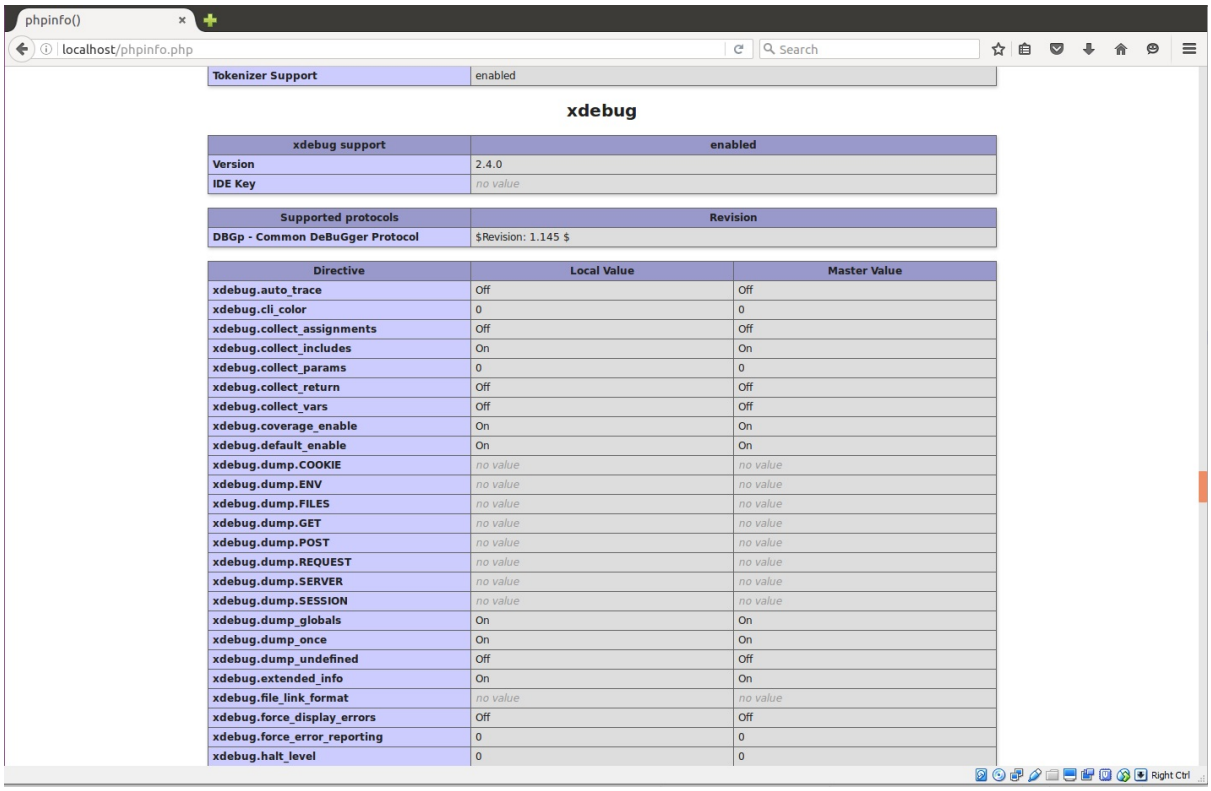
Następnie edytuj plik `/etc/php/7.0/mods-available/xdebug.ini` , wpisując:

```
sudo mcedit /etc/php/7.0/mods-available/xdebug.ini
```

Dodaj następujące linie na końcu pliku (zastąp adres IP na adres, z którego chcesz korzystać ze swojej witryny):

```
xdebug.remote_enable=1
xdebug.remote_handler=dbgp
xdebug.remote_mode=req
xdebug.remote_host=<remote_ip_address>
```

Na koniec zrestartuj serwer Apache, aby zastosować zmiany. Następnie otwórz *phpinfo.php* w przeglądarce i poszukaj sekcji XDebug (powinna wyglądać jak na rysunku):



W Fedorze, CentOS lub Red Hat Linux

W tych dystrybucjach Linuksa instalacja XDebug jest nieco trudniejsza. Zainstaluj pakiet XDebug za pomocą następującego polecenia:

```
yum install php-pecl-xdebug
```

Po instalacji wymagane jest utworzenie pliku *xdebug.ini* w katalogu */etc/php.d* :

```
mcedit/etc/php.d/xdebug.ini
```

Dodaj następujące linie na końcu pliku (zastąp adres zdalny IP adresem swojego serwera):

```
[xdebug]

zend_extension = /usr/lib/php/modules/xdebug.so
xdebug.remote_enable=1
xdebug.remote_handler=dbgp
xdebug.remote_mode=req
xdebug.remote_host=<remote_ip_address>
xdebug.remote_port=9000
```

Zrestartuj serwer WWW Apache, aby zastosować zmiany. Następnie sprawdź *phpinfo.php* w przeglądarce. Jeśli instalacja zakończyła się pomyślnie, zobaczysz informacje związane z XDebug.

Instalowanie rozszerzenia PHP XDebug w systemie Widndows

Pobierz odpowiednią bibliotekę DLL z tej witryny.

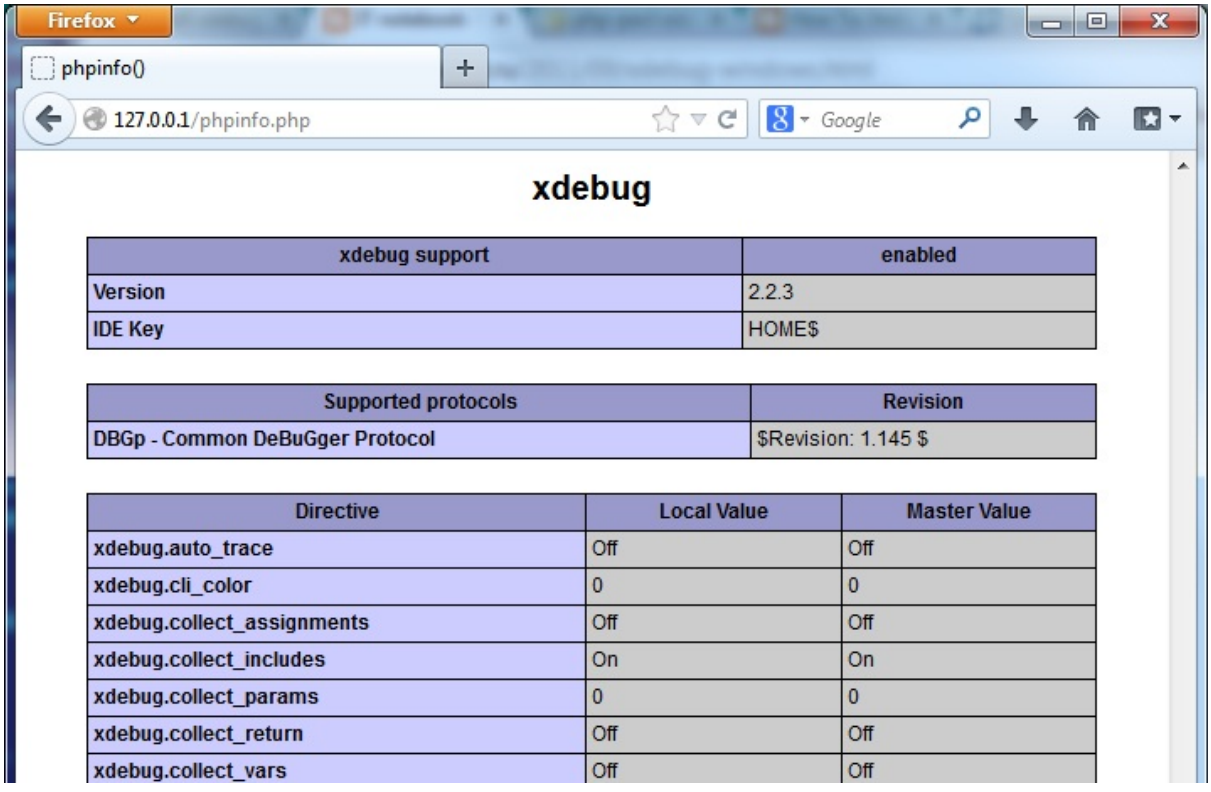
Następnie edytuj plik *php.ini* i dodaj następujący wiersz:

```
zend_extension="C:/path/to/your/xdebug.dll"
```

Dodaj następujące linie na końcu pliku (zastąp adres zdalnego IPprzez adres z którego chcesz widzieć swoją witrynę):

```
xdebug.remote_enable=on
xdebug.remote_handler=dbgp
xdebug.remote_host=<remote_ip_address>
```

Na koniec zrestartuj serwer Apache, aby zastosować zmiany. Następnie otwórz *phpinfo.php* w przeglądarce i poszukaj sekcji XDebug (powinna wyglądać jak na rysunku):



Więcej informacji:

- https://xdebug.org/wizard.php
- https://stackoverflow.com/questions/8712462/how-to-configure-xdebug-with-wamp

Fragment przykładowego pliku php.ini (dlaWAMP / Windows):

```
; XDEBUG Extension
[xdebug]
zend_extension ="C:/wamp64/bin/php/php7.2.0/zend_ext/php_xdebug-2.4.1-7.0
xdebug.remote_enable = On
xdebug.profiler_enable = off
xdebug.profiler_enable_trigger = off
xdebug.profiler_output_name = cachegrind.out.%t.%p
xdebug.profiler_output_dir ="C:/wamp64/tmp"
xdebug.show_local_vars=0
xdebug.remote_host=127.0.0.1
xdebug.remote_port=9000
xdebug.remote_autostart = 1
```

Odpowiedni fragment php.ini (do uzupełnienia po `sudo pecl install xdebug`):

```
zend_extension = /usr/lib/php/20170718/xdebug.so
xdebug.remote_enable = 1
xdebug.remote_connect_back=1
xdebug.remote_port = 9000
xdebug.scream=0
xdebug.show_local_vars=1
```

Instalowanie IDE NetBeans w systemie Linux

Możesz zainstalować IDE NetBeans używając dwóch metod: albo z repozytorium, tak jak zrobiłeś to z Apache, PHP i MySQL, albo pobierając instalator ze strony NetBeans i uruchamiając go. Pierwsza metoda jest prostsza, więc zalecamy jej użycie.

Aby zainstalować NetBeans IDE w systemie Debian lub Linux Ubuntu, wpisz następujące polecenie z powłoki poleceń:

```
sudo apt-get install netbeans
```

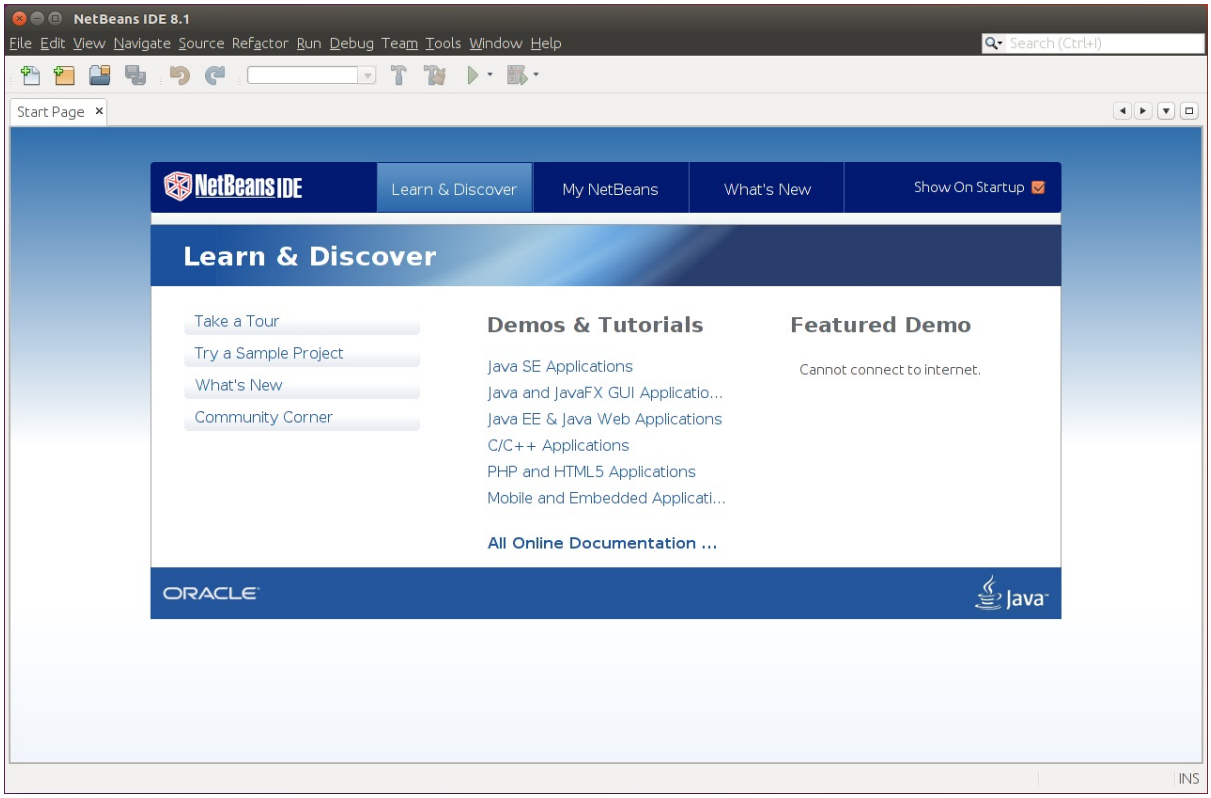
lub następujące polecenie, aby zainstalować go w Fedorze, CentOS lub Red Hat Linux:

```
sudo yum install netbeans
```

Powyższe polecenie pobiera z repozytorium i instaluje NetBeans i wszystkie jego pakiety zależne. Po zakończeniu instalacji możesz uruchomić netbeans, wpisując:

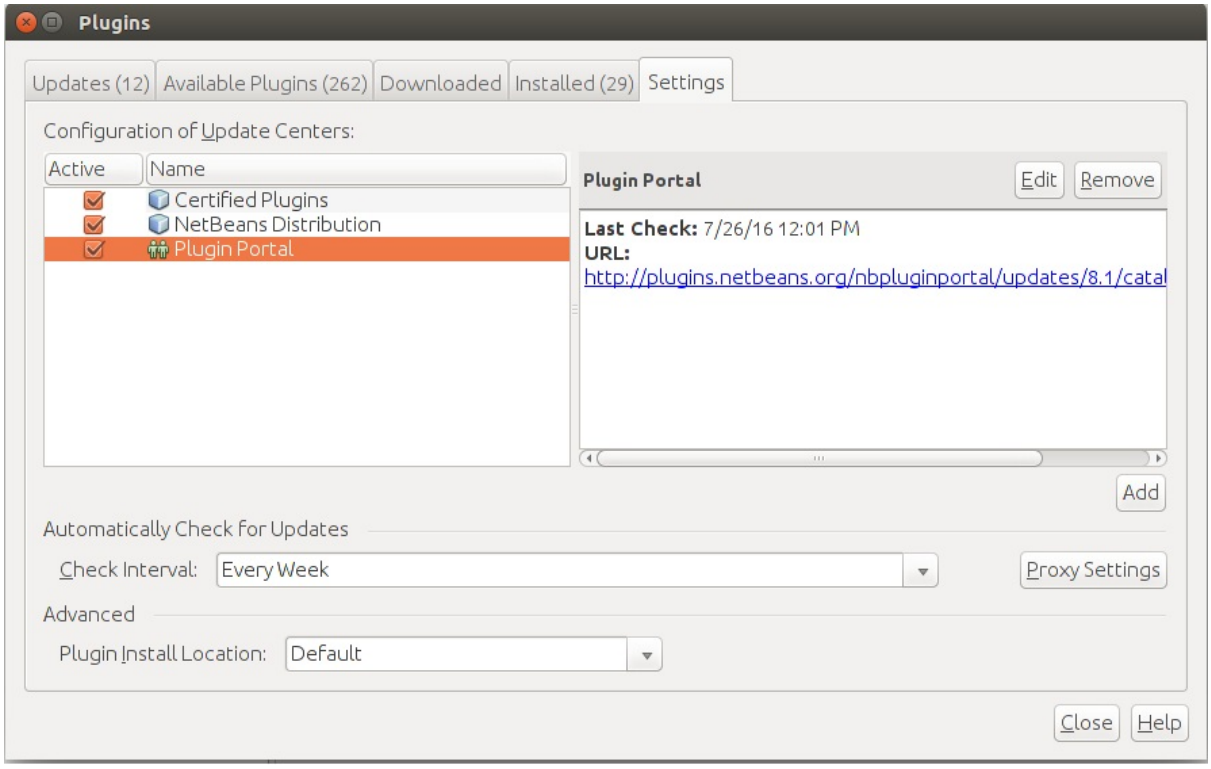
```
netbeans
```

Okno IDE NetBeans pokazano na rysunku:

