



# Model View Controller (MVC) + CodeIgniter (w bardzo wielkim skrócie)

---

dr inż. Artur Gramacki  
[a.gramacki@issi.uz.zgora.pl](mailto:a.gramacki@issi.uz.zgora.pl)

Instytut Sterowania i Systemów Informatycznych  
Uniwersytet Zielonogórski  
2017



# Metody wytwarzania aplikacji Web-owych

---

- Klasyczna
  - kody PHP osadzone są w plikach html
  - podstawowa wada: „wszystko w jednym pliku/ach”
- Z wykorzystaniem wzorców projektowych (Web Application Framework)
  - metodologia zalecana (a nawet niezbędna) przy tworzeniu dużych projektów
  - pozwala grupie programistów rozwijać wspólnie tą samą aplikację
  - łatwość i elastyczność wprowadzania zmian do aplikacji



# Web Application Framework

---

- Ułatwić wytwarzanie złożonych aplikacji internetowych
  - dominująca architektura to **M**odel **V**iew **C**ontroller (MVC)
- Ogromna ilość dostępnych rozwiązań
  - [http://en.wikipedia.org/wiki/Web\\_framework](http://en.wikipedia.org/wiki/Web_framework)
  - [http://en.wikipedia.org/wiki/Comparison\\_of\\_web\\_application\\_frameworks](http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks)
  - <http://php.pl/Wortal/Artykuly/Framework/Frameworki-dla-PHP-czyli-wydajne-tworzenie-aplikacji>
    - bardzo przystępnie napisany artykuł o tym, czym są Frameworki oraz jak zabrać się za tworzenie własnego

# Web Application Framework – rozwiązania dla PHP



Project	Start date	Current stable version	Release date	License
Agavi	2005-05	1.0.8 <sup>[15]</sup>	2015-06-29	LGPL
CakePHP	2005-08	3.5.0 <sup>[16]</sup>	2017-08-18 <sup>[±]</sup>	MIT
Codelgniter	2006-02-28	3.1.6 <sup>[17]</sup>	2017-09-25	MIT
Fat-Free	2009-09	3.6.0 <sup>[18]</sup>	2016-11-19	GPLv3
FuelPHP	2011-08	1.8 <sup>[19]</sup>	2016-04-09	MIT
Gyroscope	2008-11-20	8.8.0	2016-04-17	BSD
Jamroom	2003-07-28	6.1.0 <sup>[20]</sup>	2017-08-30	MPL
Kajona	2006	6.2 <sup>[21]</sup>	2017-06-08	LGPLv2
Kohana	2007-07	3.3.5 <sup>[22]</sup>	2016-03-10	BSD
Laravel	2011-06-11	5.5.0 <sup>[23]</sup>	2017-08-30	MIT
Li3 (Lithium)	2009-10	1.1.0 <sup>[24]</sup>	2017-04-23	BSD
Nette Framework	2006-01 <sup>[25]</sup>	2.4.0 <sup>[26]</sup>	2016-05-03	New BSD, GPLv2, GPLv3 <sup>[27]</sup>
Phalcon	2012-11-14	3.2.3 <sup>[28]</sup>	2017-10-12	BSD
Pop PHP	2012-03-19	3.6.1 <sup>[29]</sup>	2017-09-14	New BSD
PRADO	2004-01	3.3.2 <sup>[30]</sup>	2016-08-23	New BSD <sup>[31]</sup>
Silex	2011-09	2.0.0 <sup>[32]</sup>	2016-05-18	MIT
SilverStripe	2007-02-03	3.6.1 <sup>[33]</sup>	2017-06-27	BSD
Smart.Framework	2015-02-01	2.3.7.2 <sup>[34]</sup>	2016-09-27	BSD
Symfony	2005-10	3.3.9 <sup>[35]</sup>	2017-09-11	MIT
TwistPHP	2014-07	3.0.5 <sup>[36]</sup>	2017-01-11	GPLv3
TYPO3 Flow	2011-10	3.3.4 <sup>[37]</sup>	2016-09-29	LGPLv3
Yii	2008-12-03	2.0.13 <sup>[38]</sup>	2017-11-03	New BSD
Zend Framework	2006-03	3.0.0 <sup>[39]</sup>	2016-06-28	New BSD

