Adam Omelak

ZEND FRAMEWORK

PORADNIK PROGRAMISTY

Helion

Opieka redakcyjna: Ewelina Burska
Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki
Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

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

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/zef3pp
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-3283-6

Printed in Poland.

Spis treści

O autorze **7**

ROZDZIAŁ 1. Wprowadzenie **11**
1.1. Czym jest Zend Framework? 11
1.2. Komponenty 12
1.3. Dlaczego warto wybrać wersję 3.0? 13
1.4. Co nowego w ZF3? 13
1.5. Społeczność 14
1.6. Przykłady 15

ROZDZIAŁ 2. Instalacja **17**
2.1. Wymagane aplikacje 17
2.2. Ustawianie środowiska pracy 18
2.3. Ściągnięcie szkieletu aplikacji 21
2.4. Konfiguracja Zenda 22

ROZDZIAŁ 3. Struktura szkieletu aplikacji **25**
3.1. Pliki konfiguracyjne 25
3.2. Zewnętrzne biblioteki 28
3.3. Konfiguracja modułów 30
3.4. W jaki sposób działa Zend 3? 32

ROZDZIAŁ 4. Prosta aplikacja i workflow **35**
4.1. Wstępna konfiguracja 35
4.2. Modyfikacje w kontrolerze i widoku 38

ROZDZIAŁ 5. Tworzenie nowego modułu **43**
5.1. Dodanie przykładowego modułu 43
5.2. Nowy formularz 45
5.3. Dodawanie rekordów 46
5.4. Edycja rekordu 49
5.5. Usuwanie rekordu 51
5.6. Dodanie modułu przez ZF2Rapid 53
5.7. Co powinno się znajdować w module? 53

ROZDZIAŁ 6. MVC 55
6.1. Model 55
6.2. Widok 58
6.3. Kontroler 61
6.4. Router 63

ROZDZIAŁ 7. Najważniejsze komponenty 71
7.1. Event Manager 71
7.2. Module Manager 77
7.3. Service Manager 79
7.4. Hydratory 87

ROZDZIAŁ 8. Bazy danych 91
8.1. Adaptery — MariaDB, MySQL, PostgreSQL itp. 91
8.2. Proste zapytania 92
8.3. Zapytania CRUD 95
8.4. Table Gateway 97
8.5. Row Gateway 99

ROZDZIAŁ 9. Szablony widoków 101
9.1. Domyślne widoki 101
9.2. Silniki szablonów — Smarty, Twig 104
9.3. Layouty i helpery 109

ROZDZIAŁ 10. Formularze 115
10.1. Generowanie 116
10.2. Walidacja 121
10.3. Filtrowanie 125
10.4. Dekoratory 128
10.5. Wykończenie formularza użytkownika 129

ROZDZIAŁ 11. Tworzenie listy komiksów ze stronicowaniem 137
11.1. Nowy kontroler 137
11.2. Nowy model 139
11.3. Nowy widok 144
11.4. Stronicowanie 146

ROZDZIAŁ 12. Apigility 151
12.1. Ustawienie środowiska 151
12.2. Graficzny panel administracyjny 155
12.3. Tworzenie serwisów RPC 156
12.4. Tworzenie serwisów REST 162
12.5. Zabezpieczenie serwisów 172

ROZDZIAŁ 13. Tworzenie dynamicznej sondy 177
13.1. Nowy kontroler 177
13.2. Nowa biblioteka 182
13.3. Nowe endpointy API 189

ROZDZIAŁ 14. Rejestracja i logowanie 193
14.1. Rejestrowanie użytkowników 193
14.2. Logowanie 207
14.3. Sesja 211

ROZDZIAŁ 15. Tworzenie panelu administratora i CMS 217
15.1. Nowy moduł 217
15.2. Dostęp do panelu 224
15.3. CMS — system zarządzania treścią 225

ROZDZIAŁ 16. Tworzenie systemu zarządzającego dostępami użytkowników 239
16.1. Podstawowe pojęcia 239
16.2. Zabezpieczenia 240
16.3. Strategie 244
16.4. Integracja systemu autoryzacji 245

ROZDZIAŁ 17. Tworzenie modułu do debugowania i logów 251
17.1. Własny moduł debugowania 251
17.2. Obsługa błędów i biblioteka Whoops 262

ROZDZIAŁ 18. Tworzenie obsługi wielu języków 269
18.1. Integracja obiektu MvcTranslate 269
18.2. Dostępne formaty translacji 271
18.3. Dodatkowe klasy i metody lokalizacyjne 274

ROZDZIAŁ 19. Tworzenie formularzy opartych na strukturze Bootstrap TwitterCSS 279
19.1. Przygotowanie formularza i kontrolera 279
19.2. Implementacja formularza w Bootstrap 3 281

ROZDZIAŁ 20. Tworzenie własnej nawigacji i sitemapy 295
20.1. Przerobienie nawigacji 295
20.2. Dodanie patchwaya i linków 301
20.3. Dynamiczna sitemapa 303

ROZDZIAŁ 21. Tworzenie testów jednostkowych 309
21.1. Ustawienie środowiska dla testów 309
21.2. Testowanie modeli Rowset i Fixtures 312
21.3. Testowanie z użyciem baz danych 319
21.4. Testy kontrolera i mocks 324

Skorowidz 331

ROZDZIAŁ 4.