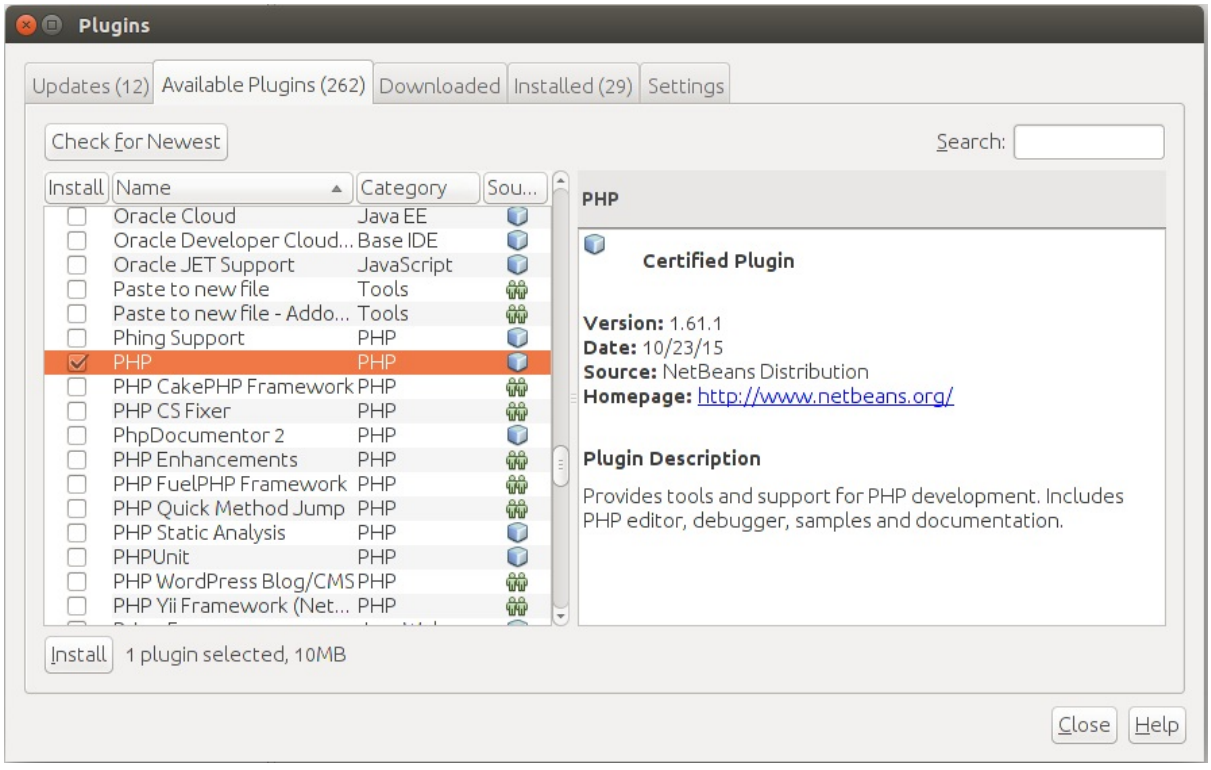


Aby móc tworzyć projekty PHP, musisz aktywować wtyczkę PHP dla NetBeans. Aby to zrobić, otwórz menu `Narzędzia-> Wtyczki (Tools -> Plugins)` pojawi się okno dialogowe Wtyczki. W wyświetlonym oknie dialogowym kliknij zakładkę *Ustawienia* i ustaw pola wyboru

dla wszystkich *Centrów aktualizacji* (patrz rysunek):



Następnie kliknij kartę *Dostępne wtyczki* . Na tej karcie kliknij przycisk *Sprawdź najnowsze* , aby utworzyć listę wszystkich dostępnych wtyczek. Następnie na liście ustaw znacznik wyboru na wtyczkę PHP i kliknij przycisk Instaluj (patrz rysunek):

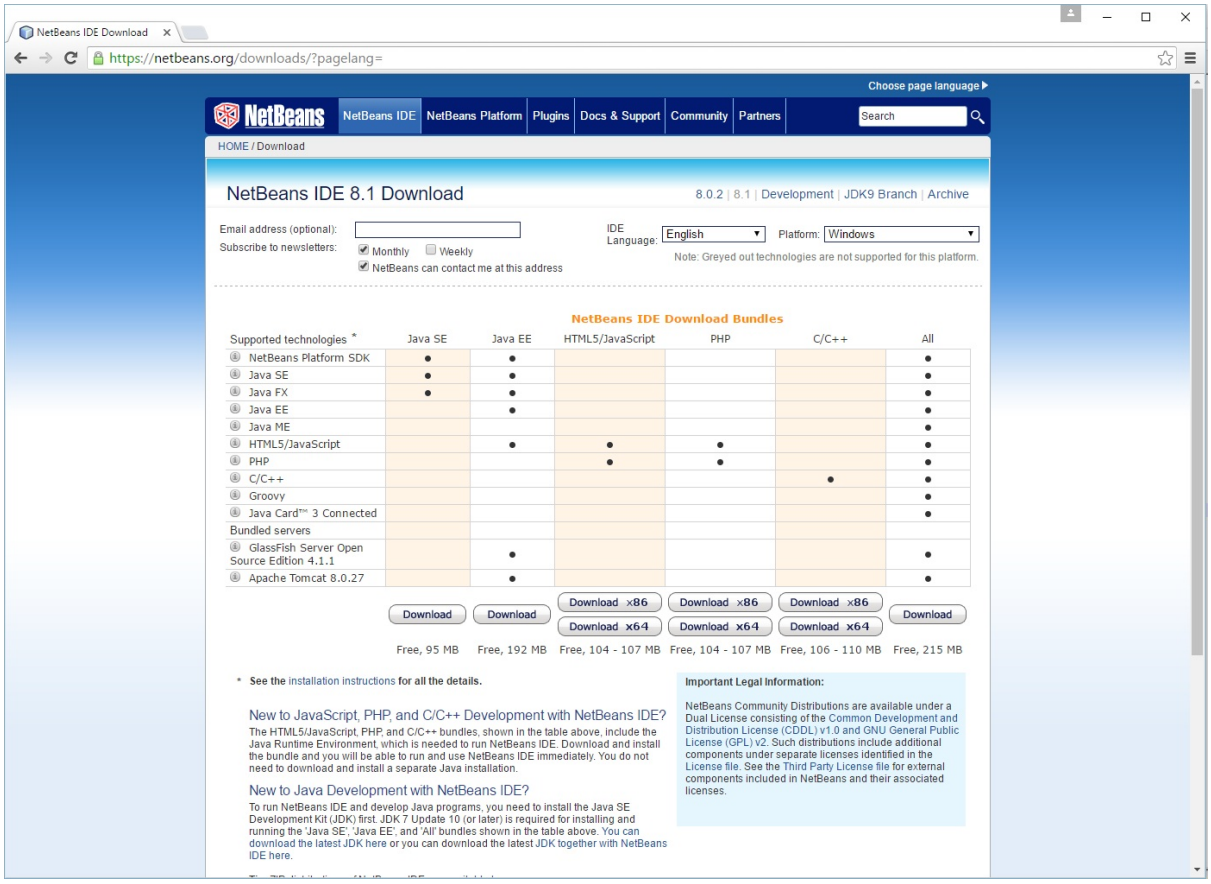


Po zakończeniu instalacji wtyczki PHP zrestartuj IDE. Wtedy powinieneś być w stanie tworzyć nowe projekty PHP z menu `New-> New Project ...` .

> Zalecane jest również zaktualizowanie IDB NetBeans do najnowszej wersji otwierając menu `Help-> Check for updates` .

Instalowanie IDE NetBeans w Windows

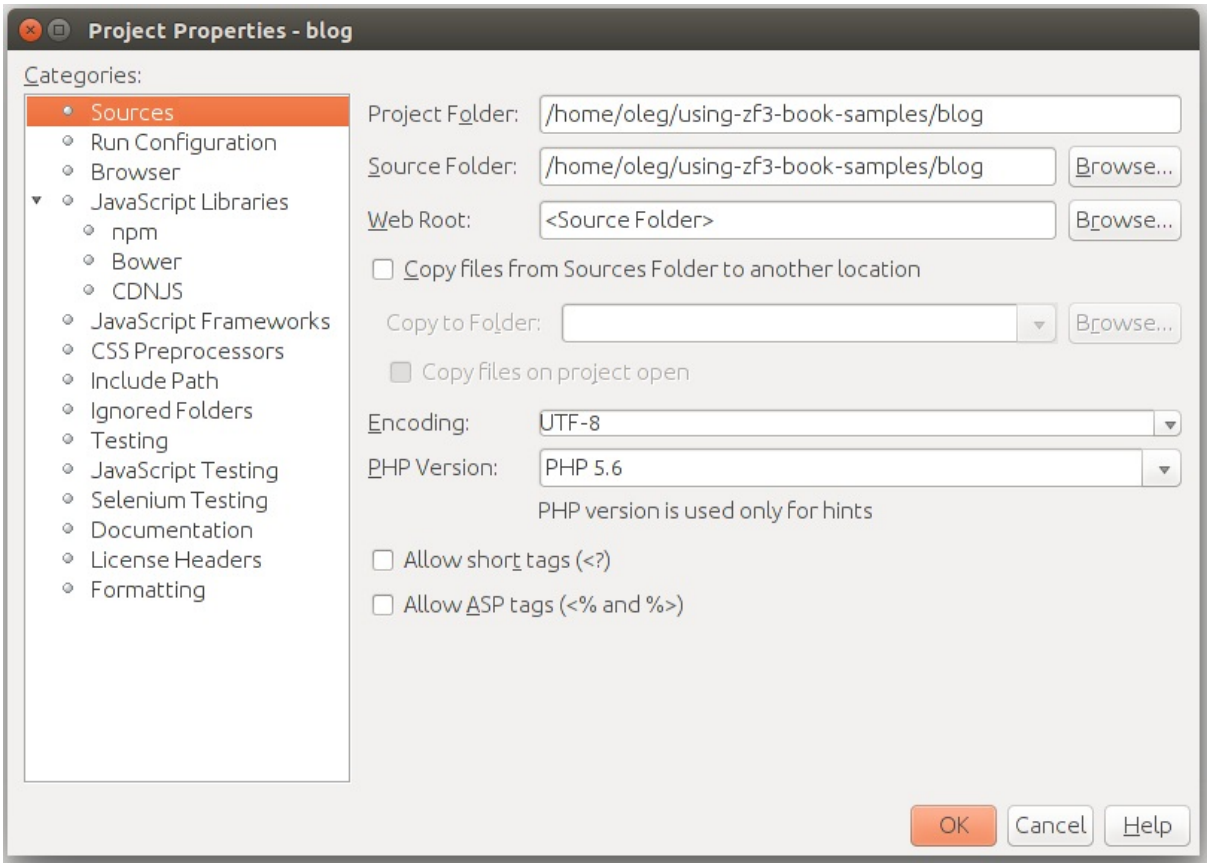
Instalacja NetBeans w Windows jest prosta. Wystarczy pobrać instalator z witryny NetBeans i uruchomić go. Możesz natknąć się na kilka pakietów NetBeans dostępnych do pobrania, a powinieneś pobrać pakiet, który jest przeznaczony do rozwoju PHP (patrz przykład).