# Frameworki dla PHP – porównanie funkcjonalności

Project	Language	AJax	MVC framework	MVC push-pull	11sn & L10n?	ORM	Testing framework(s)	DB migration framework(s)	Security framework(s)	Template framework(s)	Caching framework(s)	Form validation framework(s)	Scaffolding	RAD	Mobility
CakePHP 3	PHP >= 5.4 (M)	Any	Yes	Yes, Push & Cells	Yes	ORM, Data Mapper, Pattern, SQL, Relational Algebra, Abstraction Layer	Unit tests, object mocking, fixtures, code coverage, memory analysis with PHPUnit and Xdebug and Continuous Integration via Travis	Yes	CRUD based, ACL-based, Multiple Plugins	Themes, Layouts, Cells, Views, Elements, Plugins for Twig, Bootstrap, etc.	Memcache, Redis, XCache, APC, File	Validation via Contexts (Table (DAO), Entity (VO) & Controller, CSRF Protection	Plugin CRUD	Code Base	Mobile Agent Detection, Layouts
CodeIgniter	PHP >= 5.6.0 (M)	Any	Yes	Push	Mostly (M)	Third party only	Ready for next release	Yes	Yes	Yes	Yes	Yes	No (M)	Yes	Templates
Drupal	PHP	JQuery, jQuery UI, more	PAC	N/A	Yes	Optional module	SimpleTest	Yes	Yes	Yes	Memcache, APC, Varnish, more	Yes	No	No	Yes
FalFree Framework	PHP	Any	MVC, RMR	Push/pull	Yes	Data mappers for SQL, MongoDB, Flat-File	Built-in	Yes	Yes	Yes	APC, Memcache, XCache, WinCache, and Filesystem	Yes	No	?	?
FuelPHP	PHP >= 5.3.x	Yes	MVC, HMVC	Push	Yes	Yes	PHPUnit	Yes	Yes, Plugins available	Yes, Plugins available	File, Redis, Memcache, more	Yes	Yes	?	?
Fusebox	PHP	Yes	Not mandatory	Push	No, custom	?	?	?	Multiple plugins available	?	?	via qforms or built in PHP validation	Yes	?	?
Gyroscope	PHP >= 4	nano.js, replaceable (M)	LCHH	Push/pull	Mostly	Data-source agnostic	No	Built-in Scheme comparison tool and LDF editor	ACL-based, replaceable	Implementation-specific: helper functions and theme templates available	APC, Memcache	Yes	Interactive code generator	Yes	Dedicated mobile and tablet layouts, landscape-portrait transformation
Joomla	?	Yes	Plugin	?	?	?	?	?	?	?	?	?	?	?	?
Kajona	PHP >= 7	Any	Yes	Push	Yes	Yes	PHPUnit, Selenium, Jasmine	Yes	Yes	Yes	APC, Database, File	Yes	Yes	Yes	Bootstrap
Laravel	PHP >= 5.5.9	Any	Yes	Push	Yes	Yes	PHPUnit	Yes	Yes	Yes	APC, Database, File, Memcache, Redis	Yes	Yes	Yes	No
Li3 (Lithium)	PHP >= 5.3.6	Any	Yes	Push	Yes	Yes	Unit tests, builtin test framework or other independent	No	Yes, Plugins available	PHP, Twig, Plugin available	Memcache, Redis, XCache, APC, File	Yes, with CSRF Protection and Form Signing	No	Yes	?
Nette Framework	PHP >= 5.3.0	Toolkit-independent	MVP	Push	Yes	Third party only	Yes	No	Yes	Yes	Yes	Yes	No	?	?
Phalcon	PHP >= 5.5	Any	Yes	Push	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	?	?
Pop PHP	PHP >= 5.6.0	Any	Yes	Push	Yes	Yes	PHPUnit	Yes	ACL-based	Yes	APC, Database, File, Memcache, Redis, Session	Yes	?	Yes	?
PRADO	PHP >= 5.3.0	Prototype, script.aculo.us, own components (M)	No	Push/pull	Yes	Data access objects (DAO), active record pattern, SQLMap data mapper	PHPUnit, SimpleTest, Selenium	No	Yes	XML-based, similar to ASP.NET (M)	APC, Database, eAccelerator, Memcached, XCache	Yes (M)	Yes (M)	?	?
SilverStripe (sapphire)	PHP >= 5.2	JQuery, jQuery UI	Yes	Push/pull	Yes	Active record pattern	Unit tests, Selenium	Automatic	Incl. OpenID	Themes	Yes	Yes	Yes	Yes	Yes
Silex	PHP >= 5.3.9	Yes	Yes	Yes	Yes	Plugin exists (Doctrine)	Yes	No	Yes	PHP, Twig	Plugin exists	Yes	Plugin exists	?	?
SmartFramework	PHP >= 5.4.9	Yes	Yes	Yes	Yes	Yes (PostgresSQL, MySQL, SQLite, MongoDB, Solr, others via plugins)	Yes	No	Yes	Yes (Markers, Twig, others via plugins)	Yes (File, Redis, others via plugins)	Yes	No	Yes	Yes (jQuery mobile, Bootstrap, others via plugins)
Symfony	PHP 5	Prototype, script.aculo.us, Unobtrusive Ajax with UIJS and PUS plugins	Yes	Push	Yes	Propel, Doctrine (YAML)	Yes	Plugin exists (alpha code)	Plugin	PHP, Twig	Yes	Yes	Yes	?	?
Symfony 2	PHP >= 5.3.3	Any	Yes	Push	Yes	Propel, Doctrine (YAML)	Yes	Plugin exists	Yes	PHP, Twig	Yes	Yes	Yes	?	?
TwistPHP	PHP >= 5.3.3	Any	Yes	Push	Yes	Yes	PHPUnit via Travis	No	Yes	Yes	Yes	Yes	No	?	?
TYPO3	PHP >= 5.5	Any	Yes	Push/pull	Yes	Yes	Yes	Partial	Yes	TYPO3 Fluid	Yes	Yes	Plugin exists	Plugin exists	?
Yii	PHP >= 5.4	JQuery, jQuery UI, own components, plugins	Yes	Push/pull	Yes	Data Access Objects (DAO), Active Record Pattern, Plugins (incl. Doctrine 2.0)	PHPUnit, Selenium	Yes	ACL-based, RBAC-based, plugins	PHP-based, PRADO-like, plugins	APC, Database, eAccelerator, File, Memcache, Redis, WinCache, XCache, Zend Platform	Yes	Yes (M)	?	?
Zend Framework (M)	PHP >= 5.3	Toolkit-independent	Yes	Push/pull	Yes	Table and row data gateway or Doctrine	Unit tests, PHP Unit or other independent	Yes	ACL-based	Yes	APC, Database, File, Memcache, Zend Platform	Yes	Yes	?	?
Zend Framework 2	PHP >= 5.3.3	Toolkit-independent	Yes	Push/pull	Yes	Table and row data gateway and Doctrine 2.0 for Zend Framework 2.0	Unit tests, PHP Unit or other independent	Yes	ACL-based	Yes	APC, Database, File, Memcache, Zend Platform	Yes	Yes	?	?



# Model View Controller (MVC)

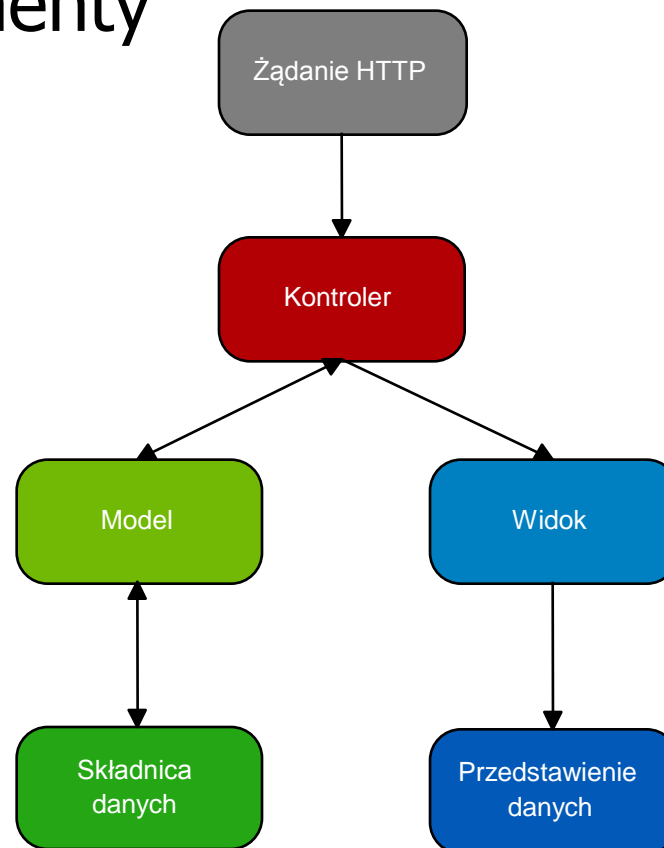
---

- Pozwala w miarę łatwo\* oddzielić od siebie następujące elementy
  - model danych
  - warstwę biznesową (logika aplikacji)
  - warstwę prezentacyjną (interfejs aplikacji)

\* „w miarę łatwo”, nie oznacza to samo, co „łatwo” ☺ Aby osiągnąć zamierzoną elastyczność, często należy się sporo napracować.

