

Łukasz Sosna

# CodeIgniter

Zaawansowane tworzenie stron w **PHP**



*Wolisz proste rozwiązania?*

*Możesz liczyć na **CodeIgniter!***



**Helion**

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

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

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

Redaktor prowadzący: Ewelina Burska

Projekt okładki: Michał Wójcik

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](mailto: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?codeig>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Materiały do książki znajdują się pod adresem:

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

ISBN: 978-83-246-4964-8

Copyright © Helion 2013

Printed in Poland.

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

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

# Spis treści

<b>Wstęp</b> .....	<b>11</b>
<b>Rozdział 1. Instalacja CodeIgnitera na serwerze</b> .....	<b>13</b>
Instalacja oprogramowania na serwerze .....	14
Aktualizacja CodeIgnitera .....	14
Praca z książką .....	15
<b>Rozdział 2. Korzystanie z frameworku</b> .....	<b>17</b>
Model, widok, kontroler .....	17
Adresy URL .....	18
Adresy URL dla wyszukiwarek .....	18
Kontroler (controller) .....	19
Definicja domyślnego kontrolera (default_controller) .....	19
Metody prywatne (private) .....	20
Umieszczanie kontrolerów w folderach .....	20
Automatyczne ładowanie (__construct) .....	21
Nazwy zastrzeżone .....	21
Widok (view) .....	21
Ładowanie widoku do kontrolera (load->view) .....	22
Ładowanie kilku widoków .....	23
Umieszczanie widoków w folderach .....	23
Dodawanie danych do widoku .....	23
Przekazywanie widoków jako danych .....	24
Model (model) .....	24
Pierwszy model .....	25
Ładowanie modelu (load->model) .....	25
Umieszczanie modeli w katalogu .....	26
Dostęp do metod modelu .....	26
Ładowanie bibliotek (load->library) .....	27
Ładowanie klas pomocniczych (load->helper) .....	28
Ładowanie sterowników (load->driver) .....	28
Automatyczne ładowanie zasobów .....	29
Routowanie URL .....	29
Ustanawianie własnych reguł .....	30
Zastrzeżone nazwy właściwości .....	30
Obsługa błędów .....	31
Funkcja show_error() .....	31
Funkcja show_404() .....	32
Funkcja log_message() .....	32

Zapisywanie stron w plikach tymczasowych (cache) .....	33
Metoda cache() .....	34
Usuwanie plików tymczasowych .....	34
Bezpieczeństwo .....	34
Zabezpieczanie adresów URL .....	35
Opcja register_globals .....	35
Opcja error_reporting .....	35
Opcja magic_quotes_runtime .....	36
Najlepsze praktyki .....	36

### **Rozdział 3. Klasy systemowe ..... 37**

Wydajność systemu (benchmark) .....	37
Wyświetlanie informacji o zużyciu pamięci podczas generowania strony .....	38
Kalendarz (calendar) .....	39
Metoda generate() .....	39
Wyszczególnianie dat w kalendarzu .....	40
Opcje wyświetlania kalendarza .....	41
Opcje konfiguracji kalendarza .....	42
Dostosowywanie wyglądu kalendarza .....	42
Konfiguracja (config) .....	43
Metoda load() .....	44
Metoda item() .....	44
Metoda set_item() .....	45
Metoda site_url() .....	45
Metoda base_url() .....	46
Metoda system_url() .....	46
E-mail (email) .....	46
Metoda from() .....	47
Metoda reply_to() .....	47
Metoda cc() .....	47
Metoda bcc() .....	47
Metoda subject() .....	47
Metoda message() .....	48
Metoda set_alt_message() .....	48
Metoda clear() .....	48
Metoda attach() .....	48
Metoda send() .....	48
Metoda print_debugger() .....	48
Ustawianie zawijania wierszy w treści wiadomości .....	49
Ustawianie opcji .....	49
Opcje wysyłania wiadomości e-mail .....	49
Kodowanie i szyfrowanie (encrypt) .....	50
Metoda encode() .....	51
Metoda decode() .....	52
Metoda sha1() .....	53
Przesyłanie plików na serwer (upload) .....	53
Ustawianie preferencji .....	55
Metoda do_upload() .....	57
Metoda display_errors() .....	57
Metoda data() .....	57
Walidacja formularzy (form_validation) .....	58
Metoda set_rules() .....	62
Metoda run() .....	62
Twoja własna metoda .....	63

Funkcja form_error()	64
Funkcja validation_errors()	64
FTP (ftp)	64
Metoda connect()	65
Metoda upload()	66
Metoda download()	66
Metoda rename()	67
Metoda move()	68
Metoda delete_file()	68
Metoda delete_dir()	69
Metoda list_files()	70
Metoda mirror()	70
Metoda mkdir()	71
Metoda chmod()	72
Metoda close()	72
Generowanie tabel HTML (table)	73
Metoda generate()	74
Metoda set_caption()	74
Metoda set_heading()	75
Metoda add_row()	75
Metoda make_columns()	75
Metoda set_template()	76
Metoda set_empty()	76
Metoda clear()	77
Metoda function()	78
Wprowadzanie danych (input)	79
Filtrowanie XSS	80
Wykorzystywanie \$_POST, \$_COOKIE i \$_SERVER	80
Metoda get()	80
Metoda get_post()	81
Metoda set_cookie()	81
Metoda ip_address()	82
Metoda valid_ip()	83
Metoda user_agent()	83
Metoda request_headers()	84
Metoda get_request_header()	84
Ładowanie klas i przekształcanie ich w obiekty	85
Metoda library()	85
Metoda view()	86
Metoda model()	86
Metoda database()	87
Metoda helper()	87
Metoda language()	87
Języki (lang)	88
Tworzenie pliku z wersją językową	88
Metoda load()	89
Metoda line()	89
Automatyczne ładowanie plików językowych	90
Generowanie strony (output)	90
Metoda set_output()	90
Metoda set_content_type()	90
Metoda set_header()	91
Metoda set_status_header()	91
Metoda enable_profiler()	91

Dzielenie treści (pagination) .....	91
Metoda initialize() .....	92
Metoda create_links() .....	93
Opcje podziału treści na strony .....	93
Bezpieczeństwo (security) .....	95
Filtrowanie XSS .....	95
Metoda xss_clean() .....	95
Metoda sanitize_filename() .....	96
Ochrona przed atakami typu CSRF .....	97
Sesja (session) .....	97
Metoda userdata() .....	98
Metoda set_userdata() .....	98
Metoda all_userdata() .....	99
Metoda unset_userdata() .....	100
Ustawienia sesji .....	100
Wygląd strony (parser) .....	101
Metoda parse() .....	102
URI (uri) .....	103
Metoda segment() .....	103
Metoda uri_to_assoc() .....	103
Metoda assoc_to_uri() .....	104
Przeglądarka użytkownika (user_agent) .....	105
Identyfikacja przeglądarki użytkownika .....	105
Metoda is_browser() .....	106
Metoda is_mobile() .....	107
Metoda is_robot() .....	107
Metoda is_referral() .....	108
Metoda browser() .....	108
Metoda version() .....	109
Metoda mobile() .....	109
Metoda robot() .....	110
Metoda platform() .....	110
Metoda referrer() .....	111
Metoda agent_string() .....	111
Metoda accept_lang() .....	112
Metoda accept_charset() .....	113
Kompresja plików (zip) .....	113
Metoda add_data() .....	114
Metoda add_dir() .....	114
Metoda read_file() .....	115
Metoda read_dir() .....	115
Metoda archive() .....	116
Metoda download() .....	116
Metoda get_zip() .....	117
Metoda clear_data() .....	117