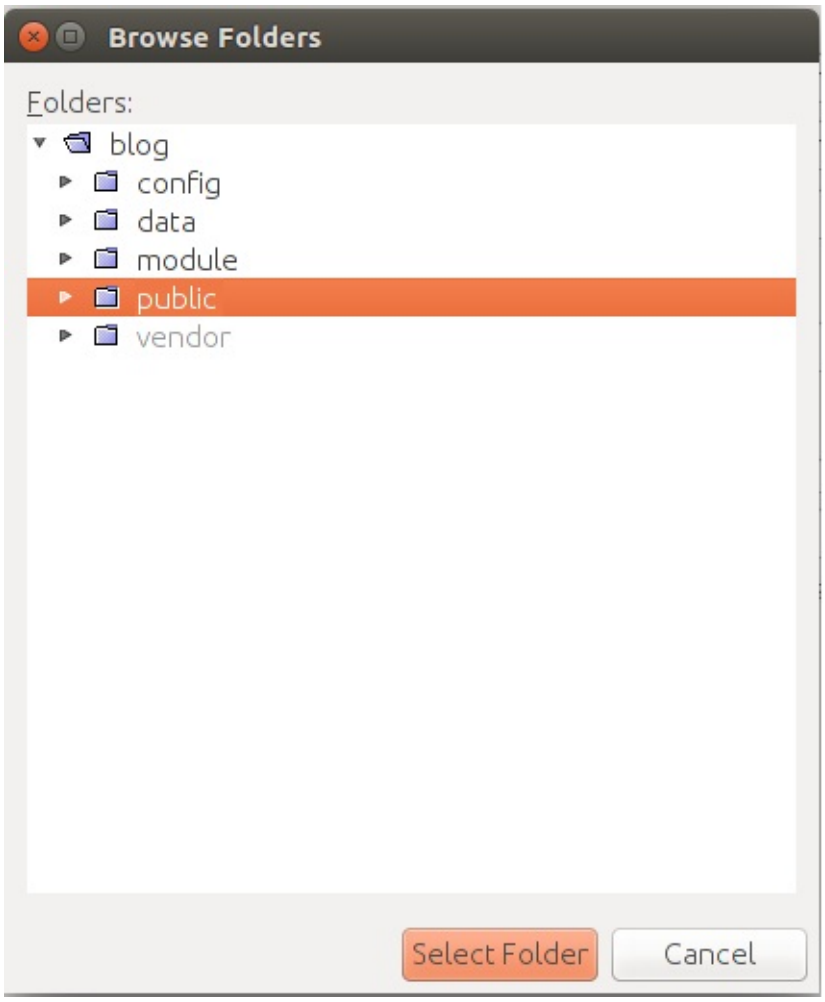
NetBeans IDE dla programisty PHP

Uruchom konfigurację

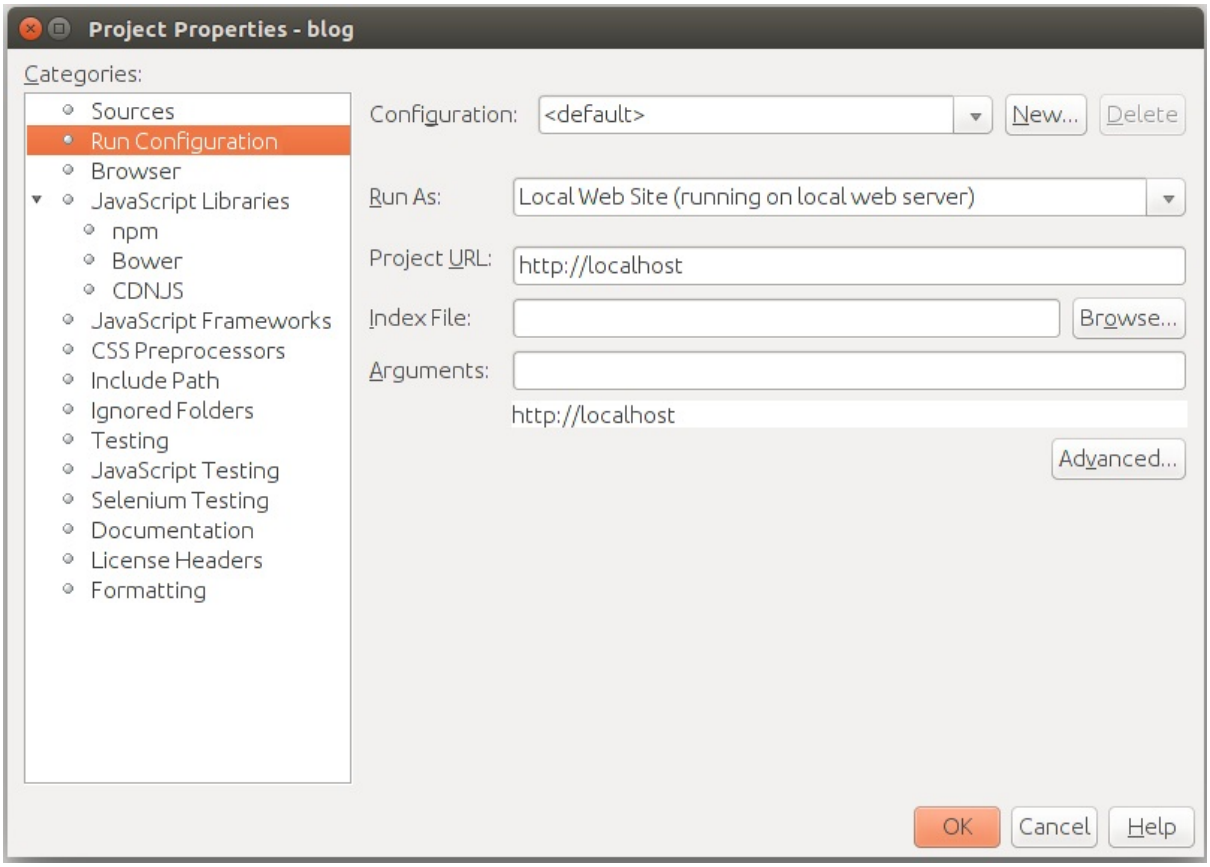
Aby móc uruchomić i debugować witrynę, najpierw musisz edytować jej właściwości. Aby to zrobić, w panelu NetBeans *Projects* kliknij prawym przyciskiem myszy nazwę projektu iz menu kontekstowego wybierz element *Properties*. Pojawi się okno dialogowe *Properties* projektu (pokazane na rysunku).



W lewym panelu okna, które się pojawi, kliknij zakładkę *Source*. W prawym okienku edytuj pole *Web Root*, aby wskazać katalog *APP_DIR/public* witryny. Możesz to zrobić, klikając przycisk *Browse* po prawej stronie pola. Następnie w oknie dialogowym kliknij katalog *public*, a następnie kliknij przycisk Wybierz przycisk folderu (pokazany na rysunku).



Następnie kliknij węzeł *Run Configuration* w lewym okienku. Prawy panel powinien wyświetlać ustawienia uruchamiania witryny (rysunek).



W prawym okienku widać, że bieżąca konfiguracja jest `domyślna` . Opcjonalnie można utworzyć kilka konfiguracji uruchamiania.

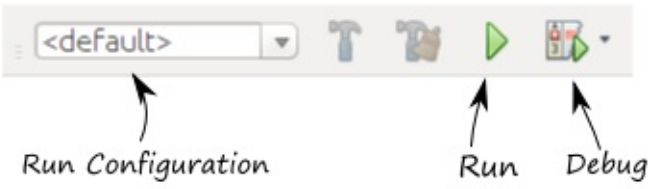
Edytuj pola w następujący sposób:

- W polu *Run As* wybierz `Witryna lokalna \(działająca na lokalnym serwerze WWW \)` .
- W polu *Project URL* wpisz `http://localhost` . Jeśli skonfigurowałeś swój wirtualny host do nasłuchu na innym porcie (na przykład na porcie 8080), wprowadź numer portu, taki jak ten `http://localhost:8080` .
- Zostaw pole *Index Plik* puste, ponieważ moduł *mod _rewrite* Apache'a zamaskuje nasz aktualny plik *index.php*.
- W polu *Arguments* można określić, które parametry GET mają być przekazywane do witryny za pośrednictwem ciągu URL. Zazwyczaj pole to pozostaje puste.

Na koniec kliknij przycisk *OK*, aby zamknąć okno dialogowe *Properties*.

Prowadzenie serwisu

Uruchomienie strony oznacza otwarcie jej w domyślnej przeglądarce internetowej. Aby uruchomić witrynę, naciśnij przycisk *Run* na pasku narzędzi *Run(rysunek B.4). Alternatywnie możesz nacisnąć przycisk _F6* na klawiaturze.



Jeśli wszystko jest w porządku z konfiguracją uruchamiania, zostanie uruchomiona domyślna przeglądarka internetowa, aw oknie przeglądarki będzie można zobaczyć stronę *Home* witryny.

Ten sam efekt miałby wpisanie w przeglądarce `http://localhost/` , ale pasek narzędzi NetBeans pozwala to zrobić jednym kliknięciem.

Site Debugging w NetBeans

Tradycyjna technika debugowania w PHP polega na umieszczeniu funkcji var_dump() w bloku kodu, który chcesz zbadać:

```
Var_dump($zmienna);
```

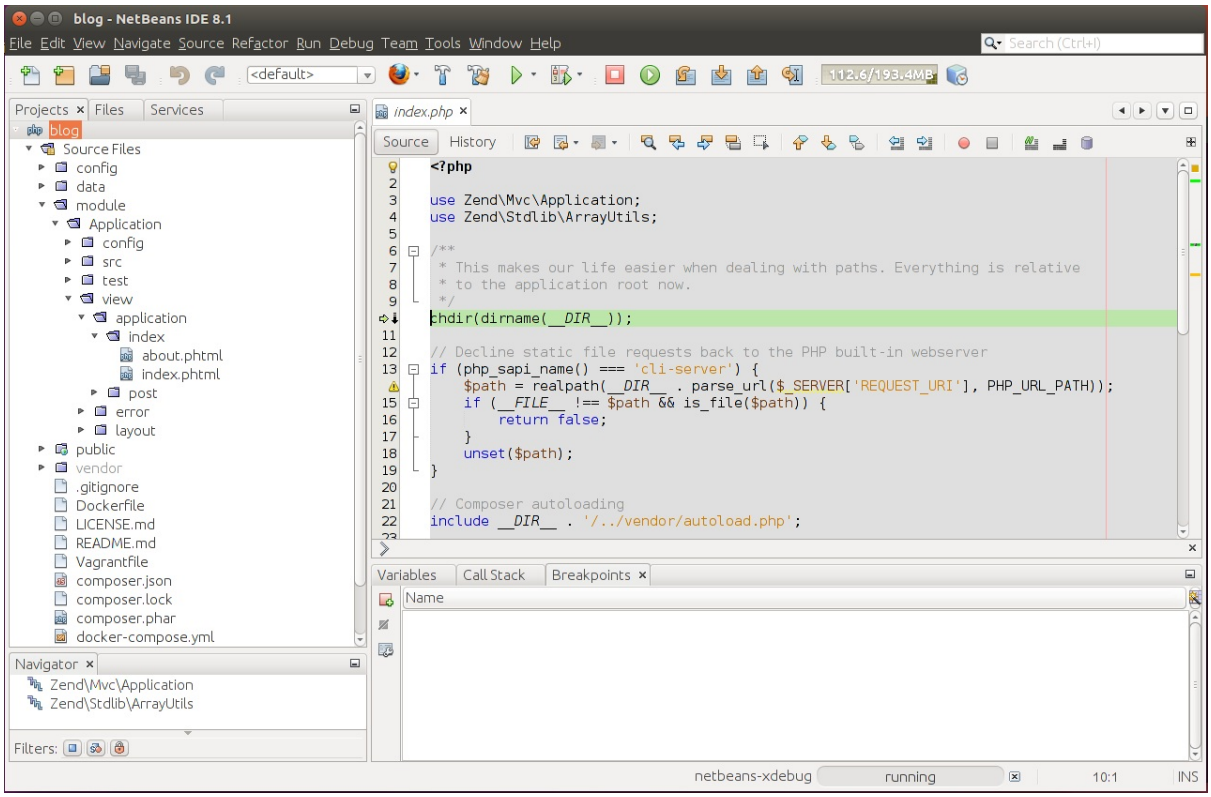
Powyższe linie wypisują wartość zmiennej `$zmienna` na wyjściu przeglądarki. Chociaż może to być użyte do debugowania nawet złożonej witryny, powoduje to, że debugowanie jest uciążliwe, ponieważ musisz wpisać polecenie zmiennej dump w pliku źródłowym PHP, a następnie odświeżyć stronę w przeglądarce, aby zobaczyć dane wyjściowe, a następnie edytować źródło ponownie plik, dopóki nie określisz przyczyny problemu.

W przeciwieństwie do tego, gdy debugujesz swoją witrynę w IDE NetBeans, interpreter PHP wstrzymuje wykonywanie programu w każdym wierszu, w którym ustawisz *breakpoint*. Umożliwia to wygodne pobieranie informacji o bieżącym stanie programu, takich jak wartości zmiennych lokalnych i stos wywołań. Informacje o debugowaniu są wyświetlane w oknie NetBeans w formie graficznej.

> Aby móc debugować witrynę, musisz mieć zainstalowane rozszerzenie XDebug (zob. początek rozdziału).

Aby rozpocząć sesję debugowania, w oknie NetBeans kliknij przycisk Debug na pasku narzędzi Run . Możesz również nacisnąć kombinację klawiszy CTRL + F5 na klawiaturze.

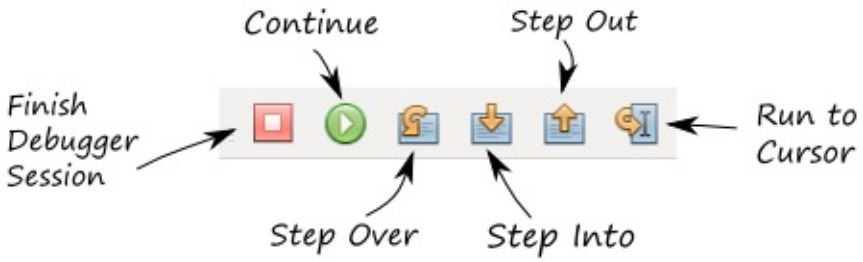
Jeśli wszystko jest w porządku, powinieneś być w stanie zobaczyć bieżący licznik programu w pierwszej linii kodu pliku *index.php* (pokazany na rysunku):



Gdy program znajduje się w stanie wstrzymania, okno przeglądarki użytkownika zostaje zamrożone, ponieważ przeglądarka oczekuje na dane z serwera WWW. Po zakończeniu wykonywania programu przeglądarka odbiera dane i wyświetla stronę internetową.

Pasek narzędzi debugowania

Możesz wznowić/zawiesić wykonanie programu za pomocą *Debug Toolbar* (patrz rysunek B.6):



Przycisk *Finish Debugger Session* na pasku narzędzi pozwala zatrzymać debugger. Naciśnij ten przycisk, gdy skończysz debugowanie programu. Ten sam efekt spowodowałby naciśnięcie kombinacji klawiszy *SHIFT + F5*.

Kliknięcie przycisku *Continue* (lub naciśnięcie klawisza *F5*) kontynuuje wykonywanie programu do następnego punktu przerwania lub do końca programu, jeśli nie ma więcej punktów przerwania.

Przycisk *Step Over* na pasku narzędziowym (lub naciśnięcie klawisza *F8*) przesuwa bieżący licznik programu do następnego wiersza programu.

Przycisk *Step Into* na pasku narzędzi (lub naciśnięcie klawisza *F7*) przesuwa bieżący licznik programu do następnego wiersza programu, a jeśli jest to punkt wejścia funkcji, wchodzi w ciało funkcji. Skorzystaj z tego, jeśli chcesz dokładniej zbadać swój kod.

Przycisk *Step Out* na pasku narzędzi (*CTRL + F7*) pozwala na kontynuowanie wykonywania programu aż do powrotu z bieżącej funkcji.

Natomiast *Run to Cursor* (*F4*) pozwala na kontynuowanie wykonywania programu aż do linii kodu, w której umieścisz kursor. Może to być wygodne, jeśli chcesz pominąć blok kodu i zatrzymać się w określonej linii programu.

Punkty przerwania

Zazwyczaj ustawia się jeden lub kilka punktów przerwania na wiersze, które mają być debugowane w trybie krok po kroku. Aby ustawić punkt przerwania, umieść mysz po lewej stronie linii kodu, w miejscu, w którym ma się pojawić punkt przerwania, i kliknij numer linii. Możesz też umieścić kursor kursora w wierszu, w którym chcesz ustawić punkt przerwania, i nacisnąć kombinację klawiszy *CTRL + F8*.

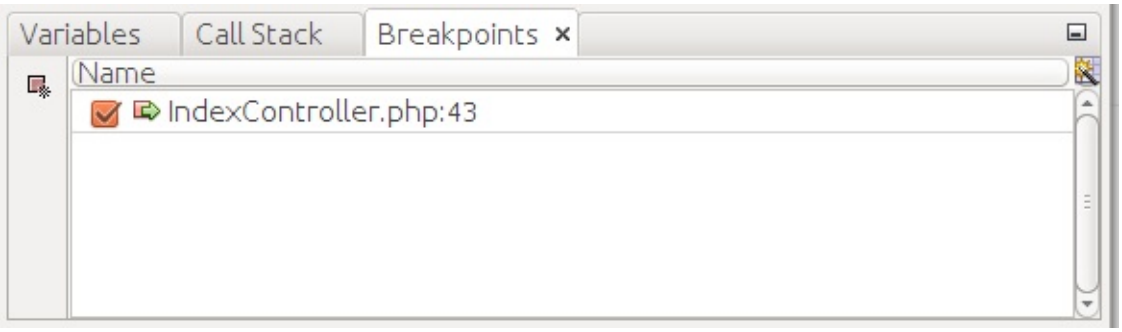
Po ustawieniu punktu przerwania linia jest oznaczona kolorem czerwonym, a po lewej stronie pojawia się mały czerwony prostokąt (pokazany na rysunku):

> Uważaj, aby nie ustawiać punktu przerwania w pustym wierszu lub w wierszu komentarza. Taki punkt przerwania zostanie zignorowany przez XDebug i będzie oznaczony przez `zepsuty` kwadrat (patrz przykład na rysunku):

```
31      */
32     public function __construct($entityManager, $postManager)
33     {
34         $this->entityManager = $entityManager;
35         $this->postManager = $postManager;
36     }
37
39     /**
40      * This is the default "index" action of the controller. It displays the
41      * Recent Posts page containing the recent blog posts.
42      */
43     public function indexAction()
44     {
45         $tagFilter = $this->params()->fromQuery('tag', null);
46
47         if ($tagFilter) {
48
49             // Filter posts by tag
50             $posts = $this->entityManager->getRepository(Post::class)
51                     ->findPostsByTag($tagFilter);
```

Możesz przechodzić pomiędzy punktami przerwania klawiszem *F5*. Ten przycisk kontynuuje wykonywanie programu, dopóki nie napotka następnego punktu przerwania. Gdy przepływ programu dojdzie do punktu przerwania, interpreter PHP zostanie wstrzymany i można przejrzeć stan programu.

