

転載禁止

GoodfindEngineer ACADEMY

参加無料

学年不問

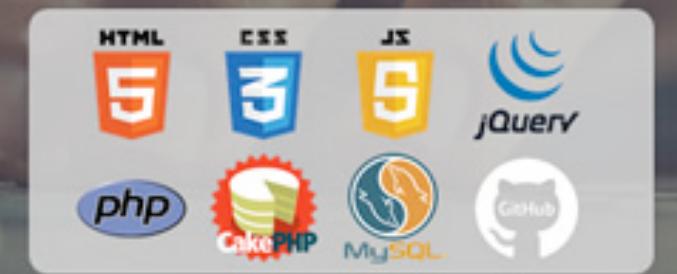
20名限定

この夏、基本からWebプログラミングを極める3日間の集中コース

- WebAPIやデザインフレームワークを使った、オリジナルのWebサービス・プロダクトを創る -

GoodfindEngineer

<3 Days= “Programming Bootcamp” >



無線LANにご接続ください

SSID : slogan-guest (- 5g)

PASS : welcomeslogan

Copyright © 2006-2015 SLOGAN Inc. All Rights Reserved.

SLOGAN

小滝 大輔

Daisuke Kotaki



Goodfind Engineer ACADEMY

スローガン株式会社

テクノロジーデザイングループ：エンジニア

早稲田大学大学院の数学研究科を修了。
業務支援パッケージソフトウェアを開発するメガベンチャーに入社し、主に販売管理の機能を開発。
人事系のシステム会社を経た後、スローガン株式会社に入社。現在はGoodfindの機能開発、セミナー・面談講師などを担当。

このセミナーの目的

- ・ Webページからのデータの操作を一通り理解する
- ・ MVCフレームワークでの効率的な開発の流れを理解する
- ・ 「なんとなく作れそう！」という自信を得る

Agenda

本日のアジェンダ

1. 更新系画面の開発
2. CakePHPの標準機能の利用

GETとPOST(データ送信の形式)

- データをURLに含めて送信するのが「GET」
- データをURLに含まずに送信するのが「POST」

POSTでデータを送信する場合は、
formタグを使う

```
<form action="/users/login" method="post">  
  
</form>
```

POSTで送信するデータの入力は、
inputタグを使う

```
<form action="/users/login" method="post">  
  
    <input name="id">  
    <input name="title">  
    <input name="body">  
  
</form>
```

データの入力形式の指定には、
type属性を使う(textarea, hiddenなど)

```
<form action="/users/login" method="post">  
  
  <input name="id" type="hidden">  
  <input name="title">  
  <input name="body" type="textarea">  
  
</form>
```

送信ボタンは
type属性にsubmitを指定する

```
<form action="/users/login" method="post">  
  
  <input name="id" type="hidden">  
  <input name="title">  
  <input name="body" type="textarea">  
  
  <input value="send" type="submit">  
</form>
```

CakePHPでは、formの生成には

FormHelperを使う！

(View上で、\$this->Form で呼び出せる)

formタグは create ~ end で生成
(createには対応するModel名を入力

```
<?php echo $this->Form->create('Topic'); ?>  
  
<?php echo $this->Form->end(); ?>
```

データの入力項目には、
input関数を使う

```
<?php echo $this->Form->create(); ?>

<?php echo $this->Form->input(); ?>

<?php echo $this->Form->end(); ?>
```

inputには、対応するカラム名を渡す
(name属性に対応)

```
<?php echo $this->Form->create('Topic'); ?>

<?php echo $this->Form->input('body'); ?>

<?php echo $this->Form->end(); ?>
```

データの入力形式の指定には、
2番目の引数でtypeを設定(text, textarea, hiddenなど)

```
<?php echo $this->Form->create('Topic'); ?>
<?php echo $this->Form->input('id',
    array('type' => 'hidden'))); ?>
<?php echo $this->Form->input('body',
    array('type' => 'textarea'))); ?>

<?php echo $this->Form->end(); ?>
```

