Gerando Código com Bake no CakePHP 3

http://book.cakephp.org/3.0/en/bake.html

A console bake do CakePHP é um outro esforço para que você esteja rodando o CakePHP rapidamente. A console do bake pode criar qualquer um dos códigos básicos do CakePHP: models, behaviors, views, helpers, controllers, components, test cases, fixtures e plugins. E não estamos falando apenas de esqueletos de classes, o bake pode criar uma aplicação inteiramente funcional em poucos minutos.

Pré-requisitos

A console do bake requer:
- PHP CLI
- extensão intl
- Pelo menos um SGBD instalado e configurado na aplicação atual

Instalação

O bake já vem instalado por padrão, mas se precisar instalar use o comando:

```
composer require --dev cakephp/bake:~1.0

ou

php composer.phar require --dev cakephp/bake:dev-master
```

Comandos disponíveis do bake. Execute:

bin\cake bake

- all
- behavior
- cell
- component
- controller
- fixture
- form
- helper
- model
- plugin
- shell
- template
- test

Usando `cake bake [name]` podemos invocar uma tarefa especificada pelo name.

Exemplo:
bin\cake bake all [name]

Ajuda
bin/cake bake controller --help

O bake é uma ferramenta espetacular. Veja que com apenas um comando geramos o aplicativo.

Com ele podemos fazer muto:

Gerar um controller:
bin/cake bake controller Clientes

Gerar controller sem actions:
bin\cake bake controller --no-actions Clientes

Gerar Componente
bin/cake bake component Calculo

Gerar um Template
bin/cake bake template Clientes

Gerar Helper
bin/cake bake helper MeuForm

Gerar Cell
bin/cake bake cell MinhaCelula

Gerar Form
bin/cake bake form MeuForm

Gerar um Model
bin/cake bake model Clientes

Gerar Behavior
bin/cake bake behavior MeuBehavior

Observe as mensagens para saber em que diretório está sendo criado.

Ajuda
bin/cake bake controller --help

Apenas controller, model e template
bin/cake bake controller clientes
bin/cake bake model clientes
bin/cake bake template clientes

Plugins

Gerando código usando plugin
bin/cake bake all --plugin CakePtbr articles

Criar plugin ContactManager com tabela contacts

cd /backup/www/cake/blog
bin/cake bake plugin ContactManager

bin/cake bake controller --plugin ContactManager Contacts
bin/cake bake model --plugin ContactManager Contacts
bin/cake bake template --plugin ContactManager Contacts

Atualizando
composer dumpautoload

Componente
bin/cake bake component Calculo

Migrate
bin/cake migrations
bin/cake migrations migrate
bin/cake migrations status --plugin PluginName

Existem alguns recursos que não são gerados ou auxiliados pelo bake, como o element.

Gerar o código completo do CRUD para uma tabela:
cd /var/www/cakeapp/app/Console
./cake bake all

Gerar o código para os actions "admin_" em todos os controllers:
./cake bake controller all --admin

cake bake controller ControllerName --plugin PluginNameInCamelCase

Gerar todas as views iniciadas com admin_:
./cake bake view all --admin

Outras formas:
./cake bake model group
./cake bake view group

./cake bake model user
./cake bake view user

./cake bake model
./cake bake controller
./cake bake view

Rodando Interativamente
./cake bake

Outras formas
./cake bake db_config
./cake bake project
./cake bake fixture
./cake bake test
./cake bake plugin plugin_name

Gerar todos os models:
./cake bake model all

Entendendo a geração de código no Cake

Quando geramos um CRUD através do bake, especialmente no que diz respeito ao model gerado, ele cria validação para os campos, mas somente para aqueles que têm alguma restrição (pelo menos NOT NULL) e também armazena no model o respectivo código para os relacionamentos (quando seguimos as convenções do Cake para relacionamentos).

Um exemplo: o model Cliente.php (cuja tabela é relacionada com a tabela pedidos).

Veja a tabela:

```
CREATE TABLE IF NOT EXISTS `clientes` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `nome` char(45) NOT NULL,
  `email` varchar(50) DEFAULT NULL,
  `data_nasc` date DEFAULT NULL,
  `cpf` char(11) NOT NULL,
  PRIMARY KEY (`id`)
);
```

Observe que os campos email e data_nasc tem NULL como default, ou seja não exigem qualquer informação. Aceitam qualquer valor para ser salvo, inclusive nenhum valor.