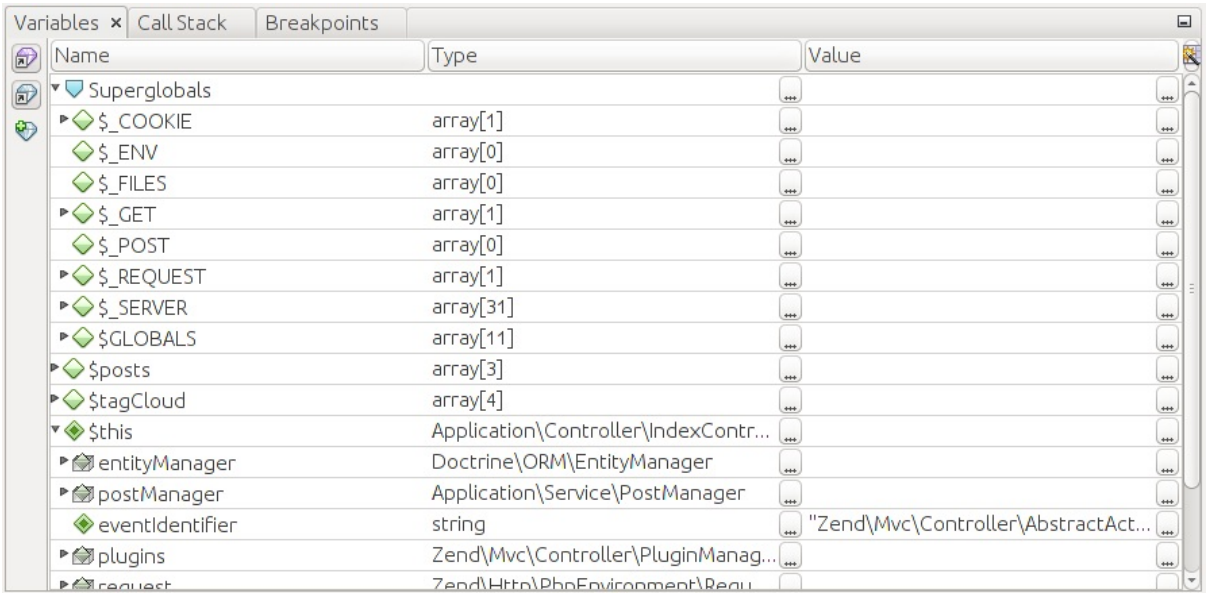
Możesz zobaczyć pełną listę punktów przerwania ustawionych w oknie *Breakpoints (patrz rysunek). _Okno _Breakpoints* znajduje się w dolnej części okna NetBeans. W tym oknie możesz dodawać nowe punkty przerwania lub anulować ustawione punkty przerwania, które zostały już ustawione.



Oglądanie zmiennych

Gdy interpreter PHP jest wstrzymany, możesz wygodnie obserwować wartości zmiennych PHP. Prostym sposobem przeglądania zmiennej jest po prostu umieszczenie kursora myszy nad nazwą zmiennej wewnątrz kodu i czekanie na sekundę. Jeśli wartość zmiennej można ocenić, zostanie ona wyświetlona wewnątrz małego wyskakującego okna.

Innym sposobem oglądania zmiennych jest okno *Variables* (pokazane na rysunku), które jest wyświetlane w dolnej części okna NetBeans. Okno *Variables* ma trzy kolumny: *Name*, *Type* i *Value*.



Przede wszystkim będziesz mieć do czynienia z trzema rodzajami zmiennych: *super globals*, *locals* i *$this*:

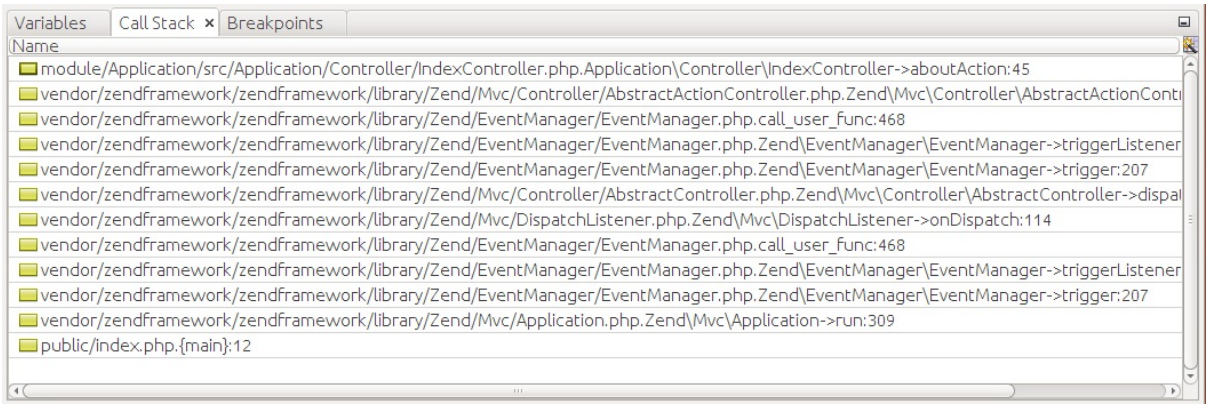
- Zmiene *Super global* są specjalnymi zmiennymi PHP, takimi jak `$_GET`, `$_POST`, `$_SERVER`, `$_COOKIES` i tak dalej. Zazwyczaj zawierają informacje o serwerze i parametry przekazywane przez przeglądarkę internetową jako część żądania HTTP.
- Zmienne lokalne są zmiennymi, które są w zakresie obecnej funkcji (lub metody klasy).
- *_Zmienna* $this_ wskazuje na bieżącą instancję (obiekt) klasy, jeśli bieżący kod jest wykonywany w kontekście klasy PHP.

Niektóre zmienne można `rozwinąć` (aby rozwinąć zmienną, należy kliknąć ikonę trójkąta po lewej stronie nazwy zmiennej). Na przykład, klikając i rozszerzając zmienną *$this*, można oglądać wszystkie pola instancji klasy. Jeśli rozwiniesz zmienną tablicową, będziesz mógł oglądać elementy tablicy.

Za pomocą okna *Variables* można nie tylko oglądać wartość zmiennej, ale także zmieniać wartość w locie. Aby to zrobić, umieść kursor myszy nad kolumną wartości i kliknij nad nią. Pojawi się pole edycji, w którym można ustawić nową wartość zmiennej.

Call Stack

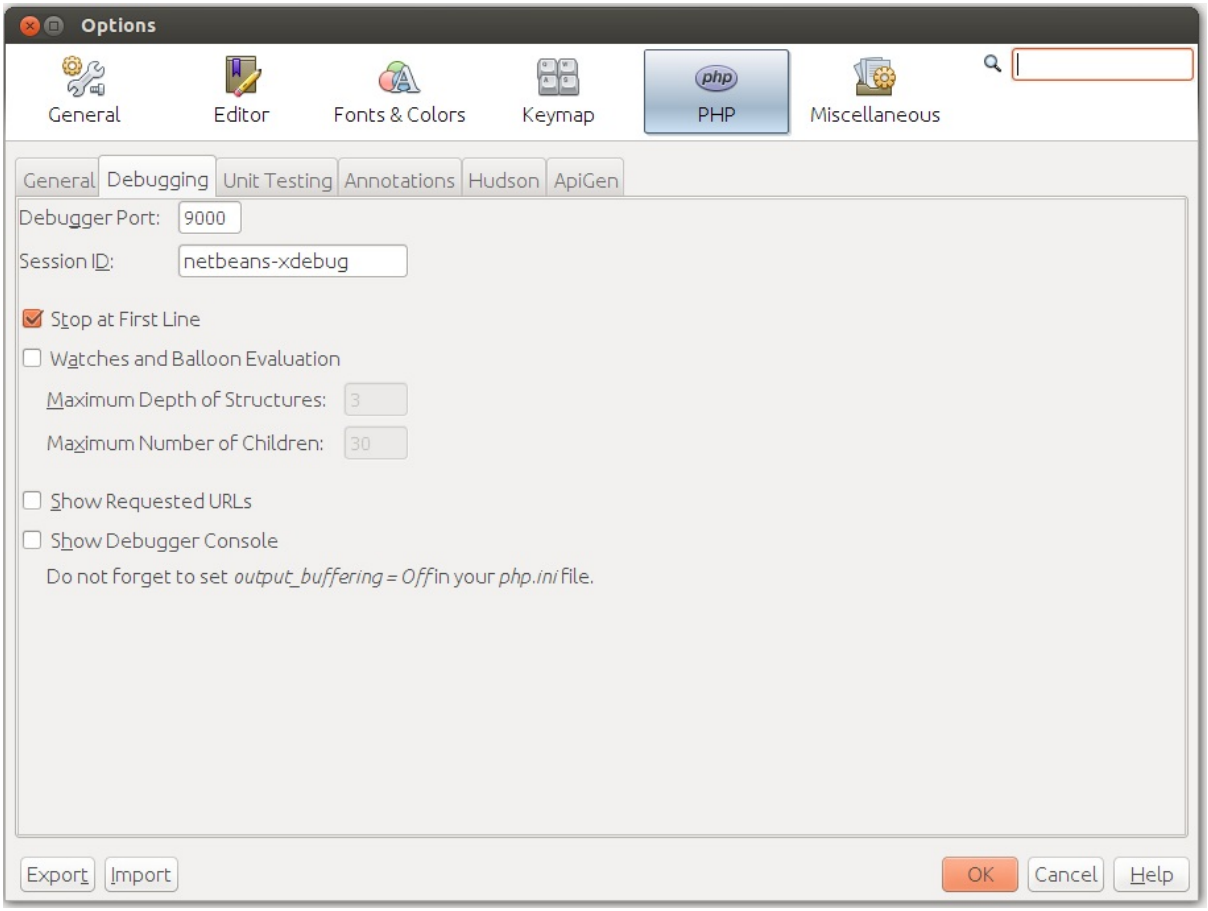
call stack wyświetla listę zagnieżdżonych funkcji, których kod jest wykonywany w chwili (pokazany na rysunku poniżej). Każda linia stosu wywołań (zwana także ramką stosu) zawiera pełną nazwę klasy, nazwę metody w klasie i numer linii. Przenosząc stos, można lepiej zrozumieć aktualny stan wykonania programu.



Na przykład na rysunku widać, że aktualnie wykonywana jest operacja `IndexController::aboutAction()`, która została wywołana przez metodę `AbstractActionController::onDispatch()` i tak dalej. Możemy przechodzić stos wywoławczy, dopóki nie dojdziemy do pliku *index.php*, który jest wierzchołkiem stosu. Możesz także kliknąć ramkę stosu, aby zobaczyć miejsce kodu, który jest aktualnie wykonywany.

Opcje debugowania

NetBeans pozwala skonfigurować niektóre aspekty działania debuggera. Aby otworzyć okno dialogowe *Opcje*, wybierz menu *Narzędzia-> Opcje*. W wyświetlonym oknie dialogo(rysunek B.12)wym kliknij kartę *PHP*, a następnie w tej zakładce wybierz podfolder *Debugging*.



Zazwyczaj nie zmieniasz większości tych opcji, musisz tylko mieć pojęcie o tym, co robią. Oto krótki opis opcji debugowania:

- Parametry *Debugger Port* oraz *Session ID* definiują sposób łączenia NetBeans z XDebug. Domyślnie numer portu to 9000. Numer portu powinien być taki sam jak port debuggera ustawiony w pliku *php.ini* podczas instalowania XDebug. Nazwa sesji jest domyślnie `netbeans-xdebug` . Zwykle nie zmieniasz tej wartości.
- Parametr *Stop at First Line* sprawia, że debugger zatrzymuje się w pierwszym wierszu pliku *index.php*, zamiast zatrzymywać się w pierwszym punkcie przerwania. Może to być denerwujące, więc możesz odznaczyć tę opcję.
- Grupa opcji *Watches and Balloon Evaluation* jest domyślnie wyłączona, ponieważ mogą one powodować błąd XDebug. Możesz włączyć te opcje tylko wtedy, gdy wiesz, co robisz.
- Parametry *Maximum Depht Structures* (*Maksymalna głębokość struktur)* określają, czy zagnieżdżone struktury (jak zagnieżdżone tablice, obiekty w obiektach itp.) Będą widoczne czy nie. Domyślnie głębokość jest ustawiona na 3.
- Opcja Maximum number of Childrens (*Maksymalna Liczba Potomków)* definiuje liczbę elementów tablicy wyświetlanych w oknie *Variables*. Jeśli ustawisz to na, powiedzmy 30, zobaczysz tylko pierwszych 30 pozycji, nawet jeśli tablica ma więcej niż 30 elementów.

- Opcja `Pokaż żądane adresy URL` po włączeniu wyświetla adres URL, który jest aktualnie przetwarzany. Wydrukuje adres URL do okna *Output*.
- Opcja *Debugger Console* pozwala zobaczyć wyjście debugowanych skryptów PHP. Dane wyjściowe są wyświetlane w oknie *Output*. Jeśli zamierzasz korzystać z tej funkcji, zalecane jest dodanie parametru `output_buffering = Off` w sekcji `[xdebug]` twojego pliku *php.ini*, w przeciwnym razie wyjście może pojawić się z opóźnieniami.

Profilowanie

Gdy Twoja strona jest gotowa i działa, zazwyczaj chcesz ją przyspieszyć. XDebug zapewnia możliwość *profilu* Twojej witryny. Profilowanie oznacza określanie, które metody klasy (lub funkcje) tracą czas. Umożliwia to określenie miejsc `wąskich gardeł` w kodzie i rozwiązywanie problemów z wydajnością.

Dla każdego żądania HTTP rozszerzenie XDebug mierzy czas wykonywania funkcji i zapisuje informacje profilowania do pliku. Zazwyczaj pliki informacyjne profilowania są umieszczane w katalogu tymczasowym systemu (w systemie Linux, do katalogu */tmp*) i mają nazwy takie jak `xdebug.out. <timestamp>` , gdzie `<timestamp>` - to znacznik czasu żądania HTTP. Wszystko, co musisz zrobić, to otworzyć plik profilujący i przeanalizować go.

> Aby włączyć profiler XDebug, należy ustawić następujący parametr konfiguracyjny XDebug w pliku *xdebug.ini*:
>
> ```
> <xdebug.profiler_enable = 1
> ```

Niestety, NetBeans for PHP nie ma osadzonego narzędzia do wizualizacji wyników profilowania. Dlatego musisz zainstalować zewnętrzne narzędzie do wizualizacji. Poniżej przedstawiamy instrukcje dotyczące instalacji prostego narzędzia internetowego o nazwie Webgrind. Webgrind może działać na dowolnej platformie, ponieważ to samo narzędzie jest napisane w PHP.

Instalacja Webgrind jest bardzo prosta.

