Curso de CakePHP 3

Índice

1 - Introdução

O CakePHP é hoje um dos frameworks php mais populares do mundo. Com bons e práticos recursos, que facilitam a criação de aplicativos.

2 - Requisitos para instalar o CakePHP

- Web server como Apache 2 com mod_rewrite
- PHP 5.6+
- extensões mbstring e intl
- Um dos 4 SGBDs suportados

Roteiro Completo: https://book.cakephp.org/3.0/en/installation.html

3 - Composer e Instalação

O método suportado de instalação pelo CakePHP 3 é usando o composer e não baixando o arquivo e descompactando.

Instalação e uso do Composer no Linux

O composer é um gerenciador de dependências que instala o CakePHP e também instala as dependências que forem necessárias.

Instalar Globalmente o Composer no Linux
Após ter instalado o servidor e ter configurado o ambiente é hora de instalar o composer.

sudo apt-get install composer

sudo su
curl -sS https://getcomposer.org/installer | php

Colocar no path do Linux
mv composer.phar /usr/local/bin/composer

Criar App
composer create-project --prefer-dist cakephp/app nome_ap

Criar script no Linux para automatizar a criação de aplicativos:
sudo nano /usr/local/bin/comp

Cole a linha:
composer create-project --prefer-dist cakephp/app $1

Execute
sudo chmod +x /usr/local/bin/comp

Usando
comp nomeapp

Desabilitando o xdebug (como recomendado) no Linux:
sudo php5dismod -s cli xdebug

Desabilitar xdebug no php7:
sudo nano /etc/php/7.0/cli/conf.d/20-xdebug.ini

Apenas comente a linha

Usando o Composer:

composer install
Este comando fará o composer ler as configurações setadas no arquivo json e instalar todas as bibliotecas/pacotes necessários para a sua aplicação e também estas mesmas bibliotecas que possuírem dependências terão as mesmas resolvidas.

composer update
composer init
composer validate
composer self-update

Detalhes
https://getcomposer.org/doc/03-cli.md
http://tableless.com.br/composer-para-iniciantes/

Removendo pacote instalado

Acesse a pasta do projeto
Edite o composer.json
Remova a linha do pacote no require ou require-dev
Execute
composer update

Removerá o pacote e suas dependências

Instalando o CakePHP no Windows via Composer

Após ter instalado o servidor (Xampp ou outro) e tudo pronto vamos instalar e usar o Composer.

Download
https://getcomposer.org/Composer-Setup.exe
https://git-scm.com/download/win

Instale o composer e o git

Criar um arquivo c:\xampp\htdocs\comp.bat, contendo a linha:

Habilitar a extensão intl no php.ini

editar c:\xampp\php\php.ini
Descomentar a linha abaixo:
extension=php_intl.dll

Removendo o ; do início
Restartar o Apache

Adicionar o php ao path
Painel de Controle - Sistema e Segurança - Sistema
Configurações Avançadas do Sistema - Variáveis de Ambiente

Clicar em Path e depois em Editar
Clique na caixa de texto e vá para o final da linha
Adicione a linha abaixo:
;C:\xampp\php

Clique em OK - Ok - OK

Usando o Composer:

composer install
Este comando fará o composer ler as configurações setadas no arquivo json e instalar todas as bibliotecas/pacotes necessários para a sua aplicação e também estas mesmas bibliotecas que possuírem dependências terão as mesmas resolvidas.

composer update
composer init
composer validate
composer self-update

Detalhes
https://getcomposer.org/doc/03-cli.md
http://tableless.com.br/composer-para-iniciantes/

4 - Convenções

A equipe do CakePHP é grande admiradora de convenção sobre configuração. Seguindo convenções você recebe funcionalidades gratuitamente e libera a si mesmo do pesadelo de manter arquivos de configuração.

Aviso Importante: Portanto, antes de começar a trabalhar com CakePHP é muito importante conhecer suas convenções para tirar delas enormes vantagens. Caso não as use o CakePHP não será de muita ajuda.