# Model View Controller (MVC)

- Wzorzec zakłada podział aplikacji na trzy funkcjonalne elementy
  - model
  - widok
  - kontroler





# Model View Controller (MVC)

---

## ■ MODEL

- odpowiada za logikę biznesową
- najczęściej reprezentuje strukturę danych z bazy
- zawierając dodatkowe metody pozwalające na wykonanie typowych operacji (pobieranie, dodawanie, aktualizowanie oraz usuwanie danych)





# Model View Controller (MVC)

---

## ■ WIDOK

- zwany jest również warstwą prezentacji
- odpowiedzialny za wyświetlanie danych (w tym interfejsu użytkownika)
- w przypadku aplikacji webowych widoki najczęściej generują kod HTML strony internetowej lub jej fragmentu, ale także dowolne inne dane, które ostatecznie mogą być interpretowane przez JavaScript po stronie klienta (np. dane w formacie XML jako odpowiedź na żądanie AJAX)



# Model View Controller (MVC)

---

## ■ KONTROLER

- odpowiada za przepływ danych z modelu do widoku
- warstwa ta odpowiedzialna jest również za obsługę żądań HTTP, którym w aplikacji internetowej może być np. odbieranie danych z formularza
- kontroler stanowi centralny punkt zarządzania programem

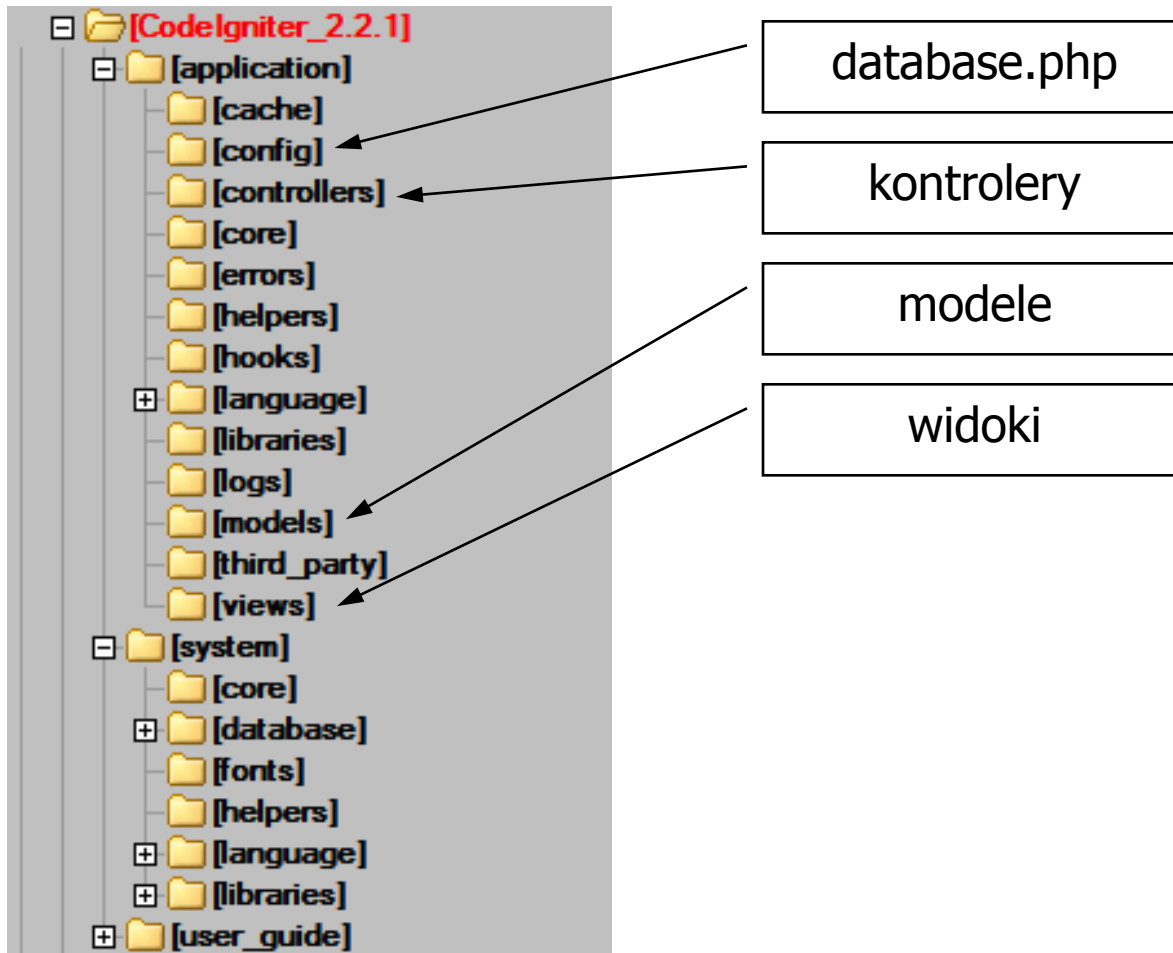


# Framework CodeIgniter



- open-source
- instalacja ogranicza się do skopiowania plików na serwer WWW obsługujący PHP (wer.  $\geq 4$ )
- architektura oparta na wzorcu MVC
- obsługuje wiele baz danych
- zarządzanie sesjami
- możliwość rozszerzenia bibliotek natywnych o dodatkową funkcjonalność
- szyfrowanie danych
- mechanizmy zabezpieczające przed atakami XSS
  - XSS – „wstrzyknięcie” szkodliwego kodu do zawartości atakowanej strony internetowej
- możliwość zapisu treści wygenerowanych widoków do plików, w celu ich przesłania przy powtórnym żądaniu
- możliwość generowania kodów HTML widoków na podstawie szablonów

# CodeIgniter – struktura katalogów (wersja 2.2.1)





# CodeIgniter, dokumentacja

## Table of Contents

---

### Basic Info

- [Server Requirements](#)
- [License Agreement](#)
- [Change Log](#)
- [Credits](#)

### Installation

- [Downloading CodeIgniter](#)
- [Installation Instructions](#)
- [Upgrading from a Previous Version](#)
- [Troubleshooting](#)

### Introduction

- [Getting Started](#)
- [CodeIgniter at a Glance](#)
- [CodeIgniter Cheatsheets](#)
- [Supported Features](#)
- [Application Flow Chart](#)
- [Model-View-Controller](#)
- [Architectural Goals](#)