## **Rozdział 4. Sterowniki systemu do obsługi bazy danych ..... 119**

Praca z bazą danych .....	119
Wczytanie sterownika (load->database) .....	120
Połączenie z bazą danych — konfiguracja danych dostępowych .....	120
Połączenie z bazą danych (database) .....	122
Metoda reconnect() .....	123
Metoda close() .....	124
Wykonywanie zapytania (query) .....	125

Metoda query()	126
Zabezpieczanie danych (escape, escape_str i escape_like_str)	126
Bezpieczne zapytania (query)	127
Generowanie rezultatów zapytania	128
Metoda result()	128
Metoda result_array()	129
Metoda num_rows()	129
Metoda free_result()	130
Metoda insert_id()	131
Metoda affected_rows()	131
Metoda count_all()	132
Metoda platform()	132
Metoda version()	133
Metoda last_query()	133
Informacje o tabelach	134
Metoda list_tables()	134
Metoda table_exists()	134
Informacje o polach tabeli	135
Metoda list_fields()	135
Metoda field_exists()	136
Metoda field_data()	137
<b>Rozdział 5. Klasy pomocnicze</b>	<b>139</b>
Tablica (array)	139
Funkcja element()	139
Funkcja random_element()	140
Funkcja elements()	141
Wysyłanie danych z formularza	142
Pliki cookie (cookie)	143
Funkcja set_cookie()	143
Funkcja get_cookie()	143
Funkcja delete_cookie()	143
Data (date)	144
Funkcja now()	144
Funkcja standard_date()	144
Funkcja local_to_gmt()	145
Funkcja unix_to_human()	146
Funkcja timespan()	146
Funkcja days_in_month()	147
Funkcja timezones()	148
Funkcja timezone_menu()	148
Katalog (directory)	150
Funkcja directory_map()	150
Pobieranie (download)	152
Funkcja force_download()	152
E-mail (email)	153
Funkcja valid_email()	153
Funkcja send_email()	154
Plik (file)	154
Funkcja write_file()	154
Funkcja read_file()	155
Funkcja delete_files()	155
Funkcja get_dir_file_info()	156
Funkcja get_file_info()	158

Funkcja <code>get_mime_by_extension()</code> .....	159
Funkcja <code>symbolic_permissions()</code> .....	159
Funkcja <code>octal_permissions()</code> .....	160
Formularz (form) .....	161
Funkcja <code>form_open()</code> .....	161
Funkcja <code>form_open_multipart()</code> .....	162
Funkcja <code>form_hidden()</code> .....	162
Funkcja <code>form_input()</code> .....	163
Funkcja <code>form_password()</code> .....	164
Funkcja <code>form_upload()</code> .....	164
Funkcja <code>form_textarea()</code> .....	165
Funkcja <code>form_dropdown()</code> .....	165
Funkcja <code>form_multiselect()</code> .....	166
Funkcje <code>form_fieldset()</code> i <code>form_fieldset_close()</code> .....	167
Funkcja <code>form_checkbox()</code> .....	167
Funkcja <code>form_radio()</code> .....	168
Funkcja <code>form_submit()</code> .....	169
Funkcja <code>form_reset()</code> .....	169
Funkcja <code>form_button()</code> .....	170
Funkcja <code>form_close()</code> .....	171
Funkcja <code>form_prep()</code> .....	171
HTML (html) .....	171
Funkcja <code>br()</code> .....	172
Funkcja <code>heading()</code> .....	172
Funkcja <code>img()</code> .....	173
Funkcja <code>link_tag()</code> .....	174
Funkcja <code>nbs()</code> .....	175
Funkcje <code>ol()</code> i <code>ul()</code> .....	175
Funkcja <code>meta()</code> .....	177
Funkcja <code>doctype()</code> .....	179
Język (language) .....	179
Funkcja <code>lang()</code> .....	180
Liczba (number) .....	181
Funkcja <code>byte_format()</code> .....	181
Ścieżka (path) .....	182
Funkcja <code>set_realpath()</code> .....	183
Bezpieczeństwo (security) .....	184
Funkcja <code>xss_clean()</code> .....	184
Funkcja <code>sanitize_filename()</code> .....	184
Funkcja <code>do_hash()</code> .....	185
Funkcja <code>strip_image_tags()</code> .....	185
Funkcja <code>encode_php_tags()</code> .....	186
Ciąg (string) .....	187
Funkcja <code>random_string()</code> .....	187
Funkcja <code>increment_string()</code> .....	188
Funkcja <code>alternator()</code> .....	188
Funkcja <code>repeater()</code> .....	189
Funkcja <code>reduce_double_slashes()</code> .....	190
Funkcja <code>trim_slashes()</code> .....	190
Funkcja <code>quotes_to_entities()</code> .....	191
Funkcja <code>strip_quotes()</code> .....	191
Tekst (text) .....	192
Funkcja <code>word_limiter()</code> .....	192
Funkcja <code>character_limiter()</code> .....	193
Funkcja <code>ascii_to_entities()</code> .....	193



Funkcja <code>entities_to_ascii()</code> .....	194
Funkcja <code>highlight_code()</code> .....	194
Funkcja <code>word_wrap()</code> .....	195
Funkcja <code>ellipsize()</code> .....	196
URL ( <code>url</code> ) .....	197
Funkcja <code>site_url()</code> .....	197
Funkcja <code>base_url()</code> .....	197
Funkcja <code>current_url()</code> .....	198
Funkcja <code>uri_string()</code> .....	198
Funkcja <code>index_page()</code> .....	199
Funkcja <code>anchor()</code> .....	199
Funkcja <code>redirect()</code> .....	200
XML ( <code>xml</code> ) .....	200
Funkcja <code>xml_convert()</code> .....	201
<b>Rozdział 6. Tworzenie pierwszej aplikacji .....</b>	<b>203</b>
Rozpoczęcie pracy .....	203
Baza danych .....	204
Główny plik aplikacji .....	204
Plik konfiguracyjny .....	204
Konfiguracja automatycznego ładowania .....	204
Konfiguracja bazy danych .....	205
Konfiguracja domyślnego kontrolera .....	205
Tworzenie pliku do przepisywania adresów .....	205
Kontroler — logowanie administratora .....	206
Kontroler — wylogowanie administratora .....	208
Kontroler — strona główna .....	209
Kontroler — kategoria wpisów .....	212
Kontroler — wpis .....	215
Kontroler — edycja wpisów .....	219
Model — logowanie użytkownika .....	227
Model — strona główna .....	228
Model — kategoria .....	229
Model — wpis .....	231
Model — polecenia współdzielone .....	233
Model — edycja .....	233
Widok — nagłówek .....	237
Widok — stopka .....	240
Widok — menu .....	240
Widok — menu administratora .....	242
Widok — menu administratora przed zalogowaniem .....	242
Widok — kategoria .....	243
Widok — edytowanie kategorii .....	245
Widok — dodawanie kategorii .....	246
Widok — wpisy .....	246
Widok — nowy wpis .....	249
Widok — edytowanie wpisu .....	251
Widok — najnowsze wpisy .....	253
Widok — zaloguj .....	254
Widok — strona główna .....	255
Widok — zmień hasło .....	256
Zakończenie .....	257

<b>Nazwy zastrzeżone .....</b>	<b>259</b>
<b>Podsumowanie .....</b>	<b>261</b>
<b>Skorowidz .....</b>	<b>263</b>

## Rozdział 4.

# Sterowniki systemu do obsługi bazy danych

Sterownik przeznaczony do obsługi bazy danych umożliwia korzystanie z jej zasobów. CodeIgniter pozwala na wykorzystywanie takich baz danych jak na przykład MySQL i PostgreSQL. Dzięki temu jest dogodną platformą do tworzenia zaawansowanych aplikacji — umożliwia przeprowadzenie różnych operacji na rekordach, takich jak umieszczanie ich w tabeli, aktualizowanie czy usuwanie.