Controllers

Nomes de classes tipo Controller devem estar no plural, ser CamelCase e terminarem com o sufixo Controller.

Exemplos: ClientesController, PeopleController e LatestArticlesController

Actions - Métodos public nos controllers são chamados de actions e se comunicam com views com mesmo nome que eles e extensão .ctp.

Um exemplo: o action Controller/ClientesController/index() é mapeado automaticamente para a view src/Template/Clientes/index.ctp.

Métodos protected ou private não podem ser acessados via Routing.

Considerações sobre URL para nomes de controllers

O ClientesController que está no arquivo ClientesController.php é chamado pelo navegador com:

http://localhost/aplicacao/clientes

Nomes de controllers com palavras compostas podem ficar assim:
• /redApples
• /RedApples
• /Red_apples
• /red_apples

Todos resolverão com o controller RedApples.

/red-apples/go-pick resolve para o action RedApplesController::goPick()

Criar links:

```
$this->Html->link('link-title', [
        'prefix' => 'MyPrefix' // CamelCased
        'plugin' => 'MyPlugin', // CamelCased
        'controller' => 'ControllerName', // CamelCased
        'action' => 'actionName' // camelCased
]
```

Namespaces

Todas as classes do core do CakePHP agora (3.x) usam namespaces e seguem as especificações de autoload (auto-carregamento) do **PSR-4**.

Por exemplo
src/Cache/Cache.php

tem o namespace
Cake\Cache\Cache

Constantes globais e métodos de helpers como __() e debug() não usam namespaces por questões de conveniência.

Nomes de arquivos e Classes:

Classe KissesAndHugsController está no arquivo KissesAnd-HugsController.php
Classe ClientesController está no arquivo ClientesController.php.

Alguns exemplos de classes e seus arquivos:

- O Controller class KissesAndHugsController deve estar no arquivo com nome KissesAndHugsController.php
- O Component class MyHandyComponent deve estar no arquivo com nome MyHandyComponent.php
- A Table class OptionValuesTable deve estar no arquivo com nome OptionValuesTable.php
- A Entity class OptionValue deve estar no arquivo com nome OptionValue.php
- O Behavior class EspeciallyFunkableBehavior deve estar no arquivo com nome EspeciallyFunkableBehavior.php
- A View class SuperSimpleView would deve estar no arquivo com nome SuperSimpleView.php
- O Helper class BestEverHelper deve estar no arquivo com nome BestEverHelper.php

Model e Bancos de Dados

Nomes de classes Table - são no plural e CamelCase

Chave Primária
Toda tabela, obrigatoriamente deve ter uma chave primária e o nome da chave deve ser id.

O Cake tem um recurso online importante para mostrar o plural de nomes:
http://inflector.cakephp.org/

Nomes válidos: Clientes, Pelople, BigPeople e ReallyBigPeople

Nomes de tabelas são em minúsculas, no plural e palavras compostas separadas por sublinhado. Nomes de tabelas para os acima:

clientes, people, big_people e really_big_people

A convenção é para usar tabelas e campos com nomes na língua inglesa.
Se por alguma razão precisar usar nomes de tabelas em outro idioma, então você deve usar a classe utility:

Cake\Utility\Inflector

Nomes de campos compostos são separados por sublinhado: first_name.

Nomes de campos são em minúsculas e quando compostos por palavras compostas são separados por sublinhado.

Se usarmos os **campos username e password** (com estes nomes) na tabela users, o Cake deve estar apto para auto-configurar muitas coisas para nós, quando implementando o user login.

Obs.: quando usar autenticação use o tamanho do campo password com varchar(255).
Também ajuda adicionar um campo chamado role na tabela users.

Relacionamentos

Chave estrangeira nos relacionamentos hasMany, belongsTo ou hasOne são reconhecidas por default com o nome (singular) da tabela relacionada seguida de "_id".

Exemplos: groups e users. Em users o campo group_id para relacionar.
Relacionamento entre articles e users. Em articles adicionar o campo user_id.