Prosta aplikacja i workflow

4.1. Wstępna konfiguracja

Domyślnie Zend przychodzi już z pierwszym modułem, o nazwie Application. Służy on głównie do wyświetlenia statycznej strony informacyjnej z linkami do modułów dokumentacji czy pomocy. Użyjemy tego właśnie modułu do modyfikacji kodu oraz wprowadzenia zmian, dzięki czemu poznamy sposób działania nowego Zend Framework.

Nasze założenie w tym rozdziale jest takie, aby zaimplementować użycie bazy danych MySQL, a następnie pobrać wartość konkretnego rekordu.

Zacznijmy od stworzenia w MySQL przykładowej tabeli `users`, która będzie miała 3 kolumny: `id`, `username` i `password`. Możemy ją utworzyć przez phpMyAdmin dołączony do pakietu XAMPP bądź też bezpośrednio przez zapytanie SQL w wierszu poleceń MySQL.

```
CREATE TABLE IF NOT EXISTS `users` (
  `id` int(11) NOT NULL,
  `username` varchar(100) NOT NULL,
  `password` char(128) NOT NULL
) ENGINE=InnoDB

ALTER TABLE `users`
 ADD PRIMARY KEY (`id`);
ALTER TABLE `users`
 MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
```

Następnie zaczniemy w głównym katalogu od modyfikacji konfiguracji naszej aplikacji. Do *config/global.php* dodajmy poniższy kod:

```
    'db' => array(
        'driver' => 'Pdo',
        'dsn' => 'mysql:dbname=zend3;host=localhost',
        'driver_options' => array(
```

```
            PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES \'UTF8\''
        )
    ),
    'service_manager' => array(
        'factories' => array(
            'Zend\Db\Adapter\Adapter' => 'Zend\Db\Adapter\AdapterServiceFactory',
        ),
    )
```

natomiast do *config/local.php.dist*, który później zmieni się na *local.php*:

```
    'db' => array(
        'username' => 'root',
        'password' => ''
    )
```

Teraz tylko włączamy tryb deweloperski w wierszu poleceń, aby nasze pliki *.dist* sklonowały się do postaci *.php*:

```
composer development-enable
```

Wszystko, co powyżej robimy, prowadzi do ustawienia hasła i nazwy użytkownika do naszej domyślnej bazy danych w MySQL. Domyślnie w XAMPP-ie nazwa głównego użytkownika to `root`, a hasło nie jest ustawione.

Tak jak w poprzednich wersja Zenda, aby połączyć się z jakąkolwiek bazą danych, musimy określić dane do połączenia. Domyślnie dla XAMPP-a `host` to `localhost`, a nazwa bazy to „zend3" (wewnątrz której już istnieje stworzona przez nas tabela o nazwie `users`). Dalej w kluczu `driver` określamy typ bazy danych: dla MySQL domyślnie jest ustawione `Pdo`. Dodatkowo określiliśmy tutaj typ kodowania znaków naszej bazy, czyli `UTF-8`; jeśli nie określimy w tym miejscu odpowiedniego kodowania, zamiast polskich znaków zobaczymy śmieszne znaczki lub np. *??*. Zwróć uwagę, że w *global.php* nie ma informacji o nazwie użytkownika i haśle, dlatego że podany plik nie powinien zawierać żadnych poufnych informacji. Wszystkie podobne informacje konfiguracyjne powinniśmy zamieszczać w *local.php*. Wynika to z systemu kontroli wersji, gdzie *global.php* jest wysyłany do repozytorium, a *local.php* dodawany do plików ignorowanych przez GIT. Dzięki temu nigdy „nie podzielisz się" swoimi prywatnymi hasłami z innymi osobami, jeśli przez przypadek wyślesz coś na publiczny serwer kontroli wersji.

Oprócz danych do bazy dodajemy również rekord o nazwie `service_manager`, który określa zależności naszego modułu od innych klas. Opcja `factories` oznacza, że będziemy tworzyć nową instancję klasy Zend\Db\Adapter\Adapter (czyli praktycznie tej samej klasy co w Zend 1). Zostanie ona utworzona z parametrem drivera Zend\Db\ AdapterAdapterServiceFactory. Dzięki temu nie musimy się przejmować tworzeniem nowych obiektów Zenda w naszych kontrolerach czy modelach. Taka klasa będzie już dostępna do użycia w innych plikach konfiguracyjnych, o których zaraz wspomnimy.

Następnym krokiem, do skonfigurowania tym razem samego modułu, jest dodanie poniższego kodu do *module/Application/config/module.config.php*:

```
'controllers' => [
        'factories' => [
            Controller\IndexController::class => function($sm) {
                $usersService = $sm->get('Application\Model\UsersTable');

                return new Controller\IndexController($usersService);
            }
        ],
    ],
```

