

# Criando Aplicativos e Extensões para o CakePHP 3

(Aplicativos, Componentes, Plugins, Layouts, Elements, ...)

In small cake recipes :)



*Ribamar FS*  
*Editora Clube de Autores*

Fortaleza, 2 de julho de 2019



### **Ficha Catalográfica**

---

S725c

Criação de Aplicativos e Extensões para o framework CakePHP 3 / Ribamar  
Ferreira de Sousa. - Fortaleza: Clube de Autores, 2019

247p.

1. Criação de Aplicativos. 2. Framework CakePHP!. Criação de Extensões. I.  
Título.

---

### **Normalização Bibliográfica**

Lúcia Maria Piancó Chaves – DNOCS-CGE/MD  
Margarida Lídia de Abreu Vieira – DNOCS-CGE/MD



*Algo só é impossível até que alguém duvide e resolva provar o contrário.*

Albert Einstein

## **Apresentação**

Como trabalho com programação também e é um das minhas principais diversões, meus livros antes de tudo são criados para mim, para servirem como guias rápidos para o meu dia-a-dia. Ao perceber que podem ajudar outros colegas, então dou uma boa organizada e publico. Foi o que aconteceu também com este, que tem a intenção de tornar seu trabalho com o CakePHP 3 ainda mais simples.

Se observarmos existe uma pequena quantidade de livros sobre o CakePHP se compararmos com outros frameworks semelhantes. Me parece que isso se deve principalmente devido a dois fatores. Primeiro a rica documentação online, que torna de certa forma desnecessário mais documentação sobre o mesmo. E segundo o bake, que cria um CRUD básico e inteiramente funcional e muitos outros códigos para o CakePHP. Estes dois fatores tornam praticamente desnecessário a existência de livros sobre nosso framework.

Então porque eu venho publicar um livro sobre o CakePHP 3? Como uso o Cake desde a versão 2, tenho acumulado boa experiência e documentação durante estes anos, chegando inclusive a estender as funcionalidades do Bake. Como a implementação de ACL no Cake atualmente é feita por um plugin via terminal/prompt, então juntei algumas peças e criei um plugin para fazer isso através de uma interface de administração web, que é o [cake-acl-br](#), hoje substituído pelo [admin-br](#), que também conta com diversos outros bons recursos e está publicado no GitHub com licença MIT.

Reuni uma boa documentação sobre o bake.

Também aprendi a criar componentes, helpers, elements, layouts, etc, que ajudam a customizar de forma segura o código do CakePHP, evitando que alteremos o core diretamente.

Outro fator é que nem toda a documentação online está traduzida. Estes e outros me levaram a compartilhar minha experiência através deste livro. Veja através do índice e alguns capítulos iniciais se vale a pena você ler.

## **Perfeição num tempo infinito**

Quando comecei a melhorar, corrigir, otimizar este livro em vários aspectos então lembrei que para deixá-lo perfeito precisaria de conhecimento e tempo ilimitados e resolvi abreviar as otimizações e deixar como está.

## **Público Alvo**

Este livro destina-se principalmente aos programadores PHP, com conhecimento de orientação a objetos e que desejam aprender a criar aplicativos com o CakePHP 3 com produtividade e facilidade.

## **Aviso Importante**

Preciso dizer com muita tranquilidade que todos os bons recursos do CakePHP citados neste livro somente serão absorvidos por quem estiver estimulado e determinado a aprender. Caso tenha a base necessária (PHP) e a vontade de aprender, então boa diversão.

*Ribamar FS  
Fortaleza, 2 de julho de 2019*

## Agradecimentos

Gostaria de agradecer a toda a equipe que criou e que toca este excelente framework, que é o CakePHP e também à comunidade que compartilha seus conhecimentos através de tutoriais, grupos de discussão, listas (perguntando e respondendo).

O programador **Michal Tatarynowicz** deu origem ao nosso framework em abril de 2005.

Abaixo podemos encontrar os principais integrantes da equipe de desenvolvimento do CakePHP:  
<https://cakephp.org/pages/team>

Mas é bom lembrar que toda a comunidade envolvida colabora com o framework.

Todos eles, todos nós, somos responsáveis por tornar a vida profissional de programadores mais simples e agradável.

Muito agradecido a todos vocês!  
Vida longa e que continuem este trabalho!





# Índice

1 - Introdução.....	11
2 – MVC no CakePHP.....	15
3 - Convenções sobre Configurações.....	19
4 – Instalação e pré-requisitos do CakePHP 3.....	25
4.1 - Algumas configurações.....	33
5 - Gerando CRUD com Bake e muito mais.....	37
5.1 - Tutorial de uso dos templates do bake no CakePHP 3.....	43
6 – Segurança.....	53
7 – Debug e Erros.....	55
8 – Detalhes sobre Models.....	57
8.1 – Validações.....	59
8.2 - O básico sobre o ORM.....	67
8.3 - Retornando dados do banco.....	73
8.4 - Table Objects.....	75
8.5 - Query Builder.....	81
8.6 – Behaviors.....	83
9 – Detalhes sobre Views.....	87
9.1 - Element.....	93
9.2 – Layout.....	95
9.3 – Helper.....	97
9.3.1 – formHelper.....	101
9.3.2 – htmlHelper.....	111
9.3.3 – TimeHelper.....	119
9.3.4 - flashHelper e Flash Component.....	125
10 – Detalhes sobre Controllers.....	127
10.1 - Componentes.....	131
11 – Plugins.....	135
11.1 - Criação de Plugin para o CakePHP 3.....	137
11.2 - Plugin DebugKit.....	143
11.3 - Criação do plugin cake-acl-br.....	145
12 - Dicas de CakePHP 3.....	155
13 - Aplicativos de Exemplo.....	161
13.1 - CakePHP Hello World.....	161
13.2 – Blog.....	163
13.3 – Aplicativo com Bootstrap.....	165
13.4 – Aplicativo via Código.....	169
13.5 - Aplicativo Finanças Pessoais.....	173
13.6 – Aplicativo usando o Plugin admin-br.....	179
13.7 - Aplicativo com uma área restrita/administrativa.....	181
13.8 – Autenticação e Autorização Simples.....	191
14 – Trabalhando com o Código do CakePHP 3.....	197
14.1 - Repassando Informações Entre Controllers.....	203
14.2 - Conhecendo o CSS default do CakePHP 3.....	207
14.3 - Trabalhando com session no CakePHP.....	213
14.4 - Comunicação entre Model e Controller.....	217
15 - Migrations.....	219
15.1 - Migrate.....	221
15.2 - Seed - inserindo registros nas tabelas.....	225

15.3 - Exemplos de uso da Faker.....	227
15.4 - Resumindo.....	231
16 – Ambiente de Desenvolvimento.....	233
17 - Referências.....	235
18 – Algo sobre o novo CakePHP 4.....	239
19 – Conclusão.....	241
20 – Sobre min.....	243
21 – Apêndice A - Dicas sobre o Composer.....	245

# 1 - Introdução

*O único lugar onde o sucesso vem antes do trabalho é no dicionário.*

Albert Einstein

## Um pouco de História

CakePHP começou em abril de 2005 quando um programador polaco chamado Michal Tatarynowicz escreveu uma versão mínima de um desenvolvedor rápido de aplicações em PHP e o apelidou de Cake. Tempos depois ele publicou o framework sob a licença MIT e abriu a comunidade online de desenvolvedores.

(<https://www.portalgsti.com.br/cakephp/sobre/>)

Em dezembro de 2005 L. Masters and G. J. Woodworth fundaram a Cake Software Foundation para promover o desenvolvimento do CakePHP. A versão 1.0 saiu em maio de 2006.

Um dos projetos que inspiraram o Cake foi o Ruby on Rails, usando muitos dos seus conceitos mas em PHP.

(<https://en.wikipedia.org/wiki/CakePHP>)

O framework CakePHP foi concebido para tornar as tarefas de desenvolvimento web mais simples e fáceis. Por fornecer um rico kit de ferramentas para você poder começar, o CakePHP funciona melhor em conjunto.

O CakePHP tem uma ótima documentação online em <https://book.cakephp.org/3.0/pt/index.html> e para download nos formatos [pdf](#) e [epub](#). O ponto forte, ao meu ver, são os vários exemplos de aplicativos que fazem parte da documentação.

## Atualmente mais de 80% dos sites da web usam PHP.

Me parece que isso é devido principalmente à facilidade de aprender a linguagem e logo já começar a criar algo útil comparando com outras linguagens. Somando a isso atualmente também temos softwares muito bons que foram criados com PHP: WordPress (sozinho é responsável por 25% dos sites da web), o Joomla, o Drupal, temos a Wikipedia, rede social como Facebook, frameworks como Laravel, Zend, CakePHP, Symfony, Slim, etc, que nos trazem código de qualidade para limpar o nome da linguagem e mostrar que com ela não se cria apenas "softwares do sobrinho".

<https://www.php.net/usage.php>

<https://insights.stackoverflow.com/survey/2019>

O framework CakePHP tem uma baixa curva de aprendizado além de excelente documentação online e uma ferramenta que automatiza a criação de aplicativos básicos, que é o bake, que nos entrega um código de qualidade. Além de seus excelentes tutoriais de criação de aplicativos de exemplo: blog, bookmarks e cms. Sem contar que boa parte da documentação já foi traduzida para o português do Brasil, como é o caso da documentação dos dois primeiros aplicativos de exemplo.

A documentação do Cake é tão forte que geralmente ocupa os primeiros retornos das pesquisas sobre algo relacionado ao Cake.

A escolha de um framework em PHP não é algo simples. Não devemos partir apenas do depoimento dos colegas que utilizam algum deles. É importante experimentar alguns deles para escolher de forma mais adequada. Dependendo da necessidade que tivermos talvez devamos escolher não somente um mas dois ou três. Um para cada necessidade.

Abaixo uma lista de 11 dos mais populares em 2019

<https://coderseye.com/best-php-frameworks-for-web-developers/>

O meu preferido é o CakePHP, mesmo não sendo o mais popular atualmente. É importante que você se informe e, principalmente, experimente na prática alguns dos mais populares para realizar uma escolha mais adequada.

Frameworks em geral ajudam a tornar mais rápido o desenvolvimento, nos entregando um código bem organizado, manutenível, reutilizável, ferramentas para manipular a segurança, boas práticas, bons padrões de projeto e muito mais.

Caso sigamos as **convenções do CakePHP**, não teremos que configurar quase nada e criaremos aplicativos com muita facilidade.

A instalação usando o composer é muito simples e rápida. Com apenas uma linha de comando instalamos o core do Cake para criar um aplicativo:

```
composer create-project --prefer-dist cakephp/app blog
```

**Licença MIT** - que é uma das licenças mais liberais entre as do software livre. Permite que o software seja reutilizado livremente tanto para uso pessoal/free quanto para uso comercial.

**Validações internas** - quando geramos o código com bake ele já insere código de validação no model Table. Este código já é um bom início para o aprendizado das validações no CakePHP.

As validações tem como objetivo fazer com que as aplicações operem de forma limpa, correta e com dados úteis. Checam se os dados são aceitáveis, se estão em certa faixa, no formato correto e bem mais.

## **Arquitetura MVC**

Todos os frameworks PHP usam este padrão de projeto. Ele irá ajudar você a criar melhor código.

## **URL limpa**

O Cake ajuda você a criar simples e limpas URLs e praticamente dá a você inteira liberdade de manipular a URL da sua aplicação. Por default a URL das aplicações do Cake tem a URL assim:  
/controller/method/parameter

Exemplo: /posts/edit/1

Será mapeado para o método edit do PostsController e passa 1 como parâmetro.

## Cache Flexível

O Cake tem 6 diferentes cache engines embutidas, sendo FileCache, Memcached, Redis e outras. Podemos mudar o cache engine a qualquer momento. Caso queira mudar basta setar ao final do config/app.php e podemos criar nosso próprio cache engine.

## Sistema de Testes Unitários integrado

O Cake usa atualmente o PHPUnit para seus testes unitários.

## Equipe de Desenvolvimento

O Cake tem uma ativa, amigável e colaborativa equipe de desenvolvimento

<https://cakephp.org/pages/team>

## Comunidade

Grande e ativa comunidade:

<https://pt.stackoverflow.com/tags/cakephp>

<https://pt.stackoverflow.com/questions/tagged/cakephp> (perguntas)

<https://stackoverflow.com/tags/cakephp>

<https://discourse.cakephp.org/>

<https://groups.google.com/forum/#!forum/cake-php>

<https://groups.google.com/forum/#!forum/cakephp-pt>

<https://www.facebook.com/groups/cake.community/>

<https://www.facebook.com/groups/cakebrasil/>

O CakePHP torna a criação de aplicativos mais simples, rápida e fácil.

## Justificando a adoção do CakePHP

Frente à boa quantidade de frameworks em PHP quero justificar a minha adoção do CakePHP. A adoção de uma ferramenta depende da finalidade e de vários fatores. No meu caso, criar aplicativos de forma produtiva e que implementem com facilidade ACL foram os dois pontos mais importantes para que eu adotasse o CakePHP como meu framework. Estive experimentando os principais: Zend, CodeIgniter, Laravel, Yii entre outros. Para meu caso, o CakePHP foi o mais produtivo, especialmente por conta da espetacular ferramenta de geração automática de aplicativo "bake" e de sua estrutura de Autenticação e Autorização que me facilitaram a criação do plugin cake-acl-br.

O CakePHP foi concebido para tornar tarefas de desenvolvimento web mais simples e fáceis, pois fornece uma caixa de ferramentas completa para você poder começar. Veja isso:

[https://www.cakedc.com/megan\\_lalk/2017/08/29/reasons-why-you-should-consider-cakephp-for-your-next-website](https://www.cakedc.com/megan_lalk/2017/08/29/reasons-why-you-should-consider-cakephp-for-your-next-website)

[https://www.cakedc.com/megan\\_lalk/2016/06/16/cakephp\\_-\\_an\\_open\\_source\\_framework\\_with\\_many\\_benefits](https://www.cakedc.com/megan_lalk/2016/06/16/cakephp_-_an_open_source_framework_with_many_benefits)

## **Objetivo principal deste livro**

O objetivo principal deste livro é o de colaborar para tornar a criação de aplicativos com o CakePHP 3 mais produtiva e agradável, mostrando dicas e informações importantes para a criação de aplicativos e também mostrar como criar plugin, component, helper e element com o intuito de estender as funcionalidades do CakePHP 3 e no capítulo 14 mostro várias dicas de como codificar no CakePHP 3 manualmente, tanto sobre o ORM quanto outras áreas, além do aplicativo para controle de finanças pessoais, onde crio código customizado (camada de negócios) para a criação do aplicativo.

## **Site oficial**

<https://cakephp.org/>

<https://book.cakephp.org/3.0/en/index.html>

<https://book.cakephp.org/3.0/pt/index.html>

<https://api.cakephp.org/3.8/>

## 2 – MVC no CakePHP

*Uma pessoa inteligente resolve um problema, um sábio previne.*  
Albert Einstein

O CakePHP usa o padrão MVC para organizar seu código, portando seguem algumas informações que tornarão mais simples o aprendizado do uso do CakePHP 3.

### **Model**

Nesta camada são realizadas as operações de validação, leitura e escrita de dados no banco de dados. É responsável por salvar e receber dados do banco de dados, como também efetua diversos processamentos com os dados.

A camada Model é o cérebro da aplicação. Basicamente qualquer coisa para ler, alterar, salvar ou excluir dados é nesta camada. A camada Model é a camada que sofreu a maior transformação na versão 3.

Nas versões anteriores o Model era representado por classes Model e Behavior. Na versão 3 agora temos as classes Table, Entity, Behavior e Query.

As classes Table representam os dados armazenados, normalmente uma tabela e são responsáveis por encontrar, salvar e validar nossos dados. Também são responsáveis por manipular grandes massas de dados, como cálculo dos totais de uma coleção de dados.

As classes Entity formam um conceito inteiramente novo e elas representam dados simples, isto é, apenas um registro na tabela. Usando classes Entity permite a você criar campos de dados virtuais muito complexos.

Classes Behavior funcionam exatamente como as da versão anterior, complementando as classes Table e Entity.

A nova classe Query dá ao CakePHP 3 uma incrível e poderosa maneira de interagir com bancos de dados relacionais padrões, através do que é chamada de uma interface fluente.

Uma boa prática é trazer para esta camada tudo que diz respeito às regras de negócio, como cálculos ou validações de integridade de dados.

Por padrão o Cake trabalha com um único model por tabela.

Caso não criemos um model em nosso aplicativo o Cake criará um dinamicamente com as funcionalidades básicas.

Os métodos que operam coleções de entidades são colocados na classe Table, enquanto os recursos pertencentes a um único registro são colocados na classe Entity.

## Controller

É o responsável pela integração entre as camadas Model e View. Basicamente a View irá realizar uma solicitação para o Controller como por exemplo uma coleção de dados ou a solicitação de remover algum item do banco e o Controller, por sua vez, irá enviar a instrução para a camada Model executar.

Um controller pode ser visto como um gerente que certifica-se que todos os recursos necessários para completar uma tarefa sejam delegados aos trabalhadores corretos.

Ele aguarda por requisições dos clientes, checa suas validades de acordo com autenticação ou regras de autorização, delega requisições ou processamento de dados para a 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.

Nesta camada (Controller) também podemos realizar verificações que não se referem às regras de negócio, visto que a boa prática é manter as regras de negócio no Model.

O Controller é o coração do aplicativo. Esta camada é a que teve a menor quantidade de alterações da versão 2 para a 3. É responsável por controlar o fluxo de dados da aplicação: recebe requisições da view e as envia para o model. Então recebe de volta do model, efetua algum processamento e envia de volta a resposta para a view.

O Controller deve ser magro enquanto que gordo deve ser o model, onde fica a lógica de negócios.

Uma solicitação de entrada do cliente é enviada para uma ação de um controller específico, que tem a lógica necessária para determinar quais dados são necessários e em qual tabela estariam. O controller verifica se o cliente solicitante tem permissão para acessar tais dados e qual view deve ser usada para a saída dos dados. A camada controller consiste de classes Controller e classes Component (opcionalmente). Um componente tem função parecida com a dos Behavior para os Model, eles auxiliam os Controllers, assim como os Behavior auxiliam os Models. O componente encapsula código que deve ser compartilhado com os Controllers.