Relacionamentos

Um - Muitos - hasMany
Se relacionamos Articles com User, inserimos um campo user_id em articles.
No model UsersTable
Um Users pode conter muitos (hasMany) Articles
Muitos Articles belongsTo Users
Ver documentação para detalhes.

Relacionamento Muitos para Muitos

Exemplo: Para relacionamento muitos para muitos de platillos com cocineros, criaremos a tabela

```
create table cocineros_platillos(
        id int unsigned auto_increment primary key,
        cocinero_id int(11) not null,
        platillo_id int(11) not null
);
```

Tipos de Relacionamentos

one to one	hasOne	A user has one profile.
one to many	hasMany	A user can have multiple articles.
many to one	belongsTo	Many articles belong to a user.
many to many	belongsToMany	Tags belong to many articles.

Para uma classe Bakers teremos uma chave estrangeira assim: baker_id.
Para uma tabela como category_types a chave estrangeira será category_type_id.

Nomes de campos especiais, que levam o Cake a tomar iniciativas importantes para nós:
title
created
modified

Convenções para as Views

As views tem nomes de arquivos em minúsculas com extensão .ctp.
O método getReady() do conroller PeopleController está associado ao template/view src/Template/People/get_ready.ctp.

Exemplo geral:

• Database table: “people”
• Table class: “PeopleTable”, found at src/Model/Table/PeopleTable.php
• Entity class: “Person”, found at src/Model/Entity/Person.php
• Controller class: “PeopleController”, found at src/Controller/PeopleController.php
• View template, found at src/Template/People/index.ctp

Usando essas convenções o CakePHP sabe que uma requisição para http://localhost/aplicativo/people mapeia para uma chamada da função index() do PeopleController, onde o model Person está automaticamente disponível (e automaticamente vinculado à tabela people do banco) e renderizada para um arquivo src/Template/People/index.ctp. Nenhum desses relacionamentos precisa ser configurado por qualquer meio mas apenas pela criação de arquivos e classes que precisamos criar sempre seguindo as convenções.

Arquivos da Aplicação

Todos os arquivos da aplicação que criamos ficam na pasta src.

Criptografia
Por padrão o CakePHP 3.x usa a criptografia bcrypt para proteger as senhas.

Documentação oficial
http://book.cakephp.org/3.0/pt/intro/conventions.html

Controllers
Nomes de classes tipo Controller devem estar no plural, CamelCase e terminam com Controller.

Exemplos: ClientesController, PeopleController e LatestArticlesController

Métodos public nos controllers são chamados de actions e são mostrados no browser através de views.

Um exemplo: a view src/Template/Clientes/index.ctp é mapeado para o action Clientes/Controller/index()

Métodos protected ou private não podem ser acessados via Routing.

Considerações sobre URL para nomes de controllers

ClientesController que está no arquivo ClientesController.php é chamado no navegador com: http://localhost/aplicacao/clientes

Nomes de controllers com palavras compostas podem ficar assim:
• /redApples
• /RedApples
• /Red_apples
• /red_apples

Todos resolverão com o controller RedApples.

/red-apples/go-pick resolve para o action RedApplesController::goPick()

Criar links:

```
$this->Html->link('link-title', [
        'prefix' => 'MyPrefix' // CamelCased
        'plugin' => 'MyPlugin', // CamelCased
        'controller' => 'ControllerName', // CamelCased
        'action' => 'actionName' // camelBacked
]
```

Namespaces

Todas as classes do core do CakePHP agora (3.x) usam namespaces e seguem as especificações de autoload (auto-carregamento) do PSR-4. Por exemplo src/Cache/Cache.php tem o namespace Cake\Cache\Cache. Constantes globais e métodos de helpers como __() e debug() não usam namespaces por questões de conveniência.