Klucz o nazwie `controllers`, jak sama nazwa wskazuje, określa wszystkie kontrolery dostępne w module. Ponieważ będziemy musieli pobrać i wyświetlić rekord z tabeli `users`, musimy mieć dostęp do nowego modułu (nazwanego `UsersTable`). Przekazujemy więc obiekt naszego modelu `$usersService` przez konstruktor *Application/src/Controller/IndexController.php*. Jedyna „magia" w tym kodzie to pobranie `UsersTable` przez `$sm` Service Managera, który jest zawsze dostępny jako pierwszy parametr, poprzez metodę `get()`. Oczywiście, obecnie Service Manager nie ma naszego modelu, który nawet jeszcze nie istnieje. Aby dodać naszą klasę do SM, potrzebujemy dodatkowego kodu w pliku *module/Application/src/Module.php*:

```
public function getServiceConfig()
{
    return array(
        'factories' => array(
            'UsersTableGateway' => function ($sm) {
                $dbAdapter = $sm->get('Zend\Db\Adapter\Adapter');

                $resultSetPrototype = new ResultSet();
                $resultSetPrototype->setArrayObjectPrototype(new User());
                return new TableGateway('users', $dbAdapter, null,
                ↪$resultSetPrototype);
            },
            'Application\Model\UsersTable' =>  function($sm) {
                $tableGateway = $sm->get('UsersTableGateway');
                $table = new UsersTable($tableGateway);

                return $table;
             }
        )
    );
}
```

Jak możesz sam zauważyć, dodaliśmy do pliku `Module` nową metodę, `getServiceConfig()`, która określa dodatkową konfigurację naszych wewnętrznych komponentów, jak modele, formularze czy zwykłe obiekty.

Aby powyższa metoda zadziałała, musimy również dodać linki do klasy (tuż pod definicją `namespace`), do której się odwołujemy:

```
use Application\Model\User;
use Application\Model\UsersTable;
use Zend\Db\ResultSet\ResultSet;
use Zend\Db\TableGateway\TableGateway;
```

Na początku definiujemy `UsersTableGateway`, która będzie obiektem zendowskim `TableGateway` nakierowanym na tabelę o nazwie `users`. Będzie on zwracał obiekty typu *Application/src/Model/User.php*, w których umieścimy metody takie jak `getId()` czy `getUsername()`. Zauważmy, że dzięki wcześniejszej deklaracji klasy Zend\Db\Adapter\Adapter w pliku *config/global.php* mamy tutaj od razu dostęp do instancji tej klasy przez Service Managera.

Następnie konfigurujemy Application\Model\UsersTable poprzez pobranie właśnie zdefiniowanego `UsersTableGateway` oraz zwrócenie obiektu klasy `UsersTable`. Teraz wystarczy już stworzyć odpowiednie pliki i zmodyfikować kontroler.

4.2. Modyfikacje w kontrolerze i widoku

Zacznijmy może od dwóch nowych klas, `User` oraz `UsersTable`.

Pierwsza reprezentuje pojedynczy rekord w bazie. Jeśli więc nasza tabela składa się z 3 kolumn, to powinniśmy mieć przynajmniej 3 metody: `getId()`, `getUsername()`, `getPassword()` plus dodatkowa metoda `exchangeArray($row)`, która przekonwertuje zwróconą tablicę asocjacyjną na zmienne klasowe. Klasa `User` z pliku *Application/src/Model/User.php* wygląda następująco:

```
namespace Application\Model;

class User
{
     public $id;
     public $username;
     public $password;

     public function exchangeArray($row)
     {
         $this->id     = (!empty($row['id'])) ? $row['id'] : null;
         $this->username = (!empty($row['username'])) ? $row['username'] : null;
         $this->password  = (!empty($row['password'])) ? $row['password'] : null;
     }

     public function getId() {
         return $this->id;
     }

     public function getUsername() {
         return $this->username;
     }

     public function getPassword() {
         return $this->password;
     }
}
```

Stwórzmy jeszcze plik klasy *UsersTable.php*, który zapiszemy w tym samym miejscu co *User.php*.

```
namespace Application\Model;

use Zend\Db\TableGateway\TableGateway;

class UsersTable
{
    public function __construct(TableGateway $tableGateway)
    {
         $this->tableGateway = $tableGateway;
    }

    public function getById($id)
    {
         $id  = (int) $id;
         $rowset = $this->tableGateway->select(array('id' => $id));
         $row = $rowset->current();

         if (!$row) {
             throw new \Exception('nie znaleziono użytkownika o id: '.$id);
         }
         return $row;
    }
}
```

`UsersTable` to nasz model, który odpowiada za komunikację z bazą danych. W tej klasie może mieć wszystkie operacje CRUD, takie jak `create()`, `replace()`, `update()` czy `delete()`, oraz — tak jak w powyższym przykładzie — `getById()`, `getByUsername()` i tak dalej. Jeśli pamiętasz plik *Module.php*, to pewnie już wiesz, że konstruktor w tej klasie oczekuje obiektu typu `TableGateway`. Dla przypomnienia:

```
$table = new UsersTable($tableGateway);
```

W taki bowiem sposób stworzyliśmy obiekt, który jest odbieralny w `__construct()`.

Wewnątrz jedynej metody `getById()` przekazujemy numer `id`, jakiego rekordu poszukujemy. Reszta to już zwykłe użycie Zend Db, z którego wywołujemy metodę `select()` z parametrem `id`. Dzięki `$rowset->current()` pobieramy pierwszy wynik zapytania. Jeśli jest on pusty, wyrzucamy wyjątek, w przeciwnym razie zwracamy pojedynczy obiekt typu Application\Model\User.

