

使ってみよう

kohana v3

kohana は MVC を使用している PHP5 のフレームワークです。セキュア、軽量、使い安さを目的としています。

一般的なデザインパターン、概念を再利用していますが、kohana が独自のものがあります

- コミュニティによる開発
- 厳密なオブジェクト指向 PHP
- 自動クラスローディング
- インターフェイス
- 抽象化
- シングルトン
- 超軽量

kohana は大型のモノリックライブラリを避けるため PECL 拡張、PEAR ライブラリとは依存関係がありません。

目次

1	はじめに	3
2	導入	4
3	設定	5
4	テストアプリケーションを作ろう	8
5	プログラムの作成.....	10
5.2	コントローラー	11
5.3	ビュー.....	12
5.4	モデル.....	13
6	エラーを出してみよう	15
6.2	「;」を忘れた	16
6.4	モデルのクラス名を間違えた	16
6.5	定義されていない関数を使用した	16
6.6	データベースの結果セットから間違った項目名で情報を取得した	16
7	モジュール.....	エラー! ブックマークが定義されていません。

1 はじめに

PHP のフレームワークは多数ありますが、私自身これらを利用したプロジェクトを行った経験がありません。

実際には、フレームワークを適用するために幾度となく、各フレームワークの機能や使い方にトライしてみましたが、いまいち気に入ったフレームワークに遭遇しませんでした。

本心は、是非ともフレームワークを使いたかったのですが！

皆さんはできるだけフレームワークを利用すべきと思います。

現在主流のフレームワークは「Symfony」「CakePHP」「Zend Framework」「ethna」です。利点と欠点がありますが、利点は欠点を補って余りあるものです。

ここでフレームワークのおさらいをして見ましょう。

利点

- MVC アーキテクチャによる機能分離

モデル	データベース
ビュー	表示
コントローラー	制御

- 生産性向上 コード量の劇的減少

オブジェクトを利用することで、書くコードの量が劇的に少なくなる

同じようなコード (DB 接続、データ読み込み、HTML 表示など) を重複して書かなくてもよい

- 均質な開発
- 保守性の向上

欠点

- フレームワークの理解コスト
- 機能不足対処

そこで kohana はどうなの？

とにかく超軽量、インストール簡単、使うのも簡単に惹かれました

2 導入

私の環境は

- OS windowsXP SP3
- XAMPP 1. 7. 1

kohana の取得は下記の URL からできます

<http://v3.kohanaphp.com/>

ダウンロードタブで安定版を選択します

kohana_v3.X.X.zip がダウンロードされます

ZIP を解凍します

出来上がったディレクトリ kohana を xampp の htdocs に移動します

確認作業

アドレスは、<http://localhost/kohana/index.php>

Environment Tests

The following tests have been run to determine if Kohana will work in your environment. If any of the tests have failed, consult the [documentation](#) for more information on how to correct the problem.

PHP Version	5.2.9
System Directory	C:\Program Files\xampp\htdocs\kohana\system\
Application Directory	C:\Program Files\xampp\htdocs\kohana\application\
Cache Directory	C:\Program Files\xampp\htdocs\kohana\application\cache\
Logs Directory	C:\Program Files\xampp\htdocs\kohana\application\logs\
PCRE UTF-8	Pass
SPL Enabled	Pass
Reflection Enabled	Pass
Filters Enabled	Pass
Iconv Extension Loaded	Pass
Mbstring Not Overloaded	Pass
URI Determination	Pass

✓ Your environment passed all requirements.
Remove or rename the install.php file now.

Optional Tests

The following extensions are not required to run the Kohana core, but if enabled can provide access to additional classes.

cURL Enabled	Pass
mcrypt Enabled	Pass
GD Enabled	Pass
PDO Enabled	Pass

これが表示されれば OK です

3 設定

kohana が動いたので設定をしましょう

kanaha/install.php を削除します

再度アクセス <http://localhost/Kohana/index.php>

「hello world」表示されました

ここで kohana のファイル構造を見て見ましょう

ディレクトリ構成が変わりました

```
root
+- application
|   +- cache
|   +- classes
|       +- controllers
|   +- config
|   +- logs
|   +- bootstrap
+- modules
|   +- auth
|   +- codebench
|   +- database
|   +- images
|   +- orm
|   +- pagination
|   +- userguide
|   +- .....
+- system
|   +- classes
|   +- config
|   +- i18n
|   +- messages
|   +- utf8
|   +- views
+- index.php
+- example.htaccess
```