## Praca z bazą danych

Zanim zaczniemy przygodę z bazą danych, musimy utworzyć dwie bazy: pierwszą o nazwie *codeigniter* oraz drugą — *codeigniter2*. Najprawdopodobniej pracujesz na oprogramowaniu XAMPP, więc w celu ich utworzenia możesz skorzystać z narzędzia phpMyAdmin, do którego dostęp uzyskasz, wpisując adres URL w pasku adresu przeglądarki internetowej: <http://localhost/phpMyAdmin>.

Następnie w pierwszej bazie o nazwie *codeigniter* należy wykonać poniższy kod zapytań SQL:

```
CREATE TABLE IF NOT EXISTS 'tabela' (  
  'id' int(11) NOT NULL AUTO_INCREMENT,  
  'tytuł' varchar(255) COLLATE utf8_unicode_ci NOT NULL,  
  PRIMARY KEY ('id')  
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci AUTO_INCREMENT=3 ;
```

```
INSERT INTO 'tabela' ('id', 'tytuł') VALUES  
(1, 'Treść 1'),  
(2, 'Treść 2');
```

```
CREATE TABLE IF NOT EXISTS 'tabela_post' (  
  'post_id' int(11) NOT NULL AUTO_INCREMENT,  
  'post_tytuł' varchar(255) COLLATE utf8_unicode_ci NOT NULL,  
  'post_tresc' text COLLATE utf8_unicode_ci NOT NULL,
```

```

    'post_aktywny' char(1) COLLATE utf8_unicode_ci NOT NULL,
    PRIMARY KEY ('post_id')
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci AUTO_INCREMENT=3 ;

INSERT INTO 'tabela_post' ('post_id', 'post_tytul', 'post_tresc', 'post_aktywny') VALUES
(1, 'Temat wiadomości', '', 't');
(2, 'Kolejny temat wiadomości', '', 't');

CREATE TABLE IF NOT EXISTS 'uzytkownicy' (
    'uzytkownik_id' int(11) NOT NULL AUTO_INCREMENT,
    'uzytkownik_nazwa' varchar(255) COLLATE utf8_unicode_ci NOT NULL,
    PRIMARY KEY ('uzytkownik_id')
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci AUTO_INCREMENT=1 ;

```

W przypadku zainstalowania oprogramowania XAMPP użytkownik mający pełen dostęp do baz danych to *root*, a jego hasło jest puste.

## Wczytanie sterownika (load->database)

Sterownik służący do obsługi bazy danych jest uruchamiany poprzez metodę `database()` z obiektu `load` (listing 4.1). Wszystkie dane potrzebne do nawiązania połączenia z bazą danych definiuje się w pliku konfiguracyjnym.

**Listing 4.1.** Ładowanie sterownika bazy danych

```

<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();
    }
}
?>

```

## Połączenie z bazą danych — konfiguracja danych dostępowych

Aby zapewnić sobie możliwość wykorzystania bazy danych, najpierw należy skonfigurować dane dostępne: określić nazwę hosta, nazwę użytkownika i jego hasło, wskazać nazwę bazy danych oraz zdefiniować inne potrzebne właściwości. W przypadku CodeIgnitera dane te są przechowywane w pliku `application/config/database.php`.

Plik ten zawiera tablicę, w której ustawia się różnego rodzaju dane dostępne do bazy danych (listing 4.2).

**Listing 4.2.** *Tablica konfiguracji danych dostępowych do bazy danych*

```
$db['default']['hostname'] = "localhost";
$db['default']['username'] = "root";
$db['default']['password'] = "";
$db['default']['database'] = "codeigniter";
$db['default']['dbdriver'] = "mysql";
$db['default']['dbprefix'] = "";
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = FALSE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = "";
$db['default']['char_set'] = "utf8";
$db['default']['dbcollat'] = "utf8_general_ci";
$db['default']['swap_pre'] = "";
$db['default']['autoinit'] = TRUE;
$db['default']['stricton'] = FALSE;
$db['default']['port'] = 3306;
```

Poniżej zostały objaśnione dostępne parametry:

- ♦ `hostname` — nazwa hosta bazy danych,
- ♦ `username` — nazwa użytkownika,
- ♦ `password` — hasło użytkownika,
- ♦ `database` — nazwa bazy danych,
- ♦ `dbdriver` — typ sterownika bazy danych, przykładowe wartości tego parametru to: *mysql*, *postgres*, *odbc* itd.,
- ♦ `dbprefix` — prefiks nazw tabel w bazie danych, który będzie automatycznie dodawany w przypadku korzystania z interfejsu *Active Record*,
- ♦ `pconnect` (TRUE/FALSE) — definiuje, czy chcemy używać stałego połączenia,
- ♦ `db_debug` (TRUE/FALSE) — wyświetlanie komunikatów o błędach bazy danych,
- ♦ `cache_on` (TRUE/FALSE) — włączenie pamięci podręcznej do przechowywania wyników zapytań,
- ♦ `cachedir` — katalog na serwerze do tymczasowego przechowywania wyników zapytań do bazy danych,
- ♦ `char_set` — zestaw znaków wykorzystywany do pracy z bazą danych,
- ♦ `dbcollat` — ustawienie kodowania pól w bazie danych,
- ♦ `swap_pre` — prefiks nazw tabel, stosowany podczas pracy nad wersją rozwojową aplikacji,
- ♦ `autoinit` — określenie automatycznego łączenia się z bazą danych po włączeniu interfejsu,

- ◆ `stricton` (TRUE/FALSE) — wymuszenie połączenia z bazą danych w trybie *Strict Mode*<sup>1</sup>,
- ◆ `port` — port wykorzystany do nawiązania połączenia z bazą danych.

Nie wszystkie opcje konfiguracyjne dostępne w pliku `database.php` są zawsze potrzebne. To, jakie dane będziesz musiał zdefiniować, zależy od rodzaju bazy danych. Przedstawiona powyżej konfiguracja jest odpowiednia dla bazy MySQL.

## Połączenie z bazą danych (database)

CodeIgniter umożliwia dwa sposoby łączenia się z bazą danych. Pierwszy z nich polega na każdorazowym załadowaniu odpowiedniego sterownika podczas uruchamiania aplikacji, co powoduje automatyczne nawiązanie połączenia z bazą danych. Opcja ta jest bardzo przydatna, gdy mamy zamiar utworzyć aplikację korzystającą z zasobów tylko jednej bazy danych.

Drugim sposobem jest ręczne zestawianie połączenia za pomocą metody `database()`. Pierwszym parametrem tej metody jest nazwa klucza tablicy, w której znajdują się dane potrzebne do połączenia z bazą danych. Jak pamiętasz, tablicę tę deklarujemy w pliku `application/config/database.php`. Kopiujemy już obecną w tym pliku tablicę `$db` ↪ ['default'] i zmieniamy jej klucz na dowolny inny, ważne jednak, by był on zgodny z nazwami kluczy w języku PHP (listing 4.3). Może się on nazywać na przykład *drugiepolaczenie* (listing 4.4).

