# DESENVOLVIMENTO PARA WEB II
# Entendendo o código gerado com CakePHP

Professor: Alexandre Strapação Guedes Vianna – IFPE Igarassu
alexandrevianna.net

# Roteiro da Aula

## Relembrando...

1. Construção da primeira aplicação
    - Configurar a senha de acesso ao banco
    - Criar um esquema no banco
    - Tutorial: construção da aplicação BookMark
        - Uso do bake para gerar as classes de modelo, controle e template (visão)

# Roteiro da Aula

1. Entender o código gerado pelo comando bake
2. Estrutura de Diretórios
3. View Templates
4. Controller
   - Index, Add, Delete, View e edit
5. Model

# Estrutura de Diretórios

**O CakePHP apresenta os seguintes diretórios:**

- bin
- config
- logs
- plugins
- src
- tests
- tmp
- vendor
- webroot
- .htaccess
- composer.json
- index.php
- README.md

# Estrutura de Diretórios

O diretório **bin** armazena os arquivos executáveis do CakePHP: server, bake,....

O diretório **config** contem os (poucos) Configuração arquivos de configuração que o CakePHP utiliza. Detalhes de conexão com banco de dados, inicialização, arquivos de configuração do núcleo da aplicação, e relacionados devem ser postos aqui.

O diretório **logs** será normalmente onde seus arquivos de log ficarão, dependendo das suas configurações.

O diretório **src** será onde você fará sua mágica: é onde os arquivos da sua aplicação serão colocados.

# Estrutura de Diretórios

O diretório **tests** será onde você colocará os testes de caso para sua aplicação.

O diretório **tmp** será onde o CakePHP armazenará dados temporários. O modo como os dados serão armazenados depende da configuração do CakePHP, mas esse diretório é comunmente usado para armazenar descrições de modelos e algumas vezes informação de sessão.

O diretório **vendor** será onde o CakePHP e outras dependências da aplicação serão instalados. Faça uma nota pessoal para não editar arquivos deste diretório. Nós não podemos ajudar se você tivé-lo feito.

O diretório **webroot** será a raíz pública de documentos da sua aplicação. Ele contem todos os arquivos que você gostaria que fossem públicos.

# View Templates

- O CakePHP tem uma extensão padrão  .ctp (CakePHP template)
- O arquivos de templates são armazenados em: src/Template
- A sintáxe é mixi de PHP + html com alguns comandos alternativos do CakePHP
- Estes arquivos contem apenas a lógica necessária para preparar/formatar os dados recebidos do controller e exibir na camada de apresentação.

# View Templates

## Variáveis da View

- Variáveis setadas com o a função set() no controller ficam disponíveis no template.

Forma com 2 parâmetros

```
$variable_to_pass = 'Fred';

$this->set('variable_to_pass', $variable_to_pass);
```

Forma com 1 parâmetro

```
$variable_to_pass = 'Fred';

$this->set(compact('variable_to_pass'));
```

# View Templates

## Variáveis da View

- Função index em ArticlesController.php

```
public function index(){
    $articles = $this->paginate($this->Articles);
    $this->set(compact('articles'));
    $this->set('_serialize', ['articles']);
}
```

# View Templates

## Variáveis da View

- Dentro da view acessamos com a função h() a variável setada no controller ou uma função de formatação de dados:
  Src/Template/Articles/Articles.ctp

```
<td><?= h($article->title) ?></td>

<td><?php echo $article->created; ?></td>
```

# View Templates

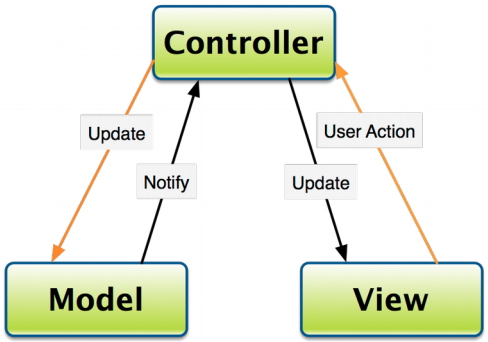
## Explorando a pasta Template gerada

- Dentro da pasta templates há subpastas para arquivos específicos de controllers. Por exemplo a pasta **Article** contém templates relacionados ao controller Article.

- **elements:** pedaços de código pequenos e reutilizáveis. Elements geralmente são renderizados dentro de views. Por exemplo: mensagens flash
- **erros:** contém os templates para páginas de erro padrão
- **:** contém os templates para páginas do controller pages, que já existe por padrão no cake PHP. Geralmente utilizamos o Pages para renderizar páginas não relacionadas a alguma entidade existente. Exemplo: home.ctp
- **E-mail**: templates para e-mails: texto puro ou html

# Controller

Os controllers (controladores) correspondem ao ‘C’ no padrão MVC. Após o roteamento ter sido aplicado e o controller correto encontrado, a ação do controller é chamada. Seu controller deve lidar com a interpretação dos dados de uma requisição, certificando-se que os models corretos são chamados e a resposta ou view esperada seja exibida. Os controllers podem ser vistos como intermediários entre a camada Model e View.