### Tutorial

- [Introduction](#)
- [Static pages](#)
- [News section](#)
- [Create news items](#)
- [Conclusion](#)

### General Topics

- [CodeIgniter URLs](#)
- [Controllers](#)
- [Reserved Names](#)
- [Views](#)
- [Models](#)
- [Helpers](#)
- [Using CodeIgniter Libraries](#)
- [Creating Your Own Libraries](#)
- [Using CodeIgniter Drivers](#)
- [Creating Your Own Drivers](#)
- [Creating Core Classes](#)
- [Hooks - Extending the Core](#)
- [Auto-loading Resources](#)
- [Common Functions](#)
- [URI Routing](#)
- [Error Handling](#)
- [Caching](#)
- [Profiling Your Application](#)
- [Running via the CLI](#)
- [Managing Applications](#)
- [Handling Multiple Environments](#)
- [Alternative PHP Syntax](#)
- [Security](#)
- [PHP Style Guide](#)
- [Writing Documentation](#)

### Additional Resources

- [Community Forums](#)
- [Community Wiki](#)

### Class Reference

- [Benchmarking Class](#)
- [Calendar Class](#)
- [Cart Class](#)
- [Config Class](#)
- [Email Class](#)
- [Encryption Class](#)
- [File Uploading Class](#)
- [Form Validation Class](#)
- [FTP Class](#)
- [HTML Table Class](#)
- [Image Manipulation Class](#)
- [Input Class](#)
- [Javascript Class](#)
- [Loader Class](#)
- [Language Class](#)
- [Migration Class](#)
- [Output Class](#)
- [Pagination Class](#)
- [Security Class](#)
- [Session Class](#)
- [Trackback Class](#)
- [Template Parser Class](#)
- [Typography Class](#)
- [Unit Testing Class](#)
- [URI Class](#)
- [User Agent Class](#)
- [XML-RPC Class](#)
- [Zip Encoding Class](#)

### Driver Reference

- [Caching Class](#)
- [Database Class](#)
- [Javascript Class](#)

### Helper Reference

- [Array Helper](#)
- [CAPTCHA Helper](#)
- [Cookie Helper](#)
- [Date Helper](#)
- [Directory Helper](#)
- [Download Helper](#)
- [Email Helper](#)
- [File Helper](#)
- [Form Helper](#)
- [HTML Helper](#)
- [Inflector Helper](#)
- [Language Helper](#)
- [Number Helper](#)
- [Path Helper](#)
- [Security Helper](#)
- [Smiley Helper](#)
- [String Helper](#)
- [Text Helper](#)
- [Typography Helper](#)
- [URL Helper](#)
- [XML Helper](#)



# CodeIgniter, różne uwagi

---

- Pliki zapisywać w UTF-8 bez BOM
  - BOM (ang. Byte Order Mark) – znacznik kolejności bajtów. Jest to znak niedrukowalny używany w wielobajtowym kodowaniu znaków, który jest zapisywany na początku strumienia bajtów (pliku) i informuje, w jakiej kolejności należy ustawić bajty, aby odczytać kod znaku
  - w wypadku kodowania UTF-8 znacznik kolejności bajtów (BOM) nie jest niezbędny, ponieważ kolejność bajtów jest jednoznaczna
- Znak końca linii w stylu UNIX-owym (LF)



# CodeIgniter, różne uwagi

- Znacznik zamykający w PHP

**INCORRECT:**

```
<?php
```

```
echo "Here's my code!";
```

```
?>
```

**CORRECT:**

```
<?php
```

```
echo "Here's my code!";
```

```
/* End of file myfile.php */
```

```
/* Location: ./system/modules/mymodule/myfile.php */
```



# CodeIgniter, różne uwagi

## ■ Porównywanie

### **INCORRECT:**

```
// If 'foo' is at the beginning of the string, strpos will return a 0,  
// resulting in this conditional evaluating as TRUE  
if (strpos($str, 'foo') == FALSE)
```

### **CORRECT:**

```
if (strpos($str, 'foo') === FALSE)
```

### **INCORRECT:**

```
function build_string($str = "")  
{  
    if ($str == "") // uh-oh! What if FALSE or the integer 0 is passed as an argument?  
    {  
    }  
}
```

### **CORRECT:**

```
function build_string($str = "")  
{  
    if ($str === "")  
    {  
    }  
}
```





# CodeIgniter, różne uwagi

- Więcej przydatnych uwag w dokumentacji [user\\_guide/general/styleguide.html](http://user_guide/general/styleguide.html)

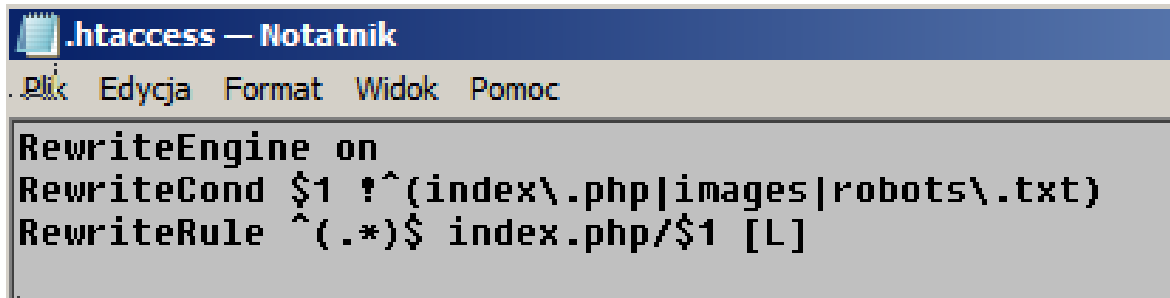
- ⇒ File Format
- ⇒ PHP Closing Tag
- ⇒ Class and Method Naming
- ⇒ Variable Names
- ⇒ Commenting
- ⇒ Constants
- ⇒ TRUE, FALSE, and NULL
- ⇒ Logical Operators
- ⇒ Comparing Return Values and Typcasting
- ⇒ Debugging Code
- ⇒ Whitespace in Files
- ⇒ Compatibility
- ⇒ Class and File Names using Common Words
- ⇒ Database Table Names

- ⇒ One File per Class
- ⇒ Whitespace
- ⇒ Line Breaks
- ⇒ Code Indenting
- ⇒ Bracket and Parenthetic Spacing
- ⇒ Localized Text
- ⇒ Private Methods and Variables
- ⇒ PHP Errors
- ⇒ Short Open Tags
- ⇒ One Statement Per Line
- ⇒ Strings
- ⇒ SQL Queries
- ⇒ Default Function Arguments



# Przykład, plik .htaccess

- Po co jest plik .htaccess?
  - usunięcie konieczności wpisywania frazy „index.php” w adresach URL (szczegóły na kolejnych slajdach)
  - czytaj user\_guide/general/urls.html
  - lokalizacja: główny katalog CI
- Przykład przetestowano na wersji CI\_2.2.1



```
.htaccess — Notatnik
Plik  Edycja  Format  Widok  Pomoc

RewriteEngine on
RewriteCond $1 !^(index\.php|images|robots\.txt)
RewriteRule ^(.*)$ index.php/$1 [L]
```

# Przykład 0

Nazwa pliku to **ctrl\_0.php**

Nazwa klasy MUSI być taka sama, jak nazwa pliku, z tą różnicą, że MUSI być pisana z dużej litery

```
1 <?php
2
3 class Ctrl_0 extends CI_Controller {
4
5     function index() {
6         echo "Hello, world!";
7     }
8 }
9
10 /* End of file ctrl_0.php */
11 /* Location: ./application/controllers/ctrl_0.php */
12
```

Domyślna metoda kontrolera.

