インストールと設定

apache の場合

http://doruby.kbmj.com/hanafubuki_on_rails/20080227/Rails_CakePHP_

■必要な環境

・PHP4.3.2 以上、あるいは PHP5 以上が動作するサーバ

■インストール

ダウンロードして解凍するだけ

■ URL と mod_rewite の設定

mod_rewrite の設定は、

```
/cake /.htaccess
/cake /app /.htaccess
/cake /app /webroot /.htaccess
```

の 3 つの .htaccess ファイルで行われています。
そして、mod_rewrite を利用できる環境では、以下のような URL が用いられます。

```
http:// 設置 URL / コントローラ / メソッド / パラメータ 1 / パラメータ 2 /...
```

例えば、

```
http://doruby.kbmj.com / members / regist / 3
```

という URL の場合、

```
members_controller.php の中の regist() メソッドに引数「3」を入れて実行する
```

という意味になり、実際に呼び出されるファイルは、

```
コントローラ： /cake /app /controllers /members_controller.php
テンプレート： /cake /app /views /regist.thtml
```

となります。

環境によっては .htaccess に RewriteBase を追加しないと動かない。

/blog/.htaccess

```
<ifmodule mod_rewrite.c>
RewriteEngine on
RewriteBase /blog
RewriteRule    ^$ app/webroot/    [L]
RewriteRule    (.*) app/webroot/$1 [L]
</ifmodule>
```

/blog/app/.htaccess

```
<ifmodule mod_rewrite.c>
RewriteEngine on
RewriteBase /blog/app
RewriteRule    ^$    webroot/    [L]
RewriteRule    (.*) webroot/$1    [L]
</ifmodule>
```

/blog/app/webroot/.htaccess

```
<ifmodule mod_rewrite.c>
RewriteEngine On
RewriteBase /blog/app/webroot
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^(.*)$ index.php?url=$1 [QSA,L]
</ifmodule>
```

■ mod_rewrite が使えない場合

mod_rewrite が利用できない場合は、以下の 3 つの作業が必要になります。

core.php の修正

```
/cake /app /config /core.php
```

の以下のコメントを外してください。

```
<?php
/**
 * If you do not have mod rewrite on your system
 * or if you prefer to use CakePHP pretty urls.
 * uncomment the line below.
 * Note: If you do have mod rewrite but prefer the
 * CakePHP pretty urls, you also have to remove the
 * .htaccess files
 * release/.htaccess
 * release/app/.htaccess
 * release/app/webroot/.htaccess
 */
// define ('BASE_URL', env('SCRIPT_NAME'));  ← ここのコメントを外す
```

.htaccess ファイルの削除

上述した 3 つの .htaccess ファイル

```
/cake /.htaccess
/cake /app /.htaccess
/cake /app /webroot /.htaccess
```

を削除してください。

URL の変更

CakePHP で用いる URL が以下のように変わります。

```
http:// 設置 URL /index.php / コントローラ名 / アクション名 /
```

nginx の場合

基本的に公式の通り。

```
server {
  listen 81;
  root /var/www/cakephp/;

  # location ~ ^/subdir/(img|css|js|files)/(.+)$ {
    # root /home/centos/public_html/cakephp/app/webroot;
    # try_files /$1/$2 =404;
  # }

  location ~ ^/(subdir)/(.*)? {
    index  index.php;

    set $new_uri /$1/app/webroot/$2;
    try_files $new_uri $new_uri/ /$1/index.php?$args;


    location ~ \.php$ {
      fastcgi_pass   127.0.0.1:9000;
      fastcgi_index  index.php;
      fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
      include        fastcgi_params;
    }
  }
}
```

Docker

■ Apache を使う場合

ディレクトリ構成

```
.
├── docker-compose.yml
├── Dockerfile
├── cakephp.conf
├── php.ini.patch
└── src
    └── cakephp
```

docker-compose.yml

```
version: '3'
services:
  web:
    build:
      context: .
      dockerfile: Dockerfile
    volumes:
      - ./src:/var/www/html
    ports:
      - 1080:80
    depends_on:
      - mysql
    command:
       httpd -DFOREGROUND
  mysql:
    image: mysql:5.7
    environment:
      - MYSQL_ROOT_PASSWORD=root
    ports:
      - 3306:3306
    volumes:
      - dbdata:/var/lib/mysql
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    environment:
      - PMA_ARBITRARY=1
      - PMA_HOST=mysql
```

```
      - PMA_USER=root
      - PMA_PASSWORD=root
    depends_on:
      - mysql
    ports:
       - 1081:80

volumes:
  dbdata:
```

Dockerfile

```
From centos:7

RUN echo "RUN は build 時に実行 "
RUN yum -y install php
RUN yum -y install php-pdo
RUN yum -y install php-mysql
RUN yum -y install patch

COPY cakephp.conf /etc/httpd/conf.d

COPY php.ini.patch /tmp
RUN patch /etc/php.ini < /tmp/php.ini.patch

CMD echo "CMD は run 時に実行 "
```

cakephp.conf

```
<Directory /var/www/html/cakephp>
    AllowOverride FileInfo
</Directory>
```

php.ini.patch

```
878c878
< ;date.timezone =
---
> date.timezone = "Asia/Tokyo"
```