送信ボタンは submit関数を利用する

```
<?php echo $this->Form->create('Topic'); ?>

<?php echo $this->Form->input('id',
    array('type' => 'hidden'))); ?>
<?php echo $this->Form->input('body',
    array('type' => 'textarea'))); ?>

<?php echo $this->Form->submit('登録'); ?>

<?php echo $this->Form->end(); ?>
```

記事を追加する画面を作成してみよう！

- URLは、
[トップページのURL]/**topics/add**
- formの中にタイトルと本文の入力欄をつくる
- デザインもできれば整えてみよう！

このような画面を
作成してみよう！

記事追加

Title

Body

追加

実装例 (app/View/Topics/add.ctp)

```
<h2>記事追加</h2>

<?php echo $this->Form->create('Topic'); ?>
<?php echo $this->Form->input('title',
    array('type' => 'text')); ?>
<?php echo $this->Form->input('body',
    array('type' => 'textarea')); ?>
<?php echo $this->Form->submit('追加'); ?>
<?php echo $this->Form->end(); ?>
```

POSTされたデータは、Controllerの
\$this->request->data に渡される

```
var_dump($this->request->data);  
  
↓  
  
array(  
    'Topic' => array(  
        'title' => '吾輩は猫である',  
        'body' => '吾輩は猫である。名前はまだない。...(略)'  
    )  
)
```

データの送信形式がPOSTかどうかは
\$this->request->is() 関数で判定する

```
if ($this->request->is('post')) {  
    (保存するときの処理)  
}  
else {  
    (表示するときの処理)  
}
```

データを保存する

Model::save()

を使う

Model::save() = INSERT or UPDATE

(どちらになるかは、'id'の値があるかで決まる)

引数には、Modelと同名のキーを含む配列を渡す
Model::save()実行前に、Model::create()が必要

```
$this->Topic->create();  
  
$this->Topic->save($this->request->data);
```

記事を追加する処理(Controller)を作成してみよう！

- URLは、
[トップページのURL]/**topics/add**
- POSTで送信された場合にのみ、保存処理を実行
- 入力項目のないカラムには仮の値(1など)を入れる

まずは、POST形式かどうかを判定 (TopicsController.php)

```
public function add() {  
    if ($this->request->is('post')) {  
    }  
}
```

save() 関数で入力したデータを保存

```
public function add() {  
    if ($this->request->is('post')) {  
  
        $this->Topic->create();  
        $this->Topic->save($this->request->data);  
    }  
}
```

category_id, user_id に仮の値を入力

```
public function add() {  
    if ($this->request->is('post')) {  
        $this->request->data['Topic']['category_id'] = 1;  
        $this->request->data['Topic']['user_id'] = 1;  
  
        $this->Topic->create();  
        $this->Topic->save($this->request->data);  
    }  
}
```

カテゴリを選択入力できるようにする

カテゴリー一覧を取得してViewにset

```
public function add() {  
    if ($this->request->is('post')) {  
        // $this->request->data['Topic']['category_id'] = 1;  
        $this->request->data['Topic']['user_id'] = 1;  
        (中略)  
    }  
    $categories = $this->Category->find('list');  
    $this->set('categories', $categories);  
}
```

カテゴリの選択欄をつくる (View/Topics/add.ctp)

```
<?php echo $this->Form->input(  
    'category_id', array('type' => 'select')  
); ?>
```

Model::save()の返り値は

成功したら true

失敗したら false

保存が成功した場合は
トップページにもどる

```
public function add() {  
    if ($this->request->is('post')) {  
        (前略)  
  
        $this->Topic->create();  
        if ($this->Topic->save($data)) {  
            $this->redirect('/');  
        }  
    }  
}
```

記事詳細にコメント追加機能を作成してみよう！

- URLは、
[トップページのURL]/**detail/[:id]**
- 名前・タイトル・コメントの入力欄をつける
- POST判定、save()関数の使い方を確認しながら…

データを削除する

Model::delete()

を使う

Model::delete() = DELETE

引数には、Modelのプライマリキーの値を渡す
(基本的にはidの値)

```
$this->Topic->delete($id);
```

Model::delete()の返り値も

成功したら true

失敗したら false

記事の削除を完成させよう！

- URLは、
[トップページのURL]/**topics/delete/[:id]**
- 記事一覧(トップページ)に、「削除」ボタンを設置
- 削除後は、トップページに戻る