Em pedidos existe um relacionamento com clientes, assim como com funcionários e produtos:

```
CREATE TABLE IF NOT EXISTS `pedidos` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `cliente_id` int(11) NOT NULL,
  `funcionario_id` int(11) NOT NULL,
  `produto_id` int(11) NOT NULL,
  `data` date NOT NULL,
  `quantidade` int(11) NOT NULL,
  `preco` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `cliente_id` (`cliente_id`),
  KEY `funcionario_id` (`funcionario_id`),
  KEY `produto_id` (`produto_id`)
);
```

Agora veja o código gerado pelo Bake para o model Cliente:

```
class Cliente extends AppModel {

	/**
	 * Validation rules
	 *
```

```
	 * @var array
	 */
		public $validate = array(
			'nome' => array(
				'notempty' => array(
					'rule' => array('notempty'),
					//'message' => 'Your custom message here',
					//'allowEmpty' => false,
					//'required' => false,
					//'last' => false, // Stop validation after this rule
					//'on' => 'create', // Limit validation to 'create' or 'update'
operations
				),
			),
			'cpf' => array(
				'notempty' => array(
					'rule' => array('notempty'),
					//'message' => 'Your custom message here',
					//'allowEmpty' => false,
					//'required' => false,
					//'last' => false, // Stop validation after this rule
					//'on' => 'create', // Limit validation to 'create' or 'update'
operations
				),
			),
		);

	/**
	 * hasMany associations
	 *
	 * @var array
	 */
		public $hasMany = array(
			'Pedido' => array(
				'className' => 'Pedido',
				'foreignKey' => 'cliente_id',
				'dependent' => false,
				'conditions' => '',
				'fields' => '',
				'order' => '',
				'limit' => '',
				'offset' => '',
				'exclusive' => '',
				'finderQuery' => '',
				'counterQuery' => ''
			)
		);

}
```

Veja, que somente tem validação os campos nome e cpf. O código da associação/relacionamento entre clientes e pedidos (hasMany - um cliente tem muitos pedidos).

Já o relacionamento entre pedidos e clientes (veja em Pedido.php) é do tipo belogsTo (pedido pertence a cliente). Veja:

```
public $belongsTo = array(
        'Cliente' => array(
                'className' => 'Cliente',
                'foreignKey' => 'cliente_id',
                'conditions' => '',
                'fields' => '',
                'order' => ''
        ),
```

Em termos de segurança dos dados no banco, o relacionamento é muito importante. Num banco de dados como o banco acima seria lamentável ter pedidos cadastrados no banco de clientes que já foram excluídos, por exemplo. O relacionamento é muito importante e deve ser implementado inclusive no banco e não somente no Cake.

Geração de todo o código: controllers, models e templates e testes
bin/cake bake all clientes
bin/cake bake all users

Apenas controller, model e template
bin/cake bake controller clientes
bin/cake bake model clientes
bin/cake bake template clientes

Plugins

Criar plugin ContactManager com tabela contacts

cd /backup/www/cake/blog
bin/cake bake plugin ContactManager
bin/cake bake controller --plugin ContactManager Contacts
bin/cake bake model --plugin ContactManager Contacts
bin/cake bake template --plugin ContactManager Contacts

Atualizando
composer dumpautoload

Componente
bin\cake bake component Calculo

Migrate
bin/cake migrations
bin/cake migrations migrate
bin/cake migrations status --plugin PluginName

Criar plugin usando o bake:
O banco precisa ter uma tabela chamada Contacts

cd /backup/www/cake/blog
bin/cake bake plugin ContactManager
bin/cake bake controller --plugin ContactManager Contacts
bin/cake bake model --plugin ContactManager Contacts
bin/cake bake template --plugin ContactManager Contacts

Gerando código para Área Administrativa

Criar aplicativo para cadastro de clientes com área administrativa acessada somente por alguns usuários.

Criar o controller para o Admin:
bin\cake bake controller --no-actions --prefix Admin Home

Criar o controller para o Site
bin\cake bake controller --no-actions --prefix Site Home

Criar um CRUD para o Admin:

Configure o banco de dados no config/app.php

Criar código para gerar classe
bin\cake bake migration CreateUsers nome:string email:string created modified

Criar tabela com código acima
bin\cake migrations migrate

Gerar o Scaffold para Usuários
bin\cake bake all Users --prefix Admin

Criar outros usos para o Bake:

Configurar o banco de dados

Configurar a carga de plugin

Criar:
- model
- controller

```
bin/cake bake controller comments --force
```

- controller sem actions
- action
- view/template
- helper
- element
- component
- criar validações interaticamente
- mudar/criar displayField
- Criar admin

Usar migrações

Criar migrações

bin\cake migrations create CreateUsersTable
bin\cake migrations create CreateGroupsTable

Criar tabelas
bin\cake migrations migrate

Usar Composer

Instalar CakePHP
composer create-project –prefer-dist cakephp/app nopeapp

chmod 550 bin\cake

Adicionando outros actions para controllers no Bake:

Yes, you can bake:

```
cake bake view users custom_func
```

For that you need to create a template in the following directory:

```
YOUR_PROJECT\lib\Cake\Console\Templates\default\views
```

```
"create a custom_func.ctp file"
```

and also create a function in (core controller file):

```
YOUR_PROJECT\lib\Cake\Console\Templates\default\actions
```

```
"add a new function custom_func"
```

Or if you don't want that function to be universal, directly write into the users controller.

