Utilização do CakePHP na UFMS/AGETIC/DIDS

Apresentação

Autor: Paulo Roberto Sampaio Bezerra

Cargo: Analista de Tecnologia da Informação - Desenvolvimento de Software

Onde: Universidade Federal do Mato Grosso do Sul

Agência de Tecnologia da Informação e Comunicação

E-mail: paulo.bezerra@ufms.br

Programação

- O que é o CakePHP3
- CakePHP num piscar de olhos
- Convenções Sobre Configuração
- A camada Model
- A camada View
- A camada Controller
- Ciclo de Requisições do CakePHP
- Mais recursos
- Visão Prática

O que é o CakePHP3?

É um framework PHP 5.5+ moderno oferecendo:

- camada de acesso a base de dados flexível e
- um sistema poderoso de scaffolding que faz a construção de sistemas mais simples, fácil e naturalmente, mais saborosa.

CakePHP torna a construção de aplicações web mais simples, mais rápida, enquanto requer menos código.

CakePHP num piscar de olhos

O CakePHP é concebido para tornar tarefas de desenvolvimento web mais simples e fáceis.

Por fornecer uma caixa de ferramentas completa para você poder começar, o CakePHP funciona bem em conjunto ou isoladamente.

O objetivo desta análise é introduzir os conceitos gerais presentes no CakePHP, e lhe dar uma rápida visão geral de como estes conceitos são implementados.

Convenções Sobre Configuração

O CakePHP provê uma estrutura organizacional básica que cobre nomenclaturas de classes, nomenclaturas de arquivos, nomenclaturas de banco de dados, e outras convenções.

Apesar das convenções levarem algum tempo para serem assimiladas, ao seguí-las o CakePHP evita configuração desnecessário e cria uma estrutura de aplicação uniforme que faz trabalhar com vários projetos uma tarefa suave.

A camada Model

A camada Model representa a parte da sua aplicação que implementa a lógica de negócio.

Ela é responsável por recuperar dados e convertê-los nos conceitos significativos primários na sua aplicação.

Isto inclui processar, validar, associar ou qualquer outra tarefa relacionada à manipulação de dados.

A camada Model

Não precisamos escrever nenhum código antes de podermos começar a trabalhar com nossos dados.

Por usar convenções, o CakePHP irá utilizar classes padrão para tabelas e entidades ainda não definidas.

```php
use Cake\ORM\TableRegistry;

$users = TableRegistry::get('Users');
$query = $users->find();
foreach ($query as $row) {
    echo $row->username;
}
```

A camada View

A View renderiza uma apresentação de dados modelados.

Estando separada dos objetos da Model, é responsável por utilizar a informação que tem disponível para produzir qualquer interface de apresentação que a sua aplicação possa precisar.

```
// No arquivo view, nós renderizaremos um 'elemento' para cada usuário.
<?php foreach ($users as $user): ?>
    <div class="user">
        <?= $this->element('user', ['user' => $user]) ?>
    </div>
<?php endforeach; ?>
```

UFMS

A camada View

A camada View provê alguma variedade de extensões como Elements e View Cells (Células de Visualização) para permitir que você reutilize sua lógica de apresentação.

A camada View não está limitada somente a HTML ou apresentação textual dos dados. Ela pode ser usada para entregar formatos de dado comuns como JSON, XML, e através de uma arquitetura encaixável qualquer outro formato que você venha precisar.

A camada Controller

A camada Controller manipula requisições dos usuários. É responsável por renderizar uma resposta com o auxílio de ambas as camadas, Model e View respectivamente.