*Do livro oficial do CakePHP 3*

Controllers no CakePHP lidam com solicitações HTTP e executam lógica de negócios contida em métodos de Model, para preparar a resposta, que será repassada para uma view. São usados para receber requisições e requisitar dados e processá-los e, se for o caso, entregar para a view respectiva.

Trabalhar com bancos de dados pode ser em controllers e models, mas em models é mais adequado e recomendado. Embora tenhamos a liberdade de trabalhar onde quisermos.

Actions são métodos public do controller que possuem rotas conectadas a eles.

### Como funciona uma requisição:

- O dispatche/despachante do CakePHP usa o roteador/router para determinar qual ação do controller deve ser chamado para servir a solicitação de entrada



- O controller, então, verificar se há detalhes de autenticação (se necessário) e quaisquer parâmetros de solicitação, tais como IDs, extensões de arquivos, etc.
- A ação/action do controller solicita os dados necessários da camada de modelo
- A camada Model recupera os dados solicitados a partir do armazenamento de dados, manipula e formata conforme necessário e retorna os dados para o controlador
- O controlador então passa os dados para a camada de visão
- A camada View torna os dados em HTML (ou JSON, PDF, XML, etc.)
- Finalmente, o controlador volta a saída da camada View para o dispatcher, que envia para o cliente.

## **View**

É a camada responsável por tudo que é visual, páginas, formulários, listagens, menus, o HTML. Tudo aquilo que interage com o usuário deve estar presente nesta camada. Representadas por HTML, classes Helpers, View, View Cells

A View não realiza operações diretamente com o banco de dados nem trata diretamente com o Model. Ela as solicita e e exibe através do Controller, que intermedia suas solicitações com o Model.

No CakePHP 3 as views agora ficam no diretório Template  
O diretório View contém apenas classes gerais.

Se o Controller é o coração, Model é o cérebro da nossa aplicação, então a View é a pele.

Para a maioria das aplicações, as views são uma mescla entre HTML e PHP, mas também podem ser distribuídas como XML, CSV, ou ainda dados binários.

Os templates da view do CakePHP são código PHP para apresentação que são inseridos dentro do layout do aplicativo.

Um layout é um código de apresentação que contém em seu interior uma view/template. Arquivos de layout contêm elementos de site comuns, como cabeçalhos, rodapés e elementos de navegação. Seu aplicativo pode ter vários layouts e você pode alternar entre eles.

Os arquivos de template do CakePHP são armazenados em src/Template dentro de uma pasta com o nome do controller ao qual eles correspondem. Então, teremos que criar uma pasta chamada "Artigos" nesse caso.

Você precisa incluir o componente Flash em qualquer controlador onde você o usará. Por conta disso ele já vem no src/Controller/AppController.php gerado pelo bake.

O método `$this->Form->control()` é usado para criar elementos de formulário com o mesmo nome. O primeiro parâmetro diz ao CakePHP a que campo eles correspondem, e o segundo parâmetro permite que você especifique uma grande variedade de opções - neste caso, o número de linhas para a área de texto. Há um pouco de introspecção e convenções usadas aqui. O `control()` gerará diferentes elementos de formulário com base no campo de modelo especificado e usará inflexão para gerar o texto do rótulo (os nomes virão da tabela no banco de dados). Você pode personalizar o rótulo, a entrada ou qualquer outro aspecto dos controles de formulário usando opções. A chamada `$this->Form->end()` fecha o formulário.

## **Resumidamente**

**Models** – consultam o banco de dados e retorna os dados necessários

**Views** – são páginas que renderizam os dados recebidos

**Controllers** – manipulam as requisições dos usuários, recebe dados dos models e passa para as views

## **Mais detalhes**

<https://book.cakephp.org/3.0/en/intro.html>

## 3 - Convenções sobre Configurações

*Você erra todo arremesso que não tenta.*  
Michal Jordan

A equipe do CakePHP é grande admiradora de convenção sobre configuração. Seguindo as convenções definidas pelo Cake você recebe funcionalidades gratuitamente e libera a si mesmo de manter arquivos de configuração e com isso economizamos tempo e esforço.

Aderir às convenções vai lhe poupar tempo.

**Aviso Importante:** Antes de começar a trabalhar com CakePHP é importante conhecer suas convenções para tirar delas as devidas 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 de classes controller:

ClientesController, PeopleController e UltimosArtigosController

**Actions** – Os actions são os métodos dos Controllers, com visibilidade public e se comunicam com views com mesmo nome que eles e extensão .ctp.

Um exemplo: action index() - view index.ctp

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:

- /vermelhaMacas
- /VermelhaMacas
- /Vermelha\_macas
- /vermelha\_macas

Todos resolverão com o controller VermelhaMacas.

## Namespaces

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

Por exemplo: (troca src por Cake e troca as barras)

```
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 `KissesAndHugsController.php`

Classe `ClientesController` está no arquivo `ClientesController.php`.

## Model e Bancos de Dados

*Nomes de classes Table* - são no plural e CamelCase

Ao nomear nosso objeto como `ArticlesTable`, o CakePHP automaticamente deduz que o mesmo utilize o `ArticlesController` e seja relacionado à tabela `articles` e aos `Template\Articles`.

O CakePHP criará automaticamente um objeto model se não puder encontrar um arquivo correspondente em **src/Model/Table**. Se você nomear incorretamente seu arquivo (isto é, `artciclestable.php` ou `ArticleTable.php`), o CakePHP não reconhecerá suas definições e usará o model gerado como alternativa.

## Chave Primária

Toda tabela, obrigatoriamente deve ter uma chave primária e o nome da chave deve ser `id` para usufruir das vantagens do CakePHP.

Lembrando que o cake trabalha com nomes em inglês. Ele tem um recurso online importante para mostrar o plural de nomes:

<http://inflector.cakephp.org/>

*Nomes válidos*: `Clientes`, `Populacao`, `GrandePopulacao` e `RealmenteGrandePopulacao`.

Pesquisando no site acima por `Clientes` e `People`, vemos que já estão no plural e seu singular é `Cliente` e `Person`

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

`clientes`, `populacao`, `grande_populacao` e `realmente_grande_populacao`

**A convenção é para usar tabelas e campos com nomes na língua inglesa.**

**Nomes de campos** em minúsculas e compostos são separados por sublinhado: name, first\_name.

Se usarmos os **campos username e password** (com estes nomes) na tabela users, o Cake deve estar apto para auto-configurar algumas 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.

Por default o Cake espera identificar o user pelos campos com nome 'username' e 'password'. Caso deseje mudar isto faça as alterações no ApplicationController:

Adicione ao initialize():

```
$this->loadComponent('Auth', [
    'authorize' => ['Controller'],
    'logoutRedirect' => [
        'controller' => 'Users',
        'action' => 'login',
        'home'
    ],
    // 'unauthorizedRedirect' => $this->referer()
    'unauthorizedRedirect' => [
        'controller' => 'Users',
        'action' => 'login',
        'prefix' => false
    ],
    'authError' => 'Você não tem permissão para acessar esta área!',
    'flash' => [
        'element' => 'error'
    ],
    'authenticate' => [
        'Form' => [ // THIS IS WHERE YOU CHANGE THE DEFAULT FIELDS
            'fields' => ['username' => 'novoUsername', 'password' => 'novoPassword']
        ]
    ]
]);
```

## Relacionamentos

Chave estrangeira nos relacionamentos hasMany, belongsTo ou hasOne são reconhecidas por default com:

nome da tabela relacionada no singular 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.

## Relacionamento Muitos para Muitos

Exemplo: Para relacionamento muitos para muitos das tabelas pratos com cozinheiros, criaremos uma tabela intermediária para relacionar pratos com cozinheiros:

```

create table cozinheiros_pratos(
    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	Um usuario tem um perfil.
one to many	hasMany	Um usuario pode ter múltiplos artigos.
many to one	belongsTo	Muitos artigos pertencem a um usuario
many to many	belongsToMany	Várias Tags pertencem a muitos artigos.

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 a nosso favor:

title

name

created

modified

### **Convenções para as Views**

As views tem nomes de arquivos em minúsculas com extensão .ctp.

O método getReady() do controller PeopleController está associado ao template/view

src/Template/People/get\_ready.ctp.

O método index() do controller ClientesController está associado a

src/Template/Clientes/index.ctp.

### **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. Uma recomendação, quando usamos bcrypt é que o tamanho do campo password deve ter 255 caracteres na tabela, para que suporte o tamanho do hash gerado.

### **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.

## Alterando convenções

Recomenda-se evitar, mas caso necessite veja abaixo um exemplo

```
<?php
namespace App\Model\Table;

use Cake\ORM\Query;
use Cake\ORM\RulesChecker;
use Cake\ORM\Table;
use Cake\Validation\Validator;

class ClientesTable extends Table
{
    public function initialize(array $config)
    {
        parent::initialize($config);

        $this->table('clientes'); // Nome da tabela. Se ausente a convenção assume uma versão em
minúsculas do prefixo deste arquivo: ClientesTable.php
        $this->displayField('cpf'); // campo usado por default na exibição em models associados, se
ausente é assumido 'id'
        $this->primaryKey('id'); // Campo Primary key da table, se ausente a convenção assume que
seja o campo 'id'

        $this->addBehavior('Timestamp'); // Permite ao model gravar timestamp na
creation/modification dos registros
    }
}
```

Documentação oficial

<http://book.cakephp.org/3.0/pt/intro/conventions.html>





## 4 – Instalação e pré-requisitos do CakePHP 3

*Talento ganha jogos, mas trabalho sério e inteligência ganha campeonato.*  
Michael Jordan

### Pré-requisitos da versão 3.7

- Web server como Apache 2 com mod\_rewrite ou Nginx ou IIS
- PHP 5.6 ou superior
- Extensões mbstring, intl, simplexml e PDO
- Um dos SGBDs suportados: MySQL, PostgreSQL, MariaDB, Microsoft SQL Server e SQLite
- Composer (opcional mas recomendado)

### Instalação do Composer

- Se tens a versão 7 ou superior do PHP instale assim (no Debian ou derivada):  
*sudo apt-get install composer*

- Para a versão 5.x assim:

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

Colocar no path do Linux na versão 5.x:

```
mv composer.phar /usr/local/bin/composer
```

- A instalação no Windows é feita baixando  
<https://getcomposer.org/Composer-Setup.exe>

### Instalando o CakePHP

Para criar um aplicativo com o CakePHP 3, instalamos sua estrutura básica e completa através do composer para então criar o aplicativo usando o bake ou manualmente

```
composer create-project --prefer-dist cakephp/app nome_app
```

A vantagem de usar o Composer é que ele irá completar automaticamente um conjunto importante de tarefas, como configurar corretamente as permissões das pastas /tmp e /logs. criar o **config/app.php** para você e setar um salt único.

Como o comando é grande e pode dar trabalho de memorizar, vamos criar um link (no Windows crie um bat com %1), no Linux para receber parâmetro é \$1.

```
sudo nano /usr/local/bin/cake
```

No windows salve na pasta <c:\windows> para que fique no path

Cole a linha

```
composer create-project --prefer-dist cakephp/app $1
```

No Windows

Crie um arquivo `c:\windows\cake.bat`

```
composer create-project --prefer-dist cakephp/app %1
```

Para instalar uma versão diferente da atual, a 3.5.13, por exemplo:

```
composer create-project --prefer-dist cakephp/cakephp:3.5.13 blog
```

**Execute assim com nosso script/batch criado:**

```
cd /var/www/html
```

```
cake cliente
```

Ou

```
cd c:\xampp\htdocs
```

```
cake.bat cliente
```

Ao finalizar o Cake já terá trazido todo o seu código para a pasta `/var/www/html/cliente` ou `c:\xampp\htdocs\cliente`.

E já vem com um controller default para que vejamos algo.

**Vejas as mensagens finais do instalador:**

```
Writing lock file
```

```
Generating autoload files
```

```
> Cake\Composer\Installer\PluginInstaller::postAutoloadDump
```

```
> App\Console\Installer::postInstall
```

```
Created `config/app.php` file
```

```
Created `/backup/www/cake/aclbr/tmp/cache/views` directory
```

```
Set Folder Permissions ? (Default to Y) [Y,n]?
```

```
Permissions set on /backup/www/cake/aclbr/tmp/cache
```

```
Permissions set on /backup/www/cake/aclbr/tmp/cache/models
```

```
Permissions set on /backup/www/cake/aclbr/tmp/cache/persistent
```

```
Permissions set on /backup/www/cake/aclbr/tmp/cache/views
```

```
Permissions set on /backup/www/cake/aclbr/tmp/sessions
```

```
Permissions set on /backup/www/cake/aclbr/tmp/tests
```

```
Permissions set on /backup/www/cake/aclbr/tmp
```

```
Permissions set on /backup/www/cake/aclbr/logs
```

```
Updated Security.salt value in config/app.php
```

Ele criou o arquivo `config/app.php`

Ajustou as permissões no `/tmp` e no `/logs`

Atualizou o salt em `config/app.php`

## Estrutura de diretórios do CakePHP 3.7.7:

`/bin` – contém todos os arquivos executáveis para a console do CakePHP  
`/config` – Contém os arquivos de configuração do usuário  
`/logs` – Contém os arquivos de log da aplicação  
`/plugins` – destinado a armazenar os plugins locais da aplicação  
`/src` – Contém os Controllers, Models, Templates e Views da aplicação  
`/tests` – Contém todos os seus arquivos de teste  
`/tmp` – Arquivos temporários são armazenados aqui  
`/vendor` – Contém todas as dependências do Cake e o próprio. Os instalados com composer ficam aqui  
`/webroot` – Contém todos os arquivos publicamente acessíveis  
`composer.json` – Arquivo do composer com a lista e versões dos arquivos que a aplicação depende  
`composer.lock` – Arquivo do composer que bloqueia específicas versões de pacotes de PHP instalados  
`index.php` - Arquivo inicial que chama o webroot  
`phpunit.xml.dist` – A ser usado pelo phpunit  
`README.md` – Informações do repositório para o GitHub

### Detalhes em:

<https://book.cakephp.org/3.0/pt/intro/cakephp-folder-structure.html>

Os diretórios **tmp** e **logs** precisam ter permissões adequadas para que possam ser alterados pelo seu servidor web. Se você usou o Composer na instalação, ele deve ter feito isso por você e confirmado com uma mensagem "Permissions set on <folder>". Se você ao invés disso, recebeu uma mensagem de erro ou se quiser fazê-lo manualmente, a melhor forma seria descobrir por qual usuário o seu servidor web é executado (`<?php echo 'whoami'; ?>`) e alterar o proprietário desses dois diretórios para este usuário. Os comandos finais a serem executados (em \*nix) podem ser algo como:

```
chown -R www-data tmp
chown -R www-data logs
```


Se por alguma razão o CakePHP não puder escrever nesses diretórios, você será informado por uma advertência enquanto não estiver em modo de produção.

### Chame pelo navegador





<http://localhost/cliente>

Aparecerá então a tela abaixo:






 Please be aware that this page will not be shown if you turn off debug mode unless you replace src/Template/Pages/home.ctp with your own version.

## Environment

-  Your version of PHP is 5.6.0 or higher (detected 5.6.30-0+deb8u1).
-  Your version of PHP has the mbstring extension loaded.
-  Your version of PHP has the openssl extension loaded.
-  Your version of PHP has the intl extension loaded.

## Filesystem

-  Your tmp directory is writable.
-  Your logs directory is writable.
-  The *FileEngine* is being used for core caching. To change the config edit config/app.php

## Database

## DebugKit

### Configurações do banco de dados no config/app.php

Procure a seção **Datasources** e altere apenas:

```
'username' => 'root',  
'password' => '',  
'database' => 'cliente',
```

Crie o banco chamado **cliente** com a tabela cliente e importe o script abaixo:

```
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`)  
);
```

```
INSERT INTO `clientes` (`id`, `nome`, `email`, `data_nasc`, `cpf`) VALUES  
(1, 'Erin Pate Skinner', 'dolor.vitae.dolor@mollisvitaeposuere.ca', '2013-10-07', '74426302285'),  
(2, 'Leonard Martinez Hays', 'dignissim.magna.a@dolorvitae.org', '2012-08-22', '75278965048'),  
(3, 'Aladdin Curry French', 'eu.augue@eutemporerat.org', '2012-10-28', '10376915676'),  
(4, 'Chloe Macdonald Dalton', 'parturient.montes@Mauris.ca', '2013-05-12', '64444679077'),
```

(5, 'Mallory Sweet Strong', 'lorem@fringillaporttitor.ca', '2013-05-19', '15687101505'),  
(6, 'Jermaine Pierce Woodward', 'mi.pede.nonummy@molestiearcu.ca', '2013-03-22', '36712502261'),  
(7, 'Bell Raymond Pruitt', 'dignissim.tempor.arcu@nuncac.org', '2013-03-09', '64629428663'),  
(8, 'Lydia Bell Whitfield', 'Sed@semper.com', '2013-12-02', '41962749289'),  
(9, 'Tad Mason Graham', 'elit.erat@vestibulum.edu', '2012-06-08', '05642745964'),  
(10, 'Felix Bradshaw Mccray', 'dui@elitCurabitursed.edu', '2013-09-16', '82071617437'),  
(11, 'Idona Jensen Garrett', 'sem@Crasvulputate.com', '2014-01-08', '07941004794'),  
(12, 'Wayne Ray Padilla', 'luctus.felis.purus@nonjustoProin.org', '2014-04-03', '60934465323'),  
(13, 'Nelle Finch Cantu', 'placerat.eget@Donec.ca', '2012-05-29', '64704574060'),  
(14, 'Maite Emerson Best', 'dui.augue@quisdiam.com', '2014-04-01', '04531857574'),  
(15, 'Jada Holman Wilkins', 'dolor@tristiquealiquet.com', '2013-01-11', '88994190741'),  
(16, 'Beverly Lane Lindsay', 'et.euismod@ametfaucibusut.com', '2013-10-22', '40194697135'),  
(17, 'Hayden Clayton Foreman', 'enim@aliquamenimnec.edu', '2013-04-16', '72583040904'),  
(18, 'Hadassah Leonard Key', 'dui.quis@augueidante.com', '2013-04-07', '72626859924'),  
(19, 'Adrian Ballard Peters', 'enim.Curabitur@faucibus.com', '2012-07-13', '50918748283'),  
(20, 'Phyllis Richmond Wynn', 'eget.laoreet@justoearcu.org', '2013-07-01', '62712888794'),  
(21, 'Amelia Baird Barrera', 'id.ante@dignissim.org', '2012-06-09', '12106836368'),  
(22, 'Whitney Mack Lamb', 'quam.Curabitur.vel@PraesentluctusCurabitur.org', '2012-06-26', '52403407001'),  
(23, 'Myra Mcmahon Valentine', 'ac.mi@fringillami.edu', '2012-07-27', '42961419194');

## Routes

Para que ao abrir o aplicativo automaticamente ele mostra o controller Clientes, em config/routes.php altere a linha abaixo:

Originalmente é assim a alinha:

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

Mudei para

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

## Geração do Código do CRUD completo com o Bake

Depois de configurados o app.php e o routes.php, então execute:

```
cd cliente  
bin/cake bake all clientes
```

**Importante.:** Lembre de mudar a barra / por \ se estiver usando o Windows.

## Como saber a versão atual do CakePHP que está instalado?

```
bin/cake version
```

## Acesse novamente pelo Navegador

Agora pode acessar seu aplicativo pelo navegador

<http://localhost/cliente>

Aparece por default o Clientes/index, como configuramos no routes

Clientes		Documentation	API		
<b>ACTIONS</b>					
New Cliente					
<b>Clientes</b>					
Id	Nome	Email	Data Nasc	Cpf	Actions
1	Erin Pate Skinner	dolor.vitae.dolor@mollisvitaeposuere.ca	10/7/13	74426302285	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
2	Leonard Martinez Hays	dignissim.magna.a@dolorvitae.org	8/22/12	75278965048	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
3	Aladdin Curry French	eu.augue@eutemporerat.org	10/28/12	10376915676	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
4	Chloe Macdonald Dalton	parturient.montes@Mauris.ca	5/12/13	64444679077	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
5	Mallory Sweet Strong	lorem@fringillaportitor.ca	5/19/13	15687101505	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
6	Jermaine Pierce Woodward	mi.pede.nonummy@molestiearcu.ca	3/22/13	36712502261	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>

Um CRUD completo e funcional com bons recursos.