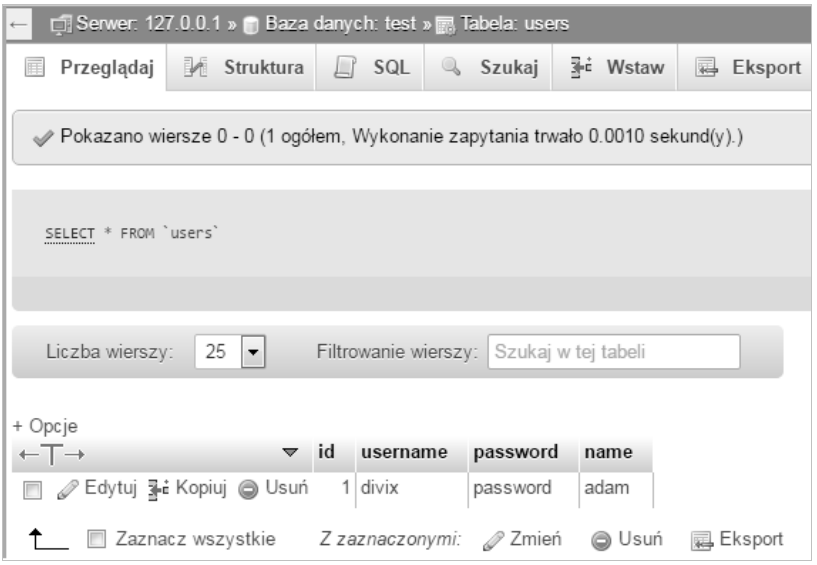
Przejdźmy teraz do kontrolera `IndexController`. Należałoby połączyć go z modelem oraz widokiem, w którym wyświetlimy dane o użytkowniku; wygląda on następująco:

```
namespace Application\Controller;

use Zend\Mvc\Controller\AbstractActionController;
use Zend\View\Model\ViewModel;
use Application\Model\UsersTable;

class IndexController extends AbstractActionController
{
```

```
    private $usersTable = null;
    public function __construct(UsersTable $usersTable)
    {
         $this->usersTable = $usersTable;
    }

    public function indexAction()
    {
        $view = new ViewModel();
        $model = $this->usersTable;
        $row = $model->getById(1);

        $view->setVariable('id', $row->getId());
        $view->setVariable('username', $row->getUsername());
        $view->setVariable('password', $row->getPassword());

        return $view;
    }
}
```

Pierwszą rzeczą, jaką wykonujemy, jest definicja klasowej zmiennej naszego modelu `$userTable`, którą otrzymamy w konstruktorze kontrolera. Dalej modyfikujemy metodę `indexAction()`, która teraz wywoła metodę `getById()` z `UsersTable` z parametrem 1. W tym momencie powinniśmy dodać jeden rekord w bazie MySQL, w tabeli o nazwie `users`: wpis z numerem identyfikacyjnym `ID = 1` i jakimiś wstępnymi danymi. W moim przypadku zrzut danych tabeli `users` wygląda następująco:

Rysunek 4.1.



Wracając do kontrolera, wywołujemy funkcję `setVariable()` z obiektu `new ViewModel()`, która wyśle nam jedną zmienną do pliku widoku *index.phtml*. Metoda ta przyjmuje najpierw nazwę zmiennej w widoku, a później samą zmienną. Zmienna `$row`, która zostaje zwrócona przez `getById()`, jest naszym obiektem Application\Model\User.

Dzięki temu jesteśmy w stanie wywołać jej metody, takie jak `getId()` czy `getUsername()`. Powtarzamy analogicznie pierwszą linię z `id` dla `username` oraz `password`.

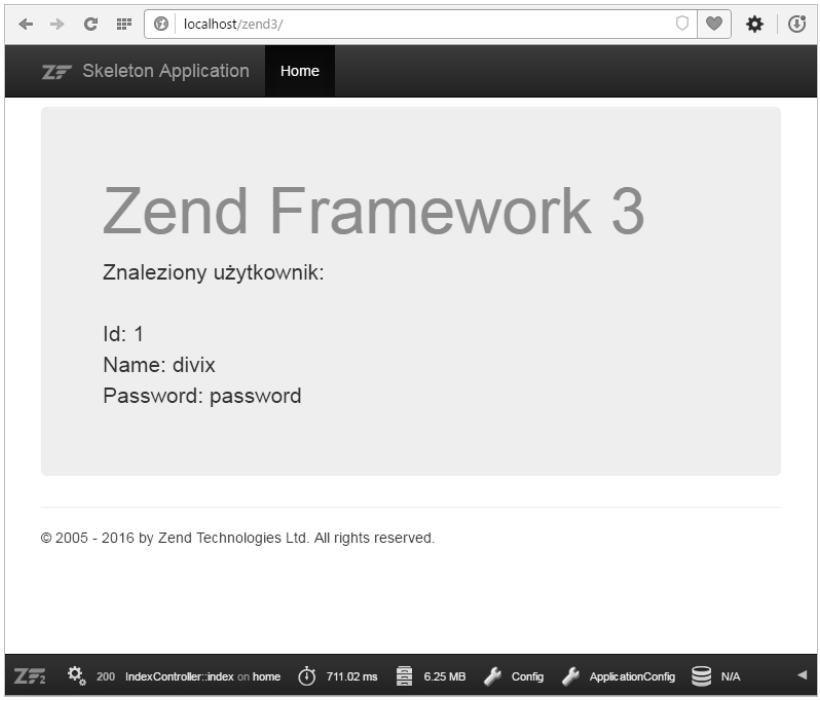
Ostatnie zadanie to wyświetlenie danych o użytkowniku w widoku *Application/view/application/index/index.phtml*:

```
<div class="jumbotron">
    <h1><span class="zf-green">Zend Framework 3</span></h1>
    <p>
        Znaleziony użytkownik:<br /><br />
        Id: <?php echo $id; ?><br />
        Username: <?php echo $username; ?><br />
        Password: <?php echo $password; ?>
    </p>
</div>
```

Możesz tutaj swobodnie zignorować znaczniki kodu HTML; interesują nas jedynie zapiski typu `<?php echo $username; ?>`. Są to nazwy kluczy, które przekazaliśmy w kontrolerze.