# Controller

Os controllers da sua aplicação são classes que estendem a classe AppController. *(src/Controller/AppController.php)* A classe AppController deve conter métodos que são compartilhadosentre todos os controllers de sua aplicação.

- Os controllers fornecem uma série de métodos que lidam com requisições. Estas são chamados de actions. Por padrão, todos os métodos públicos em um controller são uma action e acessíveis por uma URL.
  Exemplo
  - Localhost:8765/Articles/ ***padrão action index**
  - Localhost:8765/Articles/add
  - Localhost:8765/Articles/edit/[id]
  - Localhost:8765/Articles/view/[id]

# Controller

- **Definindo variáveis na View:**

O método Controller::set() é a principal maneira de enviar dados do seu controller para a sua view. Após ter usado o método Controller::set(), a variável pode ser acessada em sua view:

```
// Primeiro você passa os dados do controller:
$this->set('color', 'pink');
// Então, na view, você pode utilizar os dados:
?>
Você selecionou a cobertura <?php echo $color; ?> para o
bolo.
```

# Controller

- **Definindo variáeis na View:**

O método Controller::set() também aceita um array associativo como primeiro parâmetro. Isto pode oferecer uma forma rápida para atribuir uma série de informações para a view:

```
$data = [
    'color' => 'pink',
    'type' => 'sugar',
    'base_price' => 23.95
];

// Faça $color, $type, e $base_price
// disponíveis na view:

$this->set($data);
```

# Controller

- **Redirecionando:**

O método de controle de fluxo que você vai usar na majoritariamente é Controller::redirect(). Este método recebe seu primeiro parâmetro na forma de uma URL relativa do CakePHP. Quando um usuário executar um pedido com êxito, você pode querer redirecioná-lo para uma tela de recepção.

```
public function place_order()
{
    // Logic for finalizing order goes here
    if ($success) {
        return $this->redirect(
            ['controller' => 'Orders', 'action' => 'thanks']
        );
    }
    return $this->redirect(
        ['controller' => 'Orders', 'action' => 'confirm']
    );
}
```

# Controller

- **Redirecionando:**

Você também pode usar uma URL relativa ou absoluta como o parâmetro $url:

```
return $this->redirect('/orders/thanks');

return $this->redirect('http://www.example.com');
```

Você também pode passar dados para a action:

```
return $this->redirect(['action' => 'edit', $id]);
```

Se você precisa redirecionar o usuário de volta para a página que fez a requisição, você pode usar:

```
$this->redirect($this->referer());
```

# Controller

- **Redirecionando:**

Um exemplo usando seqüências de consulta e hash pareceria com:

```
return $this->redirect([
    'controller' => 'Orders',
    'action' => 'confirm',
    '?' => [
        'product' => 'pizza',
        'quantity' => 5
    ],
    '#' => 'top'
]);
```

A URL gerada seria:

```
http://www.example.com/orders/confirm?product=pizza&quantity=5#top
```

# Modelo

No CakePHP seu modelo de domínio da aplicação é dividido em 2 tipos de objetos principais. Os primeiros são repositories (**repositórios**) ou table objects (**objetos de tabela**). Estes objetos fornecem acesso a coleções de dados. Eles permitem a você salvar novos registros, modificar/deletar os que já existem, definir relacionamentos, e executar operações em massa. O segundo tipo de objetos são as entities (entidades). Entities representam registros individuais e permitem a você definir comportamento em nível de linha/registro e funcionalidades.

O ORM (MOR - Mapeamento Objeto-Relacional) nativo do CakePHP especializa-se em banco de dados relacionais, mas pode ser extendido para suportar fontes de dados alternativas.

# Controller

- **Por exemplo, se quiséssemos carregar alguns dados da nossa tabela articles poderíamos fazer:**

```
use Cake\ORM\TableRegistry;
$articles = TableRegistry::get('Articles');
$query = $articles->find();
foreach ($query as $row) {
    echo $row->title;
}
```

As convenções do CakePHP nos permitem **pular alguns códigos clichê**, e permitir que o framework insera classes básicas enquanto sua aplicação não criou uma classe concreta.

# Atividade Prática

**Continuar o tutorial do cake_blog**

***http://book.cakephp.org/3.0/pt/tutorials-and-examples/blog/part-three.html***

Vamos continuar o nosso aplicativo de blog e imaginar que queremos categorizar os nossos artigos. Queremos que as categorias sejam ordenadas, e para isso, vamos usar o comportamento de árvore para nos ajudar a organizar as categorias.

***Uso do Migrations**