■ Nginx を使う場合

ディレクトリ構成

```
.
├── docker-compose.yml
├── Dockerfile
├── cakephp.conf
├── nginx.repo
├── php.ini.patch
├── www.conf.patch
└── src
       └── cakephp <- CakePHP 本体
```

docker-compose.yml

```
version: '3'
services:
  web:
    build:
      context: .
```

```
    dockerfile: Dockerfile
  # user: "1000:1000"
  volumes:
    - ./src:/var/www/html
  ports:
    - 1080:80
  command:
     bash -c "php-fpm & /usr/sbin/nginx -g 'daemon off;'"
```

Dockerfile

```
From centos:7

RUN echo "RUN は build 時に実行 "
COPY nginx.repo /etc/yum.repos.d/
RUN yum -y install php
RUN yum -y install nginx
RUN yum -y install php-fpm
RUN yum -y install patch

RUN mv /etc/nginx/conf.d/default.conf /etc/nginx/conf.d/default.conf_back
COPY cakephp.conf /etc/nginx/conf.d/

COPY www.conf.patch /tmp
RUN patch /etc/php-fpm.d/www.conf < /tmp/www.conf.patch

COPY php.ini.patch /tmp
RUN patch /etc/php.ini < /tmp/php.ini.patch

COPY php-fpm.conf.patch /tmp
RUN patch /etc/php-fpm.conf < /tmp/php-fpm.conf.patch

COPY docker.conf /etc/php-fpm.d

CMD echo "CMD は run 時に実行 "
```

cakephp.conf (nginx の設定ファイル)

```
server {
  listen 80;
  root /var/www/html/;
  error_log /dev/stdout info;
  access_log /dev/stdout;

  location ~ \.php$ {
    fastcgi_pass   127.0.0.1:9000;
    fastcgi_index  index.php;
    fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
    include        fastcgi_params;
  }
}
```

nginx.repo

```
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/7/$basearch/
gpgcheck=0
enabled=1
```

php.ini.patch

```
878c878
< ;date.timezone =
---
> date.timezone = "Asia/Tokyo"
```

www.conf.patch

```
39c39
< user = apache
---
> user = nginx
41c41
< group = apache
---
> group = nginx
```

php-fpm.conf.patch

```
24c24
< error_log = /var/log/php-fpm/error.log
---
> ; error_log = /var/log/php-fpm/error.log
```

docker.conf

```
[global]
error_log = /proc/self/fd/2

[www]
; if we send this to /proc/self/fd/1, it never appears
access.log = /proc/self/fd/2

clear_env = no

; Ensure worker stdout and stderr are sent to the main error log.
catch_workers_output = yes
```

cakephp 2.3 以降の注意点

default.ctp

■コンテンツ表示

レイアウトの default.ctp の書き方が変わっているので注意。
コンテンツを表示するには、2.3 以前では

```
$content_for_layout
```

と書いていたが、2.3 以降は

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

とする。

■ META タグの文字コード

```
<?php echo $this->Html->charset(); ?>
```

とすると、 /path/to/cakephp/app/Config/core.php の

```
Configure::write('App.encoding', 'UTF-8');
```

の内容から

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

と表示してくれる。

View や Controllers のディレクトリを階層化する

Config/bootstrap.php

```
App::build(array(
 'Controller' => array(
   ROOT.DS.APP_DIR.DS.'Controller'.DS.'app1'.DS,
   ROOT.DS.APP_DIR.DS.'Controller'.DS,
   )
 ));

App::build(array(
 'View' => array(
   ROOT.DS.APP_DIR.DS.'View'.DS.'app1'.DS,
   ROOT.DS.APP_DIR.DS.'View'.DS,
   )
 ));
```

などのように書くと、コントローラやビューを探しに行ってくれる。
アクセスする際の URL は、階層化しない。

```
http://www.test/hoge/ コントローラ / メソッド
```

ではなく、

```
http://www.test/ コントローラ / メソッド
```

のままなので注意。

View や Controller に加えて URL を階層化する。（その１）

URL の階層化したい場合、上記設定に加えて、routes.php に以下の設定を加える。

```
Router::connect('/app1/:controller/:action/*', array());
Router::connect('/app1/:controller/*', array('action' => 'index'));
```

http://xxxxx/app1/controller/action
でアクセスがあった際に、bootstrap.php で指定された箇所から Controller を探す。
注意としては、Controller を探した際に同じ名前の Controller があった場合は、先に見つけた方を使う。

View や Controller に加えて URL を階層化する。（その２）

http://taka.at/blog/1234436530.html

http://blog.elkc.net/?p=515
https://liginc.co.jp/programmer/archives/1331

一つの CakePHP の下に複数のアプリケーションを含めたい場合、
Controller や View、URL を階層化したい。
上記の方法では、同じ名前の Controller や View が使えない。
同じ名前の Controller や View を使えるようにするには bootstrap.php を書き換える必要がある。

```
・アプリ１
http://hoge/Sub1/Hoge/action
・アプリ２
http://hoge/Sub2/Hoge/action
```