Jeśli wszystko dobrze wykonałeś, Twoim oczom powinna ukazać się poniższa strona bez jakichkolwiek błędów.

Rysunek 4.2.



Skorowidz

A

Active Record, 99
adapter, 91, 98
  autoryzacyjny, 246
  bazy danych, 203, 322
  budowa, 92
  DB, 246
administrator, 239, 241
  panel, 217
    dostęp, 224
    widok, 217, 221
adres
  Uri, 125
  URL, 120, 140, 240, 303
  WWW, 151
algorytm
  MD5, 195
  SHA1, 195
  SHA2, 195
Apigility, 151
  dokumentacja, 152
  konfiguracja, 151, 152, 153, 154
  UI, 152
aplikacja
  debugowanie, *Patrz:* debugowanie
  mobilna, 151
  NodeJS, *Patrz:* NodeJS
  testowanie, *Patrz:* test
architektura
  MOVE, *Patrz:* MOVE
  MVC, *Patrz:* MVC
atak
  CSRF, 120
  SQL injections, 93
autoryzacja, 194, 239, 240, *Patrz też:* zabezpieczenie
  Http Basic Authorisation, 172
  integracja, 245, 246
  przekazywanie, 243
  typ, 173, 174

B

baza danych
  adapter, 203, 322
  kodowanie, 36
  Maria DB, 91
  oparta na RDMS, 252
  połączenie, 36
  rekord, 124
    dodawanie, 46, 57, 95
    edycja, 49, 50, 57, 95
    usuwanie, 51, 95, 97
    zastępowanie, 95
  testowych, 318
  wzorzec tabel, 97
  zarządzanie, 43
biblioteka
  BjyAuthorize, 239
  Bootstrap, 279, 287
  DivixUtils, 179, 181, 182
  PHPUnit, 309
  Prophecy, 327
  punkt startowy, 54
  Redistribution package, 18
  Whoops, 265
  zewnętrzna, 29
  ZF Development Mode, 152
  ZFC RBAC, 239
    konfiguracja, 245
    strategia, *Patrz:* strategia
błąd, 253, 260, 261, 263, 265
  404, 262, 263
  500, 262
  komunikat, 265
breadcrumbs, *Patrz:* patchway

C

CAPTCHA, 120
checkbox, 119
ChromePHP, 252
ciąg znaków, 125, 127
CKEditor, 217
CMS, 225
Composer, 20, 28
Container, *Patrz:* pojemnik
CRUD, 95, 228

D

data, 120, 124
debugowanie, 251, 254, 260
  błąd, 260, 261
dekorator, 128, 129

E

edytor programisty, *Patrz:* IDE
email, 120, 251
endpoint, 151, 155
  REST, *Patrz:* REST
  RPC, *Patrz: RPC*
  testowanie, 160
event
  dispatch, 110
  renderer, 110
  response, 110

F

fabryka, 115
fieldset, 115, 117, 123, 203
filtr, 125, 127, 252, 253
FirePHP, 252, 253
Fixture, 318
format
  .phtml, 101
  CSV, 151
  GetText, 271, 272
  INI, 271
  JSON, 26, 151, 177
  XML, 26, 151, 253
formater, 253
formularz, 37, 111, 115, 129, 199, 203, 279, 281
  atrybut, 46, 48
  błąd, 222, 286, 287
  dekorator, *Patrz:* dekorator
  filtrowanie, *Patrz:* filtr
  implementacja w Bootstrap, 281
  kod HTML, 284
  struktura, 128
  tworzenie, 45, 115, 116
  walidacja, 118, 121, 123, 125, 205
  wyświetlanie, 48
funkcja, *Patrz też: metoda*
  haszująca, 194
  parametryzacja, 54

G

GIT, 153
gość, 240, 245, 300
Gravatar, 111

H

hasło, 120, 195
  haszowanie, 195
    sól, 195, 196, 200
helper, 45
  CurrencyFormat, 274, 275
HTML, 111
HTTPie, 160
hydrator, 48, 115, 162, 199, 200

I

IBAN, 124
IDE, 17
ISBN, 124

K

karta kredytowa, 124
klasa dynamiczna, 59
klucz
  adapter, 143
  charset, 59, 92
  controller_plugins, 30
  controllers, 30, 37, 303
  Database, 92
  default, 298
  driver, 36
  Driver, 92
  factories, 282, 299
  filters, 30, 31
  font, 202
  form_elements, 30, 31
  height, 203
  hostname, 92
  http-equiv, 59
  hydrators, 30
  imgDir, 203
  imgUrl, 203
  input_filters, 30, 31
  invokable, 31
  itemprop, 59
  lineNoiseLevel, 203
  log_processors, 30
  log_writers, 30
  name, 59
  navigation, 296
  Password, 92
  plugins, 143
  port, 92
  property, 59
  redirect_when_connected, 245
  route, 66, 298
  route_manager, 30
  router, 63
  serializers, 30
  service_manager, 30
  services, 31
  timeout, 202
  translate, 271
  uri, 298
  Username, 92
  validators, 30
  view_helpers, 30, 270
  width, 203
  wordLen, 202
kod kreskowy, 124
komenda composer update, 28
kontroler, 37, 40, 50, 61
  ciężki, 54
  IndexController, 39
  lekki, 54
  tworzenie, 43, 44

