# LEARNING cakephp

#cakephp

# Table of Contents

# About

# Chapter 1: Getting started with cakephp

## Remarks

This section provides an overview of what cakephp is, and why a developer might want to use it.

It should also mention any large subjects within cakephp, and link out to the related topics. Since the Documentation for cakephp is new, you may need to create initial versions of those related topics.

## Versions

| Version | Release Date |
|---|---|
| 1.2.0 | 2008-12-26 |
| 1.3.0 | 2010-04-25 |
| 2.0.0 | 2011-10-17 |
| 3.0.0 | 2015-03-22 |

## Examples

### Installation or Setup

## Requirements

The following setup guide is for cakephp 2.8 and above. All cakephp versions lower than 2.8 are not compatible with php 7

HTTP Server. For example: Apache. Having mod_rewrite is preferred, but by no means required.

- PHP 5.5.9 or greater (including PHP 7).
- mbstring PHP extension
- intl PHP extension

> **Attention!** In both XAMPP and WAMP, the mbstring extension is working by default. In XAMPP, intl extension is included but you have to uncomment extension=php_intl.dll in php.ini and restart the server through the XAMPP Control Panel. In WAMP, the intl extension is “activated” by default but not working. To make it work you have to go to php folder (by default) C:\wamp\bin\php\php{version}, copy all the files that looks like icu*.dll and paste them into the apache bin directory C:\wamp\bin\apache\apache{version}\bin. Then restart all services and it should be

OK.

While a database engine isn’t required, we imagine that most applications will utilize one. CakePHP supports a variety of database storage engines:

- MySQL (5.1.10 or greater)
- PostgreSQL
- Microsoft SQL Server (2008 or higher)
- SQLite 3

### CakePHP3 Folder Structure

After you’ve downloaded, these are the files and folders you should see:

- The **bin folder** holds the Cake console executables.
- The **config folder** holds the Configuration files CakePHP uses. Database connection details, bootstrapping, core configuration files and more should be stored here.
- The **plugins folder** is where the Plugins your application uses are stored.
- The **logs folder** normally contains your log files, depending on your log configuration.
- The **src folder** will be where your application’s files will be placed.
- The **tests folder** will be where you put the test cases for your application.
- The **tmp folder** is where CakePHP stores temporary data. The actual data it stores depends on how you have CakePHP configured, but this folder is usually used to store model descriptions and sometimes session information.
- The **vendor folder** is where CakePHP and other application dependencies will be installed. Make a personal commitment not to edit files in this folder.
- The **webroot directory** is the public document root of your application. It contains all the files you want to be publically reachable.

  Make sure that the **tmp** and **logs** folders exist and are writable, otherwise the performance of your application will be severely impacted. In debug mode, CakePHP will warn you, if it is not the case.

## Inside src Folder

CakePHP’s src folder is where you will do most of your application development.

**Console** folder contains the console commands and console tasks for your application. For more information see Shells, Tasks & Console Tools.

**Controller** folder contains your application’s controllers and their components.

**Locale** folder stores string files for internationalization.

**Model** folder contains your application’s tables, entities and behaviors.

**View -** presentational classes are placed here: cells, helpers, and template files.

**Template** - presentational files are placed here: elements, error pages, layouts, and view template files.

### Basic first empty project

## Initial Creation and Download (CakePHP 3.x)

The easiest way to create a new CakePHP project is via Composer (if you don't know about composer look here for more info)

### Install Composer

If you need to install it and are on a windows machine follow this guide

If you are on Linux/Unix/OSX follow this guide

### Create the first CakePHP Project

Open a console window and navigate to your installation of php (on Windows with the default xampp installation this is `C:\xampp\php`)

To create an empty project then run the following command:

```
php composer.phar create-project --prefer-dist cakephp/app name_of_your_project
```

### Baking/Model/View/Controllers

The magic of CakePHP is baking - an automated generation of controller, model and view code with basic CRUD options.