## Customizações

Agora já podemos melhorar seus recursos e também instalar o plugin cake-acl-br para implementar ACL com Bootstrap e outros bons recursos, ou se apenas deseja

implementar o Bootstrap instale o plugin cake-bs-br:

<https://github.com/ribafs/cake-acl-br>

<https://github.com/ribafs/cake-bs-br>

O tutorial/README.md tem um bom tutorial que mostra diversas customizações do aplicativo.

## Testando

Crie um novo registro e insira um e-mail inválido e então clique em SUBMIT. Veja que o Cake já criou o form usando HTML5, pois o campo para o e-mail é do tipo e-mail e o HTML5 não deixa inserir um e-mail inválido.

## Tome cuidado com a criação do banco de dados

O Cake faz isso com as informações que colhe do banco de dados e critica de acordo com as constraints presentes. Um exemplo é quando usamos o NOT NULL em um campo. Neste caso o Cake cria uma validação para obrigar o preenchimento do campo.

## **Problema com o mod\_rewrite?**

Ocasionalmente, novos usuários irão se atrapalhar com problemas de mod\_rewrite. Por exemplo, se a página de boas vindas do CakePHP parecer estranha (sem imagens ou estilos CSS). Isto provavelmente significa que o mod\_rewrite não está funcionando em seu servidor. Por favor, verifique a seção [Reescrita de URL](#) para obter ajuda e resolver qualquer problema relacionado.

### **Detalhes**

<https://book.cakephp.org/3.0/en/installation.html>





## 4.1 - Algumas configurações

### config/app.php

#### Debug

```
'encoding' => env('APP_ENCODING', 'UTF-8'),
'defaultLocale' => env('APP_DEFAULT_LOCALE', 'en_US'),
'defaultTimezone' => env('APP_DEFAULT_TIMEZONE', 'UTC'),
'dir' => 'src',
'webroot' => 'webroot',
'imageBaseUrl' => 'img/',
'cssBaseUrl' => 'css/',
'jsBaseUrl' => 'js/',
'paths' => [
    'plugins' => [ROOT . DS . 'plugins' . DS],
    'templates' => [APP . 'Template' . DS],
    'locales' => [APP . 'Locale' . DS],
],

'Security' => [
    'salt' => env('SECURITY_SALT',
'64ac1a00cd7dbfa490f00064ab5bbe4d642a00711be4d0be8249706974c23f31'),
],

'Cache' => [
    'default' => [
        'className' => 'Cake\Cache\Engine\FileEngine',
        'path' => CACHE,
        'url' => env('CACHE_DEFAULT_URL', null),
    ],
],

'Error' => [
    'errorLevel' => E_ALL & ~E_USER_DEPRECATED,
],

'EmailTransport' => [
    'default' => [
        'className' => 'Cake\Mailer\Transport\MailTransport',
        /*
        * The following keys are used in SMTP transports:
        */
        'host' => 'localhost',
        'port' => 25,
        'timeout' => 30,
        'username' => null,
        'password' => null,
        'client' => null,
        'tls' => null,
```

```

        'url' => env('EMAIL_TRANSPORT_DEFAULT_URL', null),
    ],
],

```

```

'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' => 'root',
        'database' => 'testes',
        /*

```

```

         * You do not need to set this flag to use full utf-8 encoding (internal default since CakePHP

```

3.6).

```

        */
        //'encoding' => 'utf8mb4',
        'timezone' => 'UTC',

```

```

'Session' => [
    'defaults' => 'php',
],

```

### config/bootstrap.php

```

* Setup detectors for mobile and tablet.
*/
ServerRequest::addDetector('mobile', function ($request) {
    $detector = new \Detection\MobileDetect();

    return $detector->isMobile();
});
ServerRequest::addDetector('tablet', function ($request) {
    $detector = new \Detection\MobileDetect();

    return $detector->isTablet();
});

```

**También podemos configurar timezone** e formato de datas no bootstrap.php, veja:

```

date_default_timezone_set('America/Fortaleza');

```

```

/**
 * Locale Formats
 */
\Cake\I18n\Time::setToStringFormat([IntlDateFormatter::MEDIUM, IntlDateFormatter::SHORT]);
\Cake\I18n\Time::setToStringFormat('dd/MM/YYYY HH:mm');

\Cake\Database\Type::build('date')
->useLocaleParser()
->setLocaleFormat('dd/MM/yyyy');

\Cake\Database\Type::build('datetime')
->useLocaleParser()
->setLocaleFormat('dd/MM/yyyy HH:mm');

\Cake\Database\Type::build('timestamp')
->useLocaleParser()
->setLocaleFormat('dd/MM/yyyy HH:mm');

\Cake\Database\Type::build('decimal')
->useLocaleParser();

\Cake\Database\Type::build('float')
->useLocaleParser();

ini_set('intl.default_locale', 'pt_BR');

/**
 * Default formats
 */
\Cake\I18n\Time::setToStringFormat('dd/MM/yyyy HH:mm:ss');
\Cake\I18n\Date::setToStringFormat('dd/MM/yyyy');
\Cake\I18n\FrozenTime::setToStringFormat('dd/MM/yyyy HH:mm:ss');
\Cake\I18n\FrozenDate::setToStringFormat('dd/MM/yyyy');

```

## O código acima vem do plugin admin-br

### config/bootstrap\_cli.php

Neste podemos alterar o comportamento default do bake. Vide admin-br.

### config/requirements.php

Neste ele verifica a versão mínima exigida do PHP e as duas extensões (intl e mbstring):

```

if (version_compare(PHP_VERSION, '5.6.0') < 0) {
    trigger_error('Your PHP version must be equal or higher than 5.6.0 to use CakePHP.' . PHP_EOL,
E_USER_ERROR);
}

/*
 * You can remove this if you are confident you have intl installed.

```

```

*/
if (!extension_loaded('intl')) {
    trigger_error('You must enable the intl extension to use CakePHP.' . PHP_EOL,
E_USER_ERROR);
}

/*
 * You can remove this if you are confident you have mbstring installed.
*/
if (!extension_loaded('mbstring')) {
    trigger_error('You must enable the mbstring extension to use CakePHP.' . PHP_EOL,
E_USER_ERROR);
}

```

### **config/routes.php**

#### **Neste ele verifica as rotas:**

```

Router::scope('/', function (RouteBuilder $routes) {
    // Register scoped middleware for in scopes.
    $routes->registerMiddleware('csrf', new CsrfProtectionMiddleware([
        'httpOnly' => true
    ]));

    /**
     * Apply a middleware to the current route scope.
     * Requires middleware to be registered via `Application::routes()` with `registerMiddleware()`
     */
    $routes->applyMiddleware('csrf');
...

```

#### **E define as rotas default e outras:**

```

$routes->connect('/', ['controller' => 'Users', 'action' => 'login']);
$routes->connect('/pages/*', ['controller' => 'Pages', 'action' => 'display']);

```

#### **Detalhes**

<https://book.cakephp.org/3.0/en/development/configuration.html>

## 5 - Gerando CRUD com Bake e muito mais

*Algumas pessoas querem que algo aconteça, outras desejam que aconteça, outras fazem acontecer.*

Michael Jordan

<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

### Executando o bake

```
cd /var/www/html/cliente  
bin/cake bake
```

Mostrará:

*The following commands can be used to generate skeleton code for your application.*

*Available bake commands:*

- *all*
- *behavior*
- *cell*
- *command*
- *component*
- *controller*
- *fixture*
- *form*
- *helper*
- *mailer*
- *middleware*
- *migration*
- *migration\_diff*
- *migration\_snapshot*
- *model*
- *plugin*

- *seed*
- *shell*
- *shell\_helper*
- *task*
- *template*
- *test*
- *twig\_template*

By using ``cake bake [name]`` you can invoke a specific bake task.

## Obtendo ajuda para controllers com o bake

`bin/cake bake controller --help`

Veja:

`bin/cake bake controller --help`  
*Bake a controller skeleton.*

Usage:

`cake bake controller [subcommand] [options] [<name>]`

Subcommands:

*all* Bake all controllers with CRUD methods.

To see help on a subcommand use ``cake bake controller [subcommand] --help``

Options:

`--actions` The comma separated list of actions to generate. You can include custom methods provided by your template set here.

`--components` The comma separated list of components to use.

`--connection, -c` The datasource connection to get data from.  
(default: default)

`--force, -f` Force overwriting existing files without prompting.

`--help, -h` Display this help.

`--helpers` The comma separated list of helpers to use.

`--no-actions` Do not generate basic CRUD action methods.

`--no-test` Do not generate a test skeleton.

`--plugin, -p` Plugin to bake into.

`--prefix` The namespace/routing prefix to use.

`--quiet, -q` Enable quiet output.

`--theme, -t` The theme to use when baking code. (choices: *Bake|Migrations|WyriHaximus/TwigView*)

`--verbose, -v` Enable verbose output.

*Arguments:*

*name* Name of the controller to bake (without the `Controller` suffix).  
You can use *Plugin.name* to bake controllers into plugins.

### **Versatilidade do bake**

#### **Ajuda para plugins**

bin/cake bake plugin --help

#### **Gerar tudo para Clientes** (controller, model, templates, etc)

bin/cake bake all Clientes

#### **Gerar novamente forçando**

Caso já tenha gerado um código e queira gerar novamente evitando as perguntas para sobrescrever os arquivos gerados, use --force ou -f, assim:

bin/cake bake all Clientes -f

#### **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

#### **Gerar apenas controller, model e template**

bin/cake bake controller clientes

bin/cake bake model clientes

bin/cake bake template clientes index (somente a index)

## Gerando plugin

```
bin/cake bake all --plugin CakePtblr articles
```

## Agora gerar o código completo para o plugin

```
bin/cake bake all -p CakeAdmin groups
```

```
bin/cake bake all -p CakeAdmin users
```

```
bin/cake bake all -p CakeAdmin permissions
```

```
bin/cake bake all -p CakeAdmin customers
```

## Criar plugin ContactManager com tabela contacts para o aplicativo cliente

```
cd /var/www/html/cliente
```

```
bin/cake bake plugin ContactManager
```

```
bin/cake bake controller --plugin ContactManager Contacts
```

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

```
bin/cake bake migration CreateGroups name:string created modified
```

```
bin/cake bake migration CreateUsers username:string password:string created modified
```

```
bin/cake migrations migrate
```

```
bin/cake bake model --plugin ContactManager Contacts
```

```
bin/cake bake template --plugin ContactManager Contacts
```

## Estrutura de Arquivos do Plugin

```
/src
/plugins
  /ContactManager
    /config
    /src
      /Plugin.php
      /Controller
        /Component
      /Model
        /Table
        /Entity
        /Behavior
      /View
        /Helper
      /Template
      /Layout
    /tests
      /TestCase
      /Fixture
    /webroot
```

## Regerando o autoloader

```
composer dumpautoload
```

## Limpendo o cache

```
composer clear-cache
```

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



## **Outros comandos**

```
bin/cake bake fixture groups  
bin/cake bake test users
```

## **Garantir que não use cache:**

```
bin/cake bake model comments --force  
bin/cake bake model issues --force  
bin/cake orm_cache clear
```

## **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 metadados 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 artigos**

```
bin/cake orm_cache clear artigos
```

## **Criando Model para o Plugin**

```
bin/cake bake model --plugin Admin Groups  
bin/cake bake model --plugin Admin Users
```

## **Criando Templates para o Plugin**

```
bin/cake bake template --plugin Admin Groups  
bin/cake bake template --plugin Admin Users
```

## **Criar somente a view login.ctp para o Plugin**

```
bin/cake bake template --plugin Admin Users login
```

## **Observação importante:**

Atualmente na versão 3, idealmente devemos criar plugins para a pasta vendor, que são hospedados num repositório público, como o GitHub e publicados num repositório do composer, como o Packagist. Assim podemos instalar e publicar o plugin pela linha de comando com o composer.

## **A oitava maravilha do mundo**

Gosto de dizer que o bake é a oitava maravilha do mundo da programação web. Realmente me maravilho com seus resultados mágicos e ele me prende ao Cake.

Mas gostaria de deixar aqui um alerta, especialmente para o programador iniciante: evite ficar preso somente ao bake e se satisfazer com o resultado dele. Ao contrário estude o código gerado por ele, que por sinal é muito bom, para então conseguir adaptar seus aplicativos para as tuas necessidades e dos teus clientes.

Comodismo é algo que acaba com o programador e me parece que com qualquer área da vida do ser humano. A curiosidade, a motivação, o interesse em estudar, aprender, mexer, customizar, isso sim, devem estar presentes na vida do programador e do ser humano, me parece.

**Detalhes - <https://book.cakephp.org/bake/1.x/en/index.html>**

## 5.1 - Tutorial de uso dos templates do bake no CakePHP 3

*A causa da derrota, não está nos obstáculos, ou no rigor das circunstâncias, está na falta de determinação e desistência da própria pessoa. (Buda)*

**Na pasta dos fontes do CakePHP 3 temos:**

```
vendor/cakephp/bake/src/Template/Bake
vendor/cakephp/bake/src/Template/Bake/Element
vendor/cakephp/bake/src/Template/Bake/Form
vendor/cakephp/bake/src/Template/Bake/Controller
vendor/cakephp/bake/src/Template/Bake/Model
```

Entre outras pastas.