L

layout, 109
lista rozwijana, 119, 147

M

manager
  ControllerManager, 30
  ControllerPluginManager, 30
  FilterManager, 30
  FormElementManager, 30
  HydratorManager, 30

InputFilterManager, 30
LogProcessorManager, 30
LogWriterManager, 30
RoutePluginManager, 30
SerializerAdapterManager, 30
ServiceLocator, 30
ValidatorManager, 30
ViewHelperManager, 30
Memecache, 13
menu
główne, 295
pojemnik główny, 299
poziom zagnieżdżenia, 299
metadane, 111, 228
metoda
__call, 59
__invoke, 59, 125
_initFrontController, 63
addChild, 102
addContent, 228
addLangContent, 228
addMetadata, 228
addPage, 228
addProcessor, 252
addWriter, 252
append, 60
appendName, 59
assertQuery, 325
assertQueryContentContains, 325
assertQueryCount, 325
assertXpathQueryCount, 325
assignContentToPage, 228
authenticate, 196, 209
between, 97
bind, 133
bootstrapForm, 283
buildSqlString, 94
captureEnd, 61, 113
captureStart, 61, 113
column, 95
configureServiceManager, 326
contentSent, 62
delete, 55, 97, 99, 100
deleteContent, 228
deletePage, 228
deleteRow, 57
deleteWith, 55
detectVersion, 62
dispatch, 61, 325, 326
displayMessages, 254
dump, 254, 257, 258, 261
equalTo, 96
escapeCss, 103
escapeHtml, 103
escapeHtmlAttr, 103
escapeJs, 103
escapeUrl, 103
exchangeArray, 47
execute, 93, 126
executeDelete, 260
executeInsert, 260
executeUpdate, 260
expression, 97
extract, 200
fetch, 93
fetchAll, 92, 141, 143
fetchOne, 92
fetchRow, 141
filter, 125, 126
formElementError, 128
formInput, 128
formRow, 128
from, 93, 94, 97
generatePassword, 196
get_object_vars, 142
getAllContentsByPageID, 230
getArrayCopy, 57
getArticleContentByPage
↪Name, 230
getArticleContentByUrl, 230
getAutoloaderConfig, 219
getavatarUrl, 56
getBy, 131
getByUserId, 131
getConfig, 219
getControllerConfig, 30
getControllerPluginConfig, 30
getCookie, 62
getEvent, 61
getEventManager, 61
getFiles, 62
getFilterConfig, 30
getForm, 188
getFormElementConfig, 30
getHeaders, 62
getHydratorConfig, 30
getInputFilter, 49
getInputFilterConfig, 30
getInputFilterSpecification, 280
getLanguageList, 231
getLogProcessorConfig, 30
getLogWriterConfig, 30
getLongMessage, 254
getMediumMessage, 254
getMessage, 188
getPage, 228
getPages, 228
getPassword, 113
getPluginManager, 61, 126
getPost, 62
getQuery, 62
getRealm, 113
getRequest, 61
getResponse, 61
getRoles, 247, 248
getRouteConfig, 30
getSerializerConfig, 30
getServiceConfig, 30, 31
getSqlString, 260
getStaticContentByPageName, 230
getTable, 97
getUploadedImages, 56
getUsername, 113
getValidatorConfig, 30
getVersion, 62
getView, 111
getViewHelperConfig, 30
greaterThan, 96
greaterThanOrEqual, 96
headersSent, 62
headScript, 60
htmlify, 301
htpasswd, 173
hydrate, 200
identity, 113
in, 97

metoda
init, 177
inlineScript, 60
insert, 97, 99
invoke, 283, 285
isActive, 300
isContentExists, 228
isDelete, 62
isFlashRequest, 62
isGet, 62
isNotNull, 97
isNull, 97
isPageExists, 228
isPost, 62
isPut, 62
isValid, 122, 124
isXmlHttpRequest, 62
join, 94
json_encode, 142
lessThan, 96
lessThanOrEqual, 96
like, 96
limit, 95
literal, 96
log, 251
mockUsersTable, 327
nest, 96
notBetween, 97
notEqualTo, 96
notIn, 97
notLike, 96
offset, 95
offsetGetName, 59
offsetSet, 60
onBoostrap, 244
onBootstrap, 211, 220
onDispatch, 61, 224
onError, 244
order, 95
phpunit, 314
populateValues, 133
POST, 46
predicate, 97
prepareStatementForSql ↪Object, 94
prepend, 60
previewAction, 233
prophesize, 327
query, 93
render, 112, 284, 285, 291
save, 55, 99, 100
saveRow, 57
seecontentsAction, 233
select, 55, 93, 97, 98
selectWith, 55, 260
send, 62
sendContent, 62
sendHeaders, 62
serialize, 143
set, 60
setAllowOverride, 326
setAttribute, 46, 48
setData, 47
setEventManager, 61, 178
setFormatter, 252
setIdentifier, 254
setInputFilters, 47
setMaxDepth, 299
setMessageCloseString, 223
setMessageOpenFormat, 222
setPartial, 300
setPluginManager, 126
setTemplate, 102
setTerminal, 103
setUiClass, 299
setup, 314
setView, 111
sha512, 196, 200
spl_object_hash, 142
start, 213
stripslashes, 231
unnest, 96
update, 55, 97, 99
updateMetadata, 228
updatePage, 228
updateWith, 55
values, 95
where, 94, 96, 97, 141, 228
mock, 322, 324, 325, 326, 327
model, 37, 39, 55, 139
ciężki, 54
lekki, 54
MongDB, 253
MongoDB, 252
MOVE, 32
MVC, 32, 61, 111, 240
MySQL Workbench, 17