Nomes de arquivos e Classes:

Classe KissesAndHugsController está no arquivo KissesAnd-HugsController.php

Convenções para Model e Bancos de Dados

Nomes de classes Table são no plural e CamelCase

Nomes válidos: Clientes, Pelople, BigPeople e ReallyBigPeople

Nomes de tabelas são no plural e palavras compostas separadas por sublinhado.
Nomes de tabelas para os acima: clientes, people, big_people e really_big_people

Nomes de campos são em minúsculas e quando compostos por palavras compostas são separados por sublinhado.

Tabela users levam vantagem se tiverem campos com nomes: username e password
Exemplos: primeiro_nome

Obs.: quando usar authenticação use o tamanho do campo password com varchar(255).
Também ajuda adicionar um campo chamado role na tabela users.

Relacionamentos
Chave estrangeira nos relacionamentos hasMany, belongsTo ou hasOne são reconhecidas por default com o nome (singular) da tabela relacionada seguida de "_id".

Relacionamentos

Um - Muitos (hasMany)
Se relacionamos Articles com User, inserimos um campo user_id em articles.
No model UsersTable
Um Users pode conter muitos (hasMany) Articles
Muitos Articles belongsTo Users
Ver documentação...

Relacionamento Muitos para Muitos
Exemplo: Para relacionamento muitos para muitos de platillos com cocineros, criaremos a tabela

```
create table cocineros_platillos(
        id int unsigned auto_increment primary key,
        cocinero_id int(11) not null,
        platillo_id int(11) not null
);
```

Para uma classe Bakers teremos uma chave estrangeira assim: baker_id.
Para uma tabela como category_types a chave estrangeira será category_type_id.

Nomes de campos especiais:
title
created
modified

Convenções para as Views

O método getReady() do conroller PeopleController está associado ao template/view src/Template/People/get_ready.ctp.

Exemplo geral:

• Database table: “people”
• Table class: “PeopleTable”, found at src/Model/Table/PeopleTable.php
• Entity class: “Person”, found at src/Model/Entity/Person.php
• Controller class: “PeopleController”, found at src/Controller/PeopleController.php
• View template, found at src/Template/People/index.ctp

Usando essas convenções o CakePHP sabe que uma requisição para http://localhost/aplicativo/people mapeia para uma chamada da função index() do PeopleController, onde o model Person está automaticamente disponível (e automaticamente vinculado à tabela people do banco) e renderizada para um arquivo. Nenhum desses relacionamentos precisa ser configurado por qualquer meio mas apenas pela criação de arquivos e classes que precisamos criar sempre.

Arquivos da Aplicação
Todos os arquivos da aplicação que criamos ficam na pasta src.

Criptografia
Por padrão o CakePHP 3.x usa a criptografia bcrypt para proteger as senhas.

5 - Configurações

Mesmo que as convenções nos livrem de estar efetuando muitas configurações no CakePHP, ainda assim precisamos configurar algumas coisas como bancos de dados, rotas e bootstrap (carga de plugins).

Todos os arquivos de configuração ficam na pasta **config**.

Os principais arquivos de configuração são:
config/app.php
config/routes.php
config/bootstrap.php

Nenhum dos 3 é obrigatório. Caso o aplicativo não use bancos de dados nem tenha plugins de terceiros não somos obrigados a alterar nenhum deles.

Quando o aplicativo usa bancos de dados precisamos alterar o app.php.

Quando desejamos personalizar as rotas também precisamos alterar o routes.php.

Quando o aplicativo usa plugins de terceiros o bootstrap.php.

app.php

Geralmente utilizado somente para configurar o banco de dados, mas existem muitas outras configurações nele. Vejamos algumas:

debug

encoding

dir – diretório da aplicação. Por default o src

webroot -

diretório base de imagens, css e javascript

salt – deve ser alterado. String usada no hash de senhas. Usando o composer ele já altera o salt