**Desde a versão 0.1.0** usa-se as tags do ASP no template do bake do CakePHP, que ainda são aceitas na versão 3, mas com previsão de serem removidas na versão 4:

- <% A Bake template php open tag
- %> A Bake template php close tag
- <%= A Bake template php short-echo tag
- <%- A Bake template php open tag, stripping any leading whitespace before the tag
- -%> A Bake template php close tag, stripping trailing whitespace after the tag

**Adicionar também login e logout na geração de código com o Bake.**

Para isso mude o arquivo `config/bootstrap_cli.php`

E deixe assim:

```
<?php
use Cake\Core\Configure;
use Cake\Core\Exception\MissingPluginException;
use Cake\Core\Plugin;
use Cake\Event\Event;
use Cake\Event\EventManager;
use Cake\Utility\Hash;

Configure::write('Log.debug.file', 'cli-debug');
Configure::write('Log.error.file', 'cli-error');

try {
    $this->addPlugin('Bake');
} catch (MissingPluginException $e) {
    // Do not halt if the plugin is missing
}

EventManager::instance()->on(
    'Bake.beforeRender.Controller.controller',
```

```
function (Event $event) {
    $view = $event->subject();
    if ($view->viewVars['name'] == 'Users') {
        // add the login and logout actions to the Users controller
        $view->viewVars['actions'] = [
            'login',
            'logout',
            'index',
            'view',
            'add',
            'edit',
            'delete'
        ];
    }
}
);
```

`$this->addPlugin('Migrations');`

Agora, ao gerar código com o bake assim:

```
bin/cake bake all users
```

Ele gerará também os actions `login()` e `logout()` e também o template `login.ctp`

**O Bake usa em seus templates atualmente, na versão 3.7.8, a sintaxe do twig, inclusive seus arquivos tem extensão twig.**

Veja por exemplo, parte do código do arquivo `vendor/cakephp/bake/src/Template/Bake/Element/form.twig` abaixo:

```
{% set fields = Bake.filterFields(fields, schema, modelObject) %}
<nav class="large-3 medium-4 columns" id="actions-sidebar">
    <ul class="side-nav">
        <li class="heading"><? = __('Actions') ?></li>
        {% if strpos(action, 'add') is same as(false) %}
            <li><? = $this->Form->postLink(
                __('Delete'),
                ['action' => 'delete', ${{ singularVar }}->{{ primaryKey[0] }},
                ['confirm' => __('Are you sure you want to delete # {0}?', ${{ singularVar }}-
                >{{ primaryKey[0] })]
            )
            ?></li>
        {% endif %}
        <li><? = $this->Html->link(__('List {{ pluralHumanName }}'), ['action' => 'index']) ?></li>
        {{- "\n" }}
```

**A versão atual (3.7.8) ainda aceita a sintaxe abaixo:**

- `<% A Bake template php open tag`

- %> A Bake template php close tag
- <%= A Bake template php short-echo tag
- <%- A Bake template php open tag, stripping any leading whitespace before the tag
- -%> A Bake template php close tag, stripping trailing whitespace after the tag

### Com estes a extensão deve ser .ctp

A partir da versão 2.0.0 do Cake a sintaxe passou a usar Twig, inclusive na extensão dos arquivos:

- {% A Bake template php open tag
- %} A Bake template php close tag
- {%= A Bake template php short-echo tag
- {%- A Bake template php open tag, stripping any leading whitespace before the tag
- -%} A Bake template php close tag, stripping trailing whitespace after the tag

### Usa também para as seções de comentários:

```
{#
e
#}
```

Agora veja o arquivo form.ctp usado no plugin ribafs/admin-br em sua versão 1.8, onde adicionei suporte para o Bootstrap 3:

```
<%
/**
 * CakePHP(tm) : Rapid Development Framework (http://cakephp.org)
 * Copyright (c) Cake Software Foundation, Inc. (http://cakefoundation.org)
 *
 * Licensed under The MIT License
 * For full copyright and license information, please see the LICENSE.txt
 * Redistributions of files must retain the above copyright notice.
 *
 * Slightly modified by Òscar Casajuana for the twbs-cake-plugin
 * also under the MIT license.
 *
 * @copyright Copyright (c) Cake Software Foundation, Inc. (http://cakefoundation.org)
 * @link http://cakephp.org CakePHP(tm) Project
 * @since 0.1.0
 * @license http://www.opensource.org/licenses/mit-license.php MIT License
 */
use Cake\Utility\Inflector;

$fields = collection($fields)
->filter(function($field) use ($schema) {
    return $schema->columnType($field) !== 'binary';
});
$pk = "\${$singularVar}->{$primaryKey[0]}";
%>
<div class="container">
```

```

<div class="actions columns col-lg-2 col-md-3">
  <h3><?= __( 'Ações' ) ?></h3>
  <ul class="nav nav-stacked nav-pills">
<% if (strpos($action, 'add') === false): %>
  <li class="active disabled"><?= $this->Html->link(__( 'Editar <%= $singularHumanName
%>' ), ['action' => 'edit', <%= $pk %>] ) ?> </li>
  <li><?= $this->Form->postLink(
    __( 'Excluir' ),
    ['action' => 'delete', <%= $pk %>],
    ['confirm' => __( 'Deseja realmente excluir # {0}?' , $<%= $singularVar %>-><%=
$primaryKey[0] %>' ), 'class' => 'btn-danger' ]
  )
  ?></li>
  <li><?= $this->Html->link(__( 'Novo(a) <%= $singularHumanName %>' ), ['action' => 'add']) ?
></li>

```

**Dica** - Quando eu estava alterando uma view gerada pelo bake original para uso com Bootstrap, troquei uma tag inicial <nav> por <div> mas esqueci de trocar a tag </nav> final então ao visualizar pelo navegador a view apareceu em apenas uma coluna. Então devemos ficar bem atentos quando alterarmos o código de um template do bake. Um detalhe importante é que a sintaxe do twig pode ser detectada pelo VSCode ao instalar a extensão abaixo:

vscode-twig

Isso já ajuda, pois o VSCode já mostra o código colorido, separado em seções e até autocompleta o código e não mais todo em cinza.

Veja o início do form.ctp do template do bake no plugin:

<https://github.com/elboletaire/twbs-cake-plugin>

Em

src/Template/Bake/Element/form.ctp:

<https://github.com/elboletaire/twbs-cake-plugin/blob/master/src/Template/Bake/Element/form.ctp>

```

<%
/**
 * CakePHP(tm) : Rapid Development Framework (http://cakephp.org)
 * Copyright (c) Cake Software Foundation, Inc. (http://cakefoundation.org)
 *
 * Licensed under The MIT License
 * For full copyright and license information, please see the LICENSE.txt
 * Redistributions of files must retain the above copyright notice.
 *
 * Slightly modified by Òscar Casajuana for the twbs-cake-plugin
 * also under the MIT license.
 *
 * @copyright Copyright (c) Cake Software Foundation, Inc. (http://cakefoundation.org)
 * @link      http://cakephp.org CakePHP(tm) Project
 * @since     0.1.0

```

```

* @license    http://www.opensource.org/licenses/mit-license.php MIT License
*/
use Cake\Utility\Inflector;
$fields = collection($fields)
    ->filter(function($field) use ($schema) {
        return $schema->columnType($field) !== 'binary';
    });
$pk = "\${$singularVar}->{$primaryKey[0]}";
%>
<div class="container">
<div class="actions columns col-lg-2 col-md-3">
    <h3><?= __('Ações') ?></h3>
    <ul class="nav nav-stacked nav-pills">
<% if (strpos($action, 'add') === false): %>

```

Veja que é um plugin para o CakePHP 3 mas ainda usa as tags do ASP.

Agora vejamos como vem o template do bake na versão atual (3.7.8) do Cake:

vendor/cake/php/bake/src/Template/Bake/Element/form.twig

```

{#
/**
 * CakePHP(tm) : Rapid Development Framework (http://cakephp.org)
 * Copyright (c) Cake Software Foundation, Inc. (http://cakefoundation.org)
 *
 * Licensed under The MIT License
 * For full copyright and license information, please see the LICENSE.txt
 * Redistributions of files must retain the above copyright notice.
 *
 * @copyright    Copyright (c) Cake Software Foundation, Inc. (http://cakefoundation.org)
 * @link         http://cakephp.org CakePHP(tm) Project
 * @since       2.0.0
 * @license     http://www.opensource.org/licenses/mit-license.php MIT License
 */
#}
{% set fields = Bake.filterFields(fields, schema, modelObject) %}
<nav class="large-3 medium-4 columns" id="actions-sidebar">
    <ul class="side-nav">
        <li class="heading"><?= __('Actions') ?></li>
    <% if strpos(action, 'add') is same as(false) %>

```

Veja que a extensão é twig e a sintaxe não mais usa as tags ASP, mas a sintaxe do twig. Então, se queremos criar um template para o bake que será suportado pela versão 4 do Cake, não devemos usar as tags do ASP mas as do twig.

**Customizar o template do bake do CakePHP 3 para usar o Twitter Bootstrap.**

Idealmente não devemos alterar o core do Cake. Para isso é importante criar plugins, helpers, componentes, behaviors, etc. Mas para facilitar e apenas exemplificar iremos alterar o core e ao final indicarei um plugin que foi criado para esta finalidade.

Minha fonte de consulta para a documentação do Bootstrap 4 é o:  
<https://www.w3schools.com/bootstrap4/default.asp>

Vou criar o aplicativo financas  
Com o banco financas e duas tabelas despesas e receitas

Configurar a rota default para Despesas/index para facilitar o acesso  
Após configurar o banco gerar o código com o template default do bake:

```
bin/cake bake all despesas  
bin/cake bake all receitas
```

Visualizar pelo navegador

<http://localhost/financas>

Deixe esta aba do navegador aberta e vamos customizar o template default.  
Para facilitar, primeiro vamos customizar o template das views geradas e depois customizaremos o template do bake.

**Comecemos por baixar o Bootstrap 4:**

<https://getbootstrap.com/docs/4.3/getting-started/download/>

**Após descompactar copiamos os arquivos:**

```
bootstrap-reboot.min.css  
bootstrap.min.css  
bootstrap-grid.min.css
```

**Para webroot/css**

Manterei o CSS default e adicionarei o do Bootstrap abaixo para que tenha prioridade.

**E alteramos o src/Template/Layout/default.ctp para que fique assim:**

```
<?php  
$cakeDescription = 'CakePHP: the rapid development php framework';  
?>  
<!DOCTYPE html>  
<html>  
<head>  
  <?= $this->Html->charset() ?>  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>  
    CakePHP 3 com Bootstrap 4  
  </title>
```



```

<?= $this->Html->meta('icon') ?>

<?= $this->Html->css('base.css') ?>
<?= $this->Html->css('style.css') ?>
<?= $this->Html->css('bootstrap-reboot.css') ?>
<?= $this->Html->css('bootstrap.min.css') ?>
<?= $this->Html->css('bootstrap-grid.min.css') ?>

<?= $this->fetch('meta') ?>
<?= $this->fetch('css') ?>
<?= $this->fetch('script') ?>
</head>
<body>
  <nav class="top-bar expanded" data-topbar role="navigation">
    <ul class="title-area col-md-d columns">
      <li class="name">
        <h1><a href="">CakePHP 3 com BootStrap 4</a></h1>
      </li>
    </ul>
  </nav>
  <?= $this->Flash->render() ?>
  <div class="container clearfix">
    <?= $this->fetch('content') ?>
  </div>
  <footer>
  </footer>
</body>
</html>

```

## Alterar src/Template/Despesas/index.ctp

### Vamos por partes. Primeiro o sidebar:

```

<nav class="large-3 medium-4 columns" id="actions-sidebar">
  <ul class="side-nav">
    <li class="heading"><?= __('Actions') ?></li>

```

### Troquemos por:

```

<div class="actions columns col-lg-2 col-md-3">
  <h3><?= __('Actions') ?></h3>
  <ul class="nav nav-stacked nav-pills">

```

### Ao final o index.ctp de Despesas usando o Botstrap ficará assim:

```

<div class="actions columns col-lg-2 col-md-3">
  <h3><?= __('Actions') ?></h3>
  <ul class="nav nav-stacked nav-pills">
    <li><?= $this->Html->link(__('New Despesa'), ['action' => 'add']) ?></li>
    <li><?= $this->Html->link(__('List Receitas'), ['controller' => 'Receitas', 'action' => 'index']) ?></li>

```

```

        <li><?=$this->Html->link(__('New Receita'), ['controller' => 'Receitas', 'action' => 'add']) ?
    ></li>
</ul>
</div>
<div class="despesas index col-lg-10 col-md-9 columns">
    <h3><?=__('Despesas') ?></h3>
    <table class="table table-hover table-striped">
        <thead>
            <tr>
                <th scope="col"><?=$this->Paginator->sort('id') ?></th>
                <th scope="col"><?=$this->Paginator->sort('descricao') ?></th>
                <th scope="col"><?=$this->Paginator->sort('valor') ?></th>
                <th scope="col"><?=$this->Paginator->sort('mes') ?></th>
                <th scope="col"><?=$this->Paginator->sort('created') ?></th>
                <th scope="col"><?=$this->Paginator->sort('modified') ?></th>
                <th scope="col"><?=$this->Paginator->sort('receita_id') ?></th>
                <th scope="col" class="actions"><?=__('Actions') ?></th>
            </tr>
        </thead>
        <tbody>
            <?php foreach ($despesas as $despesa): ?>
                <tr>
                    <td><?=$this->Number->format($despesa->id) ?></td>
                    <td><?= h($despesa->descricao) ?></td>
                    <td><?=$this->Number->format($despesa->valor) ?></td>
                    <td><?= h($despesa->mes) ?></td>
                    <td><?= h($despesa->created) ?></td>
                    <td><?= h($despesa->modified) ?></td>
                    <td><?=$despesa->has('receita') ? $this->Html->link($despesa->receita->id,
['controller' => 'Receitas', 'action' => 'view', $despesa->receita->id]) : " ?></td>
                    <td class="actions">
                        <?=$this->Html->link(__('View'), ['action' => 'view', $despesa->id]) ?>
                        <?=$this->Html->link(__('Edit'), ['action' => 'edit', $despesa->id]) ?>
                        <?=$this->Form->postLink(__('Del'), ['action' => 'delete', $despesa->id], ['confirm'
=> __('Are you sure you want to delete # {0}?', $despesa->id)]) ?>
                    </td>
                </tr>
            </tbody>
        </table>
        <div class="paginator">
            <ul class="pagination">
                <?=$this->Paginator->first('<< ' . __('first')) ?>
                <?=$this->Paginator->prev('< ' . __('previous')) ?>
                <?=$this->Paginator->numbers() ?>
                <?=$this->Paginator->next(__('next') . '>') ?>
                <?=$this->Paginator->last(__('last') . '>>') ?>
            </ul>
            <p><?=$this->Paginator->counter(['format' => __('Page {{page}} of {{pages}}, showing
{{current}} record(s) out of {{count}} total')) ?></p>

```

```
</div>  
</div>
```

E assim procedi com as outras 3 views restantes até concluir.

**Crédito:**

As classes que estou usando do Bootstrap 4, as encontrei no plugin twbs-cake <https://github.com/elboletaire/twbs-cake-plugin>. Como este plugin é para o Bootstrap 3 eu converti algumas classes que mudaram para o Bootstrap 4 com ajuda do <https://www.w3schools.com/bootstrap4/default.asp>

**Detalhes**

**Plugin que usa o BootStrap 3 com CakePHP 3 e já com Twig:**

<https://github.com/ribafs/admin-br>



## 6 – Segurança

*Eu não falhei. Apenas descobri mil maneiras que não funcionam.*  
Thomas Edison

### Usando o Componente Security

#### Customizando

O uso do componente Security geralmente é feito no método `beforeFilter()` dos Controllers.

Apenas descomente no `initialize()` do `AppController.php`, veja.

```
public function initialize()
{
    parent::initialize();

    $this->loadComponent('RequestHandler', [
        'enableBeforeRedirect' => false,
    ]);
    $this->loadComponent('Flash');

    /*
     * Enable the following component for recommended CakePHP security settings.
     * see https://book.cakephp.org/3.0/en/controllers/components/security.html
     */
    $this->loadComponent('Security'); // DESCOMENTAR ESTA LINHA
}
```

#### Validações

Para que as validações do Cake surtam efeito precisamos criar nossos forms usando o `formHelper`.

#### Salt

Há alguns outros itens que podem ser configurados. Muitos desenvolvedores completam esta lista de itens, mas os mesmos não são obrigatórios para este tutorial. Um deles é definir uma sequência personalizada (ou “salt”) para uso em hashes de segurança.

A sequência personalizada (ou salt) é utilizada para gerar hashes de segurança. Se você utilizou o Composer, ele cuidou disso para você durante a instalação. Apesar disso, você precisa alterar a sequência personalizada padrão editando o arquivo **config/app.php**. Não importa qual será o novo valor, somente deverá ser algo difícil de descobrir:

```
'Security' => [
    'salt' => 'valor longo contendo mistura aleatória de
valores.',
],
```

Quando instalamos o Cake usando o composer, o composer já cuida disso e de outras tarefas para nós.

## Componente CSRF

Cross Site Request Forgery

Ao habilitar o componente CSRF você obtém proteção contra ataques. [CSRF](#) ou Cross Site Request Forgery (Falsificação de solicitação entre sites) é uma vulnerabilidade comum nas aplicações web. Esta brecha permite que o atacante capture e responda uma requisição, e as vezes envie dados através de uma requisição usando tags de imagem ou recursos em outros domínios.

O CsrComponent trabalha setando um cookie no navegador do usuário. Quando os formulários são criados com o [Cake\View\Helper\FormHelper](#), um input hidden é adicionado contendo o token CSRF. Durante o evento Controller.startup, se a requisição for POST, PUT, DELETE ou PATCH o componente irá comparar os dados da requisição e o valor do cookie. Se um deles estiver faltando ou os dois valores forem incompatíveis o componente lançará um `Cake\Network\Exception\InvalidCsrftokenException`.

Por default o componente CSRF vem habilitado no routes.php.

## Mais detalhes

<http://book.cakephp.org/3.0/en/controllers/components/security.html>

<http://book.cakephp.org/3.0/en/core-libraries/security.html>

<http://book.cakephp.org/3.0/en/controllers/components/csrftoken.html>

## 7 – Debug e Erros

*Só existe uma maneira de evitar críticas: não fazer nada, não dizer nada e não ser nada.*  
Aristóteles

### Debug/Depuração

Depuração é uma etapa inevitável e importante de qualquer ciclo de desenvolvimento. Ainda que o CakePHP não forneça nenhuma ferramenta que se conecte com qualquer IDE ou editor de texto, este oferece várias ferramentas que auxiliam na depuração e exibição de tudo que está sendo executado "por baixo dos panos" na sua aplicação.

### Depuração Básica

Você pode usar as funções `pr()` ou `debug()` caso queira verificar erros em código ou apenas verificá-los.

```
debug(mixed $var, boolean $showHtml = null, $showFrom = true)
```

A função `debug()` é uma função de escopo global que funciona de maneira similar a função do PHP, `print_r()`. A função `debug()` exibe os conteúdos de uma variável de diversas maneiras. Primeiramente, se você deseja exibir os dados no formato HTML, defina o segundo parâmetro como `true`. A função também exibe a linha e o arquivo de onde a mesma foi chamada.

A saída da função somente é exibida caso a variável `$debug` do core esteja definida com o valor `true`.

```
debug($query) Mostra o SQL e os parâmetros incluídos, não mostra resultados.  
debug($query->all()) Mostra a propriedade ResultSet retornado pelo ORM.  
debug($query->toArray()) Um caminho mais fácil para mostrar todos os resultados.  
debug(json_encode($query, JSON_PRETTY_PRINT)) Exemplo em JSON.  
debug($query->first()) Primeiro resultado obtido na query.  
debug((string)$query->first()) Mostra as propriedades de uma única entidade em JSON.
```

Tente isto na camada Controller: `debug( $this->{EntidadeNome}->find()->all() );`

### Tratando erros

**Cada SGBD tem seus próprios códigos de erro que podem ser capturados pelo PDO.**

### Dicas sobre erros no Cake

Criar o arquivo `src/Template/Error/pdo_error.ctp`, contendo

```
<?php  
use Cake\Utility\Debugger;
```

```
?>
<h2>Erro no Cadastro</h2>
<p class="error">
  <strong>Erro: Grupo não existe na tabela estrangeira<br></strong>

  <?php
    if($error->getCode() == 23503){
      print "Este grupo não existe na tabela estrangeira.<br>Caso tenha realmente digitado de
forma correta<br>e queira adicionar cadastre-o na tabela estrangeira<br>primeiro e depois cadastre
o material com ele!";
    }
    //print $message;
  ?>
</p>
```

Tratamento de erro é uma missão nobre em aplicativos.