**Listing 4.3.** Fragment zmodyfikowanego pliku `database.php`

```
$db['default']['hostname'] = "localhost";
$db['default']['username'] = "root";
$db['default']['password'] = "";
$db['default']['database'] = "codeigniter";
$db['default']['dbdriver'] = "mysql";
$db['default']['dbprefix'] = "";
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = FALSE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = "";
$db['default']['char_set'] = "utf8";
$db['default']['dbcollat'] = "utf8_general_ci";
$db['default']['swap_pre'] = "";
$db['default']['autoinit'] = TRUE;
$db['default']['stricton'] = FALSE;
$db['default']['port'] = 3306

$db['drugiepolaczenie']['hostname'] = "localhost";
$db['drugiepolaczenie']['username'] = "root";
$db['drugiepolaczenie']['password'] = "";
```

<sup>1</sup> *Strict Mode* — tzw. tryb ścisły pracy, w którym baza danych nie będzie podejmowała prób obsłużenia danych przekazanych w nieprawidłowym formacie — *przyp. red.*

```
$db['drugiepolaczenie']['database'] = "codeigniter2";
$db['drugiepolaczenie']['dbdriver'] = "mysql";
$db['drugiepolaczenie']['dbprefix'] = "";
$db['drugiepolaczenie']['pconnect'] = TRUE;
$db['drugiepolaczenie']['db_debug'] = FALSE;
$db['drugiepolaczenie']['cache_on'] = FALSE;
$db['drugiepolaczenie']['cachedir'] = "";
$db['drugiepolaczenie']['char_set'] = "utf8";
$db['drugiepolaczenie']['dbcollat'] = "utf8_general_ci";
$db['drugiepolaczenie']['swap_pre'] = "";
$db['drugiepolaczenie']['autoinit'] = TRUE;
$db['drugiepolaczenie']['stricton'] = FALSE;
$db['drugiepolaczenie']['port'] = 3306;
```

---

**Listing 4.4.** *Ręczne łączenie się z bazą danych*

---

```
$this->load->database('drugiepolaczenie');
```

---

Na etapie tworzenia projektu aplikacji należy się dobrze zastanowić, z ilu baz danych będzie się korzystać. W zależności od tego, czy będzie to jedna baza, czy większa ich liczba, konstrukcja poleceń w metodzie będzie nieco inna (listingi 4.5 i 4.6).

---

**Listing 4.5.** *Wykonywanie zapytań w przypadku łączenia się z tylko jedną bazą danych*

---

```
$this->db->query();
$this->db->result();
```

---

**Listing 4.6.** *Wykonywanie zapytań w przypadku łączenia się z dwiema lub większą liczbą baz danych*

---

```
$DB1->query();
$DB1->result();

$DB2->query();
$DB2->result();
```

---

Pamiętaj, że poszczególne obiekty obsługujące bazy danych zawierają w swojej nazwie liczby odpowiadające kolejności dokonywania połączeń. Znajomość tych numerów umożliwi prawidłowe odwoływanie się do tych obiektów.

## Metoda reconnect()

Podczas pracy aplikacji niekiedy dochodzi do utraty połączenia z bazą danych, na przykład na skutek przekroczenia limitu bezczynności serwera bazy danych. Aby uniknąć ponownego definiowania wszystkich argumentów, w celu ponownego połączenia się z bazą wystarczy wywołać metodę `reconnect()`, która automatycznie podtrzyma połączenie lub — w przypadku jego utraty — ponownie je nawiąże (listing 4.7).

**Listing 4.7.** *Podtrzymanie lub powtórne nawiązanie połączenia z bazą danych*

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

        $Informacje = $this->db->query('SELECT * FROM tabela');

        echo '<pre>';
        print_r($Informacje);
        echo '</pre>';

        $this->db->reconnect();

        $Informacje = $this->db->query('SELECT * FROM tabela');

        echo '<pre>';
        print_r($Informacje);
        echo '</pre>';
    }
}
?>
```

## Metoda close()

CodeIgniter zapewnia automatyczne zamykanie połączenia z bazą danych, jednak jeżeli chcesz zamknąć je ręcznie, możesz wywołać metodę `close()` (listing 4.8).

**Listing 4.8.** *Zakończenie połączenia z bazą danych*

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

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



## Wykonywanie zapytania (query)

Zapytanie, które chcemy wykonać w bazie danych, musi spełniać wymagania standardu języka SQL zgodnego z bazą danych, którą będziemy wykorzystywać. Zapytanie SQL podajemy jako parametr metody `query()` z obiektu `db` (listing 4.9). Przykładowy rezultat wykonania skryptu obrazuje rysunek 4.1.

**Listing 4.9.** Wykonanie zapytania SQL

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

        $Informacje = $this->db->query('SELECT * FROM tabela');

        echo '<pre>';
        print_r($Informacje);
        echo '</pre>';
    }
}
?>
```

**Rysunek 4.1.**  
Wynik zapytania

```
CI_DB_mysql_result Object
(
    [conn_id] => Resource id #44
    [result_id] => Resource id #45
    [result_array] => Array
        (
        )
    [result_object] => Array
        (
        )
    [custom_result_object] => Array
        (
        )
    [current_row] => 0
    [num_rows] => 2
    [row_data] =>
)
)
```

## Metoda query()

Same zapytania wysyłane do bazy danych konstruuje się w języku SQL w implementacji odpowiadającej zastosowanej bazie danych. Jednak ze względu na obowiązujące standardy w większości przypadków składnia SQL jest bardzo podobna.

W języku PHP do uzyskiwania wyników zapytań SQL wykorzystuje się metodę query(). Parametrem tej metody jest zapytanie w języku SQL. Na listingu 4.10 przedstawiono przykład prostego zapytania do bazy danych.

**Listing 4.10.** Zapytanie do bazy danych

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

        $this->db->query('SELECT * FROM tabela');
    }
}
?>
```

W przypadku zapytania SQL pobierającego dane z bazy danych, takiego jak SELECT, metoda query() zwraca obiekt zbioru wyników odpowiadający wykonanemu zapytaniu. W dalszej części rozdziału pokażemy, w jaki sposób można przetworzyć ten obiekt. W pozostałych przypadkach (kiedy w wyniku zapytania nie otrzymujemy zbioru danych) wartością przekazaną przez metodę query() jest TRUE, gdy zapytanie zostanie poprawnie wykonane, lub FALSE, gdy nastąpi błąd.

## Zabezpieczanie danych (escape, escape\_str i escape\_like\_str)

Aby uchronić swoją aplikację przed atakami typu *SQL Injection*, trzeba pamiętać o prefiltrowaniu danych przed ich wprowadzeniem do zapytania SQL i wysłaniem do bazy danych. Jest to bardzo ważne ze względu na bezpieczeństwo naszej aplikacji. W tabeli 4.1 przedstawiono dostępne w CodeIgniterze metody umożliwiające zabezpieczanie danych wpisywanych przez użytkownika poprzez wstawianie lewych ukośników przed znakami specjalnymi, takimi jak apostrof czy cudzysłów.

**Tabela 4.1.** *Metody zabezpieczające dane*

Metoda	Opis	Przykład
<code>\$this-&gt;db-&gt;escape()</code>	Odpowiedni tylko dla ciągów znakowych	<code>\$Sql = "INSERT INTO tabela (tytul) VALUES (". \$this-&gt;db-&gt;escape(\$Tytul).")";</code>
<code>\$this-&gt;db-&gt;escape_str()</code>	Odpowiedni dla wszystkich danych, bez względu na ich typ	<code>\$Sql = "INSERT INTO tabela (tytul) VALUES('". \$this-&gt;db-&gt;escape_str(\$Tytul). "')";</code>
<code>\$this-&gt;db-&gt;escape_like_str()</code>	Odpowiedni dla ciągu znakowego wykorzystanego jako parametr w zapytaniach z użyciem słowa kluczowego LIKE	<code>\$Tytul = '20% upustu'; \$Sql = "SELECT id FROM tabela WHERE tytul LIKE '%". \$this-&gt;db-&gt;escape_like_str(\$Tytul). "%";</code>

## Bezpieczne zapytania (query)

Zastosowanie tego typu zapytań pozwala na zapewnienie większego bezpieczeństwa aplikacji, ponieważ programista nie musi pamiętać o sprawdzaniu danych wejściowych. Sterownik bazy danych CodeIgnitera zapewni odpowiednie filtrowanie danych (listing 4.11).

**Listing 4.11.** *Tworzenie zapytania wiązanego*