Before baking you need to have your database connection configured. To do this you need to edit the file `config/app.php` in your project.

```
'Datasources' => [
'default' => [
    'className' => 'Cake\Database\Connection',
    'driver' => 'Cake\Database\Driver\Mysql',
    'persistent' => false,
    'host' => 'localhost',
    'username' => 'my_app', //in basic xampp: root
    'password' => 'sekret', //in basic xampp: ''
    'database' => 'my_app', //name of the database you want to connect to your project
    'encoding' => 'utf8',
    'timezone' => 'UTC',
    'cacheMetadata' => true,
]
```

],

If your database is connected correctly you then enter `bin/cake bake`in the root folder of your project in a console window.

This should output something like this:

```
Welcome to CakePHP v3.1.6 Console
---------------------------------------------------------------
App : src
Path: /var/www/cakephp.dev/src/
PHP: 5.5.8
---------------------------------------------------------------
The following commands can be used to generate skeleton code for your application.

Available bake commands:

- all
- behavior
- cell
- component
- controller
- fixture
- form
- helper
- mailer
- migration
- migration_snapshot
- model
- plugin
- shell
- shell-helper
- template
- test

By using `cake bake [name]` you can invoke a specific bake task.
```

For simplicity purposes we are going to bake everything with the default settings. To do this you enter

```
cake bake all
```

This will output something along those lines:

```
Welcome to CakePHP v3.2.11 Console
---------------------------------------------------------------
App : src
Path: C:\xampp\htdocs\tipping\src\
PHP : 5.6.15
---------------------------------------------------------------
Bake All
---------------------------------------------------------------
Possible model names based on your database:
- users
- blogs
Run `cake bake all [name]` to generate skeleton files.
```

By running `cake bake all <modelNameYouWantToBake>` the model, table, controller, fixture and view files are created. Run this for every possible model name and you have a functioning Project with basic CRUD options.

Now you can open your browser and see how it looks and start extending the project by your own logic

## Requirements

```
1-HTTP Server. For example: Apache. Having mod_rewrite is preferred, but by no means required.
2-PHP 5.5.9 or greater (including PHP 7)
3-mbstring PHP extens ion
4-intl PHP extension
```

I usually make an apache and mysql installation on a linuxbox. I can use windows too, however I do not recommend it ;) So, I usually make a new entry into the /etc/hosts file to make a sitename available to cakephp.

```
127.0.0.1   localhost caketest.local
```

next step to copy all cakephp files into a subdirectory inside /home/myusername/public_html/caketest

```
app
cake
index.php
plugins
README
vendors
.htaccess
```

then I set up the site to apache (not neccessary),

<VirtualHost *:80> DocumentRoot "/home/myusername/public_html/caketest" ServerName caketest.local
# This should be omitted in the production environment SetEnv APPLICATION_ENV development

```
<Directory "/home/myusername/public_html/caketest">
Options Indexes MultiViews FollowSymLinks
AllowOverride All
Order allow,deny
Allow from all
</Directory>
```

restart apache. you also need to edit the .htaccess files and place a RewriteBase directive with hte path to the actual directory, e.g.

```
 RewriteBase /~myusername/caketest
```

create a database, set the db connection in cake config files and that's all. you can point your browser to http://caketest.local if you do not want a test site url you can skip hosts, and apache vhost creation, but the url to use should be http:/localhost/~myusername/caketest another important thing is to enable userdir modul in apache, and also check if using php is enabled in userdirs too.

### CakePHP 2.x Basic Introduction

Will talk about directory structure of CakePHP, what each folder means.