Najpierw ściągnij webgrind ze strony projektu i rozpakuj go do jakiegoś folderu. W systemie Linux możesz to zrobić za pomocą następujących poleceń:

```
cd ~
```

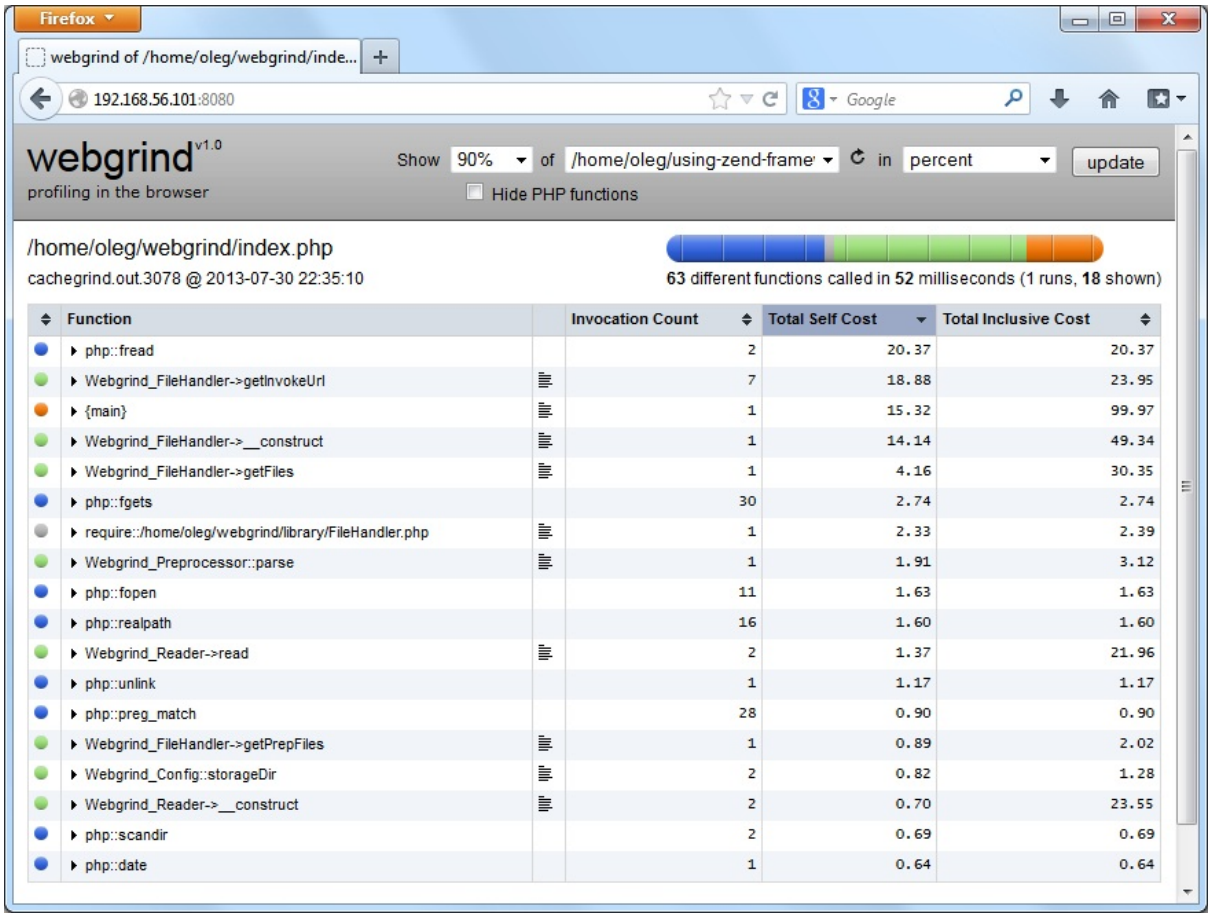
```
wget https: // github.com/jokkedk/webgrind/archive/master.zip
```

```
unzip master.zip
```

Powyższe polecenia zmienią Twój katalog roboczy na katalog domowy, a następnie ściągną archiwum Webgrind z Internetu, a następnie rozpakują archiwum.

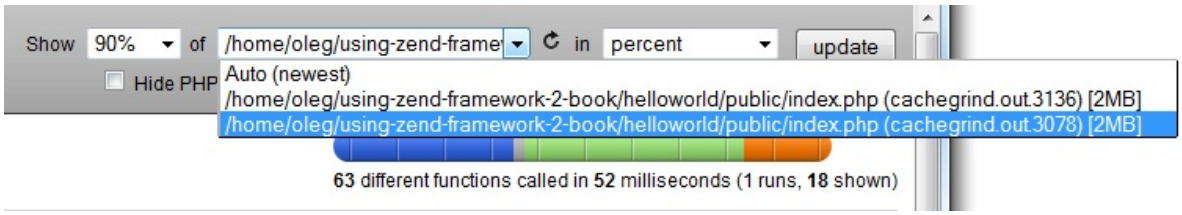
Następnie musisz poinformować serwer Apache, gdzie można znaleźć pliki Webgrind. Oznacza to, że musisz skonfigurować oddzielny host wirtualny. Nie zapomnij zrestartować serwera WWW Apache po skonfigurowaniu hosta wirtualnego.

Wreszcie, otwórz Webgrind w przeglądarce, przechodząc do adresu URL instalacji Webgrind. Na przykład, jeśli skonfigurowałeś host wirtualny do nasłuchu na porcie 8080, wpisz `http://localhost: 8080)` na pasku nawigacyjnym przeglądarki i naciśnij Enter. Strona internetowa Webgrind powinna pojawić się:



Function	Invocation Count	Total Self Cost	Total Inclusive Cost
php::fread	2	20.37	20.37
Webgrind_FileHandler->getInvokeUrl	7	18.88	23.95
{main}	1	15.32	99.97
Webgrind_FileHandler->__construct	1	14.14	49.34
Webgrind_FileHandler->getFiles	1	4.16	30.35
php::fgets	30	2.74	2.74
require::/home/oleg/webgrind/library/FileHandler.php	1	2.33	2.39
Webgrind_Preprocessor::parse	1	1.91	3.12
php::fopen	11	1.63	1.63
php::realpath	16	1.60	1.60
Webgrind_Reader->read	2	1.37	21.96
php::unlink	1	1.17	1.17
php::preg_match	28	0.90	0.90
Webgrind_FileHandler->getPrepFiles	1	0.89	2.02
Webgrind_Config::storageDir	2	0.82	1.28
Webgrind_Reader->__construct	2	0.70	23.55
php::scandir	2	0.69	0.69
php::date	1	0.64	0.64

Na górze strony Webgrind możesz wybrać procent `najcięższych` wywołań funkcji. Domyślnie jest ustawiony na 90%. Ustawienie tego na niższy procent spowoduje ukrycie funkcji o mniejszej częstotliwości.



Lista rozwijana po prawej stronie pola procentowego pozwala wybrać plik danych profilowania do analizy. Domyślnie jest ustawiony na `Auto(newest)`, co zmusza Webgrind do używania pliku z najnowszym znacznikiem czasu. Może być konieczne wybranie innego pliku, na przykład, jeśli strony internetowe korzystają z asynchronicznych żądań AJAX.

Prawe menu rozwijane pozwala ustawić jednostki, które powinny być używane do pomiaru danych. Możliwe opcje to: procent (domyślnie), milisekundy i mikrosekundy.

Po wybraniu wartości procentowej, nazwy pliku i jednostek kliknij przycisk Aktualizuj, aby umożliwić programowi Webgrind wizualizację danych dla użytkownika (obliczenia mogą zająć kilka sekund). Po zakończeniu obliczeń powinieneś zobaczyć tabelę wywołań funkcji, posortowaną w porządku malejącym według funkcji `waga` . Najcięższe funkcje będą wyświetlane u góry.

Tabela ma następujące kolumny:

- Pierwsza kolumna (*Function*) wyświetla nazwę klasy, po której następuje nazwa metody (w przypadku wywołania metody) lub nazwa funkcji (w przypadku zwykłej funkcji).
- Druga kolumna zawiera ikony `akapitu` , które można kliknąć, aby otworzyć odpowiedni plik źródłowy PHP, którego funkcja jest zdefiniowana w przeglądarce internetowej. Kolumna `_Invocation Count_` pokazuje ile razy funkcja została wywołana. Kolumna `_Total Self Cost_` pokazuje całkowity czas potrzebny na wykonanie wbudowanego kodu PHP w funkcji (z wyłączeniem czasu spędzonego na wykonywaniu innych niestandardowych funkcji). Kolumna *_Total Inclusive Cost* zawiera całkowity czas wykonania funkcji, w tym wbudowany kod PHP i inne wywoływane funkcje użytkownika.

Kliknięcie nagłówka kolumny umożliwia sortowanie danych w porządku rosnącym lub malejącym.

Możesz kliknąć ikonę trójkąta obok nazwy funkcji, aby rozwinąć listę wywołań funkcji. Ta lista pozwala zobaczyć, kto nazwał tę funkcję i ile czasu zajmuje, i zawiera następujące kolumny:

- *Calls* to funkcje `rodzica` lub metody klasy wywołujące tę funkcję (child);
- *Całkowity koszt połączenia* to całkowity czas wykonania tej funkcji, po wywołaniu z funkcji nadrzędnej;
- *Count* - liczba przypadków, w których rodzic wywołuje funkcję potomną.

Kolorowy pasek na górze strony wyświetla wkład różnych typów funkcji:

- *Blue* oznacza funkcje wewnętrzne PHP (wbudowane);
- *Lavender* to czas potrzebny na dołączenie (lub wymaganie) plików PHP;
- *Green* pokazuje udział twoich własnych metod klasowych;
- *Orange* oznacza czas zajęty na tradycyjnych `proceduralnych` funkcjach (funkcje, które nie są częścią klas PHP).

> Należy pamiętać, że profiler tworzy nowy plik danych w katalogu */tmp* dla każdego żądania HTTP do Twojej witryny. Może to spowodować wyczerpanie miejsca na dysku, które można naprawić tylko po ponownym uruchomieniu systemu.

> Po zakończeniu profilowania aplikacji zalecane jest wyłączenie profilowania poprzez edycję pliku *php.ini*, i zmienę parametru `xdebug.profiler_enable` , , a następnie ponowne uruchomienie serwera WWW Apache:
>
> ```
> xdebug.profiler_enable = 0
> ```

PHP. Podstawy.

36 prostych przykładów ilustrujących podstawowe konstrukcje języka PHP.

Źródła dostępne na: https://github.com/tenarjw/php1/tree/master/basis

Jak testować

1) z konsoli

```
$php -a
php > include "ex12.php";
albo po prostu:
$ php ex12.php
```

2) Przy pomocy serwera wbudowanego php:

```
$ php -S localhost:8000
```

W przeglądarce piszemy http://localhost:8000/ex12.php

3) Na serwerze www z PHP (kopiujemy np. przez ftp). Jeśli **ex12.php** w katalogu **dir** serwera **www.przyklady.pl**, to w przeglądarce piszemy: http://www.przyklady.pl/dir/ex12.php

Najprostsze przykłady

(01) https://ideone.com/CNR43w:

```
<html>
<head>
····<meta charset="utf8" />
</head>

<body>
<?php
echo '<h1>Można mieszać kod php i zwykły HTML</h1>';
echo "<p>Jednak nie jest to obecnie zalecanym sposobem programowania<br /
(standard MVC)";
```

```
?>
<p>Naciśnij Ctr+U by zobaczyć efekty</p>
</body>

</html>
```

==

```
<?php
print "Hello World";
echo "<br />print i echo - alternatywne  funkcje druku";
```

Łańcuchy znaków ujmujemy w apostrofy (') lub cudzysłów ("). Mogą one zajmować więcej niż 1 wiersz. (02) https://ideone.com/qT1px5:

```
<html>
<head>
<meta charset="utf8"/>
</head>
<body>
<?php
echo '<h1>Można mieszać kod php i zwykły HTML</h1>';
echo "<p>Jednak nie jest to obecnie zalecanym sposobem programowania<br /
(standard MVC)";
?>
<p>Naciśnij Ctr+U by zobaczyć efekty</p>
</body>
</html>
```

Pliki ze skryptami” możemy wstawiać jeden w drugi – za pomocą jednej z komend:

- **include**: wstaw za każdym razem, gdy wywołana instrukcja.
- **require**: tak jakinclude,alepowoduje błąd (a nie tylko ostrzeżenie) gdy brak pliku.
- **include_once**: tak jakinclude,ale tylko raz dołącza ten sam plik.
- **require_once**: tak jakrequire,ale tylko raz sołącza ten sam plik.

Komentarze (03) https://ideone.com/XJ8d6d:

```
<?php

/*
```

```
 * Znak ukośnika (slash) z gwiazdką oznacza początek komentarza.
 * W odwrotnej kolejności (gwiazdka ukośnik) - koniec komentarza.
 */

// po dwóch ukośnikach umieszczamy komentarz jednowierszowy (nie wymaga k
echo 'komentarze nie są wyświetlane na stronie'; // po komentarzu jednowi
```

Zmienne

04 https://ideone.com/XJ8d6d:

```
<?php
// zmienne: identyfikator rozpoczyna się znakiem dolara ($)
$a = 22; // znak = oznacza zmianę zmiennej (ustawienie wartości)
$b = 33;
$c = $a + $b; // można używać wyrażeń
echo $c; // echo (print) może wyświetlić zawartoś zmiennej

// poprawne identyfikatory zmiennych
$litery_cyfry123_podkreslenie = '1';
$_podkreslenie_jak_litery  = 2;
$camelCase = 3; // Duże i małe litery
/* niepoprawne:
$1cyfra = 4; // nie może zaczyna się od cyfty
$dolar$wSrodku = 5; // nie może zawierać dodatkowych dolarów
$a&^inne = 6; // ani innych znaków
$błąd = 7; // nie może zawieać znaków innych niż łacińskie (w tym polskic
$bez spacji = 8; // ani spacji
*/
```

Typy danych

- **Boolean**:logiczne (truelub false)
- **Integers**:Liczby całkowite
- **Float**:Liczby rzeczywiste (zmiennoprzecinkowe)
- **Strings**:Łańcuchy znaków

Typy zmiennych są ustawiane w chwili wstawienia do nich danych. W ten sam sposób zmieniamy typ.

05 https://ideone.com/gV9SR4:

```php
<?php
// typ i zawartoś danych można wyświetlić używając var_dump
$liczba = 123; var_dump($liczba);
$liczba = 'abc'; var_dump($liczba);
$napis = "12";
// w wyrażeniach PHP potrafi zamienić napis na liczbę
var_dump($napis+12);
var_dump(12+$napis);
var_dump($liczba>0); // logiczne
var_dump(12.2); // zmiennopozycyjne
```

Wynik:

```
int
123
string
'abc'
(length=3)
int
24
int
24
boolean
false
float
12.2
```

Operatory

Plus, minus, mnożenie, dzielenie : (+,-,*,/)modulo(%)potęga (**)negacja (-) :

06 https://ideone.com/TGNqb9

```php
<?php
$x = 12;
$y = 4;
print '<br />';print $x + $y; // 16
print '<br />';print $x - $y; // 8
print '<br />';print $x * $y; // 48
print '<br />';print $x / $y; // 3
print '<br />';print $x % $y; // 0
print '<br />';print $x ** $y; // 20736
```

```
print '<br />';print -$x; // -12
```

Operatory przypisania

[07] https://ideone.com/a18HUt:

```
<?php
$x = 12;
$x+=1; print $x; // 13
print '<br />';
$x-=3;print $x; // 10
print '<br />';
$x*=4;print $x; // 40
```

Operatory porównania

Mniejszy (<), większy (>), mniejszy – równy (<=), większy równy (>=), równy (==), identyczny (===), nierówny (!=lub <>), nieidentyczny (!==).