```

<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

        $Zapytanie = "SELECT * FROM tabela_post WHERE post_id = ? AND post_tytul = ?
        ↳AND post_aktywny = ?";
        $Parametry = array(1, 'Temat wiadomości', 't');

        $this->db->query($Zapytanie, $Parametry);
    }
}
?>

```

Metoda `query()` przyjmuje w tym przypadku dwa parametry. Pierwszym jest zapytanie do bazy danych wraz ze znakami zapytania (?) w miejscach, w których powinny się znaleźć konkretne wartości. Drugim parametrem jest tablica z tymi właśnie wartościami.

Musimy pamiętać o jeszcze jednej bardzo ważnej sprawie: liczba wartości w tablicy musi być równa liczbie znaków zapytania umieszczonych w pierwszym parametrze.

## Generowanie rezultatów zapytania

Istnieje kilka sposobów generowania rezultatu wykonanego zapytania. Wybór konkretnego sposobu zależy od tego, jaką wartość chcemy uzyskać, w jaki sposób ma zostać zaprezentowana użytkownikowi i do czego ma nam służyć.

### Metoda result()

Metoda `result()` daje dostęp do wyniku zapytania uzyskanego z wcześniej omówionej metody `query()`. Wynik ten jest udostępniany jako obiekt, który można przetworzyć w pętli `foreach`. Na listingu 4.12 pokazano przykładowy skrypt. Rezultatem jego wykonania będzie:

```
1 Temat wiadomości t
2 Kolejny temat wiadomości t
```

**Listing 4.12.** Odbieranie wyniku zapytania za pomocą metody `result()`

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

        $Rezultat = $this->db->query('SELECT * FROM tabela_post');

        foreach($Rezultat->result() as $dane)
        {
            echo $dane->post_id;
            echo ' ';
            echo $dane->post_tytul;
            echo ' ';
            echo $dane->post_aktywny;
            echo '<br />';
        }
    }
}
?>
```

## Metoda `result_array()`

Metoda `result_array()` umożliwia otrzymanie wyniku zapytania w postaci zwykłej tablicy asocjacyjnej, w której kluczami są nazwy kolumn z tabeli (listing 4.13).

**Listing 4.13.** Odbieranie wyników zapytań za pomocą metody `result_array()`

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

        $Rezultat = $this->db->query('SELECT * FROM tabela_post');

        foreach ($Rezultat->result_array() as $dane)
        {
            echo $dane['post_id'];
            echo ' ';
            echo $dane['post_tytul'];
            echo ' ';
            echo $dane['post_aktywny'];
            echo '<br />';
        }
    }
}
?>
```

## Metoda `num_rows()`

Aby otrzymać liczbę rekordów uzyskanych w wyniku zapytania, można się posłużyć metodą `num_rows()`. Wywołujemy ją jako metodę obiektu zawierającego rezultat tego zapytania. Na listingu 4.14 pokazano przykładowy skrypt. Wynikiem jego wykonania będzie liczba 2.

**Listing 4.14.** Przekazywanie liczby wybranych rekordów

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
```

```
$this->load->database();

$Rezultat = $this->db->query('SELECT * FROM tabela_post');

echo $Rezultat->num_rows();
}
}
?>
```

---

## Metoda `free_result()`

Zwykle PHP automatycznie zwalnia pamięć przy zakończeniu przetwarzania skryptu. Jeśli jednak tworzysz duży projekt i chcesz uniknąć ryzyka wystąpienia błędu PHP spowodowanego wyczerpaniem się przydzielonych aplikacji zasobów pamięci RAM serwera, warto rozważyć ręczne, a nie automatyczne zwalnianie pamięci po każdym wykonaniu zapytania. Do tego celu służy metoda `free_result()`. Na listingu 4.15 znajduje się przykład zastosowania tej metody.

### **Listing 4.15.** *Czyszczenie zawartości zapytania z pamięci serwera*

---

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

        $Rezultat = $this->db->query('SELECT * FROM tabela_post');

        foreach ($Rezultat->result_array() as $dane)
        {
            echo $dane['post_id'];
            echo ' ';
            echo $dane['post_tytul'];
            echo ' ';
            echo $dane['post_aktywny'];
            echo '<br />';
        }

        $Rezultat->free_result();
    }
}
?>
```

---

## Metoda insert\_id()

Metoda `insert_id()` przekazuje identyfikator wartości, która została ostatnio dodana do tabeli za pomocą zapytania `INSERT`. Jest on pobierany z pola typu `auto_increment`. Rezultatem wykonania skryptu z listingu 4.16 będzie liczba 3.

**Listing 4.16.** Wyświetlanie ostatniego dodanego numeru w polu `auto_increment`

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

        $Rezultat = $this->db->query('INSERT INTO tabela_post
        ↳(post_tytuł,post_aktywny) VALUES ("Inny post", "t")');

        echo $this->db->insert_id();
    }
}
?>
```

## Metoda affected\_rows()

Metoda `affected_rows()` przekazuje liczbę rekordów zmienionych podczas wykonywania ostatniego zapytania `UPDATE`. Rezultatem wykonania skryptu z listingu 4.17 będzie liczba 1.

**Listing 4.17.** Wyświetlanie liczby rekordów zmienionych podczas ostatniego zapytania

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

        $Rezultat = $this->db->query('UPDATE tabela_post SET post_tytuł = "Nowy post"
        ↳WHERE post_id = 3');

        echo $this->db->affected_rows();
    }
}
```

```
    }  
  }  
?>
```

---

## Metoda count\_all()

---

Metoda ta zlicza wszystkie rekordy tabeli, której nazwę podano w parametrze. Na listingu 4.18 znajduje się przykładowy skrypt. Rezultatem jego wykonania będzie liczba 3.

### Listing 4.18. Wyświetlanie liczby rekordów w tabeli

---

```
<?php  
if(!defined('BASEPATH'))  
{  
    exit('No direct script access allowed');  
}  
  
class Mvc extends CI_Controller  
{  
    public function index()  
    {  
        $this->load->database();  
  
        echo $this->db->count_all('tabela_post');  
    }  
}  
?>
```

---

## Metoda platform()

Metoda platform() pozwala na wyświetlenie nazwy typu bazy danych. Rezultatem wykonania skryptu z listingu 4.19 może być na przykład mysql.

### Listing 4.19. Przekazywanie informacji o typie bazy danych

---

```
<?php  
if(!defined('BASEPATH'))  
{  
    exit('No direct script access allowed');  
}  
  
class Mvc extends CI_Controller  
{  
    public function index()  
    {  
        $this->load->database();  
  
        echo $this->db->platform();  
    }  
}  
?>
```

---



## Metoda version()

Metoda `version()` przekazuje numer wersji aktualnie wykorzystywanej bazy danych. Na listingu 4.20 znajduje się odpowiedni przykład. Rezultatem wykonania tego skryptu będzie 5.5.8 (wersja bazy danych może się różnić).

**Listing 4.20.** Numer wersji bazy danych

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

        echo $this->db->version();
    }
}
?>
```

## Metoda last\_query()

Metoda `last_query()` pozwala na wyświetlenie ostatniego zapytania, jakie wykonano na bazie danych. Na listingu 4.21 pokazano przykład wykorzystania tej metody. Rezultatem wykonania tego skryptu będzie `UPDATE tabela_post SET post_tytul = "Nowy post" WHERE post_id = 3`.