Aplicativos onde o programador prevê erros em todas as situações onde ele imagina que pode haver erro são aplicativos mais robustos. Os softwares de teste como o PHPUnit tem a missão de ajudar com isso.

### **Detalhes**

<https://book.cakephp.org/3.0/en/development/debugging.html>



## 8 – Detalhes sobre Models

*Nas adversidades, uns desistem, enquanto outros batem recordes.*

Ayrton Senna

O Model representa a primeira letra do MVC.

Models (Modelos) são as classes que servem como camada de negócio na sua aplicação. Isso significa que eles devem ser responsáveis pela gestão de quase tudo o que acontece em relação a seus dados, sua validade, interações e evolução do fluxo de trabalho de informação no domínio do trabalho.

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 estendido para suportar fontes de dados alternativas.

O ORM do Cakephp toma emprestadas ideias e conceitos dos padrões ActiveRecord e Datamapper. Isso permite criar uma implementação híbrida que combina aspectos de ambos padrões para criar uma ORM rápida e simples de utilizar.

Objetos Table são usados para ser interface para coleções de objetos, ou seja, tabelas. O objeto Entity provê uma interface para um objeto individual, ou seja registros.

### **Atributos do Model**

Indicar nome de configuração do database

```
public $useDbConfig = 'alternate';
```

Prefixo das Tabelas

```
public $tablePrefix = 'alternate_';
```

Nome do campo Primary Key

```
public $primaryKey = 'example_id';
```

Display field

Nomes de campos em tabelas para tabelas relacionadas

Convenção: title ou name

Se diferente indicar no model com:

```
public $displayField = 'nomedocampo';
```

Tabela Default

```
public $useTable = 'users';
```

Caso receba a mensagem:

Erro: Table users for model User was not found in datasource default.

Isso aconteceu comigo sempre que instalei o o plugin Cakept\_br.

Adicione a variável \$use/Table contendo o nome da tabela ao model User

```
public $useTable = 'usuarios';
```

**Método \_setPassword()** - este método cria um hash da senha do usuário antes que seja salva e antes que seja validada.

## Lógica de negócios

A codificação mostrada abaixo, neste capítulo, sobre os models e o ORM, que é o padrão que o CakePHP usa para lidar com bancos de dados, é muito importante para codificar a lógica de negócios de muitos aplicativos. As operações de CRUD, que são as mais básicas, podemos criar usando o bake, mas a lógica específica de alguns aplicativos somos nós que precisaremos criar. Então devemos criá-la no model, para que esteja disponível nos controllers e possa ser solicitada pelas views.

Um exemplo deste tipo de lógica é a usada no aplicativo de exemplo Finanças, no capítulo 13.5, onde temos que retornar a soma das despesas de um certo mês, somar as receitas do mesmo mês e devolver ambas e também o saldo. O bake não faz isso, somente faz as operações básicas de CRUD, somos nós que devemos fazer manualmente e sempre irá depender de cada aplicativo como faremos isso. Por isso é muito importante conhecer como o CakePHP lida com model, controller, view e cia além do CRUD. Me parece que o pontapé inicial recomendado é seguir os tutoriais de criação de aplicativos existentes no site oficial. Depois ir em frente devorando a documentação e todo o material deste livro foi elaborado para colaborar com esta empreitada.

## Detalhes

<https://book.cakephp.org/3.0/en/orm.html>

## 8.1 – Validações

*Escolhe um trabalho de que você goste e não terá que trabalhar nem um dia na sua vida.*  
Confúcio

*(Mas se não encontrar um de que você goste, empenhe-se no que você tem no momento e fique atento para encontrar um de que gosta)*

Ribamar FS

A validação no Cake é feita no Model, mais especificamente na classe Table

Exemplo, o nosso aplicativo cliente:

`src\Model\Table\CientesTable.php`

```
...
public function validationDefault(Validator $validator)
{
    $validator
        ->integer('id')
        ->allowEmptyString('id', 'create');

    $validator
        ->scalar('nome')
        ->maxLength('nome', 45)
        ->requirePresence('nome', 'create')
        ->allowEmptyString('nome', false);

    $validator
        ->email('email')
        ->requirePresence('email', 'create')
        ->allowEmptyString('email', false);

    $validator
        ->date('data_nasc')
        ->requirePresence('data_nasc', 'create')
        ->allowEmptyDate('data_nasc', false);

    $validator
        ->scalar('cpf')
        ->maxLength('cpf', 11)
        ->requirePresence('cpf', 'create')
        ->allowEmptyString('cpf', false);

    return $validator;
}

/**
```

```

* Returns a rules checker object that will be used for validating
* application integrity.
*
* @param \Cake\ORM\RulesChecker $rules The rules object to be modified.
* @return \Cake\ORM\RulesChecker
*/
public function buildRules(RulesChecker $rules)
{
    $rules->add($rules->isUnique(['email']));

    return $rules;
}
...

```

O método `validationDefault()` diz ao CakePHP como validar seus dados quando o método `save()` no action no controller for solicitado.

### Observações:

- Vejamos o campo nome:
  - >maxLength('nome', 45)
  - >requirePresence('nome', 'create')
  - >allowEmptyString('nome', false);

De onde o Cake tirou este tamanho máximo de 45?  
Da nossa tabela. Veja que lá temos `varchar(45)`.

De onde ele tirou que deve requerer a presença na criação de novos registros?  
Também da tabela, onde viu `NOT NULL`, o que assegura a linha abaixo para não permitir vazio.

Desde que tenhamos usado o método `Cake\View\Helper\FormHelper::input()` do `FormHelper` para criar nossos elementos, nossas mensagens de alerta da validação serão exibidas automaticamente.

### Validação para o CPF

Baixar o repositório para a validação de:  
<https://github.com/gspaiva/cpfvalidator>

Ajustar num campo cujo nome seja "cpf" para que nossa `ClientesModel` seja:

```

<?php
namespace App\Model\Table;

use Cake\ORM\Query;
use Cake\ORM\RulesChecker;
use Cake\ORM\Table;
use Cake\Validation\Validator;

```

```

class ClientesTable extends Table
{
    public function initialize(array $config)
    {
        parent::initialize($config);

        $this->setTable('clientes');
        $this->setDisplayField('id');
        $this->setPrimaryKey('id');
    }

    public function validationDefault(Validator $validator)
    {
        $validator
            ->integer('id')
            ->allowEmptyString('id', 'create');

        $validator
            ->scalar('nome')
            ->maxLength('nome', 45)
            ->requirePresence('nome', 'create')
            ->allowEmptyString('nome', false);

        $validator
            ->email('email')
            ->requirePresence('email', 'create')
            ->allowEmptyString('email', false);

        $validator
            ->date('data_nasc')
            ->requirePresence('data_nasc', 'create')
            ->allowEmptyDate('data_nasc', false);

        $validator
            ->scalar('cpf')
            ->maxLength('cpf', 11)
            ->requirePresence('cpf', 'create')
            ->allowEmptyString('cpf', false);

        $validator
            ->add('cpf', 'custom',
                ['rule'=>
                    function($cpf)
                    {
                        // Verifica se um número foi informado
                        if(empty($cpf)) {
                            return false;
                        }

                        // Elimina possivel mascara

```

```

$cpf = preg_replace('[^0-9]', '', $cpf);
$cpf = str_pad($cpf, 11, '0', STR_PAD_LEFT);

// Verifica se o numero de digitos informados é igual a 11
if (strlen($cpf) != 11) {
    return false;
}
// Verifica se nenhuma das sequências invalidas abaixo foi digitada
else if ($cpf == '0000000000' ||
    $cpf == '1111111111' ||
    $cpf == '2222222222' ||
    $cpf == '3333333333' ||
    $cpf == '4444444444' ||
    $cpf == '5555555555' ||
    $cpf == '6666666666' ||
    $cpf == '7777777777' ||
    $cpf == '8888888888' ||
    $cpf == '9999999999') {
    return false;
}
// Calcula os digitos verificadores para verificar se o cpf é valido
} else {

    for ($t = 9; $t < 11; $t++) {

        for ($d = 0, $c = 0; $c < $t; $c++) {
            $d += $cpf{$c} * (($t + 1) - $c);
        }
        $d = ((10 * $d) % 11) % 10;
        if ($cpf{$c} != $d) {
            return false;
        }
    }

    return true;
}
},
// Caso retorne falso ele vai retornar uma mensagem falando que é inválido
'message'=>'=== CPF INVÁLIDO ==='
]);
return $validator;
}

public function buildRules(RulesChecker $rules)
{
    $rules->add($rules->isUnique(['email']));

    return $rules;
}
}

```

## Customizando as mensagens de erro do Cake

Caso ache necessário pode customizar o design das mensagens de erro.

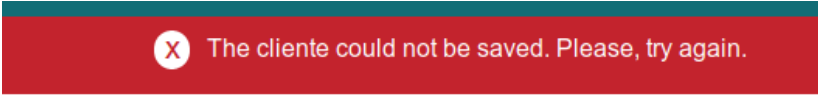
Edite o arquivo:

webroot/css/style.css

E altere o código de do seletor **.form .error-message** para este:

```
.form .error-message {
  display: block;
  padding: 0.375rem 0.5625rem 0.5625rem;
  margin-top: -1px;
  margin-bottom: 1rem;
  font-size: 1rem;
  font-weight: normal;
  font-style: italic;
  color: #FF0000;
}
```

Agora quando alguém tentar inserir um cpf inválido ou alterar um existente para outro inválido receberá:

A red banner with a white 'X' icon on the left and the text 'The cliente could not be saved. Please, try again.' in white.

**X** The cliente could not be saved. Please, try again.

### Add Cliente

Nome \*

Ribamar FS

Email \*

riba@ribaf2.com

Data Nasc \*

2019

May

16

Cpf \*

123213

*CPF Inválido*

## Outras Validações Customizadas

```
$validator->add('nascimento',[
    'notEmptyCheck'=>[
        'rule'=>[$this,'notEmptyNascimento'],
        'provider'=>'table',
        'message'=>'Favor selecionar uma data de nascimento'
    ]
]);

return $validator;
}

public function notEmptyNascimento($value,$context){
    if(empty($context['data']['nascimento'])) {
        return false;
    } else {
        return true;
    }
}
```

## Validação dos Forms

Para tirar proveito dos recursos de validação do Cake, você vai precisar usar o Form helper em suas views para criar formulários e nunca criar diretamente código HTML. O [Cake\View\Helper\FormHelper](#) está disponível por padrão em todas as views pelo uso do `$this->Form`.

Basta usar `$this->Form->...`

## Cuidados com o banco de dados

Uma preocupação com a segurança deve levar a um cuidado com a criação das tabelas no banco de dados com vários detalhes:

- Relacionamentos sempre que for importante
- Tipo de dados de cada campo selecionado com critério
- Tamanho dos campos
- Quantidade de campos
- Constraints de cada campo

## Criando nossas próprias regras de validação

O sistema de validação do Cake é rico e flexível e ainda permite que criemos nossas próprias regras de validação. Uma forma simples é usando Expressões Regulares.

Desde que estejamos usando o método `Cake\View\Helper\FormHelper::control()` do `FormHelper` para criar os elementos do form, nossas mensagens de erro de validação serão mostradas automaticamente.



Para tirar proveito dos recursos de validação do Cake, você vai precisar usar o Form helper em suas views.

## Validações Customizadas

### Validação de senha:

```
/**
 * Checks password for a single instance of each:
 * number, uppercase, lowercase, and special character
 *
 * @param type $password
 * @param array $context
 * @return boolean
 */
public function checkCharacters($password, array $context)
{
    // number
    if (!preg_match("#[0-9]#", $password)) {
        return false;
    }
    // Uppercase
    if (!preg_match("#[A-Z]#", $password)) {
        return false;
    }
    // lowercase
    if (!preg_match("#[a-z]#", $password)) {
        return false;
    }
    // special characters
    if (!preg_match("#\W+#", $password) ) {
        return false;
    }
    return true;
}

$validator
->requirePresence('password', 'create')
->notEmpty('password', 'You must enter a password', 'create')
->add('password', [
    'length' => [
        'rule' => ['minLength', 8],
        'message' => 'Passwords must be at least 8 characters long.',
    ]
])
->add('password', 'custom', [
    'rule' => [$this, 'checkCharacters'],
    'message' => 'The password must contain 1 number, 1 uppercase, 1 lowercase, and 1
special character'
]);
```

## Validação Condicional

Baseado no Entity

```
$rules->add(function ($entity, $options) use($rules) {
    if ($entity->role == 'admin') {
        $rule = $rules->existsIn('user_id', 'Admins');

        return $rule($entity, $options);
    }
    if ($entity->role == 'user') {
        $rule = $rules->existsIn('user_id', 'Users');

        return $rule($entity, $options);
    }

    return false;
}, 'userExists');
```

**Detalhes em**

<https://book.cakephp.org/3.0/en/orm/validation.html>

<https://book.cakephp.org/3.0/pt/orm/validation.html>

## 8.2 - O básico sobre o ORM

*Fazer as coisas certas é mais importante do que fazer bem feitas as coisas.*

Peter Drucker

Criar um novo aplicativo chamado **cake\_orm** e um banco também **cake\_orm**, com duas tabelas, relacionadas de acordo com a convenção do CakePHP: `articles` e `comments` para nossos testes, com o script abaixo:

*-- Primeiro, criamos a tabela articles*

```
CREATE TABLE articles (  
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
  title VARCHAR(50),  
  body TEXT,  
  created DATETIME DEFAULT NULL,  
  modified DATETIME DEFAULT NULL  
);
```

```
INSERT INTO `articles` (`id`, `title`, `body`, `created`, `modified`) VALUES  
(1, 'Titulo1', 'Corpo do primeiro artigo', '2019-06-01 10:07:39', NULL),  
(2, 'Titulo2', 'Corpo do segundo artigo.', '2019-06-01 10:07:39', NULL),  
(3, 'Titulo3', 'Corpo do terceiro artigo.', '2019-06-01 10:07:39', NULL),  
(4, 'Titulo4', 'Corpo do quarto artigo', '2019-06-02 00:00:00', NULL),  
(5, 'Titulo4', 'Corpo do quarto artigo', '2019-06-02 00:00:00', NULL);
```

```
CREATE TABLE comments (  
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
  comment VARCHAR(50),  
  created DATETIME DEFAULT NULL,  
  modified DATETIME DEFAULT NULL,  
  article_id int not null  
);
```

```
INSERT INTO comments (comment,created, article_id) VALUES ('Primeiro comantário', NOW(),  
1),  
( 'Segundo comantário', NOW(), 2),  
( 'Terceiro comantário', NOW(), 3);
```

**Configurar o aplicativo** para este banco, setar o routes para `Articles/index` e gerar os CRUDs com o `bake`:

```
bin/cake bake all articles
```

```
bin/cake bake all comments
```

## Exemplo de uso do find()

Num método, pode ser o initialize() do model ArticlesTable.php

list - retorna pares com chave e valor

```
$rets = $this->find('list', ['order'=>'id ASC']);
foreach($rets as $id=>$title){
    echo "$id : $title<br>";
}
exit;
```

Num método, pode ser o index() do controller ArticlesController.php

```
$rets = $this->Articles->find('list', ['order'=>'id ASC']);
foreach($rets as $id=>$title){
    echo "$id : $title<br>";
}
exit;
```

**Obs.:** observe que a diferença entre os usos no model ou no controller é que no controller precisamos usar Articles e no model não.

## Recebendo a conexão default:

Abra o src/Model/Table/ArticlesTable.php

Adicione ao início, abaixo dos existentes:  
use Cake\Datasource\ConnectionManager;

Crie uma nova função:

```
public function artigos(){
    $conn = ConnectionManager::get('default');
    $artigos = $conn->execute("select * from articles")->fetchAll();
    return $artigos;
}
```

Execute no initialize, ao final():

```
//debug($this->artigos()[0]);exit;
//debug($this->artigos()[0][2]);exit;
//debug($this->artigos()[0][2]);exit;
//debug($this->artigos()[2][2]);exit;
```

```
$arts = $this->artigos();
```

```

foreach($arts as $chave=>$valor){
    print 'Artigo';
    foreach($valor as $val){
        print $val.'  
';
    }
}
exit;

```

Chame pelo navegador para ver o resultado  
[http://localhost/cake\\_orm](http://localhost/cake_orm)

## Executar consultas SQL

Sempre usar no início, para dar suporte  
use Cake\Datasource\ConnectionManager;

Execute o código abaixo no initialize() da ArticlesTable

```

$conn = ConnectionManager::get('default');
$results = $conn
    ->execute('SELECT * FROM articles WHERE id = :id', ['id' => 1])
    ->fetchAll('assoc');
//debug($results);exit;
foreach($results as $chave=>$valor){
    foreach($valor as $val){
        print $val.'  
';
    }
}
exit;

```

Chame pelo navegador para ver o resultado

## Mais um exemplo:

Adicione ao final do initialize() do ArticlesTable

```

$conn = ConnectionManager::get('default');
$results = $conn
    ->execute(
        'SELECT * FROM articles WHERE title = :title', ['title'=>'Título1']
    )
    ->fetchAll('assoc');
debug($results);exit;
foreach($results as $chave=>$valor){
    foreach($valor as $val){
        print $val.'  
';
    }
}
exit;

```

Sempre Chame pelo navegador para ver o resultado

## Usando query builder

Adicione ao final do initialize()

```
$conn = ConnectionManager::get('default');
$results = $conn
    ->newQuery()
    ->select('*')
    ->from('articles')
    ->where(['title >' => 'Título1'])
    ->order(['title' => 'DESC'])
    ->execute()
    ->fetchAll('assoc');

//debug($results);exit;
foreach($results as $chave=>$valor){
    foreach($valor as $val){
        print $val.<br>;
    }
}
exit;
```

Chame pelo navegador

## Insert

```
$connection = ConnectionManager::get('default');
$connection->insert('articles', [
    'title' => 'Título4',
    'body' => 'Corpo do quarto artigo',
    'created' => '2019-06-02',
], ['created' => 'datetime']);
```

Confira no banco de dados.