## CakePHP has some main folders

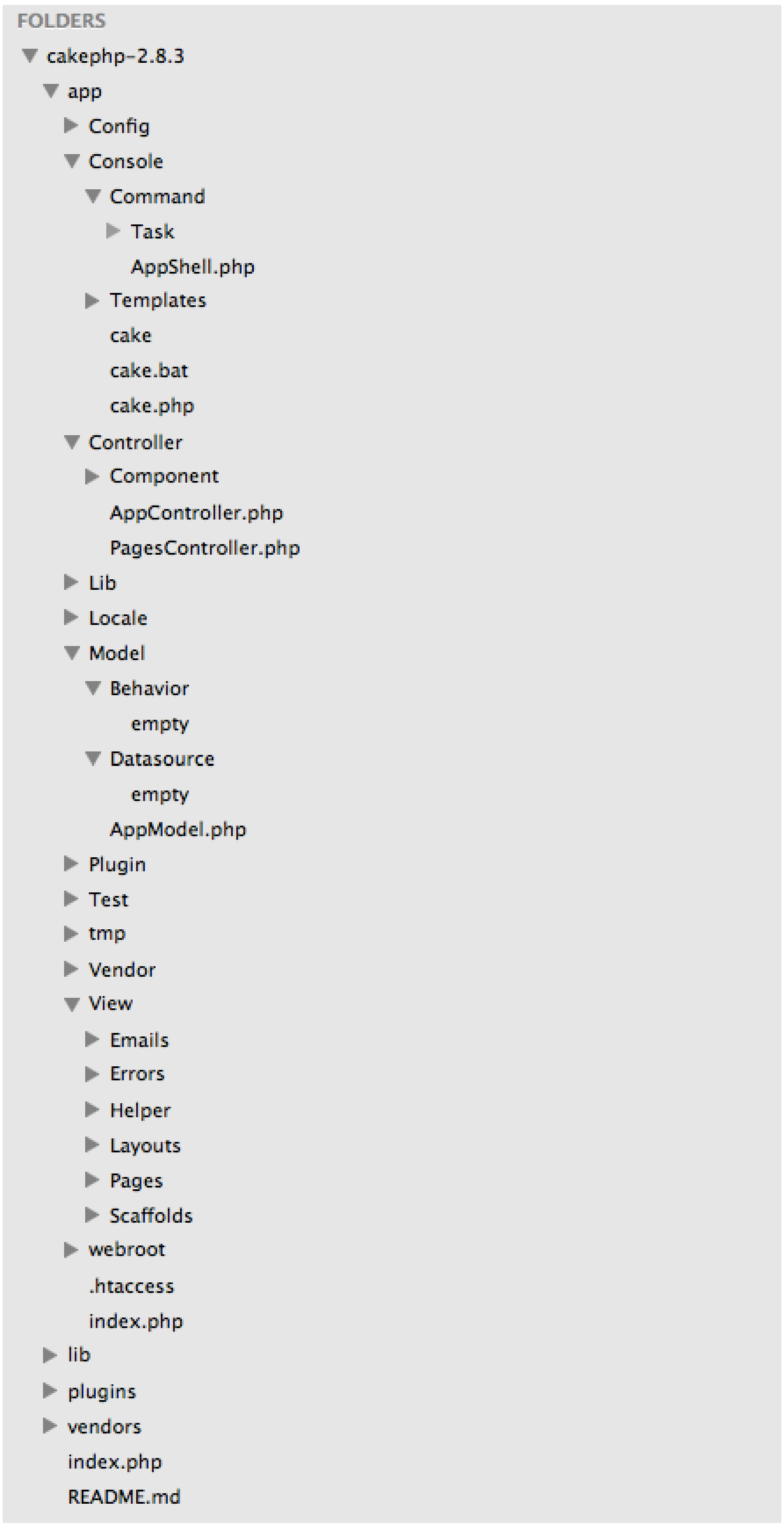
1. app - It Contains our application source code, all our code lies under this directory.
2. lib - This is the cakephp core liberary, it contains all the base cakephp library code. Editing code inside this directory is not suggested as they can cause error while upgrading the cakephp library.
3. plugins - This contains the cakephp plugins code which will be used for our applicatin.
4. vendors - This contains external code, This code will not use cakephp library.
5. index.php - This is the index file.

   We can have multiple applications hosted inside a single project. i.e they can use same lib folders, plugin and vendors.

   To Modify the lib code, best practice is to extend them in our app folder and perform the modifications.