名前の付け方ルール

- クラスは、`application/classes` の中に入れる
- アンダースコアはスラッシュに変換される
- ファイル名は小文字
- クラス記述 クラスの種類をプレフィックスでつける。
 - `controller_XXXXX`
 - `Model_XXXXX`
- Public 関数は `action_` をプレフィックスでつける

コントローラ格納場所が変更されました

「`application/classes/controller`」フォルダーにアプリケーションを作っていきます。

モデル格納場所

「`application/classes/model`」

ビュー格納場所

「`application/views`」

設定場所も変更されます

それではまず、「`konoha/application/bootstrap.php`」を修正しましょう

<code>Kohana::modules(array('database'=MODPATH.database,</code>	コメントを外し、 データベースを有効にします
--	---------------------------

次にデータベースの設定「`modules¥database¥config¥database.php`」

	<code>type</code> にデータベース種類 <code>mysql</code> <code>user</code> に使用者 <code>root</code> <code>database</code> にデータベース名 <code>testdb</code> 注意 : <code>dietdb</code> はまだ作っていません。 <code>xampp</code> の <code>phpmyadmin</code> で作成します
--	---

.htaccess の作成

example.htaccess をそのまま使ってください。windows ではリネームができません。
v.2.3.4 のものを使用しました

URI のリライトとは

<http://localhost/コントローラ名/メゾット名/引数/引数>

のように記述できます

4 テストアプリケーションを作ろう

ダイエットの記録と登録表示するアプリを作ります。

起動すると、過去の記録を表示します。

本日の記録を入力して更新ボタンを押すと登録し、ページを更新します

データベースは dietdb, テーブルは diet とします

<http://127.0.0.1> で [xampp](#) を起動します



[phpmyadmin](#) を起動し, dietdb を作ります



テーブル diet を作ります 項目数は3個です

データベース dietdb に新しいテーブルを作成する

名前:

フィールド数:

項目数を作ります、wait,percent,update

フィールド	<input type="text" value="wait"/>	<input type="text" value="percent"/>	<input type="text" value="update"/>
種別	<input type="text" value="INT"/>	<input type="text" value="INT"/>	<input type="text" value="TIMESTAMP"/>
長さ/値 ¹	<input type="text"/>	<input type="text"/>	<input type="text"/>
デフォルト値 ²	<input type="text" value="なし"/>	<input type="text" value="なし"/>	<input type="text" value="CURRENT_TIMESTAMP"/>

作成結果

	フィールド	種別	照合順序	属性	ヌル(NULL)	デフォルト値
<input type="checkbox"/>	wait	int(11)			いいえ	なし
<input type="checkbox"/>	percent	int(11)			いいえ	なし
<input type="checkbox"/>	update	timestamp			いいえ	CURRENT_TIMESTAMP

初期データを入れましょう 挿入タブ押します

サーバ: localhost ▶ データベース: dietdb ▶ テーブル: diet

表示 構造 SQL 検索 挿入 エクスポート インポート 操作

フィールド	種別	関数	ヌル(NULL)	値
wait	int(11)	<input type="text"/>		<input type="text" value="58"/>
percent	int(11)	<input type="text"/>		<input type="text" value="17"/>
update	timestamp	<input type="text"/>		<input type="text" value="CURRENT_TIMESTAMP"/>

挿入結果

サーバ: localhost ▶ データベース: dietdb ▶

表示 構造 SQL 検索 挿入

表示中の列 0 - 0 (1 合計, クエリの実行時間 0.0005 秒)

```

SELECT *
FROM `diet`
LIMIT 0 , 30
    
```

表示: 開始行

モード: (セルごとに

+ オプション

	wait	percent	update
<input type="checkbox"/> <input type="text" value="58"/> <input type="text" value="17"/> <input type="text" value="2009-08-04 09:49:54"/>			

5 プログラムの作成

5.1 kohana を使わないで PHP でデータを取得する

```
<?PHP
//(1)パラメータの取得
    $parm=$_POST['parm'];
//(2)データベース設定
    if (!$cn = mysql_connect("localhost", "root", "")) {
        print "db error";
        die;
    }
    $db="dietdb"; // MySQL DB 選択
    if (!(mysql_select_db($db))) {
        print "DB select error $db";
        die;
    }
    $sql="set names UTF8";
    mysql_query($sql);
//(3)データの取得
    $sql="select wait,percent,update from diet limit 1";
    if (!$result = mysql_query($sql)) { print "err:$sql";die;}
    $row = mysql_fetch_row($result);

    $wait=$row[0]; //
    $percent=$row[1]; //
    $update=$row[2]; //

//(4)出力
    print "<html><body>";
    print "<tabel><tr>";
    print "<td>$wait</td><td> percent</td><td>$update</td>";
    print "</tr></table>"
    print "</body></html>";
    //テンプレートエンジン(Smarty など)を使うときはここで指定
```