Może być inna nazwa, ale wówczas trzeba ją podawać jawnie w URL. Czyli zamiast:

**[your-site-url]/index.php/ctrl\_0**

trzeba podawać:

**[your-site-url]/index.php/ctrl\_0/nazwa-metody**

Brak znacznika zamykającego PHP

[http://host:port/katalog\\_aplikacji/index.php/klasa/metoda/parametr/ ...](http://host:port/katalog_aplikacji/index.php/klasa/metoda/parametr/...)

**klasa** – nazwa wywoływanego kontrolera

**metoda**– metoda (funkcja) zdefiniowana w wywoływanym kontrolerze

**parametr** – jeden lub wiele parametrów wywoływanej metody (oddzielone ukośnikami)

# Przykład 1

```
<?php
class Ctrl_1 extends CI_Controller {

    function index() {
        $this->load->view('view_1');
    }
}
```

Domyślna metoda kontrolera.

Załadowanie widoku.

Wskaźnik `$this` wskazuje na obiekt, na którym wywołujemy daną metodę. Dzięki niemu możemy dostać się do wartości przechowywanych w polach obiektu oraz wywoływać inne metody. Służy do tego specjalny operator `->`

Nazwa zgodna z istniejącą nazwą pliku w katalogu views. W przeglądarce wpisujemy:

`[your-site-url]/index.php/ctrl_1`

Zwróćmy uwagę, że fragment "index.php" jest obowiązkowy. Można to usunąć edytując (tworząc) plik `.htaccess` (uwaga na kropkę na początku, to element nazwy pliku) w głównym katalogu CodeIgniter-a i wpisując tam:

`RewriteEngine on`

`RewriteCond $1 !^(index\.php|images|robots\.txt)`

`RewriteRule ^(.*)$ index.php/$1 [L]`

```
<div>
<?php
echo "Hello, world!";
?>
</div>
```

Plik widoku (`view_1.php`)

# Przykład 2

Konstruktor w programowaniu obiektowym to specjalna **METODA** danej **KLASY**, wywoływana podczas tworzenia jej **INSTANCJI**. Podstawowym zadaniem konstruktora jest zainicjowanie obiektu. W PHP konstruktor MUSI nosić nazwę `__construct`. W nazwie są dwa (2) podkreślniki (`_`).

Wywołanie konstruktora klasy nadrzędnej.  
(CI\_Controller)

Inicjujemy zmienną `$today`

Tworzymy tablicę asocjacyjną z danymi (tu: dwa rekordy), które chcemy przesłać do widoku. Najpierw nazwa głównego widoku. Następnie zmienna z bieżącą datą (ustawiona w konstruktorze).

Teraz możemy już załadować widok z szablonem strony (plik `template.php`). Do widoku przesyłamy tablicę asocjacyjną `$data`.

```
<?php
class Ctrl_2 extends CI_Controller{

    function __construct(){
        parent::__construct();
        $this->today = date('d-m-Y');
    }

    function index(){
        $data['main_content'] = 'view_2';
        $data['today'] = $this->today;
        $this->load->view('template', $data);
    }
}
?>
```

```
<?php
$this->load->view('header');
$this->load->view($main_content);
$this->load->view('footer');
?>
```

```
<html>
<head>
    <title>Artur Gramacki</title>
</head>
<body>

</body>
</html>
```

Plik **template.php**

Plik **header.php**

Plik **footer.php**

```
<div>
<?php
echo "Dzisiejsza data to: ";
echo $today;
?>
</div>
```

Plik **view\_2.php**



## Przykład 2

---

- Tablica asocjacyjna (tablica skojarzeniowa, ang. associative array). Nazwa dla powszechnie stosowanego w informatyce abstrakcyjnego typu danych, który przechowuje pary (unikatowy klucz, wartość) i umożliwia dostęp do wartości poprzez podanie klucza.
  - [http://pl.wikipedia.org/wiki/Tablica\\_asocjacyjna](http://pl.wikipedia.org/wiki/Tablica_asocjacyjna)

# Przykład 3, plik database.php

(pokazano tylko istotny fragment pliku)

```
$db['default']['hostname'] = '  
(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP) (HOST = localhost) (PORT = 1521))  
(CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = ora10gr2)))  
';  
$db['default']['username'] = "artur";  
$db['default']['password'] = "artur";  
$db['default']['dbdriver'] = "oci8";  
$db['default']['pconnect'] = TRUE;  
$db['default']['db_debug'] = TRUE;  
$db['default']['cache_on'] = FALSE;  
$db['default']['cachedir'] = "";
```

ORACLE

```
$db['default']['hostname'] = 'localhost';  
$db['default']['username'] = 'artur';  
$db['default']['password'] = 'artur';  
$db['default']['database'] = 'artur';  
$db['default']['dbdriver'] = 'mysql';  
$db['default']['dbprefix'] = '';  
$db['default']['pconnect'] = TRUE;  
$db['default']['db_debug'] = TRUE;  
$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;
```

