

CakePHPをさらにDRYにする ドライケーキレシピ

2007.12.7

akiyan.com 秋田真宏

かなりピンポイントな
話をします

コードがたくさん出てきます

自己紹介

秋田真宏

あきやん

青い人

石川県出身

(県庁所在地：金沢)

何やってる人？

会社員

自社開発系

プログラマー

SE

Cake歴 約1年

akiyan.com 管理人



私の仕事探しは
ファインドボタンから



求人検索 | 転職 | アルバイト | 派遣

「いい言葉は、いい人生をつくる」より放さず

ブログマーケティング会議 sponsored by SONY みんなのスタートアップまとめ

あなたの知らない照明術

2007-01-16 written by akiyan (40 users)

今年初めのスゴ本に出会いました。寝る間を惜しんで長文エントリーします。



[話がよくなる照明術](#)
これまでこのサイトから 401人が購入しました
posted with amazlet on 07.01.16

稲城 秀英
PIIP研究所
売り上げランキング: 5238

おすすめ度の平均: ★★★★★
★★★★★ これて楽しく会社にいけるかな? !?
★★★★★ 灯りは明るければよいと興っていましたか
★★★★★ かべん、なるほどねえ~

[Amazon.co.jp で詳細を見る](#)

もう、全ての人が読んでもらいたいです。

特に部屋の雰囲気を変えたいなど僕然と思っている方は今すぐ読むべきだと思います。

私はとくに何も考えずに読んだのですが、読後にもたつてもいられず24時間以内に同接離席器具を3つ購入、照明リフォームを完済させていました。そしてリフォームの結果に大満足です。費用は約5000円。安いです。照明リフォーム後の部屋を人に見せたところとても好評でした。

ちなみになぜリフォームしたくなったかというと、クリエイティブ職 (= リラックスした状

akiyan.com とは

1997年開設。

管理人は秋田真宗、ニックネームは「あきやん」。

話題の中心はWeb系のブログライティング、サーバー、デザイン、マークアップなど。

書評、フィードバック、自己啓蒙、雑談などがあります。

共著書



人気の記事

gihyo.jpでCakePHPの
連載を執筆しています

自己紹介おわり

本編ここから

**Conditionsと
array_mergeで
DRYになる！**

Conditionsとは？

Cakeのモデルの
find系メソッドに
渡す検索条件

配列か文字列を渡せる

配列の例

```
$this->User->findAll(  
    array(  
        'status' => 'active',  
        'confirmed_email' => 'yes',  
    )  
)
```

文字列の例

```
$this->User->findAll(  
    " status = 'active' AND confirmed_email = 'YES' "  
)
```

今回は配列に注目します

て、

毎回Conditionsを書くのは
面倒だし、
条件が多いと書き忘れがち

findをDRYにしたい！

その前に、「DRY」って何？

DRYとは

Don't

Repeat

Yourself

の略

意味は

「繰り返して書くな」

というわけで

さっきのfindを
DRYにしてみる

ありがちな実装

モデルに専用メソッドを
定義する

```
Class User extends AppModel {  
  var $name = 'User';  
  function getActiveUser() {  
    return $this->User->findAll(array(  
      'status' => 'active',  
      'confirmed_email' => 'yes',  
    ));  
  }  
}
```

```
$this->User->getActiveUser();
```

これでもいいっちゃいいけど

LimitとOffset(page)とか
指定したくなったら？

解決例：メソッドを拡張する

```
Class User extends AppModel {  
  var $name = 'User';  
  function getActiveUser($limit = null, $page = null) {  
    return $this->User->findAll(array(  
      'status' => 'active',  
      'confirmed_email' => 'yes',  
    ), null, null, $limit, $page);  
  }  
}
```

さらにソート条件も
指定したくなったら？

やっぱりメソッドを拡張する

```
Class User extends AppModel {  
  var $name = 'User';  
  function getActiveUser($order = null, $limit = null,  
$page = null) {  
    return $this->User->findAll(array(  
      'status' => 'active',  
      'confirmed_email' => 'yes',  
    ), null, $order, $limit, $page);  
  }  
}
```

さらにさらに
取得フィールドも指定...

...って、やってらんない

機能を増やせば増やすほど
どんどん元のfindAllメソッドに
近づいていく

専用メソッドを作るたびに
findAllに似たようなものを
作るのは、
そもそもDRYじゃない

そこで

案 1

よく使うConditionsを
モデルのメンバ変数に
定義しておく

```
Class User extends AppModel {  
  var $name = 'User';  
  var $cond_activeuser = array(  
    'status' => 'active',  
    'confirmed_email' => 'yes',  
  );  
}
```

```
$this->User->findAll($this->User->cond_activeuser);
```

案1のメリット

find(all)の機能が
そのまま使える

メソッドを選ばない

(find, findAll, generateListなど)

お手軽

(メンバ変数に定義するだけ)