   Plugins and vendors folder are shared by all the applications hosted in the same directory.

   index.php is the file which is called first.



```
FOLDERS
▼ cakephp-2.8.3
  ▼ app
    ▶ Config
    ▼ Console
      ▼ Command
        ▶ Task
          AppShell.php
      ▶ Templates
        cake
        cake.bat
        cake.php
    ▼ Controller
      ▶ Component
        AppController.php
        PagesController.php
    ▶ Lib
    ▶ Locale
    ▼ Model
      ▼ Behavior
          empty
      ▼ Datasource
          empty
        AppModel.php
    ▶ Plugin
    ▶ Test
    ▶ tmp
    ▶ Vendor
    ▼ View
      ▶ Emails
      ▶ Errors
      ▶ Helper
      ▶ Layouts
      ▶ Pages
      ▶ Scaffolds
    ▶ webroot
      .htaccess
      index.php
  ▶ lib
  ▶ plugins
  ▶ vendors
    index.php
    README.md
```

## Now we should jump to our app folder

Read Getting started with cakephp online: https://riptutorial.com/cakephp/topic/958/getting-started-with-cakephp

# Chapter 2: Ajax request handling

## Examples

### Basic CakePHP 2.x example

Controller: In the Controller you have to add the RequestHandler component. This Enables CakePHP to automatically detect Ajax requests(see: http://book.cakephp.org/2.0/en/core-libraries/components/request-handling.html for more info):

```
class YourController extends AppController {
    public $components = array('RequestHandler');
    //...

    public function ajaxCall() {
        if($this->request->is('ajax'){
            // some code that should be executed
            // ...
            // variables you want to return
            $this->set(compact('firstVariable', 'secondVariable'));
            $this->set('_serialize', array('firstVariable', secondVariable))
    }
}
```

View Code (using jQuery):

```
<script>
$.ajax({
    type: 'POST',
    url: '/yourController/ajaxCall',
    success: function (result) {
        // result is a JSON array of the returned Variables
    },
    error: function (result){
        console.log(result);
    }
});
</script>
```

### Ajax Request in Cakephp 2.x

One more way to use ajax in cakephp. Cakephp provide itself for ajax request. Please see example.

```
 $data = $this->Js->get('#id')->serializeForm(array('isForm' => true, 'inline' => true));
//on click send request to controller and displays response data in view
$this->Js->get('#button-id')->event(
        'click', $this->Js->request(
                array('controller' => 'Users', 'action' => 'add'), array(
            'update' => '#time', // field you wish to update
            'data' => $data, // Form Data in serialize form
            'async' => true,
            'dataExpression'=>true,
            'method' => 'POST' // method get or post
                )
        )
);
```

Read Ajax request handling online: https://riptutorial.com/cakephp/topic/6230/ajax-request-handling

# Chapter 3: CakePHP3 Coding Tips

## Examples

### Creating new Controller

```
namespace App\Controller;

class PostsController extends AppController {

    public function initialize(){
        parent::initialize();
        // code that you want to run before every action
    }
    public function view($id) {
       //Your code here
    }
}
```

### Add beforeFilter() method in Controller

The beforeFilter() method executes before any other method executes in the controller.

First use the Event namespace before defining the class in your controller file.

```
use Cake\Event\Event;
```

In your controller, add the beforeFilter() method as shown below.

```
public function beforeFilter(Event $event) {
    parent::beforeFilter($event);
}
```

Or, You can use the `initialize()` method.

```
public function initialize(){
    parent::initialize();
}
```

### Passing variables to View

Pass each variable to view at a time

```
$this->set('color', 'pink');
$this->set('color', $color);
```

Pass multiple variables to view together via compact() function

```
$color1 = 'pink';
$color2 = 'red';
$this->set(compact('color1', 'color2'));
```

## Retrieving post data equivalent to $_POST

You can retrieve post data as Array.

```
$post_data= $this->request->data;
```

You can retrieve post data for particular key.

```
$this->request->data['field'];
```

Retrieve specific key value

```
$this->request->data('key_name');
```

Retrieve specific key value of nested array

```
$this->request->data('data.subfield');
```

the difference between the array notation and `data()` method is that `data()` is error safe and returns `null` if the key does not exist in the array

so intead of doing

```
if(isset($this->request->data['field']) && $this->request->data['field']) { ...}
```

you can do

```
if($this->request->data('field')) { ...}
```

for CakePHP 3.4.x +

get all data:

```
$this->request->getData();
```

get specific key:

```
$this->request->getData('key');
```

to set data available for getData function you have to do something like:

```
$this->request = $this->request->withData('some_key_on_the_fly', 'value');
$some_key_on_the_fly = $this->request->getData('some_key_on_the_fly');
```

useful for updating the models in controller with static data

## Load another Model into Controller

By default CakePHP loads the related model in the controller. In order to load another model in the controller, use the loadModel() method:

```
$this->loadModel('Articles');
```

or load on the fly

```
$table = TableRegistry::get('Articles');
$table->find();
```