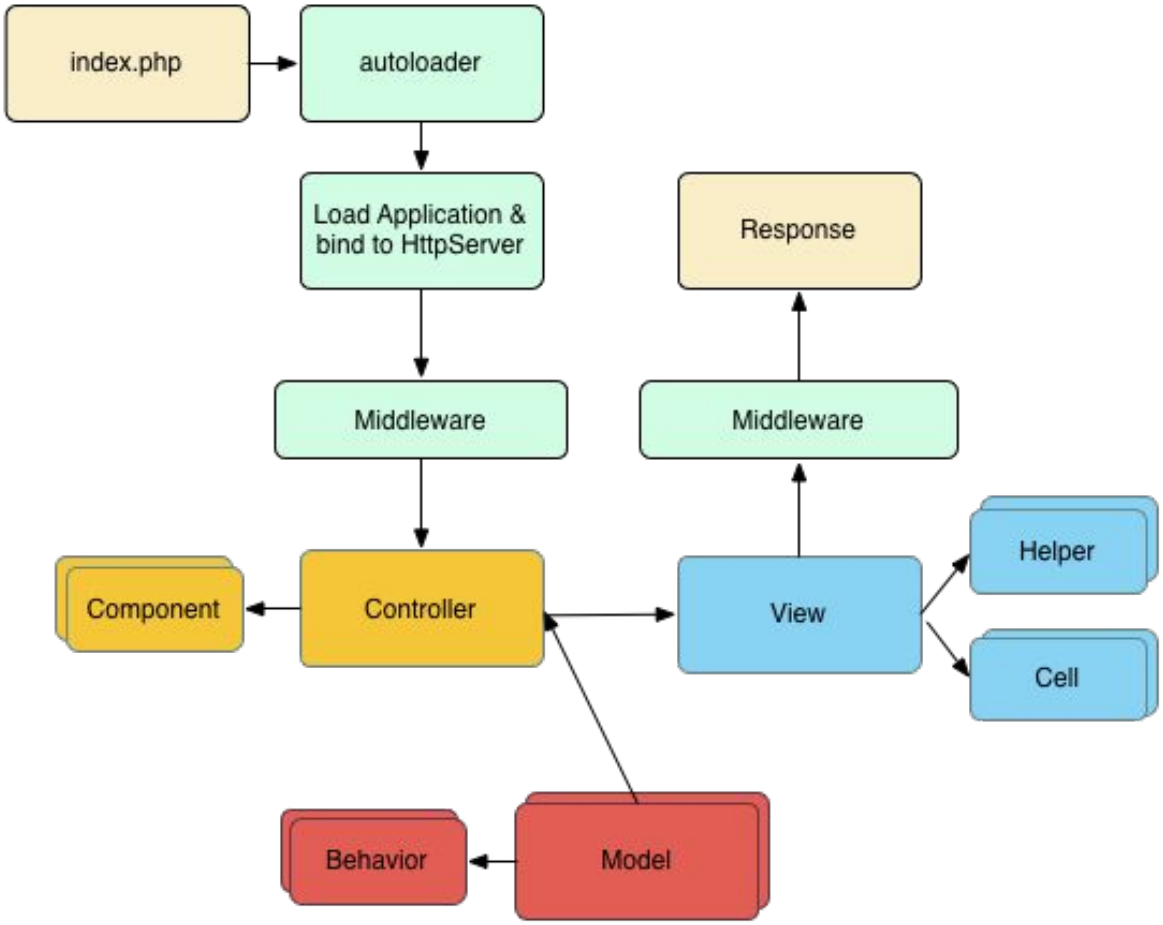
Ele aguarda por petições dos clientes, checa suas validades de acordo com autenticação ou regras de autorização, delega requisições ou processamento de dados da camada Model, selecciona o tipo de dados de apresentação que os clientes estão aceitando, e finalmente delega o processo de renderização para a camada View.

A camada Controller

```php
public function add()
{
    $user = $this->Users->newEntity();
    if ($this->request->is('post')) {
        $user = $this->Users->patchEntity($user, $this->request->data);
        if ($this->Users->save($user, ['validate' => 'registration'])) {
            $this->Flash->success(__('Você está registrado.'));
        } else {
            $this->Flash->error(__('Houve algum problema.'));
        }
    }
    $this->set('user', $user);
}
```

UFMS

Ciclo de Requisições do CakePHP



Ciclo de Requisições do CakePHP

1. As regras de reescrita do servidor encaminham a requisição para webroot/index.php.
2. Sua aplicação é carregada e vinculada a um HttpServer.
3. O middleware da sua aplicação é inicializado.
4. A requisição e a resposta são processados através do PSR-7 Middleware que sua aplicação utiliza. Normalmente isso inclui captura de erros e roteamento.
5. Se nenhuma resposta for retornada do middleware e a requisição contiver informações de rota, um Controller e uma action são acionados.
6. A action do Controller é chamada e o mesmo interage com os Models e Components requisitados.
7. O controller delega a responsabilidade de criar respostas à view, para assim gerar a saída de dados resultante do Model.
8. A View utiliza Helpers e Cells para gerar o corpo e cabeçalho das respostas.
9. A resposta é enviada de volta através do Middleware.
10. O HttpServer emite a resposta para o servidor web.

