tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

CakePHP is an open-source framework for PHP. It is intended to make developing, deploying and maintaining applications much easier.

CakePHP is based on an MVC-like architecture that is both powerful and easy to grasp. Models, Views, and Controllers guarantee a strict but natural separation of business logic from data and presentation layers.

Audience

This tutorial is meant for web developers and students who would like to learn how to develop websites using CakePHP. It will provide a good understanding of how to use this framework.

Prerequisites

Before you proceed with this tutorial, we assume that you have knowledge of HTML, Core PHP, and Advance PHP. We have used CakePHP version 3.2.7 in all the examples.

Copyright & Disclaimer

Table of Contents

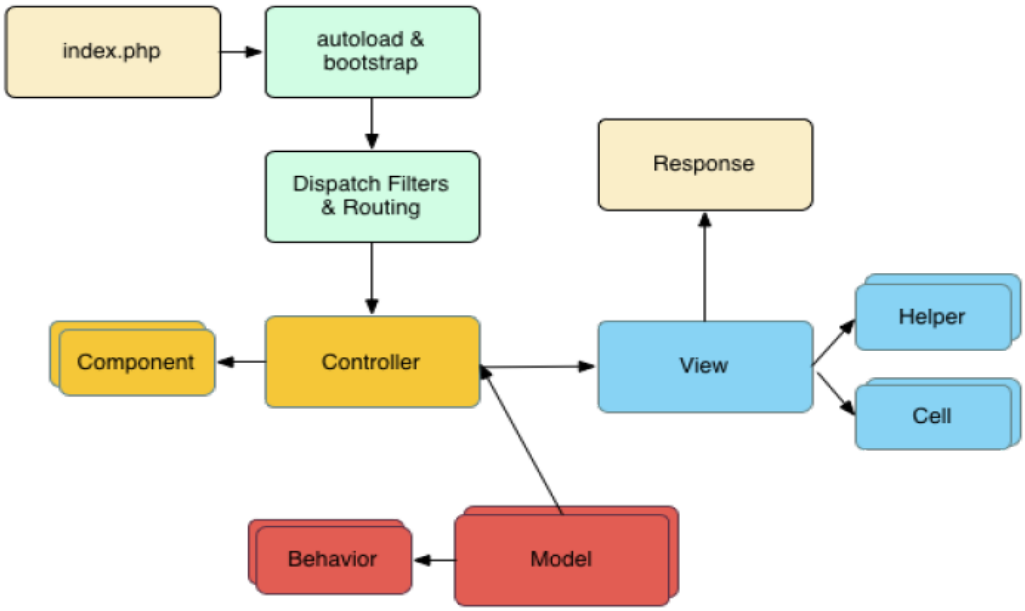
1. CakePHP — Overview

CakePHP is an open source MVC framework. It makes developing, deploying and maintaining applications much easier. CakePHP has number of libraries to reduce the overload of most common tasks. Following are the advantages of using CakePHP.

- Open Source
- MVC Framework
- Templating Engine
- Caching Operations
- Search Engine Friendly URLs
- Easy CRUD (Create, Read, Update, Delete) Database Interactions.
- Libraries and Helpers
- Built-in Validation
- Localization
- Email, Cookie, Security, Session, and Request Handling Components
- View Helpers for AJAX, JavaScript, HTML Forms and More

CakePHP Request Cycle

The following illustration describes how a Request Lifecycle works:



A typical CakePHP request cycle starts with a user requesting a page or resource in your application. At a high level, each request goes through the following steps:

- The webserver rewrite rules direct the request to webroot/index.php.
- Your application's autoloader and bootstrap files are executed.
- Any **dispatch filters** that are configured can handle the request, and optionally generate a response.
- The dispatcher selects the appropriate controller & action based on routing rules.
- The controller's action is called and the controller interacts with the required Models and Components.
- The controller delegates response creation to the **View** to generate the output resulting from the model data.
- The view uses **Helpers** and **Cells** to generate the response body and headers.
- The response is sent back to the client.