## Update

```
$connection = ConnectionManager::get('default');
$connection->update('articles', ['title' => 'Título6'], ['id' => 6]);
```

Confira no banco

## Delete

```
use Cake\Datasource\ConnectionManager;
$connection = ConnectionManager::get('default');
$connection->delete('articles', ['id' => 5]);
Confira no banco
```

## Transações

```
$conn->begin();
$conn->update('articles', ['title' => 'Título6'], ['id' => 1]);
$conn->update('articles', ['title' => 'Título7'], ['id' => 1]);
//...
$conn->commit();
```

Confira no banco

## Outro

```
$conn = ConnectionManager::get('default');

$conn->transactional(function ($conn) {
    $conn->execute('UPDATE articles SET created = "2019-06-01 10:07:39" WHERE id = 6');
    $conn->execute('UPDATE articles SET created = "2019-06-01 10:07:39" WHERE id = 4');
});
```

exit;

Confira no banco

## Executando consultas

```
Cake\Database\Connection::query($sql)
Cake\Database\Connection::execute($sql, $params, $types)
```

```
$conn = ConnectionManager::get('default');
$stmt = $conn->query('UPDATE articles SET created = "2019-06-05" WHERE id = 6');
```

exit;

```
$conn = ConnectionManager::get('default');
$stmt = $conn->execute(
    'UPDATE articles SET created = ? WHERE id = ?', ['2019-06-05', 2]
);
```

Confira no banco

## Muito mais em:

<https://book.cakephp.org/3.0/pt/orm/database-basics.html>





## 8.3 - Retornando dados do banco

*No que diz respeito ao empenho, ao compromisso, ao esforço, à dedicação, não existe meio termo. Ou você faz uma coisa bem feita ou não faz.*

Ayrton Senna

class Cake\ORM\Table

Enquanto os objetos Table fornecem uma abstração em torno de um "repositório" ou coleção de objetos, quando você consulta registros individuais, obtém objetos Entity.

Quando o ORM foi implementado no CakePHP, era muito difícil depurar os resultados obtidos. Agora existem muitas formas fáceis de inspecionar os dados retornados pelo ORM.

- debug(\$query) Mostra o SQL e os parâmetros incluídos, não mostra resultados.
- debug(\$query->all()) Mostra a propriedade ResultSet retornado pelo ORM.
- debug(\$query->toArray()) Um caminho mais fácil para mostrar todos os resultados.
- debug(json\_encode(\$query, JSON\_PRETTY\_PRINT)) Exemplo em JSON.
- debug(\$query->first()) Primeiro resultado obtido na query.
- debug((string)\$query->first()) Mostra as propriedades de uma única entidade em JSON.

### As opções suportadas por find() são:

- conditions provê acesso direto na cláusula Where.
- limit Limite o número de resultados.
- offset Uma página que você quer. Use page para cálculo simplificado.
- contain defina uma associação para carregar.
- fields Quais campos você deseja carregar somente? Quando carregar somente alguns campos o lembre-se dos plugins, callbacks.
- group adicione um GROUP BY. muito usado para funções agregadas.
- having adicionar HAVING.
- join Defina um Join específico.
- order Ordenar resultados por.

Outras opções fora dessa lista, serão passadas para o beforeFind ou outras funções de tratamento, onde podem ser usados para tratar a consulta a sua maneira. Pode usar o método getOptions() no objeto para retornar as opções utilizadas. Quando uma consulta for passada para o controller, recomendamos uma leitura sobre consultas personalizadas em Personalizando Métodos de Consulta. Usando métodos de consultas personalizados, você terá um melhor reuso de seu código, e ficará fácil para testar a sua maneira.

### Detalhes

<https://book.cakephp.org/3.0/en/orm/retrieving-data-and-resultsets.html>



## 8.4 - Table Objects

*Tente uma, duas, três vezes e se possível tente a quarta, a quinta e quantas vezes for necessário. Só não desista nas primeiras tentativas, a persistência é amiga da conquista. Se você quer chegar onde a maioria não chega, faça o que a maioria não faz.*

Bill Gates

Provê acesso para a coleção de entities/registros armazenados em uma tabela específica. Para cada tabela de um aplicativo em CakePHP devemos ter uma classe Table associada a ela que é usada para interagir com a referida tabela.

Antes de começar a usar Table Objects precisa garantir que tem configurada uma conexão com o banco de dados.

A classe Table fica em src/Model/Table.

**Por convenção objetos Table** devem usar uma tabela que é idêntica a versão em minúsculas do nome da Table. Exemplos:

ArticlesTable - artigos

BlogPostsTable - blog\_posts

**Caso queira usar um nome de tabela que não segue esta convenção** deve setar seu nome no método setTable() no método initialize() da classe Table:

```
$this->setTable('nomeTabela');
```

O ORM também espera que cada tabela tenha uma chave primária e que seu nome seja 'id'. **Caso queira usar uma PK com nome diferente de "id" use o método setPrimaryKey()** também no initialize() da Table.

```
$this->setPrimaryKey('cpf');
```

### **Classes Entity usadas em classe Table.**

Por default table objects usam uma classe Entity baseada na convenção de nomes. Por exemplo, se uma Table tem nome ClientesTable então a entity respectiva será chamada de Cliente, no singular, visto que liga com apenas um registro por vez. Caso precise mudar esta convenção use o método setEntityClass() no initialize da Table:

Para Cake com versão anterior a 3.4.0:

```
$this->entityClass('App\Model\Entity\PO');
```

Superior a 3.4.0:

```
$this->setEntityClass('App\Model\Entity\PO');
```

## Instância de uma Table:

Antes de começar a consultar uma tabela precisamos ter uma instância da mesma. Podemos fazer isso usando a classe TableRegistry:

```
// Em um método de controller ou de uma table adicione a linha  
use Cake\ORM\TableRegistry;
```

## Lifecycle Callbacks/Eventos

Os eventos são úteis se você deseja conectar-se ao ORM e adicionar lógica sem subclassificar ou substituir métodos. Ouvintes de eventos podem ser definidos em Tables ou classes de Behavior. Você também pode usar o gerenciador de eventos de uma tabela para vincular os ouvintes. São semelhantes aos eventos dos controllers e componentes.

## Lista de eventos

- Model.initialize
- Model.beforeMarshal
- Model.beforeFind
- Model.buildValidator
- Model.buildRules
- Model.beforeRules
- Model.afterRules
- Model.beforeSave
- Model.afterSave
- Model.afterSaveCommit
- Model.beforeDelete
- Model.afterDelete
- Model.afterDeleteCommit

## Alguns eventos

initialize()

```
Cake\ORM\Table::initialize(Event $event, ArrayObject $data, ArrayObject $options)
```

O evento Model.initialize é disparado após o método construtor.

## beforeFind

```
Cake\ORM\Table::beforeFind(Event $event, Query $query, ArrayObject $options, $primary)
```

The Model.beforeFind event is fired before each find operation. By stopping the event and supplying a return value you can bypass the find operation entirely.

### **buildValidator**

Cake\ORM\Table::buildValidator(Event \$event, Validator \$validator, \$name)

O evento Model.buildValidator é disparado quando o \$name validator é criado.

### **buildRules**

Cake\ORM\Table::buildRules(Event \$event, RulesChecker \$rules)

O evento Model.buildRules é disparado após uma instância de rules ser criada e após o método buildRules da Table ser chamado.

### **beforeRules**

Cake\ORM\Table::beforeRules(Event \$event, EntityInterface \$entity, ArrayObject \$options, \$operation)

O evento Model.beforeRules é disparado antes de uma entity ter a rules aplicada.

### **afterRules**

Cake\ORM\Table::afterRules(Event \$event, EntityInterface \$entity, ArrayObject \$options, \$result, \$operation)

O evento Model.afterRules é disparado após uma entity ter a rules aplicada.

### **beforeSave**

Cake\ORM\Table::beforeSave(Event \$event, EntityInterface \$entity, ArrayObject \$options)

O evento Model.beforeSave é disparado antes de cada entity ser salva.

### **afterSave**

Cake\ORM\Table::afterSave(Event \$event, EntityInterface \$entity, ArrayObject \$options)

O evento Model.afterSave é disparado após uma entity ser salva.

## **afterSaveCommit**

Cake\ORM\Table::afterSaveCommit(Event \$event, EntityInterface \$entity, ArrayObject \$options)

O evento Model.afterSaveCommit é disparado após a transação em que a operação save é embutida ser committed.

## **beforeDelete**

Cake\ORM\Table::beforeDelete(Event \$event, EntityInterface \$entity, ArrayObject \$options)

O evento Model.beforeDelete é disparado antes que uma entity seja deletada.

## **afterDelete**

Cake\ORM\Table::afterDelete(Event \$event, EntityInterface \$entity, ArrayObject \$options)

O evento Model.afterDelete é disparado após uma entity ser deletada.

## **afterDeleteCommit**

Cake\ORM\Table::afterDeleteCommit(Event \$event, EntityInterface \$entity, ArrayObject \$options)

O evento Model.afterDeleteCommit é disparado após a transação em que a operação delete está embutida for committed.

## **Prioridades nos callbacks**

Ao usar eventos em suas Tables e Behaviors, esteja ciente da prioridade a que os ouvintes de pedidos estão conectados. Eventos de Behaviors são anexados antes dos eventos da Tabela. Com as prioridades padrão, isso significa que retornos de chamada de Behavior são acionados antes do evento de Table com o mesmo nome.

Por exemplo, se sua tabela estiver usando TreeBehavior, o método TreeBehavior::beforeDelete() será chamado antes do método beforeDelete() da sua Table, e você não poderá trabalhar com os nós filhos do registro que estão sendo excluídos no método da sua Table.

Podemos gerenciar eventos de prioridades de uma das duas maneiras:

Mude a prioridade de Behavior listeners usando a opção priority. Isto irá modificar a prioridade de todos os métodos callback no Behavior:

```
// In a Table initialize() method
$this->addBehavior('Tree', [
    // Default value is 10 and listeners are dispatched from the
    // lowest to highest priority.
```

```
'priority' => 2,  
]);
```

Modifique a prioridade em sua classe Table usando o método Model.implementedEvents(). Isso permite que você atribua uma prioridade diferente por método callback:

```
// In a Table class.  
public function implementedEvents()  
{  
    $events = parent::implementedEvents();  
    $events['Model.beforeDelete'] = [  
        'callable' => 'beforeDelete',  
        'priority' => 3  
    ];  
  
    return $events;  
}
```

## Sistema de Eventos

Criar aplicativos manuteníveis é uma ciência e uma arte. É bem conhecido que uma chave para ter um código de boa qualidade é tornar seus objetos fracamente acoplados e fortemente coesos ao mesmo tempo. Coesão significa que todos os métodos e propriedades de uma classe estão fortemente relacionados à própria classe e não está tentando fazer o trabalho que outros objetos deveriam estar fazendo, enquanto o acoplamento frouxo é a medida de quão pequena uma classe é conectada a objetos externos. e quanto essa classe depende deles.

### Mais detalhes em:

<https://book.cakephp.org/3.0/en/core-libraries/events.html>

<https://book.cakephp.org/3.0/en/orm/table-objects.html>





## 8.5 - Query Builder

*Dizem que a vida é para quem sabe viver, mas ninguém nasce pronto. A vida é para quem é corajoso o suficiente para se arriscar e humilde o bastante para aprender.*

Clarice Lispector

O Query Builder do ORM fornece uma interface fluente e simples de usar para criar e executar consultas. Ao compor consultas em conjunto, você pode criar consultas avançadas usando unions e subconsultas com facilidade.

Por "baixo dos panos", o Query Builder usa instruções prepare do PDO que protegem contra ataques de injeção de SQL.

A maneira mais fácil de criar um objeto Query é usar o método `find()` a partir de um objeto `Table`. Este método retornará uma consulta incompleta pronta para ser modificada. Você também pode usar o objeto de conexão de uma tabela para acessar o Query Builder de nível inferior que não inclui recursos de ORM, se necessário.

### Quando em um controller podemos usar `Articles` ao invés de `$articles`

*// Em ArticlesController.php*

```
use Cake\ORM\TableRegistry;
$query = TableRegistry::getTableLocator()->get('Articles')->find();
```

```
foreach ($query as $article) {
    print $article->id.'-'. $article->title.'<br>';
}
exit;
```

Veja pelo navegador

Sempre usar a linha abaixo em todos os exemplos

```
use Cake\ORM\TableRegistry;
```

```
$articles = TableRegistry::getTableLocator()->get('Articles');
$resultsIteratorObject = $articles
    ->find()
    ->where(['id >' => 1])
    ->order(['id ASC'])
    ->all();
```

```
foreach ($resultsIteratorObject as $article) {
    print $article->id.'-'. $article->title.'<br>';
}
exit;
```

Veja no navegador

Outro

```
$articles = TableRegistry::getTableLocator()->get('Customers');  
$resultsArray = $articles  
    ->find()  
    ->where(['id >' => 1])  
    ->toList();  
  
foreach ($resultsArray as $article) {  
    print $article->id.'-'. $article->name.'<br>';  
}  
  
//debug($resultsArray[0]->title);  
exit;
```

**Mais detalhes:**

<https://book.cakephp.org/3.0/en/orm/query-builder.html>

## 8.6 – Behaviors

*Seja muito bom, que eles não vão ter como ignorar você.*  
Steve Martin

Criar um aplicativo para controle de receitas e despesas pessoais no CakePHP 3 usando dois behaviors, um para somar as despesas de um mês e outro para somar as receitas de um mês.

### **Criar o banco finanças**

Importar o script do capítulo 15.3

**Criar o aplicativo padrão** usando o composer na pasta finanças

Configurar o banco em config/app.php

Configurar a rota default em config/routes.php para Despesas/index

### **Gerar o código com o bake**

bin/cake bake all despesas

bin/cake bake all receitas

### **Criar os dois behaviors:**

bin/cake bake behavior Despesas

bin/cake bake behavior Receitas

Editar o src/ModelBehavior/Despesas e deixar assim:

```
<?php
namespace App\Model\Behavior;

use Cake\ORM\Behavior;
use Cake\ORM\Table;
use Cake\ORM\TableRegistry;

class DespesasBehavior extends Behavior
{
    protected $_defaultConfig = [];

    public function despesasMes($mes){

        $despesas = TableRegistry::get('Despesas')->find();
        $res = $despesas->select(['total_sum' =>$despesas->func()->sum('Despesas.valor')]-
>where(['Despesas.mes' => $mes])->first();
        $total = $res->total_sum;
        return $total;
    }
}
```

Agora edite o behavior Receitas e deixe assim:

```
<?php
namespace App\Model\Behavior;

use Cake\ORM\Behavior;
use Cake\ORM\Table;
use Cake\ORM\TableRegistry;

class ReceitasBehavior extends Behavior
{
    protected $_defaultConfig = [];

    public function receitaMes($mes){
        $receitas = TableRegistry::get('Receitas')->find();
        $res = $receitas->select(['total_sum' => $receitas->func()->sum('Receitas.valor')]-
>where(['Receitas.mes' => $mes])
->first()); //perform the sum operation
        $total = $res->total_sum;
        return $total;
    }
}
```

**Edite o src/Model/Table/DespesasTable.php** e adicione logo abaixo do behavior Timestamp:

```
$this->addBehavior('Despesas');
```

**Também no ReceitasTable.php:**

```
$this->addBehavior('Receitas');
```

**Customizar o controller** src/Controller/DespesasController.php, adicionando ao início do arquivo:

```
use Cake\ORM\TableRegistry;
```

No início da classe:

```
public function despesasMes()
{
    $mes = $this->request->data('mes');
    $despesas = TableRegistry::get('Despesas');
    $total = $despesas->despesasMes($mes);
    $this->set('total', $total);
    $this->set('mes', $mes);
}
```

```
public function saldoMes()
{
    $mes = $this->request->data('mes');
```

```

        $this->loadModel('Receitas');// Como não é o model relacionado com este controller,
precisamos usar LoadModel para carregá-lo
        $receitas = $this->Receitas->receitaMes($mes);

        $despesas = TableRegistry::get('Despesas');
        $total = $despesas->despesasMes($mes);

        $saldo = $receitas - $total;
        $this->set('saldo',$saldo);
        $this->set('mes',$mes);
    }

```

### Adicionar ao src/Template/Despesas:

#### despesas\_mes.ctp

```

<?php ?>
<nav class="large-3 medium-4 columns" id="actions-sidebar">
    <ul class="side-nav">
        <h3><?= __('Despesas') ?></h3>
    <?php
print '<b>Mês: </b>'. $mes. '<br><b>Total: </b>'. $total;
?>
<br>
<br>
<br>
<li><?= $this->Html->link(__('Voltar'), ['action' => 'index']) ?></li>

</nav>

```

#### saldo\_mes.ctp

```

<?php ?>
<nav class="large-3 medium-4 columns" id="actions-sidebar">
    <ul class="side-nav">
        <h3><?= __('Despesas') ?></h3>
    <?php
print '<b>Mês: </b>'. $mes. '<br><b>Saldol: </b>'. $saldo;
?>
<br>
<br>
<br>
<li><?= $this->Html->link(__('Voltar'), ['action' => 'index']) ?></li>

</nav>

```

### Altere o index.ctp, adicionando ao início:

```

...
<nav class="large-3 medium-4 columns" id="actions-sidebar">
    <ul class="side-nav">

```

```

<li class="heading"><?=__('Actions') ?></li>
<li><?=$this->Html->link(__('New Despesa'), ['action' => 'add']) ?></li>
<li><?=$this->Html->link(__('List Receitas'), ['controller' => 'Receitas', 'action' => 'index']) ?
></li>
<li><?=$this->Html->link(__('New Receita'), ['controller' => 'Receitas', 'action' => 'add']) ?
></li>

<li><b>Despesas de um mês</b>
<?php
    echo $this->Form->create("Despesas",['url'=>'/despesas/despesas_mes']);
    echo $this->Form->input('mes');
    echo $this->Form->button('Submit');
    echo $this->Form->end();
?></li>
<li><b>Saldo de um mês</b>
<?php
    echo $this->Form->create("Saldo",['url'=>'/despesas/saldo_mes']);
    echo $this->Form->input('mes');
    echo $this->Form->button('Submit');
    echo $this->Form->end();
?></li>
</ul>
</nav>
...

```

### Chame pelo navegador:

<http://localhost/financas>

Use o pequeno form da esquerda para cálculo das Despesas de um mês. Informe o mês '05/2017'  
 Observe que o campo mês é do tipo string, com o formato mes/ano.