Mais recursos

Alguns outros grandes recursos no CakePHP são:

- Framework de cache que integra com Memcached, Redis e outros backends.
- Poderosas ferramentas de geração de código para você sair em disparada.
- Framework de teste integrado para você assegurar-se que seu código funciona perfeitamente.

Arquitetura NTI

CakePHP com Bootstrap

Bootstrap

- O Bootstrap é o framework HTML, CSS e JS mais popular para o desenvolvimento de projetos responsivos e mobile first na web.

- Originalmente criado pela equipe de desenvolvimento do Twitter com o nome de Twitter Bootstrap, atualmente o projeto recebe o nome apenas de Bootstrap.

- A Agetic utiliza o Bootstrap para o seu layout padrão de sistemas:
  - https://sistemas.ufms.br/layout/

Plugin NtiLayout

- Nossos projetos utilizam o layout NTI através de um plugin do CakePHP desenvolvido por nos mesmos.

- O plugin pode ser utilizado juntamente a ferramenta Cake Bake para gerar os CRUD’s no padrão do layout do NTI.

- Para executar o bake utilizando um plugin de layou é necessário passar o seguinte comando:
  - bin/cake bake all --theme NomeDoPlugin

    OU

  - bin/cake bake all -t NomeDoPlugin

Autenticação

- A autenticação dos sistemas é feita por um único serviço de autenticação.

- Dentro dos projetos em Cake existem duas classes que fazem a comunicação com os serviços de autenticação:
  - src/Auth/AuthUfmsAuthenticate.php
  - src/Auth/AuthUfmsAuthorize.php

- No arquivo config/boostrap.php é configurada a url do serviço de autenticação e identificador do sistema.

Autenticação

- Como resposta ao serviço de autenticação é devolvida uma lista de permissões do sistema para aquele usuário.

- Ou é negada a entrada do usuário no sistema devolvendo um erro 500 no status do http de resposta.

Visão Prática

Criar CRUDs utilizando apenas linha de comando e autenticação

Prática

1. Utilizar Migration para criar a estrutura de banco de dados

2. Utilizar Bake para criação de CRUD

3. Rodar a aplicação e testar CRUD

Prática - Instalação

$ php composer.phar install

$ php bin\cake server

Prática - Banco de Dados

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    email VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    created DATETIME,
    modified DATETIME
);

CREATE TABLE bookmarks (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    title VARCHAR(50),
    description TEXT,
    url TEXT,
    created DATETIME,
    modified DATETIME,
    FOREIGN KEY user_key (user_id) REFERENCES users(id)
);
```

Prática - Banco de Dados

```
CREATE TABLE tags (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255),
    created DATETIME,
    modified DATETIME,
    UNIQUE KEY (title)
);

CREATE TABLE bookmarks_tags (
    bookmark_id INT NOT NULL,
    tag_id INT NOT NULL,
    PRIMARY KEY (bookmark_id, tag_id),
    INDEX tag_idx (tag_id, bookmark_id),
    FOREIGN KEY tag_key(tag_id) REFERENCES tags(id),
    FOREIGN KEY bookmark_key(bookmark_id) REFERENCES bookmarks(id)
);
```

Prática - Migrações

$ bin/cake bake migration CreateUsers email:string password:string created modified

$ bin/cake bake migration CreateBookmarks user_id:integer title:string description:text url:string created modified

$ bin/cake bake migration CreateTags title:string created modified

$ bin/cake bake migration CreateBookmarksTags bookmark_id:integer tag_id:integer

Prática - Migrações

Chave primaria composta:

```
$table = $this->table('bookmarks_tags', ['id'=>false, 'primary_key' =>
['bookmark_id', 'tag_id']]);
```

Chaves Estrangeiras:

```
$table->addForeignKey('bookmark_id', 'bookmarks', 'id', array('delete'=>
'SET_NULL', 'update'=> 'NO_ACTION'));

$table->addForeignKey('tag_id', 'tags', 'id', array('delete'=> 'SET_NULL',
'update'=> 'NO_ACTION'));

$table->addForeignKey('user_id', 'users', 'id', array('delete'=> 'SET_NULL',
'update'=> 'NO_ACTION'));
```

Prática - Criar CRUDs com bake

Executar migrações

$ bin/cake migrations migrate

Criar CRUDs

$ bin/cake bake all users --theme NtiLayout
$ bin/cake bake all bookmarks --theme NtiLayout
$ bin/cake bake all tags --theme NtiLayout

Links

Todo o conteúdo extraído de cakephp.org

- http://book.cakephp.org/3.0/pt/intro.html

Fim