## Redirecting to another page from Controller

Redirect to within application (another action of specific controller).

```
return $this->redirect([
    'controller' => 'myController',
    'action' => 'myAction'
]);
```

Redirect to referrer page

```
return $this->redirect($this->referer());
```

Redirect to outside of application or specific URL

```
return $this->redirect("http://stackoverflow.com/users/1793428/haresh-vidja");
```

## Passing Variable to Action from URL with redirect

Passing Variable in URL as a **method's parameter**

```
return $this->redirect([
    'controller' => 'users',
    'action' => 'profile',
    $id
]);
```

Url should be looks like this http://your_app_url/users/profile/{id}

in UsersController.php file in profile() method

```
class UsersController extends Controller {
    public function profile($id=null) {
        $userData=$this->Users->get($id);
    }
}
```

Passing variable in URL as a **Query String**

```
return $this->redirect([
    'controller' => 'users',
    'action' => 'profile',
    '?'=>['id'=>$id]
]);
```

Url should be looks like this http://your_app_url/users/profile/?id={id}

in UsersController.php file in profile() method

```
class UsersController extends Controller {
    public function profile() {
        $userData=$this->Users->get($this->request->query('id'));
    }
}
```

## Set or Change Layout of application

Set default layout **for entire application**. i.e, created layout file in /src/Template/Layout/admin.ctp

```
class AppsController extends Controller {

    public function beforeFilter(Event $event) {
        parent::beforeFilter($event);
        $this->viewBuilder()->layout('admin'); // For Version >= 3.1 or
        $this->layout = 'admin'; // for version < 3.1

        // your other code should be here
    }
}
```

Set default layout **for specific action in application**. i.e, application have different layout in login page in /src/Template/Layout/login.ctp

```
class UsersController extends Controller {

    public function login() {

        $this->viewBuilder()->layout('login'); // For Version >= 3.1 or
        $this->layout = 'login'; // for version < 3.1

        //your other code should be here
    }
}
```

Change Layout **for specific Controller**. for i.e, you need different layout for all method of specific controller

class UsersController extends Controller {

```
    public function beforeFilter(Event $event) {
        parent::beforeFilter($event);

        $this->viewBuilder()->layout('user_layout'); // For Version >= 3.1 or
        $this->layout = 'user_layout'; // for version < 3.1

        //your other code should be here
    }
}
```

## Set Ajax Request Layout

Generally in AJAX request no need of load CSS, JS. Also omitting other HTML code.

Make ajax.ctp file in /src/Template/Layout, and code should be

```
<?php
    $this->fetch('content');
```

Set AJAX based layout for entire application, in AppsController.php

class AppsController extends Controller {

```
public function beforeFilter(Event $event) {
    parent::beforeFilter($event);
    if($this->request->isAjax())
    {
        $this->viewBuilder()->layout('ajax'); // For Version >= 3.1 or
        $this->layout = 'ajax'; // for version < 3.1

    }
    else
    {
        $this->viewBuilder()->layout('admin'); // For Version >= 3.1 or
        $this->layout = 'admin'; // for version < 3.1
    }

    // your other code should be here
}
```

}

## Load Components in CakePHP

We can load components in two ways.

1. By initialize or override $components property in Controller
2. By using loadComponent() method in initialize() method of Controller.

**Way-1** It should be override loading component by AppsController.php load one or more component

```
class UsersController extends AppController {
    public $components = ['RequestHandler','Auth','Flash'];
}
```

**Way-2** Use this way when you need load component dynamically for specific controller. Load One component

```
class UsersController extends AppController {
    public function initialize() {
        parent::initialize();
        $this->loadComponent("RequestHandler"); // load specific component
        $this->loadComponent(["RequestHandler","Auth","Flash"]); // load specific component
    }
}
```

## What is initilaize() method?

initialize() is introduced in CakePHP version > 3.0

As a code structure, it looks like same as beforeFilter() method. but there is many differences between beforeFilter() and initialize().