Conditionsのカスタマイズも
そこそこ柔軟

例えば、性別条件を
追加するとき

サンプル

```
// 性別条件を追加
$this->User->findAll(
    am(
        $this->User->cond_activeuser,
        array('sex' => 'male')
    )
);
```

解説

```
// 性別条件を追加
$this->User->findAll(
    am(
        $this->User->cond_activeuser,
        array('sex' => 'male')
    )
);
```

am = cake/basics.phpで定義された、
array_mergeへのエイリアス

```
// 性別条件を追加
$this->User->findAll(
    am(
        $this->User->cond_activeuser,
        array('sex' => 'male')
    )
);
```

cond_activeuserの配列に
'sex' キーで 'male' の値をマージ。

```
// 結果
```

```
Array(  
  [status] => active  
  [confirmed_email] => yes  
  [sex] => male  
)
```

とりあえずは案 1 で
結構いける

が

Conditionsが複雑になると
ちょっと非力

例えば
複数条件(or)があるような
Conditionsとか

例

```
Class User extends AppModel {  
  var $name = 'User';  
  var $cond_activeuser = array(  
    'or' => array(  
      array(  
        'status' => 'active',  
        'confirmed_email' => 'yes',  
      ),  
      array(  
        'status' => 'active',  
        'confirmed_phone' => 'yes',  
      ),  
    ),  
  );  
};  
}
```

わかりにくいので
print_rした形で

```
Array
(
  [or] => Array
    (
      [0] => Array
        (
          [status] => active
          [confirmed_email] => yes
        )
      [1] => Array
        (
          [status] => active
          [confirmed_phone] => yes
        )
    )
)
```

このConditionsに適切に
性別条件を追加するには？

array_mergeでは、まず無理

array_merge_recursiveも
試してみたけど
できなかった

失敗例

```
array_merge_recursive($this->User->cond_activeuser,  
    array(  
        'or' => array(  
            array('sex' => 'male'),  
        )  
    )  
)
```

結果

```
Array
(
  [or] => Array
    (
      [0] => Array
        (
          [status] => active
          [confirmed_email] => yes
        )
      [1] => Array
        (
          [status] => active
          [confirmed_phone] => yes
        )
      [2] => Array
        (
          [sex] => male
        )
    )
)
```

惜しい orz

お寒い成功例

```
$cond = $this->User->cond_activeuser;  
$cond['or'][0]['sex'] = 'male';  
$cond['or'][1]['sex'] = 'male';
```

元のConditionsによって
条件を追加する方法が
変わってしまう

そもそも中間変数を必要としてしまっている

ありえない

そこで、

案 2

Conditionsを返す関数を
用意する

引数に追加条件を渡すと
よしなにマージしてくれる
機能つき

実装例 1

```
Class User extends AppModel {  
  var $name = 'User';  
  function getActiveCond($merger) {  
    return am(array(  
      'status' => 'active',  
      'confirmed_email' => 'yes',  
    ), $merger);  
  }  
}
```

```
$this->User->getActiveCond(array('sex' => 'male'));
```

```
Array(  
    [status] => active  
    [confirmed_email] => yes  
    [sex] => male  
)
```

この例はメンバ変数版との
差はほとんど無い

(array_mergeでも普通にできちゃうから)

複数条件のあるConditionsで
真価を発揮

実装例 2

複数条件

```
Class User extends AppModel {
  var $name = 'User';
  function getActiveCond($merger) {
    return array(
      'or' => array(
        am(array(
          'status' => 'active',
          'confirmed_email' => 'yes',
        ), $merger),
        am(array(
          'status' => 'active',
          'confirmed_phone' => 'yes',
        ), $merger),
      ),
    );
  }
}
```

使ってみる

```
$this->User->getActiveCond(array('sex' => 'male'));
```

```
Array
```

```
(  
  [or] => Array  
  (  
    [0] => Array  
    (  
      [status] => active  
      [confirmed_email] => yes  
      [sex] => male  
    )  
    [1] => Array  
    (  
      [status] => active  
      [confirmed_phone] => yes  
      [sex] => male  
    )  
  )  
)
```

できた！

案2のメリット

メソッドを選ばない

(案1と同じく)

複雑な条件でも、
追加条件を渡すだけで
Conditionsをカスタム可能

元の条件が変化しても
コントローラー側のコードは
変更しなくてもよい

それって超DRY！

ドライケーキつ焼き上がり！



← 銀座 洋菓子舗 ウエストに
Dry cake というケーキがあるらしい

<http://www.ginza-west.co.jp/>

いろいろやってみて
気づいたこと

レコード検索の条件指定が
連想配列だと
とても加工しやすい

SQLをsprintfとかで
組み上げていた頃からは
考えられない柔軟性

(prepareもしかり)

SQLをコードで組み立てるのは、
別の言語のコードを
コードによって組み立てるような
非常に煩雑な作業なのでは...？

Conditions 万歳(^o^)

以上、
Conditionsとarray_merge
でDRYになる！
でした

ご静聴ありがとうございました