Também teste o Saldo de um mês.

Quando temos um model table com muitos métodos podemos criar behaviors para modular mais o código e mantê-lo mais organizado e de fácil manutenção.

### Behavior Timestamp

O behavior Timestamp que irá preencher automaticamente os campos created e modified da nossa tabela.

### Mais detalhes:

<https://book.cakephp.org/3.0/pt/orm/behaviors.html>

<https://book.cakephp.org/3.0/en/orm/behaviors.html>

## 9 – Detalhes sobre Views

*Eu gosto do impossível porque lá a concorrência é menor.*

Walt Disney

As Views representam o V de MVC. Elas são responsáveis por gerar a saída específica para a requisição do usuário. Geralmente a saída é em HTML, mas também pode ser em XML, JSON, PDF e outras.

O CakePHP conta com uma grande quantidade de classes embutidas para manipulação o mais comum dos cenários a renderizar:

- Para criar webservers XML ou JSON que podemos usar nas views
- Para servir arquivos protegidos, ou arquivos gerados dinamicamente
- Para criar múltiplas views com templates

### A AppView

A classe AppView é a classe View default. Em si ela estende a classe View e é definida em src\View\AppView.php assim:

```
<?php
namespace App\View;

use Cake\View\View;

class AppView extends View
{
}
```

Podemos usar AppView para carregar Helper (no método initialize()), mas geralmente os helpers criados e adicionados a src\View\Helper já ficam automaticamente disponíveis para as views dos templates.

### Views Template

A camada de views do CakePHP é a forma de nos comunicarmos com o usuário da aplicação.

A extensão default das views é .ctp (**C**ake**P**HP **T**emplate) e utiliza uma sintaxe alternativa em PHP para estruturas de controle e saída. Estes arquivos contem a lógica necessária para preparar os dados recebidos do controller no formato de apresentação que está pronto para a nossa audiência.

O CakePHP espera que os nossos templates sigam a convenção de nomenclatura onde o nome do template é a versão minúscula e grifada do nome do respectivo action do controller.

## Partes de uma View

A camada View no CakePHP pode ser formada por algumas partes. Cada parte tem diferentes usos e será coberta neste tutorial:

- **templates**: Templates são a parte da página que é única para o action começar a rodar; Elas são a parte principal da resposta do aplicativo. Arquivos de template do CakePHP são armazenados em **src/Template** dentro de uma pasta com o nome do controller correspondente e com extensão **.ctp**.

<https://book.cakephp.org/3.0/en/views.html#view-templates>

- **elements**: pequenos, reusáveis pedaços de código da view. Elements geralmente são renderizados dentro da view. Documentação oficial:

<http://book.cakephp.org/3.0/en/views.html#elements>

- **layouts**: são arquivos de templates que contém código de apresentação que envolve muitas interfaces na sua aplicação. Muitas views são renderizadas dentro de um layout. Os layouts incorporam templates: index, add, edit, login, etc.

Um layout é um conjunto de códigos encontrado ao redor das views. Múltiplos layouts podem ser definidos, e você pode alterar entre eles, mas agora, vamos usar o default, localizado em **src/Template/Layout/default.ctp**.

<http://book.cakephp.org/3.0/en/views.html#layouts>

- **helpers**: estas classes encapsulam lógica de views que são necessárias em muitos lugares na camada view. Entre outras coisas os helpers no CakePHP podem ajudar você a construir forms, criar funcionalidade AJAX, paginar modelo de dados, etc.

<http://book.cakephp.org/3.0/en/views/helpers.html>

- **cells**: estas classes oferecem uma miniatura tipo as características dos controllers para a criação de componentes UI auto contidos. Veja a documentação para mais informações:

<http://book.cakephp.org/3.0/en/views/cells.html>

## Dicas sobre Views

### Título do Aplicativo

- Adicionar ao AppController.php:

```
function beforeFilter(Event $event){  
    $this->set('title_for_layout','Controle de Finanças Pessoais');  
}
```

- Será usado no Layout/default.ctp, aqui assim:

```
<title>  
    <?= $cakeDescription ?>:  
    <?php echo $title_for_layout;?>  
</title>
```



E no body

...

```
<li class="name">
  <h1><a href=""><?php echo $title_for_layout;?></a></h1>
</li>
```

Isso mostrará no title do navegador a variável \$title\_for\_layout definida no ApplicationController e também no body, no h1.

### Trocando campo do tipo text por select:

```
$options = array("=>'Selecione','colgioelias' => 'Colégio do Elias', 'eliasmesada'=>'Elias
Mesada','eliasmerenda'=>'Elias Merenda','meire' => 'Meire Salário', 'coelce'=>'COELCE',
'condominio'=>'Condomínio Ferreira', 'gvt'=>'GVT',
'carnes'=>'Carnes','mercantil'=>'Mercantil','almoco'=>'Almoço
Trabalho','passagens'=>'Passagens','remedio'=>'Remédio');
```

Usando o \$options:

```
echo $this->Form->input('descricao', array('type'=>'select','label' => 'Descrição da
Despesa','options' => $options,'default'=>'0'));
```

### Algumas views típicas:

```
<!-- File: src/Template/Articles/add.ctp -->
```

```
<h1>Add Article</h1>
```

```
<?php
echo $this->Form->create($article);
echo $this->Form->control('title');
echo $this->Form->control('body', ['rows' => '3']);
echo $this->Form->button(__('Save Article'));
echo $this->Form->end();
?>
```

```
<!-- File: src/Template/Articles/edit.ctp -->
```

```
<h1>Edit Article</h1>
```

```
<?php
echo $this->Form->create($article);
echo $this->Form->control('title');
echo $this->Form->control('body', ['rows' => '3']);
echo $this->Form->button(__('Save Article'));
echo $this->Form->end();
?>
```

```
<!-- File: src/Template/Articles/view.ctp -->
```

```
<h1><?= h($article->title) ?></h1>
<p><?= h($article->body) ?></p>
<p><small>Created: <?= $article->created->format(DATE_RFC850) ?></small></p>
```

```
<!-- File: src/Template/Articles/index.ctp (edit links added) -->
```

```
<!-- File: src/Template/Articles/index.ctp (delete links added) -->
```

```
<h1>Blog articles</h1>
<p><?= $this->Html->link('Add Article', ['action' => 'add']) ?></p>
<table>
  <tr>
    <th>Id</th>
    <th>Title</th>
    <th>Created</th>
    <th>Actions</th>
  </tr>
```

```
<!-- Here's where we loop through our $articles query object, printing out article info -->
```

```
<?php foreach ($articles as $article): ?>
  <tr>
    <td><?= $article->id ?></td>
    <td>
      <?= $this->Html->link($article->title, ['action' => 'view', $article->id]) ?>
    </td>
    <td>
      <?= $article->created->format(DATE_RFC850) ?>
    </td>
    <td>
      <?= $this->Form->postLink(
        'Delete',
        ['action' => 'delete', $article->id],
        ['confirm' => 'Are you sure?'])
      ?>
      <?= $this->Html->link('Edit', ['action' => 'edit', $article->id]) ?>
    </td>
  </tr>
<?php endforeach; ?>
```

```
</table>
```

```
<!-- File: src/Template/Users/login.ctp -->
<div class="users form">
  <?= $this->Flash->render() ?>
  <?= $this->Form->create() ?>
  <fieldset>
    <legend><?= __('Please enter your username and password') ?></legend>
    <?= $this->Form->control('username') ?>
```

```
<?= $this->Form->control('password') ?>
</fieldset>
<?= $this->Form->button(__('Login')); ?>
<?= $this->Form->end() ?>
</div>
```

**Detalhes em**

<http://book.cakephp.org/3.0/en/views.html>





Customizando o css para exibir melhor nosso menu:

Mude a linha com a para:

```
a {  
    color: black;  
}
```

## Usando

Para usar adicione a linha no src/Template/Layout/default.ctp, como indicado abaixo:

```
<body>  
    <nav class="top-bar expanded" data-topbar role="navigation">  
        <?php echo $this->element('topmenu');?>  
    </nav>  
</body>
```

**Veja que é um element bem simples mas pode usar o conhecimento e criatividade para fazer algo melhor. Também pode ver um exemplo mais interessante no plugin admin-br.**

## Dicas sobre Element no Cake

Muitas aplicações possuem pequenos blocos de código de apresentação que precisam ser repetidos a cada página, às vezes em diferentes lugares no layout. O CakePHP ajuda você a repetir partes do seu website que precisam ser reutilizados. Estas partes reutilizáveis são chamadas de Elements (ou Elementos). Propagandas, caixas de ajuda, controles de navegação, menus extras, formulários de login e chamadas geralmente são implementadas como elements. Um element é basicamente uma mini-view que pode ser incluída em outras views, layouts e até mesmo em outros elements. Elements podem ser usados para criar uma view mais legível, colocando o processamento de elementos repetidos em seu próprio arquivo. Eles também podem ajudá-lo a reusar conteúdos fragmentados pela sua aplicação.

**Element** - são pequenos trechos de código que podem ser usados nas views para qualquer utilidade.

Um element pode ser acessado por qualquer view.

No local da view onde queremos que apareça o código do elemento inserimos:

```
echo $this->element('espacos');
```

Podemos passar informações para um element através do seu segundo parâmetro:

```
echo $this->element('helpbox', array(  
    "textoajuda" => "Oh, this text is very helpful."  
));  
echo $textoajuda;
```

São blocos de código tipo view reutilizáveis que criamos uma vez usamos várias vezes e em qualquer lugar do site (geralmente aplicados no layout)

**Detalhes** - <https://book.cakephp.org/3.0/en/views.html#elements>

## 9.2 – Layout

*A grande glória da vida não está em nunca cair, mas em se levantar a cada vez que cair.*

Nelson Mandela

Layout contém código de apresentação que é incorporado na view. Qualquer coisa que você deseje ver em todas as suas views deve ser colocada em um layout.

No CakePHP 3 o arquivo default de layout fica em  
src\Template\Layout\default.ctp

Se você deseja mudar a aparência da sua aplicação (cores, fontes e posições, imagens, etc) então o layout é o lugar para mexer.

Outros arquivos de layout devem ser criados em  
src\Template\Layout\

Aqui está o layout default.ctp do CakePHP 3.8.0 com leves modificações:

```
<!DOCTYPE html>
<html>
<head>
  <?=$this->Html->charset() ?>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>
    <?=$this->fetch('title') ?>
  </title>
  <?=$this->Html->meta('icon') ?>

  <?=$this->Html->css('base.css') ?>
  <?=$this->Html->css('style.css') ?>

  <?=$this->fetch('meta') ?>
  <?=$this->fetch('css') ?>
  <?=$this->fetch('script') ?>
</head>
<body>
  <nav class="top-bar expanded" data-topbar role="navigation">
    <ul class="title-area large-3 medium-4 columns">
      <li class="name">
        <h1><a href=""><?=$this->fetch('title') ?></a></h1>
      </li>
    </ul>
  </nav>
  <?=$this->Flash->render() ?>
  <div class="container clearfix">
    <?=$this->fetch('content') ?>
  </div>
</body>
```

```
</footer>
</body>
</html>
```

Podemos criar tantos layouts quantos desejarmos. Apenas os crie em:  
src\Template\Layout\

**E chame no action respectivo com:**

```
$this->viewBuilder()->layout('admin');
```

**Na view use:**

```
$this->layout = 'loggedin';
```

Veja no plugin [admin-br](#) o uso de dois layouts e sua chamada no ApplicationController.

Carregando num controller

```
public function initialize()
{
    $this->viewBuilder()->setLayout('admin');
}
```

Ou

```
if($loguser == 'user' || $loguser == 'manager'){
    $this->viewBuilder()->setLayout('CakeAclBr.default');
}else{
    $this->viewBuilder()->setLayout('CakeAclBr.admin');
}
```

**Layout** - contém código de apresentação da view. Tudo que vemos nas views está incorporado num layout.

Estão em src/Template/Layout

O cake já vem com um layout padrão, que é o default.ctp.

Como setar outro layout no controller. Exemplo admin.ctp:

```
public function initialize()
{
    $this->viewBuilder()-setLayout('admin');
    // Para plugin
    // $this->viewBuilder()-setLayout('AdminBr.admin');
}
```

**Detalhes**

<https://book.cakephp.org/3.0/en/views.html#layouts>



## 9.3 – Helper

*Você não pode mudar o vento, mas pode ajustar as velas do barco para chegar onde quer*  
Confúcio

Os helpers contém código de apresentação que são compartilhados entre muitas views, elements ou layouts.

**Veja a relação de helper nativos do CakePHP 3:**

- [Breadcrumbs](#)
- [Form](#)
- [Html](#)
- [Number](#)
- [Paginator](#)
- [Rss](#)
- [Session](#)
- [Text](#)
- [Time](#)
- [Url](#)

**Carregamos Helpers** no Cake declarando em classes views.

src\View

```
Class AppView extends View
{
    public function initialize()
    {
        parent::initialize();
        $this->loadHelper('Html');
        $this->loadHelper('Form');
        $this->loadHelper('Flash');
    }
}
```

### Usando o helper

Também podemos usar o método `beforeRender()` dos controllers para carregar Helpers.

```
class ArticlesController extends AppController
{
    public function beforeRender(Event $event)
    {
        parent::beforeRender($event);
    }
}
```

```

    $this->viewBuilder()->helpers(['MeuHelper']);
}
}

```

## Criando Helpers

Podemos criar classes Helper para usar em aplicações ou em plugins.

Os Helpers têm algumas convenções que nos ajuda se as seguirmos:

- Arquivos de classe Helper devem ficar em src/View/Helper. Exemplo:  
src/View/Helper/MeuHelper.php
- Classes helpers devem ser sufixadas com Helper. Exemplo:  
MeuHelper
- Quando referenciar classes helper deve omitir o sufixo Helper:  
\$this->loadHelper('Meu');

## Criando o esqueleto de um Helper chamado Mensagem com o bake:

```
cd /var/www/html/cliente
```

```
bin/cake bake helper Mensagem
```

Editar o src\View\Helper\MensagemHelper.php e adicionar a função msg(), assim:

```

<?php
namespace App\View\Helper;

use Cake\View\Helper;
use Cake\View\View;

class MensagemHelper extends Helper
{
    protected $_defaultConfig = [];

    public function msg($msg)
    {
        return '<h2>'.$msg.'</h2>';
    }
}

```

## Usando o Helper

Editar a view index.ctp e adicionar em algum lugar:

```
<?=$this->Mensagem->msg('Minha Mensagem Helper') ?>
```

Ao chamar no navegador verá a mensagem com título h2.

Um helper bem simples para mostrar o caminho das pedras.

## Máscaras no CakePHP

- Máscara na view index para o campo CPF  
Enquanto não descubro uma função do Cake que faz isso...

O CPF é um campo texto mas constituído somente de números. A listagem (index.ctp) mostra a relação de números sem nenhuma formatação. Ajudaria se formatássemos usando uma máscara adequada para CPF, ajudaria a visualizar.

Vamos fazer isso agora para a view Clientes/index.ctp:

Substitua a linha

```
<td><?php echo h($cliente['Cliente']['cpf']); ?>&nbsp;  </td>
```

Por este código:

```
<tbody>
  <?php foreach ($customers as $customer): ?>
<?php
$scpfmask1 = substr($customer->cpf, 0,3);
$scpfmask2 = substr($customer->cpf, 3,3);
$scpfmask3 = substr($customer->cpf, 6,3);
$scpfmask4 = substr($customer->cpf, 9,2);
$scpfmask = $scpfmask1.'!'.$scpfmask2.'!'.$scpfmask3.'-!'.$scpfmask4;
?>
  <tr>
    <td><?= $this->Number->format($customer->id) ?></td>
    <td><?= h($customer->name) ?></td>
    <td><?= h($customer->birthday) ?></td>
    <td><?= h($scpfmask) ?></td>
```

Com isso a listagem mostrará o CPF com a sua máscara.  
Caso existam registros eles não serão alterados, apenas os novos.

### Dicas sobre data e hora

Ano mínimo sendo 16 anos antes do atual e máximo sendo 100 anos antes do atual em actions add e edit

```
echo $this->Form->input('nascimento',['label' => 'Nascimento',
'dateFormat' => 'DMY',
'minYear' => date('Y') - 100,
'maxYear' => date('Y') - 16,
'empty' => [
'day' => 'Dia',
'month' => 'Mês',
'year' => 'Ano'
]
]);
```

**Detalhes em:**

<http://book.cakephp.org/3.0/en/views/helpers.html>

## 9.3.1 – formHelper

*Você nunca sabe a força que tem, até que a sua única alternativa é ser forte.*

Johnny Depp

Tradução simplificada e resumida do original em:

<http://book.cakephp.org/3.0/en/views/helpers/form.html>

```
class Cake\View\Helper\FormHelper(View $view, array $config = [])
```

O FormHelper faz a maior parte do trabalho pesado na criação de formulários. O FormHelper foca na criação de formulários de forma rápida, de uma forma que irá agilizar a validação, re-população e layout. O FormHelper também é flexível - ele vai fazer quase tudo por você, utilizando convenções, ou você pode usar métodos específicos para obter apenas o que você precisa.

### Iniciando um Formulário

```
Cake\View\Helper\FormHelper::create(mixed $model = null, array $options = [])
```

O primeiro método que você deve usar para tirar vantagem do FormHelper é o create(). Este método mostra uma tag de abertura de formulário.

Todos os parâmetros são opcionais. Caso create() seja chamado sem nenhum parâmetro, ele assume que você está construindo um form que submete para o controller atual, via URL atual. O método default para a submissão do form é o POST. Caso você chame o create dentro da view para UsersController::add(), com \$this->Form->create() você deve ver a seguinte saída na view renderizada:

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

Se create() é chamado sem parâmetros fornecidos, assume-se a construção de um formulário que submete dados via POST para a action add() (ou edit() no caso de um id estar incluído nos dados do formulário).