MySQL

# Przykład 3, plik database.php

```
| ['hostname'] The hostname of your database server.
| ['username'] The username used to connect to the database
| ['password'] The password used to connect to the database
| ['database'] The name of the database you want to connect to
| ['dbdriver'] The database type. ie: mysql. Currently supported:
|               mysql, mysqli, postgres, odbc, mssql, sqlite, oci8
| ['dbprefix'] You can add an optional prefix, which will be added
|               to the table name when using the Active Record class
| ['pconnect'] TRUE/FALSE - Whether to use a persistent connection
| ['db_debug'] TRUE/FALSE - Whether database errors should be displayed.
| ['cache_on'] TRUE/FALSE - Enables/disables query caching
| ['cachedir'] The path to the folder where cache files should be stored
| ['char_set'] The character set used in communicating with the database
| ['dbcollat'] The character collation used in communicating with the database
|               NOTE: For MySQL and MySQLi databases, this setting is only used
|               as a backup if your server is running PHP < 5.2.3 or MySQL < 5.0.7
|               (and in table creation queries made with DB Forge).
|               There is an incompatibility in PHP with mysql_real_escape_string() which
|               can make your site vulnerable to SQL injection if you are using a
|               multi-byte character set and are running versions lower than these.
|               Sites using Latin-1 or UTF-8 database character set and collation are unaffected.
| ['swap_pre'] A default table prefix that should be swapped with the dbprefix
| ['autoinit'] Whether or not to automatically initialize the database.
| ['stricton'] TRUE/FALSE - forces 'Strict Mode' connections
|               - good for ensuring strict SQL while developing
|
| The $active_group variable lets you choose which connection group to
| make active. By default there is only one group (the 'default' group).
|
| The $active_record variables lets you determine whether or not to load
| the active record class
```



# Przykład 3, tabela bazodanowa

```
CREATE TABLE emp (  
    empno INT PRIMARY KEY,  
    ename VARCHAR(10),  
    job VARCHAR(9),  
    mgr INT,  
    hiredate DATE,  
    sal NUMERIC(7,2),  
    comm NUMERIC(7,2),  
    deptno INT  
) ENGINE = InnoDB DEFAULT CHARSET=utf8;  
  
INSERT INTO emp VALUES (7369, 'SMITH', 'CLERK', 7902, '1980-12-17', 800, NULL, 20);  
INSERT INTO emp VALUES (7499, 'ALLEN', 'SALESMAN', 7698, '1981-02-20', 1600, 300, 30);  
INSERT INTO emp VALUES (7521, 'WARD', 'SALESMAN', 7698, '1981-02-22', 1250, 500, 30);  
INSERT INTO emp VALUES (7566, 'JONES', 'MANAGER', 7839, '1981-04-02', 2975, NULL, 20);  
INSERT INTO emp VALUES (7654, 'MARTIN', 'SALESMAN', 7698, '1981-09-28', 1250, 1400, 30);  
INSERT INTO emp VALUES (7698, 'BLAKE', 'MANAGER', 7839, '1981-05-01', 2850, NULL, 30);  
INSERT INTO emp VALUES (7782, 'CLARK', 'MANAGER', 7839, '1981-06-09', 2450, NULL, 10);  
INSERT INTO emp VALUES (7788, 'SCOTT', 'ANALYST', 7566, '1987-07-13', 3000, NULL, 20);  
INSERT INTO emp VALUES (7839, 'KING', 'PRESIDENT', NULL, '1981-11-17', 5000, NULL, 10);  
INSERT INTO emp VALUES (7844, 'TURNER', 'SALESMAN', 7698, '1981-09-08', 1500, 0, 30);  
INSERT INTO emp VALUES (7876, 'ADAMS', 'CLERK', 7788, '1987-07-13', 1100, NULL, 20);  
INSERT INTO emp VALUES (7900, 'JAMES', 'CLERK', 7698, '1981-12-03', 950, NULL, 30);  
INSERT INTO emp VALUES (7902, 'FORD', 'ANALYST', 7566, '1981-12-03', 3000, NULL, 20);  
INSERT INTO emp VALUES (7934, 'MILLER', 'CLERK', 7782, '1982-01-23', 1300, NULL, 10);
```

UTF-8

# Przykład 3, kontroler

```
<?php
class Emp_controller extends CI_Controller {
    function __construct() {
        parent::__construct();
        // Załadowanie całej infrastruktury potrzebnej do współpracy z bazami danych
        $this->load->database();
    }

    // Domyślna metoda kontrolera.
    function index() {
        // Załadowanie modelu - nazwa zgodna z istniejącą
        // nazwą pliku w katalogu models.
        $this->load->model('emp_model');

        // Odwołanie się do metody (zdefiniowanej w emp_model), która zwraca listę pracowników.
        // Wynik zapisujemy do tablicy asocjacyjnej.
        //
        // Tablica z wynikami dostępna jest w miejscu wywołania metody modelu,
        // czyli w kontrolerze, który następnie przekazuje ją do widoku.
        $view_data['EMP_LIST'] = $this->emp_model->get_emp();

        // Dodatkowo tworzymy sobie zmienną z bieżącą datą.
        $view_data['TODAY'] = date('d-m-Y');

        // Załadowanie widoku, do którego prześlemy listę pracowników.
        // Do widoku przesyłamy zdefiniowaną wcześniej tablicę z danymi.
        // Nazwa zgodna z istniejącą nazwą pliku w katalogu views.
        $this->load->view('emp_view', $view_data);
    }
}
?>
```

# Przykład 3, model

```
<?php
class Emp_model extends CI_Model {

    function __construct() {
        parent::__construct();
    }

    // Metoda zwraca listę pracowników z tabeli PRAC.
    function get_emp() {

        // Wybór pól dla klauzuli SELECT.
        $this->db->select('ENAME, HIREDATE, SAL');

        // Wybór tabeli dla klauzuli FROM.
        $this->db->from('EMP');

        // Wybór kolumny dla klauzuli ORDER BY.
        $this->db->order_by('ENAME');

        // Pobranie danych z bazy.
        $query = $this->db->get();

        // Zwraca tablicę z listą pracowników.
        // Jeśli pobrano przynajmniej jeden rekord.
        if ($query->num_rows() > 0)
            $smp = $query->result_array();

        // Zwraca tablicę z pracownikami.
        return $smp;
    }
}
```

Używamy klasy ActiveRecord.

ActiveRecord kolekcjonuje szereg akcji, które w momencie wywołania metody `get()` konwertowane są na kod SQL zgodny z używaną bazą danych.

Jeśli wszystkie zapytania do bazy danych realizowane są przy pomocy ActiveRecord aplikacja staje się **niezależna od zastosowanego systemu bazodanowego**.

Wywołanie metody `get()`

Uwaga, CI definiuje też metodę `result()`. Patrz:

[http://www.codeigniter.com/user\\_guide/database/results.html](http://www.codeigniter.com/user_guide/database/results.html)

Wówczas w widoku zamiast `$EL['ENAME']` będzie trzeba wpisać `$EL->ENAME`.