各アプリケーションで同じ名前の Controller や View を使う。

■ Config/bootstrap.php

```
$base = $_SERVER["SCRIPT_NAME"];
$indexUrl = "app/webroot/index.php";
$base = mb_substr($base, 0, mb_strlen($base) - mb_strlen($indexUrl));
$subDir = mb_substr($_SERVER["REQUEST_URI"], mb_strlen($base));
$sepPos = mb_strpos($subDir, DS);
if ($sepPos !== false) {
  $subDir = mb_substr($subDir, 0, $sepPos);
}
$isSubDir = false;

if ($subDir === "Sub1" || $subDir === "Sub2") {
  $isSubDir = true;
}
if ($isSubDir) {
  App::build(array(
  'Controller' => array(
    ROOT.DS.APP_DIR.DS.'Controller'.DS.$subDir.DS,
    ROOT.DS.APP_DIR.DS.'Controller'.DS,
    )
  ));

App::build(array(
  'View' => array(
    ROOT.DS.APP_DIR.DS.'View'.DS.$subDir.DS,
    ROOT.DS.APP_DIR.DS.'View'.DS,
    )
  ));
}
```

■ Config/routes.php

```
Router::connect('/Sub1/:controller/:action/*', array());
Router::connect('/Sub1/:controller/*', array('action' => 'index'));
Router::connect('/Sub2/:controller/:action/*', array());
Router::connect('/Sub2/:controller/*', array('action' => 'index'));
```

■ Config/core.php

Cache::config('_cake_core_'.... の引数を条件によって変更する。
Sub1 や Sub2 の場合はキャッシュ先を変更する
$subApps = array("Sub1", "Sub2");

```
$base = $_SERVER["SCRIPT_NAME"];
$indexUrl = "app/webroot/index.php";
$base = mb_substr($base, 0, mb_strlen($base) - mb_strlen($indexUrl));
$subDir = mb_substr($_SERVER["REQUEST_URI"], mb_strlen($base));
$sepPos = mb_strpos($subDir, DS);
```

```
if ($sepPos !== false) {
  $subDir = mb_substr($subDir, 0, $sepPos);
}

if (array_search($subDir, $subApps) !== false) {
  print "hoge<br>\n";
  print dirname(CACHE).DS.$subDir.DS."<br>\n";
  print CACHE."<br>\n";
  Cache::config('_cake_core_', array(
    'engine' => $engine,
    'prefix' => $prefix . 'cake_core_',
    'path' => CACHE . 'persistent' . DS.$subDir.DS,
    'serialize' => ($engine === 'File'),
    'duration' => $duration
    ));
} else {
  Cache::config('_cake_core_', array(
    'engine' => $engine,
    'prefix' => $prefix . 'cake_core_',
    'path' => CACHE . 'persistent' . DS,
    'serialize' => ($engine === 'File'),
    'duration' => $duration
    ));
}
```

テーマの利用

■ディレクトリ構成

app 以下の View ディレクトリに Themed を作成する。
各テーマ以下は webroot と同じ構成にする。
各テーマのディレクトリ名は大文字ではじめること。

```
View
  `-- Themed
      `-- HelloTheme
          `-- webroot
              `-- css
                  |-- cake.hello.css
```

■テーマの利用

コントローラで

```
$this->theme = 'helloTheme';
```

とすれば、テーマが適用される。

注意点

■無駄な改行について

Controller の

```
<?php
 処理
?>
```

のあとに無駄な改行があると、そのまま改行が出力されるので注意

■古い cake と新しい PHP の組み合わせ

エラーの種類として
・E_NOTICE

・ E_DEPRECATED
が追加されている PHP で、古い cake を動かすとメッセージが大漁に表示される。
cake 内の

```
error_reporting
```

を
E_ALL & ~E_NOTICE & ~E_DEPRECATED

```
E_ALL & ~E_STRICT & ~E_DEPRECATED
```

するとメッセージが消える。

■出力の最後の埋め込まれる実行時間について

cakephp のバージョンによっては、実行時間がソースに吐き出されることがある。
邪魔な場合は

```
/app/webroot/index.php
```

にある

```
・・・・・・s -->
```

が書かれている行を消す

モデルを複合キーに対応させる

CakePHP 3.X からはモデルが複合キーに対応できるようになるようだが
2.X では、複合キーに対応していない。

http://piyopiyocs.blog115.fc2.com/blog-entry-358.html
を参考に複合キーに対応したモデルを作成すると良い。

Oracle などに接続する際に文字コードを指定する

大きく 2 つの方法がある。
どちらの方法でも良い。状況によって使い分ける。

接続時に文字コードを指定する

```
    public $default = array(
'datasource' => 'hogehoge',
            ..............
            'charset' => 'AL32UTF8',
);
```

環境変数に文字コードを指定する

具体的には apache が実行される際に設定される環境変数に NLS_LANG を設定する。
/etc/sysconfig/httpd

```
export NLS_LANG=Japanese_Japan.UTF8
```