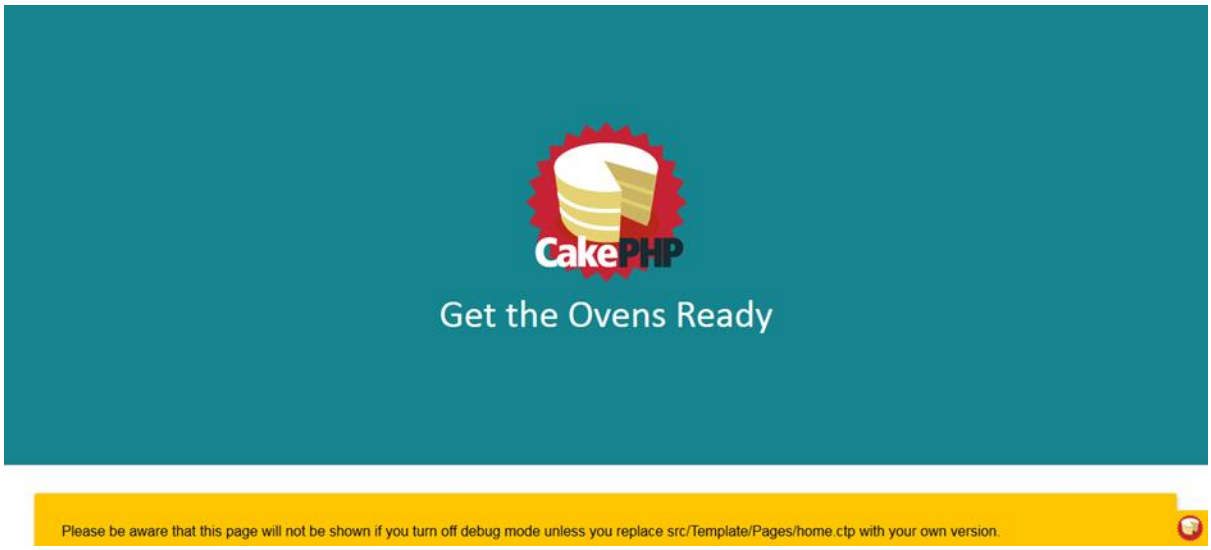
2. CakePHP – Installation

Installing CakePHP is simple and easy. You can install it from composer or you can download it from github — https://github.com/cakephp/cakephp/releases. We will further understand how to install CakePHP in WampServer. After downloading it from github, extract all the files in a folder called "CakePHP" in WampServer. You can give custom name to folder but we have used "CakePHP".

Make sure that the directories **logs**, **tmp** and all its subdirectories have **write permission** as CakePHP uses these directories for various operations.

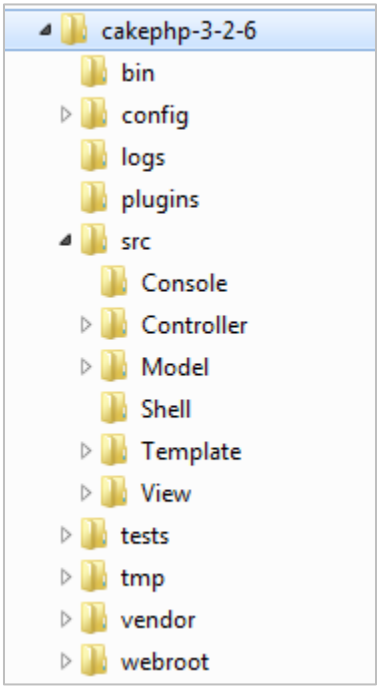
After extracting it, let's check whether it has been installed correctly or not by visiting the following URL in browser: http://localhost:85/CakePHP/

The above URL will direct you to the screen as shown below. This shows that CakePHP has successfully been installed.



3. CakePHP — Folder Structure

Take a look at the following screenshot. It shows the folder structure of CakePHP.



The following table describes the role of each folder:

Folder Name	Description
bin	The bin folder holds the Cake console executables.
config	The config folder holds the (few) configuration files CakePHP uses. Database connection details, bootstrapping, core configuration files and more should be stored here.
logs	The logs folder normally contains your log files, depending on your log configuration.
plugins	The plugins folder is where the Plugins your application uses are stored.
src	The src folder will be where you work your magic: It is where your application's files will be placed. CakePHP's src folder is where you will do most of your application development. Let's look a little closer at the folders inside src. • Console Contains the console commands and console tasks for your application. • Controller Contains your application's controllers and their components.

	• Locale Stores string files for internationalization. • Model 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.
tests	The tests folder will be where you put the test cases for your application.
tmp	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.
vendor	The vendor folder is where CakePHP and other application dependencies will be installed. Make a personal commitment not to edit files in this folder. We can't help you if you've modified the core.
webroot	The webroot directory is the public document root of your application. It contains all the files you want to be publically reachable.

4. CakePHP – Configuration

CakePHP comes with one configuration file by default and we can modify it according to our needs. There is one dedicated folder **"config"** for this purpose. CakePHP comes with different configuration options.

General Configuration

The following table describes the role of various variables and how they affect your CakePHP application.

Variable Name	Description
debug	Changes CakePHP debugging output. false = Production mode. No error messages, errors, or warnings shown. true = Errors and warnings shown.
App.namespace	The namespace to find app classes under.
App.baseUrl	Un-comment this definition if you don't plan to use Apache's mod_rewrite with CakePHP. Don't forget to remove your .htaccess files too.
App.base	The base directory the app resides in. If false, this will be auto detected.
App.encoding	Define what encoding your application uses. This encoding is used to generate the charset in the layout, and encode entities. It should match the encoding values specified for your database.
App.webroot	The webroot directory.
App.wwwRoot	The file path to webroot.
App.fullBaseUrl	The fully qualified domain name (including protocol) to your application's root.
App.imageBaseUrl	Web path to the public images directory under webroot.
App.cssBaseUrl	Web path to the public css directory under webroot.
App.jsBaseUrl	Web path to the public js directory under webroot.
App.paths	Configure paths for non-class based resources. Supports the **plugins, templates, locales** subkeys, which allow the definition of paths for plugins, view templates and locale files respectively.
Security.salt	A random string used in hashing. This value is also used as the HMAC salt when doing symmetric encryption.
Asset.timestamp	Appends a timestamp which is last modified time of the particular file at the end of asset files URLs (CSS, JavaScript, Image) when using proper helpers. Valid values: • (bool) false - Doesn't do anything (default) • (bool) true - Appends the timestamp when debug is true