**Listing 4.21.** Wyświetlanie ostatniego wykonanego zapytania

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

        $Rezultat = $this->db->query('UPDATE tabela_post SET post_tytul = "Nowy post"
↳WHERE post_id = 3');

        echo $this->db->last_query();
    }
}
?>
```

# Informacje o tabelach

W ramach obsługi baz danych CodeIgniter udostępnia również metody pozwalające na uzyskanie pewnych informacji o tabelach w bazie danych.

## Metoda `list_tables()`

Za pomocą metody `list_tables()` możemy uzyskać nazwy wszystkich tabel z aktualnie wykorzystywanej bazy danych. Na listingu 4.22 pokazano odpowiedni przykład. Rezultatem wykonania tego skryptu będzie wynik:

```
tabela
tabela_post
uzytkownicy
```

**Listing 4.22.** Wyświetlanie nazw tabel z obecnie wykorzystywanej bazy danych

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

        $Tabele = $this->db->list_tables();

        foreach ($Tabele as $Tabela)
        {
            echo $Tabela;
            echo '<br />';
        }
    }
}
?>
```

## Metoda `table_exists()`

Metoda `table_exists()` umożliwi sprawdzenie, czy tabela o nazwie podanej jako parametr tej metody istnieje w bazie danych (listing 4.23). Rezultatem wykonania skryptu będzie: Tabela istnieje w bazie danych, jeżeli się tam rzeczywiście znajduje, lub komunikat: Tabela NIE istnieje w bazie danych w przypadku jej braku.

**Listing 4.23.** Sprawdzenie, czy dana tabela istnieje w bazie danych

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

        if($this->db->table_exists('tabela'))
        {
            echo 'Tabela istnieje w bazie danych';
        }
        else
        {
            echo 'Tabela NIE istnieje w bazie danych';
        }
    }
}
?>
```

## Informacje o polach tabeli

Twórcy CodeIgnitera udostępnili programistom metody pomocne w uzyskiwaniu dodatkowych informacji na temat pól (kolumn) w tabelach bazy danych.

### Metoda `list_fields()`

Metoda `list_fields()` przekazuje listę wszystkich pól w tabeli, której nazwa została podana jako parametr tej metody. Na listingu 4.24 znajduje się przykładowy kod. Rezultatem jego wykonania będzie:

```
post_id
post_tytul
post_tresc
post_aktywny
```

**Listing 4.24.** Informacje o polach tabeli

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
```

```
{
    public function index()
    {
        $this->load->database();

        $Pola = $this->db->list_fields('tabela_post');

        foreach ($Pola as $Pole)
        {
            echo $Pole;
            echo '<br />';
        }
    }
}
?>
```

---

## Metoda `field_exists()`

Metoda `field_exists()` umożliwia sprawdzenie, czy w tabeli istnieje dane pole. Pierwszym parametrem metody jest nazwa kolumny, a drugim — nazwa tabeli (listing 4.25). Rezultatem wykonania skryptu będzie kolumna `post_tytul` istnieje w tabeli w przypadku stwierdzenia jej obecności.

### **Listing 4.25.** *Informacja o określonym polu w tabeli*

---

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

        if($this->db->field_exists('post_tytul', 'tabela_post'))
        {
            echo 'Kolumna post_tytul istnieje w tabeli';
        }
        else
        {
            echo 'Kolumna post_tytul NIE istnieje w tabeli';
        }
    }
}
?>
```

---

## Metoda `field_data()`

Metoda `field_data()` przekazuje podstawowe informacje na temat pola w tabeli, którego nazwę podano jako parametr tej metody. Odpowiedni przykład znajduje się na listingu 4.26. Rezultatem jego wykonania będzie:

```
post_id int 11 1
post_tytul varchar 255 0
post_tresc text 0
post_aktywny char 1 0
```

---

### Listing 4.26. Wyświetlanie informacji o polach tabeli

---

```
<?php
if(!defined('BASEPATH'))
{
    exit('No direct script access allowed');
}

class Mvc extends CI_Controller
{
    public function index()
    {
        $this->load->database();

        $Pola = $this->db->field_data('tabela_post');