# Przykład 3, widok

```
<html>
<head>
  <title>Pracownicy</title>
</head>
<body>
  <?php echo $TODAY; ?>
  <table>
    <tr>
      <th> # </th>
      <th> ENAME </th>
      <th> HIREDATE </th>
      <th> SAL </th>
    </tr>
    <?php $i = 1; ?>
    <!-- Odwołujemy się do zmiennej EMP_LIST -->
    <?php foreach ($EMP_LIST as $EL) { ?>
      <tr>
        <td><?php echo ($i); $i = $i + 1 ?></td>
        <td><?php echo $EL['ENAME'] ?></td>
        <td><?php echo $EL['HIREDATE'] ?></td>
        <td><?php echo $EL['SAL'] ?></td>
      </tr>
    <?php } ?>
  </table>
</body>
</html>
```

Plik widoku nie jest klasą, lecz szablonem HTML, który pomiędzy kodem HTML generuje dynamiczną zawartość na podstawie danych dostarczonych w tablicy asocjacyjnej przesłanej jako parametr

Odwołujemy się do zmiennej `$TODAY`, która została przesłana do widoku za pośrednictwem tablicy asocjacyjnej `$view_data`.

Elementy tablicy asocjacyjnej przesłanej do widoku przypisywane są do zmiennych, których nazwy odpowiadają kluczom tej tablicy

Odwołujemy się do zmiennej `$EMP_LIST`, która została przesłana do widoku za pośrednictwem tablicy asocjacyjnej `$view_data`.

# Przykład 3, kontroler + parser

```
<?php
class Emp_controller extends CI_Controller {
    function __construct() {
        parent::__construct();
        $this->load->database();
    }

    // Domyślna metoda kontrolera.
    function index() {
        $this->load->model('emp_model');
        $view_data['EMP_LIST'] = $this->emp_model->get_emp();
        $view_data['TODAY'] = date('d-m-Y');

        $this->load->view('emp_view', $view_data);

        // Inny sposób obsługi widoków z użyciem "Template Parser Class".
        // Umożliwia wyświetlanie treści przekazanych do widoku w sposób
        // nieco bardziej estetyczny, niż mieszanina kodu HTML z topornymi
        // wstawkami PHP.
        // Załadowanie biblioteki parsującej plik widoku.
        $this->load->library('parser');
        $this->parser->parse('emp_view_2', $view_data);
    }
}
```

Tak było poprzednio, bez  
używania parsera.

Tak jest teraz

# Przykład 3, widok + parser

```
<!--
Klamra {EMP_LIST}/{EMP_LIST}, jest odpowiednikiem pętli po wszystkich elementach
tablicy $view_data['EMP_LIST']. Jeśli w zawartości, która ujęta została w tą klamrę,
umieszczone zostaną kolejne pseudo-zmienne, parser zinterpretuje je jako nazwy kluczy
tablicy asocjacyjnej, będącej jednym z elementów tablicy $view_data['EMP_LIST']. Stąd
odwołanie do {NAZWA} jest równoznaczne z $view_data['EMP_LIST'][index]['NAZWA'].
-->
<html>
<head>
  <title>Pracownicy</title>
</head>
<body>
  <?php echo $TODAY; ?>
  <table>
    <tr>
      <th> ENAME </th>
      <th> HIREDATE </th>
      <th> SAL </th>
    </tr>
    {EMP_LIST}
    <tr>
      <td>{ENAME}</td>
      <td>{HIREDATE}</td>
      <td>{SAL}</td>
    </tr>
  </EMP_LIST>
  </table>
</body>
</html>
```

```
<html>
<head>
  <title>Pracownicy</title>
</head>
<body>
  <?php echo $TODAY; ?>
  <table>
    <tr>
      <th> # </th>
      <th> ENAME </th>
      <th> HIREDATE </th>
      <th> SAL </th>
    </tr>
    <?php $i = 1; ?>
    <!-- Odwołujemy się do zmiennej EMP_LIST -->
    <?php foreach ($EMP_LIST as $EL) { ?>
      <tr>
        <td><?php echo ($i); $i = $i + 1 ?></td>
        <td><?php echo $EL['ENAME'] ?></td>
        <td><?php echo $EL['HIREDATE'] ?></td>
        <td><?php echo $EL['SAL'] ?></td>
      </tr>
    <?php } ?>
  </table>
</body>
</html>
```

# Przykład 3, wynik

26-02-2015

#	ENAME	HIREDATE	SAL
1	ADAMS	1987-07-13	1100.00
2	ALLEN	1981-02-20	1600.00
3	BLAKE	1981-05-01	2850.00
4	CLARK	1981-06-09	2450.00
5	FORD	1981-12-03	3000.00
6	JAMES	1981-12-03	950.00
7	JONES	1981-04-02	2975.00
8	KING	1981-11-17	5000.00
9	MARTIN	1981-09-28	1250.00
10	MILLER	1982-01-23	1300.00
11	SCOTT	1987-07-13	3000.00
12	SMITH	1980-12-17	800.00
13	TURNER	1981-09-08	1500.00
14	WARD	1981-02-22	1250.00

Wersja bez parsera.

26-02-2015

ENAME	HIREDATE	SAL
ADAMS	1987-07-13	1100.00
ALLEN	1981-02-20	1600.00
BLAKE	1981-05-01	2850.00
CLARK	1981-06-09	2450.00
FORD	1981-12-03	3000.00
JAMES	1981-12-03	950.00
JONES	1981-04-02	2975.00
KING	1981-11-17	5000.00
MARTIN	1981-09-28	1250.00
MILLER	1982-01-23	1300.00
SCOTT	1987-07-13	3000.00
SMITH	1980-12-17	800.00
TURNER	1981-09-08	1500.00
WARD	1981-02-22	1250.00

Wersja z parserem. Nie ma tutaj jawnej pętli w pliku widoku, więc numery kolejne rekordów trzeba by nieco inaczej stworzyć (w pliku modelu, nieco inaczej konstruując zapytanie SQL).

# Przykład 4, bardziej złożony

## # ENAME HIREDATE SAL

1	ADAMS	1987-07-13	1100.00	<a href="#">Edytuj</a>	<a href="#">Kasuj</a>
2	ALLEN	1981-02-20	1600.00	<a href="#">Edytuj</a>	<a href="#">Kasuj</a>
3	BLAKE	1981-05-01	2850.00	<a href="#">Edytuj</a>	<a href="#">Kasuj</a>
4	CLARK	1981-06-09	2450.00	<a href="#">Edytuj</a>	<a href="#">Kasuj</a>
5	FORD	1981-12-03	3000.00	<a href="#">Edytuj</a>	<a href="#">Kasuj</a>
6	JAMES	1981-12-03	950.00	<a href="#">Edytuj</a>	<a href="#">Kasuj</a>
7	JONES	1981-04-02	2975.00	<a href="#">Edytuj</a>	<a href="#">Kasuj</a>
8	KING	1981-11-17	5000.00	<a href="#">Edytuj</a>	<a href="#">Kasuj</a>
9	MARTIN	1981-09-28	1250.00	<a href="#">Edytuj</a>	<a href="#">Kasuj</a>
10	MILLER	1982-01-23	1300.00	<a href="#">Edytuj</a>	<a href="#">Kasuj</a>
11	SCOTT	1987-07-13	3000.00	<a href="#">Edytuj</a>	<a href="#">Kasuj</a>
12	SMITH	1980-12-17	800.00	<a href="#">Edytuj</a>	<a href="#">Kasuj</a>
13	TURNER	1981-09-08	1500.00	<a href="#">Edytuj</a>	<a href="#">Kasuj</a>
14	WARD	1981-02-22	1250.00	<a href="#">Edytuj</a>	<a href="#">Kasuj</a>