[08] https://ideone.com/NBX8xl:

```
<?php
$x = 12; $y=12.0;
var_dump($x==$y); // równe
var_dump($x===$y); // ale nie identyczne
var_dump($x<$y);
var_dump($x<=$y);
var_dump($x!=$y);
var_dump($x<>$y);
```

Wynik:

```
boolean
true
boolean
false
boolean
false
boolean
true
```

Operatory logiczne

[09] https://ideone.com/E4s81d:

```php
<?php
$x = 12; $y=12.0;
var_dump($x>$y || $x<=$y); // lub - true
var_dump($x>$y && $x<$y); // i - false
var_dump(! $x>$y); // negacja - false
```

Incrementi decrement

[10] https://ideone.com/WoOSIg

```php
<?php
$x = 13;
var_dump($x++); // druk 13, potem $x = $x+1
$y = ++$x; // $x i $y zwiększone o 1 (15)
var_dump($x, $y);
$y = $x--; // zmienia się tylko $x (znaki po zmiennej)
var_dump($x, $y);
```

Pierwszeństwo operatorów

Operator	Type
**	Arithmetic
++,--	Increasing/decreasing
!	Logical
*,/,%	Arithmetic
+,-	Arithmetic
<,<=,>,>=	Comparison
==,!=,===,!==	Comparison
&&	Logical
\	\		Logical
=,+=,-=,*=,/=,%=,**=	Assignment

[11] https://ideone.com/jAc1IO

```php
<?php
$a = 1;
$b = 3;
$c = true;
$d = false;
$e = $a + $b > 5 || $c; // true
var_dump($e);
$f = $e == true && !$d; // true
var_dump($f);
$g = ($a + $b) * 2 + 3 * 4; // 20
var_dump($g);
```

Łańcuchy znaków

Funkcje [12] https://ideone.com/4zMvWM:

http://php.net/manual/en/ref.strings.php,

```
<?php
$text = ' Napisy nazywamy łańcuchami znaków (ang. string) .<br /> ';
echo $text;
print strlen($text); // długość
echo '<br />';
$text = trim($text);
echo $text;echo strlen($text); // trim obcina spacje?
echo '<br />';
echo strtoupper($text); // duże znaki
echo strtolower($text); // małe
$text = str_replace('napis', 'Napis', $text); // zamiana
echo $text;  //
print substr($text, 2, 6); // fragment (substring)
var_dump(strpos($text, 'am')); // pierwsze wysątąpienie
var_dump(strpos($text, 'łań')); // jeśli dobrz skonfigurowane - uwzględni
var_dump(strpos($text, 'Droga')); // false
```

Wstawianie zmiennych [13] https://ideone.com/0y5OD9:

```
<?php
$name = 'Jan Kowalski';
print "Łańcuchy znaków w cudzysłowach (\")
ze zmiennymi są przetwarzane
- nazwy zmiennych astępowane ich wartością:
$name<br />";
print 'Łańcuchy apostrofach (\')- nie: $name<br />';
print 'Backsash (\\) = znak "ucieczki" (escape)';
```

Konkatenacja [14] https://ideone.com/OrcSaJ:

```
<?php
$nazwisko = 'Kowalski';
$imie = 'Jan';
print $nazwisko.' '.$imie; // . = znak konkatenacji
```

Tablice – LISTY i SŁOWNIKI

http://php.net/manual/en/book.array.php

[15] https://ideone.com/91PVqE:

```php
<?php
$lista1 = []; // pusta lista
$lista2 = array(); // to samo
$imiona = ['Robert', 'Adam', 'Magda'];
var_dump($imiona);
var_dump(array('Robert', 'Adam', 'Magda'));
$osoba1 = [ // słownik (mapa, tablica asocjacyjna)
'name' => 'Jan Kowalski', 'plec' => 'M'
];
var_dump($osoba1);
var_dump(array( 'name' => 'Magda Kowalska', 'plec' => 'K'));
$lista1=$imiona;
$lista1[1]='Adaś';
var_dump($lista1);
// modyfikowana (not immutable), zmieniono 2-gi (!) element
var_dump($imiona); // $imiona bez zmian
```

Złożone struktury (16) https://ideone.com/m9ayT1:

```php
<?php
$books = [
  [ 'title' => 'Pan Tadeusz',
    'authors' => 'Adam Mickiewicz'
  ],
  [ 'title' => 'Przedwiośnie',
    'authors' => 'Stefan Żeromski'
  ]
];
var_dump($books);
print $books[1]['title'];
echo '<br />';
print_r($books); // wypisuje strukturę - bez formatowania (jak var_dump)
$books=array_merge(['from' => 'Biblioteczka'],$books); // łączenie strukt
var_dump($books);
$json = json_encode($books); // zamienia na łańcuch znaków w standarzie J
var_dump($json);  // zwróć uwagę na sposób kodowania polskich liter
```

Zmiany tablic

[17] https://ideone.com/7YHAH3:

```php
<?php
$book = [ 'title' => 'Treny',
    'authors' => 'Jan Kochanowski'
  ]; // wstawianie
$books=[];
var_dump(empty($books)); // funkcja empty - pusty
$books[] = $book; // dodawanie
$books[] = $book;
$books[1]['title'] = 'TRENY';
var_dump($books);
$ile=count($books);
print $ile;
unset($books[$ile-1]['title']); // usun wartość (nie element)
var_dump($books);
var_dump(isset($books[$ile-1]['title']));
// isset - funkcja - czy zdefiniowana wartość
```

Przeszukiwanie i porządkowanie tabel

http://php.net/manual/en/array.sorting.php,

Funkcja	Sortuj według klucza / wartości	Słownik	Kolejność
sort	Value	No	Lowtohigh
rsort	Value	No	Hightolow
asort	Value	Yes	Lowtohigh
arsort	Value	Yes	Hightolow
ksort	Key	Yes	Lowtohigh
krsort	Key	Yes	Hightolow

[18] https://ideone.com/RweFih

```php
<?php
$imiona = ['Robert', 'Adam', 'Magda'];
var_dump(in_array('Adam', $imiona)); // true
var_dump(array_search('Adam', $imiona)); // 1
sort($imiona); // sortuje w tym samym miejscu (przek. przez referencje)
var_dump($imiona); // posortowany
```

```php
$books = [
    [ 'title' => 'Pan Tadeusz',
      'authors' => 'Adam Mickiewicz'
    ],
    [ 'title' => 'Przedwiośnie',
      'authors' => 'Stefan Żeromski'
    ]
  ];
arsort($books); // k - klucz, r - rewersm, a - wartosc slownika
var_dump($books);
```

Algorytmy – instrukcje

Warunkowe

[19] https://ideone.com/9VWsjQ

```php
<?php
$a=-1;
echo '$a='.$a;
if ($a > 0) {
    echo " - dodatnia";
} elseif ($a<0) {
    echo " - ujemna";
}
else {
    echo " - zero";
}
```

Switch…case

[20] https://ideone.com/b7tGid

```php
<?php
$a=1;
echo '$a='.$a;
// zamiast wielu if - case .. switch
switch ($a){
    case 0:
    echo " - zero";
    break;
    case 1:
    echo " - jeden";
//    break; - pominięcie break powoduje przejście dalej
    default: // pozostałe przypadki (domyślnie
    echo " - reszta";
}
```

Pętle for, while, do … while

[21] https://ideone.com/0Yo8IH

```php
<?php
$i = 1;
while ($i < 4) {
    echo $i++ .' '; // drukuje 1 2 3
}
echo "<br />wstecz:<br />";
do {
    echo $i;
    $i--;
} while ($i>0); // od 4 do 1
echo "<br />od zera do 3:<br />";
for ($i=0;$i<=4;$i++) {
    echo $i;
}
```

Foreach

[22] https://ideone.com/2trZRo

```php
<?php
$names = ['Robert', 'Adam', 'Magda'];
foreach ($names as $name) { // pobiera kolejne elementy tabel
  echo $name . "<br />";
}
foreach ($names as $key => $name) { // inna forma: klucz => wartość
    echo $key . " -> " . $name . "<br /> ";
}
```

Inne formy zapisu instrukcji

Stosowane głównie w szablonach [23] https://ideone.com/LfEFpf

```php
<?php $names = ['Robert', 'Adam', 'Magda']; ?>

<?php if (!$names): ?>
    <b>Pusta lista</b>
<?php else: ?>
    <ul>
    <?php foreach ($names as $name):?>
        <li><?= $name ?></li>
    <?php endforeach; ?>
    </ul>
```

```
<?php endif; ?>
```

Funkcje

Predefiniowane funkcje używaliśmy w prostszych przykładach (empty,in_array,var_dump). Możemy definiować własne funkcje[24] https://ideone.com/XQG3Eo.

```php
<?php
function fullName($osoba) {
    return $osoba['imie'].' '.$osoba['nazwisko']; // return - zwraca wyni
}

$osoby = [
    ['imie' => 'Jan',
     'nazwisko' => 'Kowalski'],
    ['imie' => 'Robert',
     'nazwisko' => 'Nowak']
];

// funkcji możemy używać tak - jak wyrażenia
foreach ($osoby as $osoba) print fullName($osoba).'<br />';
// gdy jedna instrukcja - nie musi być {}
```

Argumenty domyślne funkcji [25] https://ideone.com/5I2NJL

```php
<?php
function wartosc($liczba,$system=10) {
    $wartosc=0;
    foreach ($liczba as $cyfra) {
        $wartosc=$system*$wartosc+$cyfra;
    }
    return $wartosc;
}
// $wartość i wartosc() nie mylą się - znak dolara

echo 'W "101" systemie dziesiętnym to '.wartosc([1,0,1]).
     ', ale w systemie dwójkowym to '.wartosc([1,0,1],2);
```

Pliki

http://uk1.php.net/manual/en/book.filesystem.php

[26] https://ideone.com/g4uWJ9

```php
<?php
// prosty przykład - zwraca plik json z książkami
// gdy go nie ma - tworzy przykładowy
$nazwa_pliku = __DIR__.'/books.json';

if (! file_exists($nazwa_pliku) ) {
    // file_exists - czy istnieje; jeśli nie - przykładowe dwie książki
    $books = [
        [ 'title' => 'Pan Tadeusz',
          'authors' => 'Adam Mickiewicz'
        ],
        [ 'title' => 'Przedwiośnie',
          'authors' => 'Stefan Żeromski'
        ]
    ];
    file_put_contents( $nazwa_pliku, json_encode($books) );
} else {
    $books = json_decode(file_get_contents($nazwa_pliku), true);
    // json_decode - tworzy strukturę z łańcucha znaków
    // drugi parametr - czy tablica socjacyjna (słownik)
}

// ewentualne dodanie nowej książki
if (is_writable($nazwa_pliku)) file_put_contents( $nazwa_pliku, json_enco
// zwracamy:
echo json_encode($books);
```

PHPa komunikacja w internecie

FormularzeHTML

[27]

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Książki</title>
<style> td {padding: 10px; border:1px solid gray;} </style>
</head>
<body>
<?php
$nazwa_pliku = __DIR__.'/books.json'; // plik z książkami na swerwerze
// ewentualne dodanie nowej książki
if (! file_exists($nazwa_pliku) )  $books=[];
else {
    $books = json_decode(file_get_contents($nazwa_pliku), true);
}
$nowa = isset($_GET['title']) || isset($_GET['authors']);
if ($nowa) {
    $books[] =
        [ 'title' => $_GET['title'],
          'authors' => $_GET['authors']
        ];
    if (is_writable($nazwa_pliku))
      file_put_contents( $nazwa_pliku, json_encode($books) );
}
if ($books) { // zwracam tabelę książek
?>
    <table>
    <?php foreach ($books as $book): ?>
    <tr><td><?= $book['title'] ?></td><td><?= $book['authors'] ?></td></t
    <?php endforeach ?>
    </table>
<?php
}
?>
<form action="ex27.php" method="get">
<!-- metoda get lub post; get - w adresie URL wartości -->
<h3>Nowa książka</h3>
```

```
<div>Autorzy:<input type="text" name="authors" /></div>
<div>Tytuł:<input type="text" name="title" /></div>
<input type="submit" value="Zapisz"/>
</form>
</body>
</html>
```

Protokół HTTP

Sposób przesyłania danych w internecie definiują standardy komunikacyjne (protokoły). Dane w internecie są przesyłane przy użyciu standardu TCP/IP (czasem UDP). Standardy regulują jednak nie tylko sam proces przesyłania danych, ale także to co z tymi danymi robią aplikacje (na przykład inaczej są obsługiwane przesyłane pliki, a inaczej komendy). Dlatego różne standardy mogą regulować komunikację na różnym poziomie (zob. Model Warstwowy). Dla programisty aplikacji najważniejsza jest oczywiście warstwa najwyższa (warstwa aplikacji). Mamy w niej protokoły HTTP (strony internetowe), FTP (pliki), DNS (adresy komputerów) etc… Najczęściej wykorzystywanym standardem (protokołem) komunikacji jest obecnie HTTP (lub z szyfrowaniem: HTTPS). Pierwotnie służył on jedynie do przesyłania stron internetowych oraz danych z formularzy umieszczonych na tych stronach. Obecnie służy także do komunikacji między aplikacjami. Protokół ten jest bezstanowy. Oznacza to, że uzyskane dane nie zależą od wcześniejszej historii (stanu) połączenia. Dlatego gdy potrzebujemy zapamiętać stan komunikacji, stosuje się dodatkowe mechanizmy (sesje i ciasteczka). Adres internetowy z podaniem protokołu nazywa się URL. Na przykład: **https://www.google.com**

MVC

Najprostsza implementacja Model – View – Controler musi składać się z 3 plików [28]:

Model:

```php
<?php

function modelPobierz($nazwa_pliku=__DIR__.'/books.json') {
  if ( file_exists($nazwa_pliku) )  {
    $_books = json_decode(file_get_contents($nazwa_pliku), true);
  }
  return $_books;
}

function modelNowa(&$books,
                   $title, $authors,
                   $nazwa_pliku=__DIR__.'/books.json') {
  $books[] =
        [ 'title' => $title,
          'authors' => $authors
        ];
  return file_put_contents( $nazwa_pliku, json_encode($books) );
}
```

Widok:

```php
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Książki</title>
<style> td {padding: 10px; border:1px solid gray;} </style>
</head>
<body>
<?php
if (count($books)>0) {
?>
    <table>
    <?php foreach ($books as $book): ?>
    <tr><td><?= $book['title'] ?></td><td><?= $book['authors'] ?></td></t
    <?php endforeach ?>
    </table>
```

```
<?php
}
?>
<form action="ex28c.php" method="get">
<!-- metoda get lub post; get - w adresie URL wartości -->
<h3>Nowa książka</h3>
<div>Autorzy:<input type="text" name="authors" /></div>
<div>Tytuł:<input type="text" name="title" /></div>
<input type="submit" value="Zapisz"/>
</form>
</body>
</html>
```

Kontroler:

```
<?php
// Prosta implementacja MVC = model / widok / kontroler
include "ex28m.php"; // model

$books = modelPobierz();
// $books widoczna w widoku
// http://php.net/manual/pl/language.variables.scope.php

// poniżej kontroler (główny moduł)
$nowa = isset($_GET['title']) || isset($_GET['authors']);
if ($nowa) {
  modelNowa($books, $_GET['title'], $_GET['authors']);
}
include "ex28v.php"; // widok
```

Wywołanie

```
<?php
include 'ex28c.php';
```

Użycie szablonów - na przykładzie Smarty

Widok z poprzedniego przykładu możemy zastąpić przez wywołanie szablonu Smarty [29] [https://pl.wikibooks.org/wiki/PHP/Smarty]:

```
<?php
// Smarty http://www.smarty.net
require_once("smarty/libs/Smarty.class.php");
$smarty = new Smarty;
//$smarty->force_compile = true;
$smarty->debugging = true;
$smarty->caching = false;
// $smarty->cache_lifetime = 120;

$smarty->assign('books', $books);
$smarty->assign('title', 'Katalog Książek');
$smarty->assign('charset', 'utf8');

$smarty->display('ex29.tpl');
?>
```

Przykładowy szablon (e29.tpl):

```
{include file="header.tpl"}

<h1> {#title#|capitalize}</h1>
<div>Data {$smarty.now|date_format:"%Y-%m-%d %H:%M:%S"}</div>


<form action="ex29.php" method="get">
<h3>Nowa książka</h3>
<div>Autorzy:<input type="text" name="authors" /></div>
<div>Tytuł:<input type="text" name="title" /></div>
<input type="submit" value="Zapisz"/>
</form>


<table>

{section name=b loop=$books}
{if $smarty.section.b.index is odd by 1}
    <tr class="odd">
    <td>
```

```
    {$books[b].title}
    </td>
    <td>
    {$books[b].authors}
    </td>
  </tr>
{else}
  <tr class="even">
    <td>
    {$books[b].title}
    </td>
    <td>
    {$books[b].authors}
    </td>
  </tr>
{/if}
{sectionelse}
<tr><td> zbiór pusty </td></tr>
{/section}
</tables>
{include file="footer.tpl"}
```

Jedną pożytecznych kwestii w szablonach jest możliwość wydzielenia ich fragentówdo innych plików (tu: header.tpl I footer.tpl).

Header:

```
<HTML>
<HEAD>
<TITLE>{$title}</TITLE>
<meta charset={$charset}" />
<style type="text/css">
{literal}
  td{padding: 10px; border:1px solid gray;} .odd {background-color:lightg
{/literal}
</style>
</HEAD>
<BODY bgcolor="#ffffff">
```

Footer:

```
</BODY>
</HTML>
```

Zwróć uwagę na konieczność ujęcia CSS w sekcji {literal} - umożliwia to użycie nawiasów {}, które w szablonie mają inne znaczenie (literal - nie jest analizowane).