        foreach($Pola as $Pole)
        {
            echo $Pole->name:
            echo ' ';
            echo $Pole->type:
            echo ' ';
            echo $Pole->max_length:
            echo ' ';
            echo $Pole->primary_key:
            echo '<br />';
        }
    }
}
?>
```

---

Poniżej omówiono przekazywane wartości:

- ♦ `name` — nazwa pola,
- ♦ `type` — typ pola,
- ♦ `max_length` — maksymalna długość danych w polu (wielkość przechowanych informacji),
- ♦ `primary_key` — jeżeli dane pole stanowi klucz główny tabeli, wartość tego parametru wyniesie 1.



# Skorowidz

## A

- adres
  - e-mail, 153
  - IP, 82
  - URL, 18
- aktualizacja CodeIgnitera, 14
- arkusz stylów, 174
- atak
  - CSRF, 97
  - directory travealsal, 184
  - XSS, 79
- automatyczne ładowanie, 21, 204

## B

- baza danych, 12, 119, 204
- benchmark, 37
- bezpieczeństwo, 34, 95, 184
- bezpieczne zapytania, 127
- biblioteka, 27
  - lang, 88
  - pagination, 92, 213
- błąd 404, 31
- buforowanie stron, 34

## C

- ciąg, 187
- CodeIgniter, 11
- CSRF, Cross-Site Request Forgeries, 97
- czas generowania strony, 37
- czyszczenie zawartości zapytania, 130

## D

- dane sesji, 99
- data, 144
- dodawanie
  - danych do widoku, 23
  - pliku do archiwum, 114
- dostęp do
  - bazy danych, 121
  - metod modelu, 26
- dzielenie treści, 91

## E

- e-mail, 46, 153

## F

- filtrowanie
  - danych, 95, 142
  - XSS, 80, 95
- format daty, 145
- formularz, 54, 55, 58, 161
- framework
  - CodeIgniter, 11
  - Symfony, 261
  - Yii, 261
  - Zend, 11, 261
- FTP, 64
- funkcja
  - alternator(), 188
  - anchor(), 199
  - ascii\_to\_entities(), 193
  - base\_url(), 197
  - br(), 172
  - byte\_format(), 181
  - character\_limiter(), 193
  - current\_url(), 198

## funkcja

days\_in\_month(), 147  
 delete\_cookie(), 143  
 delete\_files(), 155  
 directory\_map(), 150  
 do\_hash(), 185  
 doctype(), 179  
 element(), 139  
 elements(), 141  
 ellipsis(), 196  
 encode\_php\_tags(), 186  
 entities\_to\_ascii(), 194  
 force\_download(), 152  
 form\_button(), 170  
 form\_checkbox(), 167  
 form\_close(), 171  
 form\_dropdown(), 165  
 form\_error(), 64  
 form\_fieldset(), 167  
 form\_fieldset\_close(), 167  
 form\_hidden(), 162  
 form\_input(), 163  
 form\_multiselect(), 166  
 form\_open(), 161  
 form\_open\_multipart(), 162  
 form\_password(), 164  
 form\_prep(), 171  
 form\_radio(), 168  
 form\_reset(), 169  
 form\_submit(), 169  
 form\_textarea(), 165  
 form\_upload(), 164  
 get\_cookie(), 143  
 get\_dir\_file\_info(), 156  
 get\_file\_info(), 158  
 get\_mime\_by\_extension(), 159  
 heading(), 172  
 highlight\_code(), 194  
 img(), 173  
 increment\_string(), 188  
 index\_page(), 199  
 lang(), 180  
 link\_tag(), 174  
 local\_to\_gmt(), 145  
 log\_message(), 32  
 meta(), 177  
 nbs(), 175  
 now(), 144  
 octal\_permissions(), 160  
 ol(), 175  
 quotes\_to\_entities(), 191  
 random\_element(), 140  
 random\_string(), 187  
 read\_file(), 155  
 redirect(), 200  
 reduce\_double\_slashes(), 190  
 repeater(), 189  
 sanitize\_filename(), 184  
 send\_email(), 154  
 set\_cookie(), 143  
 set\_realpath(), 183  
 show\_404(), 32  
 show\_error(), 31  
 site\_url(), 197  
 standard\_date(), 144  
 strip\_image\_tags(), 185  
 strip\_quotes(), 191  
 symbolic\_permissions(), 159  
 timespan(), 146  
 timezone\_menu(), 148  
 timezones(), 148  
 trim\_slashes(), 190  
 ul(), 175  
 unix\_to\_human(), 146  
 uri\_string(), 198  
 valid\_email(), 153  
 validation\_errors(), 64  
 word\_limiter(), 192  
 word\_wrap(), 195  
 write\_file(), 154  
 xml\_convert(), 201  
 xss\_clean(), 184

**G**

## generowanie

adresu URL, 197  
 ciągu URI, 104  
 łączy, 93  
 nagłówek, 91  
 przycisku, 170  
 rezultatów zapytania, 128  
 strony, 90

grupowanie pól, 167

**H**

HTML, 171

**I**

identyfikacja przeglądarki, 105

informacje o

plikach, 157  
 polach tabeli, 135  
 przeglądarce, 112  
 tabelach, 134  
 typie bazy danych, 132  
 instalacja CodeIgnitera, 14



**J**

język, 88, 179

**K**

kalendarz, 39  
  opcje konfiguracji, 42  
  opcje szablonu, 43  
  opcje wyświetlania, 41  
  wyszczególnianie dat, 40  
katalog, 150  
  cache, 33  
  CodeIgniter, 19  
  CodeIgniterBlog, 203  
  controllers, 18  
  helpers, 28  
  libraries, 27  
  logs, 33  
  uploads, 55  
  views, 54  
klasa  
  array, 139  
  CI\_Model, 25  
  cookie, 143  
  date, 144  
  directory, 150  
  download, 152  
  email, 153  
  file, 154  
  form, 161  
  html, 171  
  language, 179  
  number, 181  
  parser, 101  
  path, 182  
  security, 95, 184  
  session, 97  
  string, 187  
  text, 192  
  uri, 103  
  url, 197  
  user\_agent, 105  
  xml, 200  
  zip, 113  
klasy  
  pomocnicze, 28, 139  
  systemowe, 37  
kodowanie, 53  
  UTF-8, 113  
  wyrazów, 185  
kolorowanie składni, 194  
kompresja plików, 113

konfiguracja, 43  
  automatycznego ładowania, 204  
  bazy danych, 205  
  domyślnego kontrolera, 205  
  kalendarza, 42  
  obiektu email, 49  
  obiektu upload, 56  
  sesji, 100  
kontroler, 11, 17, 19  
  edycja wpisów, 219  
  domyślny, 19  
  formularza, 59  
  kategoria wpisów, 212  
  logowanie administratora, 206  
  strona główna, 209  
  wpis, 215  
  wylogowanie administratora, 208  
konwersja znaczników, 186

**L**

LAMP, 203  
liczba, 181  
liczba rekordów, 132  
lista  
  rozwijana, 166  
  wielokrotnego wyboru, 166  
  wypunktowana, 176

**Ł**

ładowanie  
  automatyczne zasobów, 29  
  biblioteki, 27, 53  
  biblioteki dzielenia treści, 92  
  biblioteki email, 85  
  biblioteki table, 86  
  klasy, 85  
  klasy pomocniczej, 28, 87  
  modelu, 25, 87  
  obiektu encrypt, 51  
  obiektu form\_validation, 58  
  obiektu ftp, 64  
  obiektu zip, 113  
  pliku językowego, 88, 90  
  pliku konfiguracyjnego, 44  
  preferencji, 41  
  sterownika, 29  
  sterownika bazy danych, 120  
  wersji językowej, 89  
  widoku, 22, 23, 86  
łącza do stron, 93, 199

**M**

## metoda

- accept\_charset(), 113
- accept\_lang(), 112
- add\_data(), 114
- add\_dir(), 114
- add\_row(), 75
- affected\_rows(), 131
- agent\_string(), 111
- all\_userdata(), 99
- archive(), 116
- assoc\_to\_uri(), 104
- attach(), 48
- base\_url(), 46
- bcc(), 47
- browser(), 108
- cache(), 34
- cc(), 47
- chmod(), 72
- clear(), 48, 77
- clear\_data(), 117
- close(), 72, 124
- connect(), 65
- count\_all(), 132
- create\_links(), 93
- data(), 57
- database(), 87, 120, 122
- decode(), 52
- delete\_dir(), 69
- delete\_file(), 68
- display\_errors(), 57
- do\_upload(), 57
- download(), 66, 116
- enable\_profiler(), 91
- encode(), 51
- field\_data(), 137
- field\_exists(), 136
- free\_result(), 130
- from(), 47
- function(), 78
- generate(), 39, 74
- get(), 80
- get\_post(), 81
- get\_request\_header(), 84
- get\_zip(), 117
- helper(), 28, 87
- initialize(), 92
- insert\_id(), 131
- ip\_address(), 82
- is\_browser(), 106
- is\_mobile(), 107
- is\_referral(), 108
- is\_robot(), 107
- item(), 44
- language(), 87
- last\_query(), 133
- library(), 85
- line(), 89
- list\_fields(), 135
- list\_files(), 70
- list\_tables(), 134
- load(), 44, 89
- log\_message(), 33
- make\_columns(), 75
- message(), 48
- mirror(), 70