1. initialize() is always called after constructor is called. but beforeFilter() is not calling in case of action method not found in particular controller.
2. initialize() method is generally used for initialize something like add new components and helpers. but in beforeFilter() is generally used for execute some global logic part.

## Retrive query string data equivalent to $_GET

You can retrieve query string data as Array.

```
$post_data= $this->request->query;
```

You can retrieve post data for particular key.

```
$this->request->query['field'];
```

Retrieve specific key value

```
$this->request->query('key_name');
```

Retrieve specific key value of nested array

```
$this->request->query('data.subfield');
```

## Creating Table(Model) Class

How to create User Model Class

```
namespace App\Model\Table;
use Cake\ORM\Table;

class UsersTable extends Table {
    public function initialize(array $config) {
        $this->table('users'); //define table name
        $this->displayField('username'); // unique or other special field of users table
        $this->primaryKey('id'); // primary key of users table
        $this->tablePrefix('prefix_'); // if prefix set tablename should be prefix_users

        // your other code here
    }

    // your other methods here
}
```

## Model Associations in CakePHP

There are 4 types of associations(relationships) we can define in CakePHP

```
class PostsTable extends Table {
    public function initialize(array $config) {

        // table initialization code should be here

        $this->belongsTo('Authors', [
            'className' => 'Authors',
            'foreignKey' => 'author_id',
            'joinType' => 'INNER',
        ]);
        $this->hasMany('Tags');
        $this->hasOne('Categories');
        $this->hasAndBelongsToMany('Topics');
    }
}
```

In above example, you can see 4 types of relationships

Post **belongs to** Author **(One to One)**, it means in `posts` table has one foreign key `author_id` which is associated with `id` of `authors` table.

Post **has many** Tags **(One to Many)**, it means in `tags` table has one foreign key `post_id` which is associated with `id` of `posts` table.

Post **has one** Tags **(Many to One or One to One)**, it means in `posts` table has one foreign key `category_id` which is associated with `id` of `categories` table.

Post **has and belongs to** Topics **(Many to Many)**, this is many to many relationship between `posts` and `topics` table. for maintain many to many relationship must need to create third table, table, table name should be `posts_categories`. Fields of this table as mentioned below

1. id (primary key of table)
2. post_id (foreign key of `posts` table)
3. topic_id (foreign key of `topics` table)

Read CakePHP3 Coding Tips online: https://riptutorial.com/cakephp/topic/3341/cakephp3-coding-tips

# Chapter 4: Instantiating models from another datasource

## Remarks

There will come a time where your CakePHP application will need to query more than one database. The method for requesting Models from non-default databases is not present in the official documentation.

## Examples

### Instantiating uses App::uses

```
App::uses('YourModel', 'Model');
$model_1 = new YourModel(array('ds' => 'default'));
$model_2 = new YourModel(array('ds' => 'database2'));
```

### On the fly database changes for modal

For multiple databases, you have the database.php file where you can set as many databases as you need.

If you want to "switch" a database for a specific model on the fly, use the setDataSource() method.

For example, if you have two databases, you can define them in the database.php file as "default" and "sandbox", as an example.

Then, in your code:

$this->MyModal->setDataSource('sandbox');

The sandbox is the name of the configuration, and the actual name of the database is written only once in the database.php file.

Read Instantiating models from another datasource online: https://riptutorial.com/cakephp/topic/1953/instantiating-models-from-another-datasource

# Credits

| S. No | Chapters | Contributors |
|---|---|---|
| 1 | Getting started with cakephp | burzum, Community, Deejay, gins t, Gorakh Yadav, Max90, mcgowan.b |
| 2 | Ajax request handling | Max90, Rashid Khan |
| 3 | CakePHP3 Coding Tips | arilia, CodeZilla, Haresh Vidja, IWillScoop, Max90, Miheretab Alemu, parsaya |
| 4 | Instantiating models from another datasource | Holt, RavatSinh Sisodiya, Zetaphor |