編集・削除のボタンを 記事一覧に設置

記事タイトル 記事タイトル

[Edit](#) [Delete](#)

テキストテキストテキストテキストテキストテキストテキストテキストテキストテキストテキストテキスト
テキストテキストテキストテキストテキストテキストテキストテキストテキストテキストテキストテキスト
テキストテキストテキストテキストテキストテキストテキストテキストテキストテキストテキストテキスト
テキストテキストテキストテキストテキストテキストテキストテキストテキストテキストテキストテキスト

[Continue reading](#)

[日記](#)

delete.ctp を作る必要はない！

これだけでもOK (idが正常値なら)

(TopicsController.php)

```
public function delete($id = null) {  
    $this->Topic->delete($id);  
    $this->redirect('/');  
}
```

Model::delete() の実行結果により メッセージを分ける

```
public function delete($id = null) {  
    if ($this->Topic->delete($id)) {  
        $this->Session->setFlash('成功しました');  
    } else {  
        $this->Session->setFlash('失敗しました');  
    }  
    $this->redirect('/');  
}
```

トップページで実行結果のメッセージを表示 (View/Pages/index.ctp)

(略)

```
<div class="alert alert-success">
    <?php echo $this->Session->flash(); ?>
</div>
<?php foreach ($topics as $topic): ?>
```

(略)

記事の編集画面を完成させよう！

- URLは、
[トップページのURL]/topics/detail/[**:id**]
- POSTでない場合はModel::findを使ってSELECT
POSTの場合はModel::saveを使ってUPDATE
- input hiddenタグなどを使って**id**をPOST送信しないと、
Model::saveではINSERT扱いになるので注意！

Model::find() = SELECT

Model::save() = INSERT or UPDATE

Model::delete() = DELETE

データの操作がすべて揃った！

Agenda

本日のアジェンダ

1. 更新系画面の開発
2. CakePHPの標準機能の利用

AuthComponentを使った認証

AuthComponent::login() = ログイン実行

AuthComponent::logout() = ログアウト実行

AuthComponent::user() = ユーザー情報取得

AuthComponent::password() = パスワード暗号化

AuthComponent::allow() = 未ログイン者の利用許可

ログイン画面を作成しよう！

- URLは、
[トップページのURL]/**users/login**
- login() を実行し、結果によって処理を分ける

まずはViewを作成 (Users/login.ctp)

```
<h2>ログイン</h2>
<?php echo $this->Form->create('User');
<?php echo $this->Form->input('username'); ?>
<?php echo $this->Form->input('password'); ?>
<?php echo $this->Form->submit('ログイン'); ?>
```

このような画面になればOK

ログイン

Username

Password

ログイン

POSTの場合のみログインを試す (UsersController.ctp)

```
public function login(){
    if($this->request->is('post')) {
        if($this->Auth->login()) {
            return $this->redirect('/');
        } else {
            $this->Session->setFlash('ログイン失敗');
        }
    }
}
```

ログアウト機能を作成しよう！

- URLは、
[トップページのURL]/**users/logout**
- logout() を実行し、トップページへリダイレクト

POSTの場合のみログインを試す (UsersController.ctp)

```
public function logout(){
    $this->Auth->logout();
    $this->redirect('/');
}
```

ユーザー登録画面を作成しよう！

- URLは、
[トップページのURL]/**users/register**
- AuthComponent::password() 関数による
パスワードの暗号化が必要

まずはViewを作成 (Users/register.ctp)

```
<h2>新規登録</h2>
<?php echo $this->Form->create('User');
<?php echo $this->Form->input('username'); ?>
<?php echo $this->Form->input('password'); ?>
<?php echo $this->Form->submit('登録'); ?>
```

POSTの場合は入力値を保存 (UsersController.php)

```
public function register(){
    if($this->request->is('post')) {
        $data = $this->request->data;
        $data['User']['password']
            = $this->Auth->password($data['User']['password']);
        if ($this->User->save($data)) {
            $this->Auth->login();
            $this->redirect('/');
        }
    }
}
```