[Dodaj](#)

.../emp\_controller\_2/c\_select

## Tworzenie nowego rekordu w tablicy EMP

EMPNO

ENAME

.../emp\_controller\_2/c\_insert

## Edycja pracownika o numerze ID = 7876

EMPNO

ENAME

JOB

MGR

HIREDATE

SAL

COMM

.../emp\_controller\_2/c\_update\_form/7876



# Przykład 4, bardziej złożony

```
<?php
class Emp_controller_2 extends CI_Controller {
    function __construct() {
        parent::__construct();
        $this->load->database();
        $this->load->model('emp_model_2');
    }

    // Wyświetlanie danych z tabeli.
    function c_select() {
        ...
    }

    // Wyświetlanie rekordu o wskazanym ID.
    function c_select_by_id($id) {
        ...
    }

    // Dodawanie nowego rekordu.
    function c_insert() {
        ...
    }

    // Formularz do edycji rekordów.
    function c_update_form($id) {
        ...
    }

    // Edycja rekordu.
    function c_update() {
        ...
    }

    // Kasowanie rekordu o wskazanym ID.
    function c_delete($id) {
        ...
    }
}
```

```
<?php
class Emp_model_2 extends CI_Model {
    function __construct() {
        parent::__construct();
    }

    // Metoda zwraca listę pracowników
    function m_select() {
        ...
    }

    // Pobieramy jeden rekord.
    function m_select_by_id($id) {
        ...
    }

    // Wstawiamy jeden rekord.
    function m_insert() {
        ...
    }

    // Modyfikujemy jeden rekord.
    function m_update() {
        ...
    }

    // Kasujemy jeden rekord.
    function m_delete($id) {
        ...
    }
}
```

Uwaga: w kontrolerze NIE  
ZDEFINIOWANO domyślnej metody  
index (nie ma takiego wymogu).

emp_view_2_failure.php	40
emp_view_2_footer.php	171
emp_view_2_header.php	825
emp_view_2_insert.php	641
emp_view_2_link.php	119
emp_view_2_select.php	1 047
emp_view_2_select_by_id.php	453
emp_view_2_success.php	38
emp_view_2_update.php	1 468

# Przykład 4, bardziej złożony

```
// Wyświetlanie danych z tabeli.
```

```
function c_select() {  
    $view_data['emp_list'] = $this->emp_model_2->m_select();  
    $this->load->view('emp_view_2_header');  
    $this->load->view('emp_view_2_select', $view_data);  
    $this->load->view('emp_view_2_footer');  
}
```

emp\_view\_2\_select.php  
(fragment)

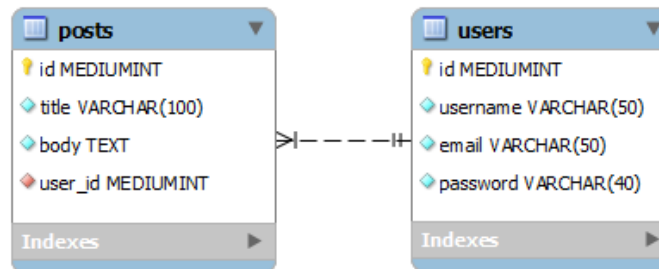
```
// Metoda zwraca listę pracowników z tabeli PRAC.
```

```
function m_select() {  
  
    $this->db->select('empno, ename, hiredate, sal');  
    $this->db->from('emp');  
    $this->db->order_by('ename');  
    $query = $this->db->get();  
  
    // Jeśli pobrano przynajmniej jeden rekord.  
    if ($query->num_rows() > 0)  
        $emp = $query->result_array();  
  
    return $emp;  
}
```

```
<table>  
    <tr>  
        <th> # </th>  
        <th> ENAME </th>  
        <th> HIREDATE </th>  
        <th> SAL </th>  
        <th> &nbsp; </th>  
    </tr>  
    <?php $i = 1; ?>  
    <!-- Odwołujemy się do zmiennej EMP_LIST -->  
    <?php foreach ($emp_list as $sel) { ?>  
        <tr>  
            <td> <?php echo ($i); $i = $i + 1 ?> </td>  
            <td> <?php echo $sel['ename'] ?> </td>  
            <td> <?php echo $sel['hiredate'] ?> </td>  
            <td> <?php echo $sel['sal'] ?> </td>  
            <td> <a href="c_update_form/<?php echo $sel['empno'] ?>">Aktualizuj </a>  
            <td> <a href="c_delete/<?php echo $sel['empno'] ?>">Usuń </a>  
        </tr>  
    <?php } ?>  
    <tr><td colspan="4">  
        <a href="c_insert"> Dodaj </a>  
    </td></tr>  
</table>
```

# Przykład 5, prosty blog

- [www.codeigniter.org.pl/category/tutoriale/](http://www.codeigniter.org.pl/category/tutoriale/)
- Prosta obsługa blogów
- Aplikacja „z logowaniem” i obsługą bazy danych



- Korzysta z frameworka **Bootstrap**
  - bardzo ułatwia tworzenie bardzo atrakcyjnych wizualnie aplikacji internetowych

# Przykład 5, prosty blog

Blog tutorial

Logowanie

Email

Hasło

Zaloguj

© 2012 CodeIgniter.org.pl

Blog tutorial

Logowanie przebiegło pomyślnie!

Brak wpisów w bazie danych.

© 2012 CodeIgniter.org.pl

Blog tutorial

Lista

Dodaj

Wyloguj

## Dodaj wpis

Tytuł

Mój pierwszy blog

Treść

Dodaj tekst...

Dodaj wpis

Anuluj

## Dostępne wpisy

### Mój pierwszy blog

CodeIgniter is a toolkit for people who build web applications using PHP. Its goal is to enable you to develop projects much faster than you could if you were writing code from scratch, by providing a rich set of libraries for commonly needed tasks, as well as a simple interface and logical structure to access these libraries. CodeIgniter lets you creatively focus on your project by minimizing the amount of code needed for a given task.

Edytuj