```
<?php
chdir('ex29smarty');
include 'ex29c.php';
```

Cookies i pamiętane dane

Jeśli chcemy, aby formularz był częściowo wypełniony – używamy „ciasteczek” (cookies) [30].

```php
<?php
include "ex28m.php"; // model

$books = modelPobierz();

$nowa = isset($_GET['title']) || isset($_GET['authors']);
if ($nowa) {
  modelNowa($books, $_GET['title'], $_GET['authors']);
  $authors=$_GET['authors'];
  setcookie('authors', $authors, time() + 86400); // na 1 godzinę
} else {
  $authors = htmlspecialchars($_COOKIE['authors']);
}

include "ex30v.php";
```

W pliku widoku (na ex30v - na bazie ex28v.php) zmieniamy jeden wiersz:

```
<div>Autorzy:<inputtype="text"name="authors"value="<?=$authors?>"/></div>
```

Tablica $_COOKIE należy do tak zwanych „superglobals”:

http://php.net/manual/en/language.variables.superglobals.php

- $_GET – z formularza GET
- $_POST – z formularza POST
- $_COOKIE – ciasteczka
- $_SERVER – dane serwera
- $_REQUEST – $_GET, $_POST i $_COOKIE razem

Klasy i obiekty - podstawy

[31]

```php
<?php

class Book { // klasa obiektów
    public $authors, $title; // własności

    // metoda klasy
    // $this  = aktualny obiekt (instancja)
    public function description() {
        return $this->title.', <i>'.$this->authors.'</i>';
    }
}

// obiekty danej klasy
$book = new Book();
$book1 = new Book();

$book->title = "1984";
$book->authors = "George Orwell";
var_dump($book);

$book1 = new Book();
$book1->title = "Pan Tadeusz";
var_dump($book1);

echo $book->description();
echo '<br />';
echo $book1->description();
```

Widoczność (powyżej użyto public):

- private: Tylko dla obiektów danej klasy.
- protected: Także dla klas pochodnych (zob. dalej)
- public: Publiczne (dla wszystkich).

Dziedziczenie [32]

```php
<?php
```

```php
/*
Dziedziczenie zachowuje metody i własności przodka.

Metody magiczne - nie są wywoływane "wprost", ale z chwilą operowania obi

- __construct - gdy tworzymy obiekt
- __destruct - gdy "zabijamy obiekt"
- __clone - przy kopiowaniu
- __toString - gdy tworzymy jego reprezentację w postaci łańcucha znaków
*/

class Book {
    public $authors, $title, $exlibris;

    public function __toString() {
        return $this->title.', <i>'.$this->authors.'</i>';
    }

    public function opis():string {
        return 'ok';
    }
}

class MyBook extends Book {
    public $exlibris;

    public function __construct(){
        $this->exlibris = 'JW';
    }
    public function __toString() {
        if ($this->exlibris)
            return parent::__toString().' ['. $this->exlibris.']';
        else
            return parent::__toString();
    }

}

$book = new Book();
$book1 = new MyBook();

var_dump($book->opis());
$book->title = "1984";
$book->authors = "George Orwell";
$book1->title = "Pan Tadeusz";
```

```
print (string)$book;
// (string) - rzutowanie na łańcuch znaków
print '<br />';
print (string)$book1;
```

Konstruktor jako „metoda magiczna”

Metody i własności statyczne

[33]

```
<?php
/*
Metody statyczne - mogą być używane do włąsności / metod
niezależnych od obiektów (instancji) - dla całej klasy.
Na przykład - jednoznaczny identyfikator.
*/

class Book {

    private static $lastId = 0;

    private static function nextId() {
        return ++self::$lastId;
    }

    public function __construct(string $title, string $authors) {
        $this->id=self::nextId();
        $this->title = $title;
        $this->authors = $authors;
    }
}

$book1 = new Book('Ogniem i Mieczem', 'Henryk Sienkiewicz');
$book2 = new Book('Raport z oblężonego miasta', 'Zbigniew Herbert');

var_dump($book1);
var_dump($book2);

print "<br />book1->id = ".$book1->id;
print "<br />book2->id = ".$book2->id;
```

[34]

```php
<?php
/*
Metody statyczne i encasulation
*/

class camelCase {

    public static function fromUnderscore($ident) {
        if (!$ident) return $ident;
        $fragments = explode("_", $ident);
        var_dump($fragments);
        print lcfirst($fragment[0]);
        for ($i=1;$i<count($fragments);$i++) {
            print ucfirst($fragments[i]);
        }
        var_dump($fragments);
        $result='';
        foreach ($fragments as $f) {
            $result.=$f;
        }
        print 'Result='.$result;
        return $result;
    }

}

echo(camelCase::fromUnderscore('Duze_piwo'));
```

Metody statyczne często stosuje się także do funkcji / zmiennych, które mogłyby być globalne, ale zostały “zamknięte” w klasach (encapsulation).

Wyjątki

[35]

```php
<?php
/*
  throw - zgłoś wyjątek
  chatch - łap wyjątek jaki się zdarzył w bloku try {}
*/

function Koduj(int $cyfra) : int {
  $kod=10-$cyfra;
  if ($kod <= 0) {
    throw new Exception('To nie jest cyfra! ['.$cyfra.']');
  }
  return $kod;
}

$komunikat = [9,2,1,10,9,2];

try {
  for ($i=0; $i<count($komunikat); $i++) {
    $komunikat[$i]=Koduj($komunikat[$i]);
  }
} catch (Exception $e) {
    echo 'Nastąpił wyjątek: '.$e->getMessage();
}
```

Do konstrukcji try {} można dodać wiele sekcji catch (dla różnych wyjątków) oraz finally {} - dla końcowej obsługi (także w sytuacji, gdy wczęsniej obsłużono wyjątek).

```php
<?php
/*
  throw - zgłoś wyjątek
  charch - łap wyjątek jaki się zdarzył w bloku try {}
*/

function Koduj(int $cyfra) : int {
  $kod=10-$cyfra;
  if ($kod <= 0) {
    throw new Exception('To nie jest cyfra! ['.$cyfra.']');
  }
```

```
  return $kod;
}

$komunikat = [9,2,1,10,9,2];

try {
  for ($i=0; $i<count($komunikat); $i++) {
    $komunikat[$i]=Koduj($komunikat[$i]);
  }
} catch (Exception $e) {
    echo 'Nastąpił wyjątek: '.$e->getMessage();
}
```

W PHP dostępne są następujące wbudowane podklasy klasy Exception :

- Exception (http://php.net/manual/class.exception.php),
- ErrorException (http://php.net/manual/class.errorexception.php).

  Biblioteka standardowa PHP (http://php.net/manual/book.spl.php) zawiera następujące podklasy wyjątków:

- LogicException (http://php.net/manual/class.logicexception.php),
- BadFunctionCallException (http://php.net/manual/class.badfunctioncallexception.php) ,
- BadMethodCallException (http://php.net/manual/class.badmethodcallexception.php),
- DomainException (http://php.net/manual/class.domainexception.php),
- InvalidArgumentException (http://php.net/manual/class.invalidargumentexception.php),
- LengthException (http://php.net/manual/class.lengthexception.php),
- OutOfRangeException (http://php.net/manual/class.outofrangeexception.php),
- RuntimeException (http://php.net/manual/class.runtimeexception.php)
  - OutOfBoundsException (http://php.net/manual/class.outofboundsexception.php),
  - OverflowException (http://php.net/manual/class.overflowexception.php),
  - RangeException (http://php.net/manual/class.rangeexception.php),
  - UnderflowException (http://php.net/manual/class.underflowexception.php),
- UnexpectedValueException (http://php.net/manual/class.unexpectedvalueexception.php).

Composer - wprowadzenie

Jedną z zalet otwartych narzędzi programowania takich jak Python czy PHP jest możliwość wykorzystania bibliotek, które są rozwijane przez różnych programistów. Pojawia się w związku z tym problem zależności różnych wersji języka i bibliotek. Zostały stworzone dwa systemy obsługi bibliotek, które rozwiązują ten problem:

- PEAR: http://pear.php.net
- Composer: https://packagist.org

Różnica między nimi jest taka, że Pear jest używany na poziomie systemu operacyjnego, a więc wszystkie aplikacje na danym serwerze uzyskują dostęp do tych samych bibliotek. Composer natomiast działa w obrębie konkretnej aplikacji.

Composer jest skryptem napisanym w PHP o nazwie composer.phar. Można go pobrać ze strony:https://getcomposer.org/. Wywołuje się go poleceniem:

```
php composer.phar
```

Zarówno w systemie Windowa jak i Linux może on zostać zintegrowany z systemem i obudowany skryptem o nazwie composer. Wywołujemy więc po prostu polecenie composer: (lub wybieramy prawym przyciskiem myszy z menu kontektowaego Windows).

Konfigurację Composera zapisuje się w pliku tekstowym, w standardzie json. Można go zainicjować wykonując polecenie:

```
composer init
```

W trakcie inicjowania można wyszukać biblioteki jakie są potrzebne. Na przykład po wyborze yii2 dostajemy plik:

Przykładowy wynik:

```
{
"name": "galicea/test1",
"description": "testowa aplikacja",
"require": {
"yiisoft/yii2": "^2.0"
},
"authors": [
```

```
{
"name": "dev",
"email": "dev@s.pwste.edu.pl"
}
]
}
```

następnie wykonujemy instalację:

```
composer install
```

Paczki są instalowane w podkatalogu **vendor**.

Pełny opis Composera i pliku konfiguracyjnego:http://composer.json.jolicode.com/

Spis najważniejszych kluczy w pliku conposer.json:

name– nazwa pakietu

description– zwięzły opis pakietu

keywords– słowa kluczowe (dla wyszukiwarek)

license– licencja

require– lista wymaganych paczek wraz z ichwersjami.

autoload– definicje dla ładowarki (zob. odrębną część lekcji)

scripts– możliwość rozszerzenia działania composera (zobaczwięcej).

Pełen spis wszystkich możliwy kluczy oraz ich znaczeń znajduje się pod tym linkiem.

Zobacz też:

http://pl.phptherightway.com

http://itcraftsman.pl/composer-czyli-jak-zarzadzac-zaleznosciami-w-php/

Przestrzenie nazw i ładowarki

Przestrzenie nazw

Może się zdarzyć tak, że użyjemy takiego samego identyfikatora w dwóch różnych modułach php. Wówczas może dojść do nieoczekiwanych efektów i trudnych w diagnostyce błędów. Dlatego wprowadzono mechanizm przestrzeni nazw (namespaces). Wybierając przestrzeń nazw w jakiej ma operować nasz moduł, świadomie decydujemy jakie identyfikatory są dla niego widoczne.

Polecenie namespace jest umieszczane zawsze na początku modułu.

Przetestujmy, tworząc dwa moduły:

1) test1.php:

```
<?php
namespace ns1;
class test1 { public function id() { echo 'test1'; } }
```

2) test2.php:

```
<?php
namespace ns2;
class test1 { public function id() { echo 'test2'; }}
```

Stwórzy teraz moduł testowy

```
namespace ns1;
require("test1.php"); require("test2.php");
test1::id();
```

Jego uruchomienie powoduje wypisanie napisu 'test1'.

Jeśli w miejsce ns1 wpiszemy ns2, to wynikiem będzie napis 'test2'.

Zastosowanie przestrzeni nazw

Przestrzenie nazw umożliwiają rozwiązywanie kolizji nazw między komponentami kodu i zapewniają możliwość skrócenia długich nazw. Można stwierdzić, że przestrzeń nazw jest "kontenerem" grupy nazw i innych przestrzeni nazw. Przestrzenie nazw mogą bowiem zawierać inne przestrzenie nazw - tworząc drzewo. Jego korzeniem jest przestrzeń nazw *global* do której należą klasy nie zawarte w przestrzeniach nazw podrzędnych - na przykład `Exception` i `DateTime` .

Nazwy zagnieżdżonych przestrzeni nazw oddzielamy znakiem ukośnika (\). Rozpoczęcie nazwy od ukośnika oznacza, że zaczynamy od przestrzeni zawartej w globalnej przestrzeni nazw. Bez ukośnika odwołujemy się do "bieżącej" (czyli takiej w której znajduje się miejsce odwołania) przestrzeni nazw. Na przykład moduł należący do przestrzeni nazw \Aplikacja może odwoływać się do podrzędnej przestrzeni `Moduly\Logowanie` poprzez powyższą nazwę (zaczynającą się od "Moduly"), albo nazwę "**kwalifikowaną**" `\Aplikacja\Moduly\Logowanie` .

Instrukcja use (wstawiana na początku modułu) pozwala na tworzenie *aliasu* (nazwy skróconej). Dla powyższego przykładu możemy użyć opcjonalnie jednej z dwóch zapisów:

```
<?php
use \Aplikacja\Modul\Logowanie;

$log1 = new Logowanie();
$log2 = new \Aplikacja\Modul\Logowanie();
```

Autoloader

Instrukcja **use** korzysta z wprowadzonego w wersji 5.0 PHP mechanizmu automatycznego ładowania klas (zob.https://pl.wikibooks.org/wiki/PHP/Automatyczne_%C5%82adowanie). Umożliwia on ładowanie modułów zawierających definicję klas bez stosowania **require** lub **include**. Ładowanie w chwili użycia obiektu wykonuje funkcja **__autoload**.

Stwórzmy na przykład moduł definiujący klasę test. Jego nazwa powinna być zgodna z nazwą klasy, czyli test.php:

```
<?php
class test { public function id() { echo 'test'; }}
```

Moduł testowy:

```
<?php
function __autoload($class_name) {
include $class_name . '.php';
```

```
}
$obj = new test();
$obj->id();
```

W wersji php 5.1 wprowadzono możliwość używania wielu „ładowarek”. Informujemy o nich system funkcją **spl_autoload_register()**. Wywołując ją bez parametrów, nakazujemy użycie standardowej ładowarki:

```
spl_autoload_register();
test::id();
```

Jeśli używamy przestrzeni nazw, to standardowa ładowarka zakłada, że umieszczamy moduły w odpowiednich podkatalogach – o nazwie równej nazwie przestrzeni nazw. Możemy też używać innego rozszerzenia, niż php – na przykład class.php:

```
namespace ns1;
set_include_path('.');
spl_autoload_extensions('.class.php');
spl_autoload_register();
test1::id();
```

Przykład własnego autoloadera:

```
define ("CLASS_ROOT", 'p6');
spl_autoload_register(
        function ($class) {
            $filename = CLASS_ROOT.DIRECTORY_SEPARATOR.
                        str_replace('\\', DIRECTORY_SEPARATOR, $class).'.php'
            if (file_exists($filename)) {
                require $filename;
                return true;
            }
            return false;
        }
);
```

Autoloader i Composer

Autoloader jest jedną z funkcji Composera. W sekcji „autoload” pliku **composer.json** definiujemy co ma być ładowane. Używany jest do tego standard PSR-0 lub PSR-4. W obu wypadkach definiowany jest alias, którego możemy używać w miejsce ścieżki do katalogu. Na przykład:

```
use Jaspersoft\Client\Client;
```

wskazuje, że chcemy użyć modułu Client.php z przestrzeni nazw Jaspersoft/Client.

Przykład definicji w pliku composer.json:

```
"autoload": {
"psr-0": { "Album": "module/Album/src/" }
},
```

Composer na podstawie podanych defincji tworzy autoloadera w pliku vendor/autoload.php.

Zend 3

Przykład zastosowania przestrzeni nazw we Frameworku Zend3 (pochodzi z komponentu `Zend\Mvc`):

```
<?php
namespace Zend\Mvc;

/**
 * Main application class for invoking applications.
 */
class Application
{
    // ... class members were omitted for simplicity ...
}
```

W Zend Framework 3 wszystkie klasy należą do przestrzeni nazw najwyższego poziomu *Zend*. Przestrzeń nazw *Mvc*, która jest zagnieżdżona w przestrzeni nazw *Zend*. Wszystkie klasy komponentu Mvc (w tym klasa aplikacji *Zend\Mvc\Application* pokazana w tym przykładzie należą do tej przestrzeni nazw. Do klasy `Application` [`\Zend\Mvc\Application`], można odwołać się używając kwalifikowanej nazwy:

```
<?php
$application = new \Zend\Mvc\Application();
```

Yii1 i composer

Jeśli tworzymy aplikację która nie korzysta z Composera, należy przesunąć katalog vendor i plik composera do miejsca z którego będzie można przyłączać biblioteki. Poniżej mamy przykład dla aplikacji w yii1 (choć dla nowych aplikacji chyba lepiej od razu użyć yii2).

Tworzymy aplikację:

```
vendor/yiisoft/yii/framework/yiic webapp <nazwa aplikacji>
```

Teraz powinniśmy przesunąć pliki konfiguracyjne Composera (wraz z podkatalogiem vendor) do katalogu **protected**. Wówczas w pliku index.php możemy zdefiniować:

```
$yii=dirname(__FILE__).'/protected/vendor/yiisoft/yii/framework/yii.php';
```

Istnieje zminimalizowana wersja yii1, którą można użyć w miejsce dystrybucji standardowej:

https://github.com/square1-io/yii-framework

Wtedy mamy w pliku composer.json:

```
"require": {
"square1-io/yii-framework": "1.1.14"
}
```

Aby wykorzystać inne biblioteki, możemy użyć polecenia **composer require**. Na przykład:

```
composer require clevertech/yii-booster
```

Sekcja „require w naszym pliku composer.json zmieni się na:

```
    "require": {
        "square1-io/yii-framework": "1.1.14",
        "clevertech/yii-booster": "^4.0"
    },
```

Oczywiście musimy odpowiednio skonfigurować aliasy. W pliku konfiguracyjnym config/main.php dodajemy na początku:

```
Yii::setPathOfAlias('booster', dirname(__FILE__).'/../vendor/clevertech/y
```

Dodatkowo:

```
'preload'=>array('log','bootstrap'),
…
'components'=>array(
'bootstrap' => array(
'class' => 'booster.components.Booster'
),
```

Możemy już przetestować kontrolki (widgets) Bootstrapa. Na przykład dodając do views/site/index.php:

```
$this->widget('booster.widgets.TbButton', array(
'buttonType'=>'submit',
'context'=>'primary',
'label'=>'OK',
));
```

Joomla i Composer

Komponenty dla CMS Joomla można tworzyć także z użyciem Composera. Pomocny może być w tym projekt https://github.com/galicea/jcc (plik Composera znajdziesz w katalogu: https://github.com/galicea/jcc/tree/master/src/library).

Autoloader w Yii

Framework Yii daje możliwość używania wielu ładowarek (autoloader) dopiero w wersji 2.0. W wersji 1.0 mamy tylko jedną ładowarkę (używaną przezYii::import()) i jedną przestrzeń nazw (http://www.yiiframework.com/doc/guide/1.0/pl/basics.namespace).

Chcąc używać bibliotek z własnymi przestrzeniami nazw trzeba nieco zmienić moduł yii.php - wykorzystując moduł Gautoloader: https://gist.github.com/mindplay-dk/4234540.

Po tej modyfikacji możemy zaimplementować rozszerzenie do Yii do obsługi raportów Jasperreport:

```
<?php
use Jaspersoft\Client\Client;
class jrs extends CComponent {
public function report($server="http://jasper.pwste.edu.pl:8080/jasperser
                       $username="jasperadmin",
                       $password="<haslo>",
                       $reportUri='/reports/report001') {
  $cl = new Client($server, $username,$password,"");
  $cl->setRequestTimeout(60);
  $report = $cl->reportService()->runReport($reportUri, 'pdf');
  ....
 }
}
```

W pliku konfiguracyjnym main.php możemy zarejestrować biblitekę Jaspersoft:

```
Yii::getAutoloader()->addNamespace('Jaspersoft', 'protected/extensions/jr
```

```
return array(
…
'import'=>array(
'ext.jrs.*',
...
),
'components'=>array(
'jrs'=>array('class'=>'ext.jrs.jrs'),
...
```

Standard PSR-4

Celem ujednolicenia nazewnictwa stosowanego w bibliotekach wprowadzono standard PSR-4 (PSR oznacza PHP Standards Recommendation).

Standard PSR-4 określa zalecaną strukturę kodu, którą musi stosować aplikacja lub biblioteka, aby zagwarantować poprawne działanie autoloadera. Podstawowym wymaganiem jest definicja przestrzeni nazw zgodnie ze schematem:

```
\<Vendor Name>\( <Namespace> )*\<Class Name>
```

- Obszary nazw mogą mieć tyle poziomów zagnieżdżenia, ile potrzeba, ale *Nazwa dostawcy* (Vendor Name) powinna być przestrzenią nazw najwyższego poziomu.
- Przestrzenie nazw powinny być odwzorowane na strukturę katalogów. Każdy ukośnik w definicji przestrzeni nazw (`\`) jest w procesie ładowania plików konwertowany na stałą systemową `DIRECTORY_SEPARATOR` specyficzną dla systemu operacyjnego.
- Uzyskana w powyższy sposób nazwa jest uzupełniana rozszerzeniem *.php* aby uzyskać nazwę pliku w którym zdefiniowano klasę..

Na przykład dla klasy `Dostawca\Biblioteka\Klasa` należy stworzyć następującą strukturę katalogów:

```
/Dostawca
  /Biblioteka
     Klasa.php
```

Ten sposób przekładania przestrzeni nazw na nazwę pliku został wprowadzony w standardzoe PSR-0. nazw plików Co jednak zrobić, jeśli chcemy na rzykład aby w katalogu /Dostawca/Biblioteka pojawił się podkatalog src, a w nim dopiero moduły? To właśnie definiuje standard PSR-4. Wprowadza on powiązania prefiksu ścieżki z nazwą (przestrzenią nazw).

Na przykład tak (fragment pliku composer.json):

```
    "autoload": {
        "psr-4": {
            "Application\\": "module/Application/src/"
        }
    },
```

Teraz autoloader będzie wiedział, że definicję klasy \Application\Module znajdzie w pliku module/Application/src/Module.php.

Przykład autoloadera dla projektu Zend Framework (bez mapy klas tworzonej przez Composera)

```php
<?php

// "Standard" autoloader function.
function standardAutoloadFunc($className)
{
    // Replace the namespace prefix with base directory.
    $prefix = '\\Zend\\Mvc';
    $baseDir = '/path/to/zendframework/zend-mvc/src/';
    if (substr($className, 0, strlen($prefix)) == $prefix) {
        $className = substr($className, strlen($prefix)+1);
        $className = $baseDir . $className;
    }

    // Replace namespace separators in class name with directory separato
    $className = str_replace('\\', DIRECTORY_SEPARATOR, $className);

    // Add the .php extension.
    $fileName = $className . ".php";

    // Check if file exists and is readable.
    if (is_readable($fileName)) {
        // Include the file.
        require $fileName;
    }
}

// Register the autoloader function.
spl_autoload_register("standardAutoloadFunc");
```

Ten autoloader sprawdza po kolei podkatalogi w poszukiwaniu miejsca położenia modułu. Jest to oczywiście rozwiązanie wolniejsze niż niż autoloader z mapą klas. Jednak w trakcie opracowywania nowego kodu dla aplikacji może to być wygodniejsze.

Autoloader i Composer

Tworenie Autoloadera jest jedną z funkcji Composera. W pliku composer.json definiujemy co ma być ładowane. Używany jest do tego standard PSR-0 lub PSR-4. W obydwu wypadkach definiowany jest alias, którego możemy używać w miejsce ścieżki do katalogu.

Przykład (Mailing):

Definicja w pliku composer.json:

```
"require": {
"swiftmailer/swiftmailer": "^6.0"
},
```

Composer na podstawie podanych defincji tworzy autoloadera w pliku vendor/autoload.php.

Aby go użyć – dodajemy na początku skryptu.

Przykład:

```
<?php
require 'vendor/autoload.php';
function msend($par_mail, $adres, $subject, $body) {
$transport = (new Swift_SmtpTransport(
$par_mail['host'], $par_mail['port']))
->setUsername($par_mail['user'])
->setPassword($par_mail['pass']);
$mailer = new Swift_Mailer($transport);
// Create a message
$message = (new Swift_Message($subject))
->setFrom($par_mail['user'])
->setTo([$adres])
->setBody($body)
;
// Send the message
try {
return $mailer->send($message);
} catch (Exception $e) {
return 'Błąd:'.$e->getMessage();
}
}
```

Klasy PHP

PHP obsługuje obiektowy styl programowania (OOP). W programowaniu obiektowym głównym elementem kodu jest *klasa* . Klasa może mieć *właściwości* i *metody* . Na przykład stwórzmy skrypt PHP o nazwie *Person.php* i określmy prostą klasę o nazwie `Person` w tym pliku:

```
<?php

class Person
{
    private $fullName;

    public function __construct()
    {
        // Some initialization code.
        $this->fullName = 'Unknown person';
    }

    public function getFullName()
    {
        return $this->fullName;
    }

    public function setFullName($fullName)
    {
        $this->fullName = $fullName;
    }
}
```

Klasa `Person` powyżej ma prywatną właściwość `$fullName` i trzy metody:

- Metoda `__construct()` jest specjalną metodą o nazwie *konstruktor* . Jest używana do zainicjowania własności klasy.
- `getFullName()` i `setFullName()` są publicznymi metodami używanymi do zrobienia czegoś z klasą.

Po zdefiniowaniu klasy możesz utworzyć *obiekty* tej klasy za pomocą operatora `new` w następujący sposób:

```
<?php

// Instantiate the Person.
```

```php
$person = new Person();

// Set full name.
$person->setFullName('John Doe');

// Print person's full name to screen.
echo "Person's full name is: " . $person->getFullName() . "\n";
```

Interfejsy PHP

W PHP *interfejsy* pozwalają określić, jakie metody powinna mieć klasa, ale bez implementacji tych metod. Przykład (interfejs aplikacji Zend3):

```
<?php
namespace Zend\Mvc;

interface ApplicationInterface
{
    // Retrieves the service manager.
    public function getServiceManager();

    // Retrieves the HTTP request object.
    public function getRequest();

    // Retrieves the HTTP response object.
    public function getResponse();

    // Runs the application.
    public function run();
}
```

Jak widać na powyższym przykładzie, interfejs jest definiowany za pomocą słowa kluczowego `interface`, prawie w ten sam sposób, w jaki definiujesz standardową klasę PHP.

Klasa implementująca interfejs nazywana jest *konkretyzacją*.

Przykład:

```
<?php
namespace Zend\Mvc;

class Application implements ApplicationInterface
{
    // Implement the interface's methods here

    public function getServiceManager()
    {
        // Provide some implementation...
    }

    public function getRequest()
```

```
    {
        // Provide some implementation...
    }

    public function getResponse()
    {
        // Provide some implementation...
    }

    public function run()
    {
        // Provide some implementation...
    }
}
```

Konkretyzację interfejsu definiuje słowo kluczowe `implements` .

Różne mechanizmy dostępu do bazy

1. Mechanizm mysql

Obecnie nie istnieje!!!!

```php
<?php
// ustawienie zmiennych
$user="root";
$password="???";
$database="books";
$host="localhost";
// polaczenie z serwerem bazy danych
mysql_connect($host,$user,$password);
// wybor bazy danych
@mysql_select_db($database) or die( "Unable to select database");
// @ - powoduje wylaczenie kontroli bledów -
// dzieki temu mozemy wyswietlic tylko swój komunikat (die)
$query="select * from book"; // zapytanie
$res=mysql_query($query); // wykonanie zapytania
print mysql_error();
while ($row=mysql_fetch_array($res)) {
print_r($row); print '<br /><br />';
}
mysql_close(); // zamkniecie polaczenia
?>
```

2. Mechanizm mysqli - funkcyjnie

```php
<?php
$user="root";
$password="???";
$database="books";
$host="localhost";
$db=mysqli_connect($host,$user,$password,$database);// polaczenie z serwe
$q="select * from book";
$res=mysqli_query($db,$q); // zapytanie
while ($row=mysqli_fetch_array($res)) {
print_r($row); print '<br /><br />';
```

```
}
mysqli_close($db); // zamkniecie polaczenia
?>
```

3. Mechanizm mysqli – obiektowo

```
<?php
$user="root";
$password="???";
$database="books";
$host="localhost";
$db = new mysqli($host,$user,$password,$database);
if (mysqli_connect_errno()) {
printf("Connect failed: %s\n", mysqli_connect_error());
exit();
}
$query="SELECT * FROM book";
$res=$db->query($query); // zapytanie
while ($row=mysqli_fetch_array($res)) {
print_r($row);print '<br /><br />';
}
$db->close();
?>
```

Mechanizm obiektowo z użyciem pętli foreach i funckji printf

```
<?php
$user="root";
$password="???";
$database="books";
$host="localhost";
$db = new mysqli($host,$user,$password,$database);
if (mysqli_connect_errno()) {
printf("Connect failed: %s\n", mysqli_connect_error());
exit();
}
$query="SELECT * FROM book";
foreach ( $db->query($query) as $row ) {
printf("%s %s \n
<br>Title: %s \n
<br><hr><br>",
$row['title']);
```

```
}
$db->close();
?>
```

4. PDO

```
$dbname='books';
$dbuser="root";
$dbpass="????";
try {
  $db = new PDO("mysql:host=localhost;dbname=books", $dbuser, $dbpass);
} catch(PDOException $e) {
  echo $e->getMessage();
  die(); // kończymy aplikacje
}
$q_insert='insert into book(title, authors) values ("Świtezianka", "Adam
$q_select='select * from book';
$q_delete='delete from book';
try {
  foreach($db->query($q_select) as $row) {
    print_r($row);
  }
} catch (PDOException $e) {
  print "Error!: " . $e->getMessage() . "<br/>";
}
try {
  $db->exec($q_create);
  $result = $db->query($q_insert);
  $result = $db->query($q_select);
  foreach ($result->fetchAll() as $row) {
    print_r($row);
  }
  // $result = $db->exec($q_delete);
} catch (Exceprion $e) {
  echo $e->getMessage();
}
// preparowane
$q = $db->prepare("select * from book where authors=:autor ");
$autor='Adam Mickiewicz';
$q->bindParam(':autor',$autor); //drugi parametr musi byc zmienną!
$q->execute();
foreach ($q->fetchAll() as $row) {
  print_r($row);
}
```