- mkdir(), 71
- mobile(), 109
- model(), 86
- move(), 68
- num\_rows(), 129
- parse(), 102
- platform(), 110, 132
- print\_debugger(), 48
- query(), 125, 126
- read\_dir(), 115
- read\_file(), 115
- reconnect(), 123
- referrer(), 111
- rename(), 67
- reply\_to(), 47
- request\_headers(), 84
- result(), 128
- result\_array(), 129
- robot(), 110
- run(), 62
- sanitize\_filename(), 96
- segment(), 103
- send(), 48
- set\_alt\_message(), 48
- set\_caption(), 74
- set\_content\_type(), 90
- set\_cookie(), 81
- set\_empty(), 76
- set\_header(), 91
- set\_heading(), 75
- set\_item(), 45
- set\_output(), 90
- set\_rules(), 62
- set\_status\_header(), 91
- set\_template(), 76
- set\_userdata(), 98
- sha1(), 53
- site\_url(), 45
- subject(), 47
- system\_url(), 46
- table\_exists(), 134
- unset\_userdata(), 100
- upload(), 66

- uri\_to\_assoc(), 103
- user\_agent(), 83
- userdata(), 98
- valid\_ip(), 83
- version(), 109, 133
- view(), 86
- xss\_clean(), 95
- metody
  - prywatne, 20
  - zabezpieczające dane, 127
- model, 11, 17, 24
  - edycja, 233
  - kategoria, 229
  - logowanie użytkownika, 227
  - polecenia współdzielone, 233
  - strona główna, 228
  - wpis, 231
- MVC, Model–View–Controller, 11, 17

## N

- nagłówek MIME, 159
- nagłówki, 84, 91
- narzędzie phpMyAdmin, 119, 204
- nazwy zastrzeżone, 259
- nazwy zastrzeżone właściwości, 30

## O

- obiekt
  - benchmark, 37
  - calendar, 39
  - email, 46, 49
  - encrypt, 50
  - form\_validation, 58
  - ftp, 64
  - input, 79
  - load, 85
  - modelu, 25
  - output, 90
  - table, 73
  - upload, 53, 56
  - zip, 113
- obsługa
  - bazy danych, 119
  - błędów, 31
- odczytywanie zawartości pliku, 155
- odszyfrowywanie, 52
- opcja
  - error\_reporting, 35
  - magic\_quotes\_runtime, 36
  - register\_globals, 35
- opcje walidacji formularza, 61

## P

- pamięć, 38
- plik, 154
  - .htaccess, 18, 205
  - autoload.php, 29, 44, 90, 204
  - config.php, 18, 204
  - database.php, 120, 122, 205
  - index.php, 31, 204
  - routes.php, 19, 30, 205
- pliki
  - cookie, 81, 143
  - dziennika, 33
  - konfiguracyjne, 43
  - kontrolerów, 20
  - modelu, 26
  - tymczasowe, 33, 35
  - widoku, 22, 102
- pobieranie, 152
  - adresu IP, 82
  - archiwum, 116
  - pliku, 66
  - pliku generowanego w locie, 152
  - zmiennej, 80
- podział na strony, 92
- poła tabeli, 135
- pole
  - tekstowe, 163
  - typu checkbox, 168
  - typu file, 164
  - typu radio, 168
  - ukryte, 162
  - wieloliniowe, 165
- połączenie z bazą danych, 120, 122
- pozycjonowanie strony, 29
- prawa dostępu do pliku, 160
- przeglądarka użytkownika, 105
- przekazywanie widoków, 24
- przenoszenie pliku, 68
- przesyłanie pliku, 53, 66

## R

- reguły walidacji, 59
- resetowanie formularza, 170
- rezultaty zapytań, 128

## S

- sesja, 97
- spolszczenie, 13
- sprawdzanie ścieżki dostępu, 183
- sterownik bazy danych, 120
- strefy czasowe, 149

struktura katalogu, 150  
szablon  
  kalendarza, 42  
  tabeli, 77  
szyfrowanie, 51

**Ś**

ścieżka, 182  
środowisko pracy, 203

**T**

tabele, 73, 134  
tablica, 139  
  \$\_POST, 80  
  \$config[], 43  
  \$route, 19, 30  
tekst, 192  
tworzenie  
  aplikacji, 203  
  archiwów, 116, 118  
  bazy danych, 204  
  formularza, 161  
  katalogu, 71  
  kolumn, 75  
  kopii katalogu, 71  
  listy, 176  
  łącza, 199

**U**

URI, 103  
URL, 197  
urządzenie mobilne, 107  
usuwanie  
  apostrofów, 191  
  danych z sesji, 100  
  katalogu, 69  
  plików, 68, 156  
  plików tymczasowych, 34  
  ukośników, 190  
  znaczników, 186

**V**

VPS, Virtual Private Server, 11

**W**

walidacja  
  adresu e-mail, 153  
  formularzy, 58

widok, 11, 17, 21  
  dodawanie kategorii, 246  
  edytowanie kategorii, 245  
  edytowanie wpisu, 251  
  formularz, 54  
  kategoria, 243  
  menu, 240  
  menu administratora, 242  
  nagłówek, 237  
  najnowsze wpisy, 253  
  nowy wpis, 249  
  stopka, 240  
  strona główna, 255  
  wpisy, 246  
  załoguj, 254  
  zmień hasło, 256  
wieloznaczniki, wildcards, 30  
własna metoda sprawdzająca, 63  
wprowadzanie danych, 79  
wydajność systemu, 37  
wydruk  
  nagłówka, 172  
  wartości pola, 171  
wygląd strony, 101  
wyrażenia regularne, 30  
wysyłanie  
  danych z formularza, 142, 169  
  wiadomości, 46, 49  
wyświetlanie  
  daty, 145  
  struktury katalogu, 150  
  wpisów, 212  
  znacznika czasu, 145

**X**

XAMPP, 203  
XML, 200  
XSS, Cross-Site Scripting, 79

**Z**

zabezpieczanie  
  danych, 126  
  nazwy pliku, 184  
zamykanie  
  formularza, 171  
  połączenia, 72  
zapis  
  do sesji, 98  
  tekstu do pliku, 154  
zapytania SQL, 119, 125  
zapytanie wiązane, 127

- zastrzeżone nazwy
  - funkcji, 259
  - kontrolerów, 259
  - stałych, 260
  - zmiennych, 260
- zawijanie wierszy, 49
- zmiana
  - nazwy pliku, 67
  - praw katalogu, 72
- zmienna
  - \$c, 18
  - \$m, 18
- znacznik
  - <br />, 172
  - <link />., 174
  - <META>, 177
  - DTD, 179
  - IMG, 173
- znaki ASCII, 193
- zużycie pamięci, 38



# PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



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

Zmień swoją stronę WWW  
w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

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

**CodeIgniter** to ni mniej, ni więcej, tylko doskonały framework, który przyda się wszystkim programistom języka PHP. Pozwala on na szybkie i efektywne zaimplementowanie tych elementów stron czy aplikacji internetowych, których samodzielne przygotowanie jest trudne i czasochłonne. Jego główne zalety to niewielkie wymagania co do serwera, na którym ma działać oprogramowanie, zapewnienie logicznego i wygodnego rozdziału elementów strony, brak potrzeby konfiguracji oraz świetna dokumentacja i życzliwa społeczność użytkowników, chętnie dzielących się z innymi swoim doświadczeniem w pracy z programem.

Ta książka, przeznaczona zarówno dla początkujących, jak i zaawansowanych programistów PHP, pozwoli Ci zapoznać się z wieloma aspektami używania CodeIgniter. Znajdziesz tu informacje o instalacji frameworka i jego strukturze. Zobaczysz, jak działają klasy systemu, jakie funkcje oraz metody warto wykorzystać w różnych sytuacjach i jak działają sterowniki systemu do bazy danych. Będziesz mógł także przeciwżyć tworzenie aplikacji z użyciem tego zestawu bibliotek. Programowanie z frameworkiem CodeIgniter to czysta przyjemność!

- Instalacja CodeIgniter na serwerze
- Używanie frameworka (model, widok, kontroler)
- Klasy systemu
- Sterowniki systemu dla bazy danych
- Pomocnicy (funkcje)
- Tworzenie pierwszej aplikacji

**Zyskaj czas z CodeIgniter!**

**helion.pl**  
księgarnia  
internetowa

Nr katalogowy: 8668



Księgarnia internetowa  
<http://helion.pl>



Zamówienia telefoniczne:  
**0 801 339900**



**0 601 339900**

Informatyka w najlepszym wydaniu



**Helion**

Sprawdź najnowsze promocje:

- <http://helion.pl/promocje>
- Książki najchętniej czytane:
- <http://helion.pl/bestsellery>
- Zamów informacje o nowościach:
- <http://helion.pl/novosci>

Helion SA  
ul. Kościuszki 1c, 44-100 Gliwice  
tel.: 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
<http://helion.pl>

sięgnij po WIĘCEJ



KOD KORZYŚCI

ISBN 978-83-246-4964-8



9 788324 649648

Cena: 47,00 zł