	• (string) 'force' - Always appends the timestamp

Databases Configuration

Database can be configured in **config/app.php** file. This file contains a default connection with provided parameters which can be modified as per our choice. The below screenshot shows the default parameters and values which should be modified as per the requirement.

```
/**
 * Connection information used by the ORM to connect
 * to your application's datastores.
 * Drivers include Mysql Postgres Sqlite Sqlserver
 * See vendor\cakephp\cakephp\src\Database\Driver for complete list
 */
'Datasources' => [
    'default' => [
        'className' => 'Cake\Database\Connection',
        'driver' => 'Cake\Database\Driver\Mysql',
        'persistent' => false,
        'host' => 'localhost',
        /**
         * CakePHP will use the default DB port based on the driver selected
         * MySQL on MAMP uses port 8889, MAMP users will want to uncomment
         * the following line and set the port accordingly
         */
        //'port' => 'non_standard_port_number',
        'username' => 'my_app',
        'password' => 'secret',
        'database' => 'my_app',
        'encoding' => 'utf8',
        'timezone' => 'UTC',
        'flags' => [],
        'cacheMetadata' => true,
        'log' => false,
```

Let's understand each parameter in detail:

Key	Description
className	The fully namespaced class name of the class that represents the connection to a database server. This class is responsible for loading the database driver, providing SQL transaction mechanisms and preparing SQL statements among other things.
driver	The class name of the driver used to implements all specificities for a database engine. This can either be a short classname using plugin syntax, a fully namespaced name, or a constructed driver instance. Examples of short classnames are Mysql, Sqlite, Postgres, and Sqlserver.
persistent	Whether or not to use a persistent connection to the database.
host	The database server's hostname (or IP address).
username	Database username
password	Database password

database	Name of Database
port (optional)	The TCP port or Unix socket used to connect to the server.
encoding	Indicates the character set to use when sending SQL statements to the server like 'utf8' etc.
timezone	Server timezone to set.
schema	Used in PostgreSQL database setups to specify which schema to use.
unix_socket	Used by drivers that support it to connect via Unix socket files. If you are using PostgreSQL and want to use Unix sockets, leave the host key blank.
ssl_key	The file path to the SSL key file. (Only supported by MySQL).
ssl_cert	The file path to the SSL certificate file. (Only supported by MySQL).
ssl_ca	The file path to the SSL certificate authority. (Only supported by MySQL).
init	A list of queries that should be sent to the database server as when the connection is created.
log	Set to true to enable query logging. When enabled queries will be logged at a debug level with the queriesLog scope.
quoteIdentifiers	Set to true if you are using reserved words or special characters in your table or column names. Enabling this setting will result in queries built using the Query Builder having identifiers quoted when creating SQL. It decreases performance.
flags	An associative array of PDO constants that should be passed to the underlying PDO instance.
cacheMetadata	Either boolean true, or a string containing the cache configuration to store meta data in. Having metadata caching disable is not advised and can result in very poor performance.

5. CakePHP – Email Configuration

Email can be configured in file **config/app.php**. It is not required to define email configuration in config/app.php. Email can be used without it; just use the respective methods to set all configurations separately or load an array of configs. Configuration for Email defaults is created using **config()** and **configTransport()**.

Email Configuration Transport

By defining transports separately from delivery profiles, you can easily re-use transport configuration across multiple profiles. You can specify multiple configurations for production, development and testing. Each transport needs a **className**. Valid options are as follows:

- Mail — Send using PHP mail function
-
- Smtp — Send using SMTP
-
- Debug — Do not send the email, just return the result

You can add custom transports (or override existing transports) by adding the appropriate file to **src/Mailer/Transport**. Transports should be named **YourTransport.php**, where **'Your'** is the name of the transport. Following is the example of Email configuration transport.

Example

```
'EmailTransport' => [
        'default' => [
            'className' => 'Mail',
            // The following keys are used in SMTP transports
            'host' => 'localhost',
            'port' => 25,
            'timeout' => 30,
            'username' => 'user',
            'password' => 'secret',
            'client' => null,
            'tls' => null,
            'url' => env('EMAIL_TRANSPORT_DEFAULT_URL', null),
        ],
    ],
```

Email Delivery Profiles

Delivery profiles allow you to predefine various properties about email messages from your application and give the settings a name. This saves duplication across your application and makes maintenance and development easier. Each profile accepts a number of keys. Following is an example of Email delivery profiles.

Example

```
'Email' => [
        'default' => [
            'transport' => 'default',
            'from' => 'you@localhost',
        ],
    ],
```

End of ebook preview

If you liked what you saw…

Buy it from our store @ **https://store.tutorialspoint.com**