Error -

Email configuração

Datasources

Log

Session

Configurando o Banco de Dados

Para configurar o banco devemos indicar apenas o **username, password** e **database** na configuração **default**.

```
    'Datasources' => [
        'default' => [
            'className' => 'Cake\Database\Connection',
            'driver' => 'Cake\Database\Driver\Mysql',
            'persistent' => false,
            'host' => 'localhost',
            /**
             * CakePHP will use the default DB port based on the driver selected
             * MySQL on MAMP uses port 8889, MAMP users will want to uncomment
             * the following line and set the port accordingly
             */
            //'port' => 'non_standard_port_number',
            'username' => 'root',
            'password' => '',
            'database' => 'blog',
            'encoding' => 'utf8',
            'timezone' => 'UTC',
            'flags' => [],
            'cacheMetadata' => true,
            'log' => false,
```

Uma configuração importante aqui também é o driver do SGBD. O CakePHP 3 trabalha atualmente com os SGBDs: Mysql, Postgres, Sqlite e Sqlserver. Caso queira criar o aplicativo acessando um banco do PostgreSQL, então a linha driver assim:

```
'driver' => 'Cake\Database\Driver\Postgres',
```

Claro que também precisa alterar username, password e database.

routes.php

Neste arquivo você configura as rotas para os actions e controllers. Rotas são um importante mecanismo que permite que você conecte para diferentes URLs apenas escolhendo os controllers e seus actions.

Para uma configuração simples, suponha que você criou um aplicativo chamado cliente, que ficou na pasta cliente e criou os controllers e seus actions padrões: index(), view(), etc. Então vamos configurar o routes.php.

Logo que criou o controller ClientesController e seus actions e acessar o aplicativo assim:

http://localhost/cliente

Quem responderá é o action display do controller PagesController, que é um controller e action padrão, que já vem com o Cake e estão pré-configurados no routes.php como a rota padrão. Veja:

$routes->connect('/', ['controller' => 'Pages', 'action' => 'display', 'home']);

Para que seu controller Clientes seja mostrado juntamente com seus actions, precisamos dizer isso ao CakePHP, assim. Adicione esta linha **antes** da linha anterior:

$routes->connect('/pages/*', ['controller' => 'Clientes', 'action' => 'index']);

Chame seu aplicativo novamente com:

http://localhost/cliente

Agora sim, ele mostra o resultado do action index() do controller Clientes.

Existem muitas configurações para as rotas. Para mais detalhes veja:
http://book.cakephp.org/3.0/en/development/routing.html

bootstrap.php

Veja as mensagens iniciais que ele pode disparar, acusando que este é um dos arquivos iniciais no boot do CakePHP 3.

Observe que ele já vem com várias configurações prontas. Já vem com alguns plugins carregados, como o Migrations e o DebugKit.

Carregando Plugins

Caso queira carregar um plugin de terceiro precisa adicionar uma linha logo abaixo desta:
Plugin::load('Migrations');

Supondo que precise carregar um plugin chamado Acl:

Plugin::load('Acl');

Checando para ver se uma configuração foi definida

static `Cake\Core\Configure::check`(*$key*)

```
$exists = Configure::check('Company.name');
```

Deletando Configuração

static `Cake\Core\Configure::delete`(*$key*)

```
Configure::delete('Company.name');
```

6 – Extrutura MVC

O CakePHP segue o padrão de projeto MVC.

6.1 – Model

A camada model é responsável por lidar com os dados.

A camada model contém vários elementos, **Behavior, Entity, Table e ORM**, que são importantes no Cake.

6.2 – Controller

O controller é quem controla as duas outras camadas. Recebe do usuário, passa para o model e devolve para a view. Contém um exemento importante e complementar, que é o **component**.

6.3 – View

É quem entrega para o usuário as informações num formato adequado. Contém os elementos Elements, Layouts e Helpers.

7 - Plugins

Os plugins são como pequenas aplicações dentro da aplicação mestre.

Veja estes dois plugins muito úteis:

7.1 – CakeControl

Download - https://github.com/ribafs/cake-control

Tutorial – http://ribafs.org/cakephp/cake-control.pdf

7.2 – CakeControlBr

Download - https://github.com/ribafs/cake-control-br

Tutorial – http://ribafs.org/cakephp/cake-control-br.pdf

8 – Dicas

Link

http://ribafs.org/cakephp/dicas