このように何もかも、コーディングしなければなりません。

これが kohana を使うとどのようになるのでしょうか？

5.2 コントローラー

次は MVC のコントローラを作ります。これが WEB からアクセスする基点になります

konoha/application/classes/controller/**diet**.php を作成します

まず helloworld を表示しましょう

ここで注意しなければならないのは、

クラスの名前は、ontroller プレフィックスの次に PHP の名前と同じ、かつ、クラスの名前文字を大文字にしなければなりません

```
<?php defined('SYSPATH') OR die('No direct access allowed.');
```

```
class Controller_Diet extends Controller  
{  
    public function action_index()  
    {  
        $this->request->response = 'hello, world!';  
    }  
}
```

```
?>
```

さあ実行してみましょう

<http://127.0.0.1/konoha/diet>

Hello World が表示されました

実行されるのはコントローラーの関数 `index` です

<http://127.0.0.1/konoha/diet/index> でも同じです

説明をすると

<http://localhost/konoha/コントローラ名/メソッド名/引数/引数>

5.3 ビュー

出力する HTML を用意します

```
<?php defined('SYSPATH') OR die('No direct access allowed.');?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>

    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
</head>
<body>
    <?php echo $content ?>
</body>
</html></body>
```

```
<html>
<body>
<table><tr>
<?PHP
foreach ($data as $d) {
    print "<tr><td>$d->wait</td> <td>$d->percent </td><td>$d->update</td></tr>";
}
?>
</tr></table>
</body>
</html>
```

ビューのファイルを保存

kohana/application/**views/diet**template.php を作成します

kohana/application/**views/diet**.php を作成します

HTML のなかに PHP を書くのは、テンプレートを使ってきた方には、違和感を感じる方がいる
と思います。これも軽量化のためです。我慢しましょう
記述された変数は、コントローラから渡されます。

5. 4 モデル

データベースをアクセスする部分を作りましょう

konohaapplication/classes/model/diet.php を作成します

クラス名の書き方が変更になりました

query の引数も変更になりました

type sql の種類を入れなければなりません select:,insert....

Query 同じ

Object true:stdClass

関数 GetData で情報を取得します

```
<?php defined('SYSPATH') or die('No direct script access.');?>

class Model_diet extends Model {
    protected $db;
    public function __construct()
    {
        parent::__construct();
        $this->db = Database::instance();
    }
    public function getData() {
        $sql="select wait,percent,update from diet order by update DESC";
        $result=$this->db->query(Database::SELECT,$sql,true);
        return $result;
    }
}
?>
```

さあ、準備ができました。コントローラーを修正して、モデル、ビューを使えるようにしましょう

5. 5 コントローラの修正

konoha/application/classes/controller/**diet**.php を修正します

```
<?php defined('SYSPATH') or die('No direct script access.');
```

```
class Controller_diet extends Controller_Template  
{  
    public $template = 'diettemplate';  
  
    public function action_index()  
    {  
        $this->template->content = new View('diet');           //ビュー  
        $dat = new model_diet;                                //モデル  
        $result=$dat->Getdata();                               //データ取得  
        $this->template->content->data=$result;  
    }  
}  
?>
```

アクセスしてみましょう

<http://127.0.0.1/konoha/diet>

どうですか? データは表示されましたか?

- kohana を使うと、ファイルが3個に分割され、それぞれの役割分担ができています
- データベースの出力は `stdClass` になっている。テーブルの項目名で参照できるは、ミスが少なくなっていて便利です
- データベース接続などは書かなくてもいいです
- 分割されているので、単体検査が楽になります