N

narzędzie
bower, 153
grunt, 153
nawigacja, 295
link HTML, 302
patchway, *Patrz:* patchway
pomocnicza, 299
pozycja aktywna, 296, 300
priorytet, 307
skalowalność, 296
NPM, 152
numer IP, 124

O

obraz, 120

P

page, *Patrz:* strona
paginacja, *Patrz:* widok paginacji
patchway, 301
plik
.phtml, 101
404.phtml, 262, 263
application.tests.config.php, 319
composer.json, 28, 310
config/global.php, 35
config/local.php.dist, 36
config/Module.config.php, 30
CSS, 111, 152
delete.phtml, 51
edit.phtml, 49
format, *Patrz:* format
global.php, 295, 296, 299, 310
INI, 272
JS, 152
konfiguracyjny, 25, 27
layout.phtml, 296
local.php, 36
MO, 273
module.config.php, 63, 131, 271
modules.config.php, 151

phpunit.xml, 310
phpunit.xml.dist, 310
PO, 272
POT, 272
public/index.php, 32
szablonu, 104
szablonu tłumaczeń, *Patrz:* plik POT
widoku, 40, 47, 49, 102
XML, 303
zfc_rbac.global.php.dist, 245
pojemnik, 295, 300
domyślny, 298
główny, 299
pole
tekstowe, 120
wyszukiwania, 120
polityka zabezpieczeń, 240
Postman Launcher, 154, 160
Protection Policy, *Patrz:* polityka zabezpieczeń
przycisk, 120

R

RBAC, 239, 247
strategia, *Patrz:* strategia
typ używania, 240
refaktoryzacja, 54
reguła
dodawanie, 63
zarządzanie, 64
REST, 155
autoryzacja, 172, 173, 174
tworzenie, 162, 163, 166
RESTClient, 160
router, 63
Hostname, 65
Literal, 66
logowania, 207
Method, 67
Regex, 67
Scheme, 68
Segment, 68
RPC, 155
opcje, 161
testowanie, 160, 161
tworzenie, 156, 157, 158, 189

S

serwer, 124
silnik szablonów, 104, 107
Smarty, 104, 106, 113
sonda, 180, 183
aktywna, 187
głosowanie, 188
lista, 186
standard RFC-3164, 252
strategia przekierowania, 244
RedirectStrategy, 244, 246, 247
UnauthorizedStrategy, 244, 245
strona, 295
mapy, 303
podstrona, 298
strumień danych PHP, 252
Syslog, 252
system
CMS, *Patrz:* CMS
debugowania, *Patrz:* debugowanie
translacji tekstów, 269, 270, 272
format, 271
plugin, 274, 275, 276, 277
szyfrowanie jednostronne, 195

Ś

ścieżka
absolutna, 127
bazowa, 56, 58

T

tabela, 35, 97
tag HTML, 111
Template Engine, *Patrz:* silnik szablonów
test, 309
aktualizacji użytkownika, 329
funkcyjny, 322
imitowanie zależności, 324
integracyjny, 322
jednostkowy, 309, 310, 321
modelu
Fixtures, 312, 317, 318
Rowset, 312, 313, 314, 316, 317
parametry, 317
uruchamianie, 311
usunięcia użytkownika, 330
z użyciem baz danych, 319
translator, 269, 270, 271, 272
dat, 274, 276
liczby mnogie dla określonego języka, 274, 277
walut, 274, 275
trigger, 54
Twig, 107, 108, 113

U

użytkownik, 113, 199
administrator, *Patrz:* administrator
edytor, 239
gość, *Patrz:* gość
hasło, *Patrz:* hasło
logowanie, 207, 210
ranga
admin, 241
tworzenie, 245
rejestrowanie, 193
sesja, 211, 212, 214, 223
zalogowany, 244, 247, 300

V

view helper, 128, 129

W

walidator, 124, 125
warstwa
API, *Patrz:* Apigility
kontrolera, *Patrz:* kontroler
modelu, *Patrz:* model
widoku, *Patrz:* widok
wiadomość, 112
widok, 39, 44, 58, 102, 111
definicja układu, *Patrz:* layout
klasa pomocnicza, 279, 283

warstwa
paginacji, 146, 147, 149
link, 148
panelu administratora, *Patrz:* administrator panel widok
szablon, 101
tworzenie, 144
wiersz poleceń, 152
wyrażenie regularne, 67, 125