ログイン状態をもとにメニューを変えよう！

- AuthComponent::user() 関数を使って、
ユーザー情報を取得
- ユーザー情報が空の場合は
ログインしていない人と判断
- メニューを記述している場所は？

各アクションに共通の処理は

Controller::beforeFilter

に記述する

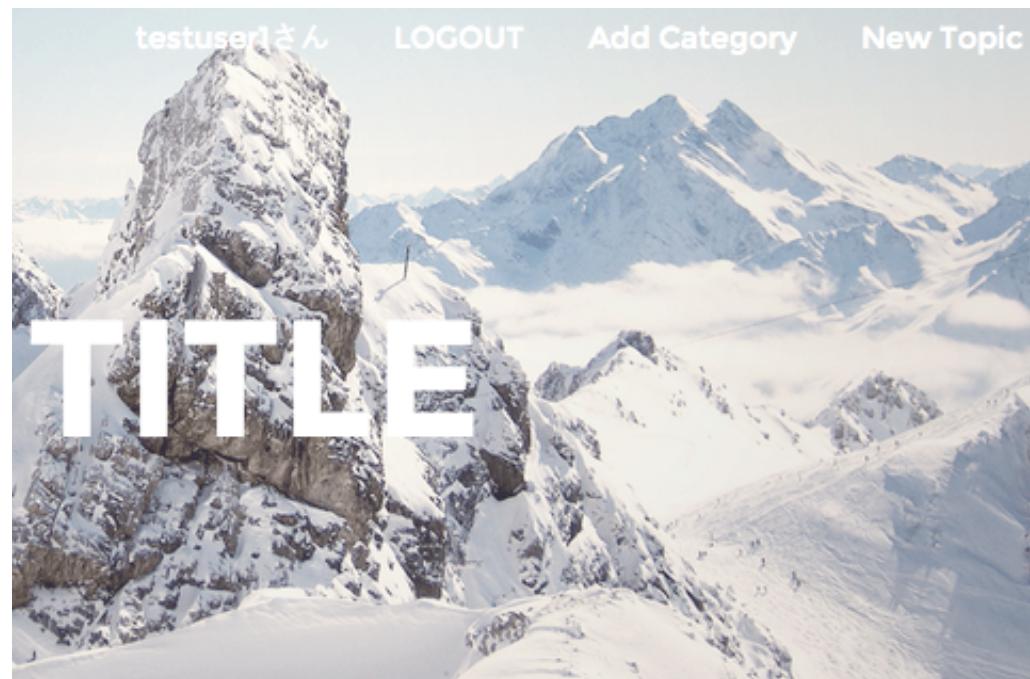
Controllerでログイン状態を確認 (PagesController.php, TopicsController.php)

```
public function beforeFilter() {  
  
    parent::beforeFilter();  
    $user = $this->Auth->user();  
    $this->set('user', $user);  
}
```

ユーザー情報の有無でメニューを分岐 (View/Elements/navbar.ctp)

```
<?php if(!empty($user)): ?>  
  
    (ログインした人のメニュー)  
  
<?php else: ?>  
  
    (ログインしていない人のメニュー)  
  
<?php endif; ?>
```

ログインした人のメニュー



ログインしていない人のメニュー



ログイン状態をもとに表示を変えよう！

- AuthComponent::allow() 関数を使って、
ログインしていないなくても使える画面を記述
- AuthComponent::user() 関数を使って、
ログイン状態・ユーザーの情報を取得

CakePHPによるWebアプリケーション開発

Goodfind Engineer ACADEMY

ブログ、できましたか？

早く終わってしまった方に

機能を拡張してみよう

- カテゴリの一覧・追加・編集・削除などを作成
- 記事の編集・削除をログインした人しかできないように
- データベースにある情報以外のものを追加してみる
(タグ、ブックマーク etc.)

SLOGAN

Goodfind Engineer ACADEMY

本日はご参加ありがとうございました。

講師の連絡先はこちら

小滝 大輔

Daisuke Kotaki

<http://www.facebook.com/daisuke.kotaki>

Email : kotaki.daisuke@slogan.jp

Copyright © 2006-2015 SLOGAN Inc. All Rights Reserved.