6 エラーを出してみよう

プログラムを作成する段階で、必ず行うことがシンタックスのデバッグです。

たとえば

- 変数名を間違った
- SQL 文を間違った
- 「;」を忘れた
- 関数名を間違った
- PHP の書き方間違い

まあ、数え上げればきりがありません

このようなときに、kohana はどのようなエラーを返してくれるのでしょうか。結構、いい情報を出してくれるので、開発の生産性に寄与します。

出力例

controller/diet.php で定義されていない変数\$results を使ったときです。実際はミススペルで \$result でした

v2.3.4 より見やすくなりました

```
ErrorException [ Notice ] Undefined variable: results

APPSPATH/classes/controller/diet.php [ 12 ]

7         public function action_index()
8         {
9
10
11
12     $this->template->content->data=$results;
13 //  firebug::debug($this->obj2arr($result[0]), __FILE__, __LINE__);
14
15
16     $view->data=$result;
17     $view->render();

1. APPSPATH/classes/controller/diet.php [ 12 ] » Kohana_Core::error_handler()
2. {PHP internal call} » Controller_diet->action_index()
3. SYSPATH/classes/kohana/request.php [ 850 ] » ReflectionMethod->invokeArgs(arguments)
4. APPSPATH/bootstrap.php [ 76 ] » Kohana_Request->execute()
5. DOCROOT/index.php [ 106 ] » require(arguments)

Environment
```

6. 1 定義されていない変数を使った。変数のミススペル

6. 2 「;」を忘れた

通常の PHP エラーが出ます

6. 3 定義されていないビューを指定した

6. 4 モデルのクラス名を間違えた

クラス名がないと表示される model/diet.php 内に `Class Dietm_model()`がない。

6. 5 定義されていない関数を使用した

model/diet.php に `Getdata1` 関数がない

6. 6 データベースの結果セットから間違った項目名で情報を取得した

間違った項目名が表示されます

6. 7 データベースからデータが取得できなかったの結果セットからデータを取り出そうとしたこのような例が、一番面倒です

具体的な行や、変数名などが表示されません

テーブル

スキーマ

```
CREATE TABLE IF NOT EXISTS `roles` (  
  `id` int(11) UNSIGNED NOT NULL AUTO_INCREMENT,  
  `name` varchar(32) NOT NULL,  
  `description` varchar(255) NOT NULL,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `uniq_name` (`name`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
INSERT INTO `roles` (`id`, `name`, `description`) VALUES(1, 'login', 'Login  
privileges, granted after account confirmation');  
INSERT INTO `roles` (`id`, `name`, `description`) VALUES(2, 'admin',  
'Administrative user, has access to everything.');
```

```
CREATE TABLE IF NOT EXISTS `roles_users` (  
  `user_id` int(10) UNSIGNED NOT NULL,  
  `role_id` int(10) UNSIGNED NOT NULL,  
  PRIMARY KEY (`user_id`, `role_id`),  
  KEY `fk_role_id` (`role_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE IF NOT EXISTS `users` (  
  `id` int(11) UNSIGNED NOT NULL AUTO_INCREMENT,  
  `email` varchar(127) NOT NULL,  
  `username` varchar(32) NOT NULL DEFAULT '',  
  `password` char(50) NOT NULL,  
  `logins` int(10) UNSIGNED NOT NULL DEFAULT '0',  
  `last_login` int(10) UNSIGNED,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `uniq_username` (`username`),  
  UNIQUE KEY `uniq_email` (`email`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE IF NOT EXISTS `user_tokens` (  
  `id` int(11) UNSIGNED NOT NULL AUTO_INCREMENT,  
  `user_id` int(11) UNSIGNED NOT NULL,  
  `user_agent` varchar(40) NOT NULL,
```

```

`token` varchar(32) NOT NULL,
`created` int(10) UNSIGNED NOT NULL,
`expires` int(10) UNSIGNED NOT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `uniq_token` (`token`),
KEY `fk_user_id` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `roles_users`
  ADD CONSTRAINT `roles_users_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES
`users` (`id`) ON DELETE CASCADE,
  ADD CONSTRAINT `roles_users_ibfk_2` FOREIGN KEY (`role_id`) REFERENCES
`roles` (`id`) ON DELETE CASCADE;

ALTER TABLE `user_tokens`
  ADD CONSTRAINT `user_tokens_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES
`users` (`id`) ON DELETE CASCADE;

```

データ

Users								
			id	email	username	password	logins	last_login
<input type="checkbox"/>			1		test	tes	42	1250045553
<input type="checkbox"/>			2	q	admin	tes	0	NULL

Roles_user					
			id	name	description
<input type="checkbox"/>			1	login	Login privileges, granted after account confirmati...
<input type="checkbox"/>			2	admin	Administrative user, has access to everything.

Roles				
			user_id	role_id
<input type="checkbox"/>			1	1