X

XAMPP, 20, 193, 266
instalowanie, 18

Z

zabezpieczenie, 240
ControllerGuard, 242, 243
ControllerPermissionsGuard, 242
RouteGuard, 240
RoutePermissionGuard, 241, 243
zapytanie, 62, 92, 154
adres, 162
AJAX, 151, 181
CRUD, 95
DELETE, 163
GET, 155, 163
mapowanie, 65
parametr, 93, 141
PATCH, 163
POST, 155
przekierowanie, 64
PUT, 155, 163
SELECT, 260
typ, 155
zdarzenie
EVENT_DISPATCH_ERROR, 244
Zend
instalowanie, 21
komponent, 11
konfiguracja, 22, 25, 35
dodawanie, 26
moduł, 30
plik konfiguracyjny, *Patrz:* plik konfiguracyjny
plugin, 11
społeczność, 14
tryb deweloperski, 25
walidator, 124, 125
wersja, 13
Zend ACL, 239
Zend Authentication, 12
Zend Captcha, 193
Zend Certified Engineer, 14
Zend Code, 154
Zend Config, 12
Zend Crypt, 12
Zend Debug, 254
Zend Dom, 325
Zend DOM, 12
Zend Event Manager, 12, 62, 178
instancja, 61
Zend File, 12
Zend Filter, 154
Zwnd FlashMessenger, 222
Zend Form, 115, 120
Zend Framework, *Patrz:* Zend
Zend Hydrator, 154
Zend InputFilter, 154
Zend Log, 251
Zend Mail, 12
Zend Module Manager, 13
Zend Monitor, 252
Zend MVC, 12
Zend MVC-i18n integration, 12, 269
Zend Navigation, 295, 301, 302
Zend Paginator, 139
Zend Plugin Manager, 111, 112, 113
Zend Router, 32
Zend Row Gateway, 99
Zend Serializer, 139
Zend Service Manager, 12, 13, 32, 56, 243
Zend SQL Abstraction, 93
Zend Table Gateway, 97
Zend Toolbar, 12
Zend Validator, 12, 154
Zend View, 12, 101, 103, 105, 113
Zend Xml2Json, 177, 186
ZendApplication, 35
ZF2Rapid, 53
ZFC Admin, 217
ZFTool, 53

Ż

żądanie nieautoryzowane, 244

ROZPRACUJ ZEND FRAMEWORK
— POZNAJ ŚRODOWISKO STWORZONE PRZEZ TWÓRCÓW PHP!

Jeśli zetknąłeś się już kiedyś z programowaniem w języku PHP, wiesz, że jest on niezastąpiony przy tworzeniu stron internetowych i aplikacji sieciowych. Wiesz także, że istnieją frameworki znacznie ułatwiające generowanie kodu w tym języku. Jednym z nich, być może najbardziej przydatnym i wygodnym w użyciu, jest Zend — środowisko pracy stworzone m.in. przez Matthew Weiera O'Phinneya i Enrica Zimuela. Kto lepiej niż twórcy PHP rozumie potrzeby programistów i mógłby zaprojektować środowisko idealnie dostosowane do charakteru ich pracy? Z pewnością nikt — i dlatego właśnie nadszedł czas, byś nauczył się w pełni korzystać z dobrodziejstw Zend Framework.

Z tej książki dowiesz się, jak działa ZF i jak używać jego komponentów: Zend Form (do generowania i obsługi formularzy), Zend Session (do kontrolowania sesji użytkownika) czy też Zend DB (do komunikacji z bazą danych). Zobaczysz także, dlaczego warto korzystać z całego pakietu Zend Framework, który oferuje pełną integrację komponentów, a ponadto zawiera moduł o nazwie MVC, znacznie skracający czas tworzenia własnych aplikacji. Ponadto nauczysz się radzić sobie z konkretnymi problemami programistycznymi, z debugowaniem i testowaniem aplikacji, z widokami, z tworzeniem kont użytkownika i panelu administratora oraz z tysiącem innych rzeczy — szybko, wydajnie i bez stresu.

- **Wypróbuj Zend Framework 3! Instalacja, struktura szkieletu aplikacji, prosta aplikacja i Workflow**
- **Tworzenie nowego modułu i wykorzystanie MVC**
- **Najważniejsze komponenty i bazy danych**
- **Szablony widoków, formularze i lista komiksów ze stronicowaniem**
- **Apigility oraz tworzenie dynamicznej sondy**
- **Rejestracja i logowanie, tworzenie systemu zarządzającego dostępami użytkowników**
- **Tworzenie panelu administratora i CMS oraz modułu do debugowania i logów**
- **Obsługa wielu języków i formularze oparte na Bootstrapie Twitter CSS**
- **Własna nawigacja i sitemapy oraz testy jednostkowe**

Adam Omelak — od 10 lat programuje aplikacje oraz strony WWW na potrzeby internetu w językach: PHP, JavaScript, Java i ActionScript. Jest twórcą takich serwisów oraz aplikacji jak: Funkcje.net, ZaplanujTransport.pl, GazetkiSklepowe.pl, Polska Lista Zakupów Android oraz platformy e-learningowej dla sektora edukacji w Wielkiej Brytanii, Danii i Malezji (12 mln użytkowników). Pracował w Portal Technology Ltd. przy użyciu systemu hybris, a obecnie pracuje dla firmy Frog Education Ltd. Ponadto prowadzi własną firmę konsultingową oraz projektującą strony internetowe i aplikacje na Androida.

Helion

księgarnia internetowa

http://helion.pl

zamówienia telefoniczne

0 801 339900

0 601 339900

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

ISBN 978-83-283-3283-6

Informatyka w najlepszym wydaniu

cena: 59,00 zł