Garantir que não use cache:

You will also need to regenerate your model classes and clear out the cache:

bin/cake bake model comments –force
bin/cake bake model issues –force
bin/cake orm_cache clear

2.0

Modify default HTML produced by “baked” templates

If you wish to modify the default HTML output produced by the “bake” command, follow these simple steps:

For baking custom views

1. Go into: lib/Cake/Console/Templates/default/views
2. Notice the 4 files there
3. Copy them to your: app/Console/Templates/[themename]/views
4. Make changes to the HTML output to control the way “bake” builds your views

The `[themename]` path segment should be the name of the bake theme that you are creating. Bake theme names need to be unique, so don’t use ‘default’.

For baking custom projects

1. Go into: lib/Cake/Console/Templates/skel

2. Notice the base application files there
3. Copy them to your: app/Console/Templates/skel
4. Make changes to the HTML output to control the way “bake” builds your views
5. Pass the skeleton path parameter to the project task

```
cake bake project --skel Console/Templates/skel
```

Note

- You must run the specific project task `cake bake project` so that the path parameter can be passed.
- The template path is relative to the current path of the Command Line Interface.
- Since the full path to the skeleton needs to be manually entered, you can specify any directory holding your template build you want, including using multiple templates. (Unless CakePHP starts supporting overriding the skel folder like it does for views)

Cache do ORM

Reconstruir o cache dos metadados de todas as tabelas da conexão default
bin/cake orm_cache build --connection default

Para reconstruir o cache dos emtadados apenas da tabela artigos
bin/cake orm_cache build --connection default artigos

Remover o cache de todos os metadados:
bin/cake orm_cache clear

Remover o cache de todos os metadados da tabela aratigos
bin/cake orm_cache clear artigos

Plugins e Outros com o Bake

bin\cake bake plugin Admin

Com isso ele cria a pasta plugins\Admin contendo config, src, webroot e composer.json.

Em src ele cria a pasta Controller e dentro o arquivo AppController.php com uma classe AppController vazia.

Carregando Plugin no bootstrap.php com o bake

bin\cake plugin load Admin

Agora podemos continuar e criar controllers, models, etc, como fazemos com aplicativos do CakePHP, assim:

Criando um controller

bin\cake bake controller --plugin Admin Groups
bin\cake bake controller --plugin Admin Users
bin\cake bake controller --plugin Admin Privileges

Alerta: cuidado ao copiar e colar código de processadores de texto. O exemplo acima acaba de me gerar um problema, pois os dois hifens antes de plugin foram para a console como apenas um traço e deu erro.

Criando Componente para o Plugin

bin\cake bake component --plugin Admin AccessControl

Com o comando acima ele cria um controller completo, com todos os actions default. Verifique.

Componentes, Helpers e Behaviors

Um plugin pode conter componentes, helper e behaviors criados da mesma forma que para uma aplicação normal.

Criando Componente com bake
Antes criar o Controler
bin\cake bake component AccessControl

Criando Helper com bake
Antes criar o Template
bin\cake bake helper MeuHelper

Criando Behavior com o bake
Antes criar o Model
bin\cake bake behavior MeuBehavior

Criar o arquivo de migração

```
bin\cake bake migration CreateContacts title:string body:text created modified
```

Com isso ele cria um arquivo em config/Migrations contendo uma classe e uma função com a definição da tabela a ser criada sugerida por nós.

Para criar a tabela execute:

```
bin\cake migrations migrate
```

Pode verificar agora a tabela contacts criada em nosso banco atual.

Criando Model e Template para o Plugin

bin\cake bake model --plugin Admin Groups
bin\cake bake model --plugin Admin Users

bin\cake bake model --plugin Admin Privileges

Criando Templates para o Plugin

bin\cake bake template --plugin Admin Groups
bin\cake bake template --plugin Admin Users
bin\cake bake template --plugin Admin Privileges

Criar somente a view login.ctp para o Plugin

bin\cake bake template --plugin Admin Users login

Agora nosso plugin tem uma estrutura básica completa, inclusive com bancos e podemos inserir registros para ver como se comporta.

Chamando nosso plugin pelo navegador

http://localhost/aplicativo/admin/groups

Ele irá mostrar a index.ctp do controller Contacts do nosso plugin.

É como se nosso plugin Admin fosse outro aplicativo rodando sobre o aplicativo atual e estamos chamando seu controller contacts.

Procurei esclarecer e melhorar um pouco o tutorial oficial do site e está aí, podendo servir para a criação de um plugin bem útil no CakePHP 3.

Também contei com a ajuda do plugin migrations para criar a tabela no tutorial de criação de Blog no site do Cake.