O argumento \$model é usado como forma de contexto. Existem vários contextos embutidos para formulário e você pode adicionar seu próprio, o que nós vamos cobrir na próxima seção. O provedores internos mapeiam para os seguintes valores de \$model:

- Uma instância de entidade ou um mapa de iterator para o EntityContext, neste contexto permite FormHelper trabalhar com os resultados do ORM embutido.
- Um array que contém a chave de esquema, mapeia para ArrayContext que permite a criação de estruturas de dados simples para construir forms contra.
- null e false mapeiam para o NullContext, esta classe de contexto simplesmente satisfizer a interface que a FormHelper requer. Este contexto é útil se você quiser construir um pequeno formulário que não requer persistência ORM.

Para criar um form para uma entity faça o seguinte:

```
// Se você está em /articles/add
// $article deve ser uma entidade vazia de Article.
echo $this->Form->create($article);
```

Isto irá postar os dados do formulário para o action add () de ArticlesController. No entanto, você também pode usar a mesma lógica para criar um formulário de edição. O FormHelper utiliza o objeto de entidade para detectar automaticamente se criará um form de adicionar ou editar. Se a entidade fornecido não é "novo", o formulário será criado como um formulário de edição. Se for novo será add.

Por exemplo, se navegar para

**<http://example.org/articles/edit/5>**, nós devemos fazer o seguinte:

```
// src/Controller/ArticlesController.php:
```

```
public function edit($id = null)
{
    if (empty($id)) {
        throw new NotFoundException;
    }
    $article = $this->Articles->get($id);
    // Save logic goes here
    $this->set('article', $article);
}
```

```
// View/Articles/edit.ctp:
```

```
// Since $article->isNew() is false, we will get an edit form
```

```
<?=$this->Form->create($article) ?>
```

### **Mudando o Método para o Form**

```
echo $this->Form->create($article, ['type' => 'get']);
```

Quando existir algum campo do tipo file, precisará ser assim:

```
echo $this->Form->create($article, ['type' => 'file']);
```

### **Configurando a URL para o Form**

```
echo $this->Form->create($article, ['url' => ['action' => 'login']]);
```

Caso o desejado form action não seja do controller atual, podemos especificar a completa URL para a action do form:

```
echo $this->Form->create(null, [
    'url' => ['controller' => 'Articles', 'action' => 'publish']
]);
```

Ou pode apontar para uma URL externa:

```
echo $this->Form->create(null, [  
    'url' => 'http://www.google.com/search',  
    'type' => 'get'  
]);
```

## Criando Controles para Forms

```
Cake\View\Helper\FormHelper::control(string $fieldName, array $options = [])
```

- `$fieldName` – O nome do campo do form `'Modelname.fieldname'`.
- `$options` - Um array opcional que pode incluir ambos [Options for Control](#), e options dos outros métodos (que o `control()` emprega internamente para gerar vários elementos HTML) como também atributos HTML válidos.

Em versões anteriores o Cake usava `$this->form->input()`, agora usa `$this->form->control()`.

O método `control()` permite gerar todos os controles/inputs do formulário. Esses controles incluirão uma div para quebra automática, um rótulo, um widget de controle e validação de erro, se necessário. Usando os metadados no contexto do formulário, esse método escolherá um tipo de controle apropriado para cada campo. Internamente `control()` usa os outros métodos do `FormHelper`.

Observe que enquanto os campos gerados pelo método `control()` são chamados genericamente de “inputs” nesta página, tecnicamente falando, o método `control()` pode gerar não apenas todos os elementos de tipo `input` do HTML, mas também outros elementos de formulário HTML (por exemplo, `selects`, `button`, `textarea`).

Por default o método `control()` deve empregar os seguintes widget templates:

```
'inputContainer' => '<div class="input {{type}}{{required}}">{{content}}</div>'  
'input' => '<input type="{{type}}" name="{{name}}"{{attrs}}/>'
```

Em caso de validação de erros ele também usa:

```
'inputContainerError' => '<div class="input {{type}}{{required}}  
error">{{content}}{{error}}</div>'
```

O tipo de controle criado (quando nós não provemos nenhum options adicional para especificar o tipo de elemento gerado) é inferido via introspecção do model e depende do tipo de dados do campo:

### Column Type - Resulting Form Field

string, uuid (char, varchar, etc.) - text

boolean, tinyint(1) - checkbox

decimal - number

float - number

integer - number

text - textarea

text, with name of password, passwd - password

text, with name of email - email

text, with name of tel, telephone, or phone - tel

date - day, month, and year selects  
datetime, timestamp - day, month, year, hour, minute, and meridian selects  
time - hour, minute, and meridian selects  
binary - file

O parâmetro `$options` segue sua escolha de um controle específico tipo caso necessite:

```
echo $this->Form->control('published', ['type' => 'checkbox']);
```

### **Exemplo de form:**

Crie uma nova view em um template, chamada form.ctp, contendo

```
<?php
    echo $this->Form->create($user);
    // The following generates a Text input
    echo $this->Form->control('username');
    // The following generates a Password input
    echo $this->Form->control('password');
    // Assuming 'approved' is a datetime or timestamp field the following
    //generates: Day, Month, Year, Hour, Minute
    echo $this->Form->control('approved');
    // The following generates a Textarea element
    echo $this->Form->control('quote');
    echo $this->Form->button('Add');
    echo $this->Form->end();
?>
```

### **Adicione um action ao controller correspondente, assim:**

```
public function form()
{
    $customers = $this->Customers;

    $this->set(compact('customers'));
}
```

Então chame pelo navegador assim:

<http://localhost/clientes/controllerNome/form>

Podemos especificar qualquer opção para o tipo do input e qualquer atributo HTML.



## Criar um Select

Se você deseja criar um campo select enquanto usando uma relação belongTo ou hasOne você pode adicionar o seguinte para seu UsersController (assumindo que User belongTo Group):

```
$this->set('groups', $this->Users->Groups->find('list'));
```

Então na view do Template:

```
echo $this->Form->input('group_id', ['options' => $groups]);
```

Para um select em uma associação belongsToMany Groups você pode adicionar o seguinte para seu UsersController:

```
$this->set('groups', $this->Users->Groups->find('list'));
```

Na view:

```
echo $this->Form->input('groups._ids', ['options' => $groups]);
```

Se o nome do model consiste em duas ou mais palavras, por exemplo, "UserGroup", ao passar os dados usando set() você deve nomear os seus dados com um formato pluralizado e camelCase como se segue:

```
$this->set('userGroups', $this->UserGroups->find('list'));
```

## Não usar input() para gerar submits

Para isso usar o método \View\Helper\FormHelper::submit().

## Options do Input

*\$options['type']*

```
echo $this->Form->input('field', ['type' => 'file']);  
echo $this->Form->input('email', ['type' => 'email']);
```

*\$options['label']*

```
echo $this->Form->input('name', [  
    'label' => 'The User Alias'  
]);
```

Outros:

```
echo $this->Form->input('name', [  
    'label' => [  
        'class' => 'thingy',  
        'text' => 'The User Alias'  
    ]  
]);
```

Tipo select:

```
$sizes = ['s' => 'Small', 'm' => 'Medium', 'l' => 'Large'];  
echo $this->Form->select('size', $sizes, ['default' => 'm']);
```

```
$options['value']
```

```
echo $this->Form->time('close_time', [  
    'value' => '13:30:00'  
]);
```

```
echo $this->Form->select('rooms', [  
    'multiple' => true,  
    // options with values 1 and 3 will be selected as default  
    'default' => [1, 3]  
]);
```

```
$options['empty']
```

```
echo $this->Form->select(  
    'field',  
    [1, 2, 3, 4, 5],  
    ['empty' => '(choose one)']  
);
```

```
$options['datetime']
```

```
echo $this->Form->input('time', [  
    'type' => 'time',  
    'interval' => 15  
]);
```

**Criando dois selects estáticos** para os campos controller e action em Permissions/add.ctp

```
$options =
```

```
['Customers'=>'Customers','Groups'=>'Groups','Users'=>'Users','Permissions'=>'Permissions','P  
roducts'=>'Products','ProductItems'=>'ProductItems','value'=>'Selecione'];
```

```
echo $this->Form->input('controller',['options'=>$options,'required'=>'false', 'class'=>'col-md-  
12','empty'=>'Selecione']);
```

```
$options2 = ['index'=>'index','add'=>'add','edit'=>'edit','view'=>'view','delete'=>'delete'];
```

```
echo $this->Form->input('action', ['options'=>$options2,'required'=>'false', 'class'=>'col-md-12',  
'empty'=>'Selecione']);
```

São estáticos, portanto sempre que precisar adicionar um novo controller ou action, precisa alterar este código.

**Para que Html->Link permita CSS:**

```
<? = $this->Html->Link('<span class="glyphicon glyphicon-plus"></span> Novo',  
    ['controller'=>'Bookmarks','action'=>'add'],
```

```
['class'=>'btn btn-primary pull-right']);
```

Assim ele mostrará o código `<span...</span>`

Para que permita o CSS, use:

```
<?= $this->Html->link('<span class="glyphicon glyphicon-plus"></span> Novo',  
['controller'=>'Bookmarks','action'=>'add'],  
['class'=>'btn btn-primary pull-right', 'escape'=>false]);
```

Observação:

classe Html, método link. Classe com inicial maiúscula e método tudo em minúsculas.

## Mudar o tipo de um campo

O Cake gerou um campo com o tipo textarea em um form.

Para mudar podemos fazer isso:

```
print $this->Form->input('url',['label', 'URL']);
```

Mudar para tipo text (campo texto) assim:

```
print $this->Form->input('url',['type'=>'text','label', 'URL']);
```

## Pegando os Erros do Formulário

Uma vez sido validado, o formulário pode recuperar seus próprios erros:

```
$errors = $form->errors();  
/* $errors contains  
[  
  'email' => ['A valid email address is required']  
]  
*/
```

## Form Helper

Form

`$this->Form->control()` - usado para criar elementos com o mesmo nome. Tem dois parâmetros:

Primeiro - nome do campo

Segundo - opcional, permite usar arrays com múltiplas opções

Para o CakePHP os campos do tipo data, datetime ou timestamp obrigatoriamente devem usar DEFAULT NULL:

nascimento date DEFAULT NULL

E também não podemos alterar sua validação para exigir preenchimento com notEmpty ou notBlank. Caso contrário o Cake não reconhece e não adiciona o registro.

src/Template/Cientes/add.ctp ou edit.ctp

Ano mínimo sendo 13 anos antes do atual e máximo sendo 100 anos antes do atual, ou seja, como estou em 2016 de 1916 até 2003

**Select múltiplo** (permite selecionar várias opções)

```
print $this->Form->input('pilot_ratings',[
    'type' => 'select',
    'class' => 'listbox',
    'size' => 5,
    'id' => 'pilot_ratings',
    'multiple' => 'multiple',
    'options' => [
        ['name' => 'Habilitación de Vuelo Nocturno Local', 'value' => '1'],
        ['name' => 'Habilitación Cat. II / Cat. III', 'value' => '2'],
        ['name' => 'Habilitación de Remolque de Planeador', 'value' => '5']
    ]
]);
```

src/Template/Cientes/add.ctp ou edit.ctp

### Dicas sobre data e hora

Para o CakePHP os campos do tipo data, datetime ou timestamp obrigatoriamente devem usar DEFAULT NULL:

nascimento date DEFAULT NULL

E também não podemos alterar sua validação para exigir preenchimento com notEmpty ou notBlank. Caso contrário o Cake não reconhece e não adiciona o registro.

src/Template/Cientes/add.ctp ou edit.ctp

Por padrão o Cake mostra apenas os anos de 2011 até 2021 na combo Ano.

Vamos alterar para que o ano mínimo seja 13 anos antes do atual e máximo seja 100 anos antes do atual, ou seja, como estou em 2016, que mostre de 1916 até 2003, mas isso deve ser pensado para atender ao requisito da tabela/aplicativo. No nosso caso, do DNOCS, devemos usar 18 anos antes, ou mais para o primeiro?

## Criar o select controller e o action

```
<?php
    $controls = ['Groups'=>'Groups', 'Users'=>'Users', 'Permissions'=>'Permissions',
'Customers'=>'Customers'];
    $actions = ['index'=>'index','add'=>'add','edit'=>'edit','delete'=>'delete'];
    echo $this->Form->input('group_id', ['options' => $groups,
'empty'=>'Grupo','class'=>'col2']);
    echo $this->Form->input('controller',
['options'=>$controls,'class'=>'col2','empty'=>'Controller']);
    echo $this->Form->input('action',
['options'=>$actions,'class'=>'col2','empty'=>'Action']);
?>
```

## Actions em form

```
echo $form->create('Post', ['action' => 'whatever']);
echo $form->create('Post', ['url' => '/controller_name/action_name']);
echo $this->Form->create("Despesas",['url'=>'/despesas/despesas_mes']);
```

Quando usamos o método set() em nosso controller, definimos variáveis específicas para serem enviadas para a view/template correspondente. A view/template fará com que todas as variáveis passadas estejam disponíveis no escopo do template como variáveis locais.

## Mudando largura de campos

```
echo $this->Form->input['Busca',['type'=>'text','maxlength'=>'8','style'=>'width:50px;
height:20px;']];
```

## Criando link com o HtmlHelper:

Para o controller atual

```
<?= $this->Html->link(__('Lista de Usuário'), ['action' => 'index']) ?>
```

Para outro controller

```
<?= $this->Html->link(__('Lista de Usuário'), ['controller'=>'Users','action' => 'index']) ?>
```

## Criando formulário com o FormHelper

```
<?= $this->Form->create($user) ?>
<legend><?= __('Editar Usuário') ?></legend>
<?php
    echo $this->Form->input('username');
    echo $this->Form->input('password');
    echo $this->Form->input('role', ['admin'=>'Administrador','user'=>'Usuário']);
?>
<?= $this->Form->button(__('Submit')) ?>
<?= $this->Form->end() ?>
```

### **Campo input simular select em FormHelper e mudando o label:**

```
echo $this->Form->label('Tipo'); // Criando o Label
echo $this->Form->select('role', ['user'=>'Usuário','admin'=>'Administrador'], ['default' =>
'admin']);
```

### **Mais detalhes em:**

<http://book.cakephp.org/3.0/en/views/helpers/form.html>

<https://book.cakephp.org/3.0/pt/core-libraries/form.html>

## 9.3.2 – htmlHelper

*Não perca a motivação só porque as coisas não estão correndo como o previsto. Adversidade gera sabedoria e é isso que levará você ao sucesso.*

```
class Cake\View\Helper\HtmlHelper(View $view, array $config = [])
```

O papel do HtmlHelper no CakePHP é fazer opções relacionados a HTML mais fácil, mais rápido e mais resistente à mudança. Usando este helper permitirá que sua aplicação seja mais luz em seus pés, e mais flexível de onde ele é colocado em relação à raiz de um domínio.

Você deve ter notado o uso de um objeto chamado `$this->Html`, uma instância da classe [Cake\View\Helper\HtmlHelper](#) do CakePHP. O CakePHP vem com um conjunto de view helpers que simplificam tarefas como gerar links e formulários. Você pode aprender como usá-los em [Helpers \(Facilitadores\)](#), mas aqui é importante notar que o método `link()` irá gerar um link HTML com o referido título (primeiro parâmetro) e URL (segundo parâmetro).

Muitos métodos HtmlHelper incluem um parâmetro `$attributes`, que lhe permitem aderência em quaisquer atributo extras em suas tags. Aqui estão alguns exemplos de como usar o parâmetro `$attributes`:

*Desired attributes:* `<tag class="someClass" />`

*Array parameter:* `['class' => 'someClass']`

*Desired attributes:* `<tag name="foo" value="bar" />`

*Array parameter:* `['name' => 'foo', 'value' => 'bar']`

### Inserindo Elementos Bem Formatados

A tarefa mais importante que o HtmlHelper realiza é a criação de marcação bem formado. Esta seção irá cobrir alguns dos métodos do HtmlHelper e como usá-los.

#### Criando a tag Charset

```
Cake\View\Helper\HtmlHelper::charset($charset=null)
```

Usado para criar a meta tag especificando o charset do documento. O default valor é UTF-8. Veja um exemplo:

```
echo $this->Html->charset();
```

Uma alternativa:

```
echo $this->Html->charset('ISO-8859-1');
```

## Linkando para Arquivos CSS

```
Cake\View\Helper\HtmlHelper::css(mixed $path, array $options = [])
```

Cria um link para um CSS. Caso o bloco option seja configurado para true, a tag link é adicionada para o bloco CSS que você pode imprimir dentro da tag head do documento.

Este método de inclusão do CSS assume que o CSS especificado reside dentro do webroot/css e não inicia com '/':

```
echo $this->Html->css('forms');
```

Deve gerar:

```
<link rel="stylesheet" href="/css/forms.css" />
```

O primeiro parâmetro deve ser um array para incluir múltiplos arquivos:

```
echo $this->Html->css(['forms', 'tables', 'menu']);
```

Saída:

```
<link rel="stylesheet" href="/css/forms.css" />
```

```
<link rel="stylesheet" href="/css/tables.css" />
```

```
<link rel="stylesheet" href="/css/menu.css" />
```

Adicionando CSS de um Plugin:

```
echo $this->Html->css('DebugKit.toolbar.css');
```

## Criando Meta tags

```
<?= $this->Html->meta(  
    'favicon.ico',  
    '/favicon.ico',  
    ['type' => 'icon']  
);  
?>
```

```
<?= $this->Html->meta(  
    'keywords',  
    'enter any meta keyword here'  
);  
?>
```

```
<?= $this->Html->meta(  
    'description',  
    'enter any meta description here'  
);  
?>
```



## Linkando para Imagens

```
Cake\View\Helper\HtmlHelper::image(string $path, array $options = [])  
  
echo $this->Html->image('cake_logo.png', ['alt' => 'CakePHP']);
```

Gera:

```

```

```
echo $this->Html->image("recipes/6.jpg", [  
    "alt" => "Brownies",  
    'url' => ['controller' => 'Recipes', 'action' => 'view', 6]  
]);
```

Plugin:

```
echo $this->Html->image('DebugKit.icon.png');
```

## Criando Links

```
echo $this->Html->link(  
    'Enter',  
    '/pages/home',  
    ['class' => 'button', 'target' => '_blank']  
);
```

```
echo $this->Html->link(  
    'Dashboard',  
    ['controller' => 'Dashboards', 'action' => 'index', '_full' => true]  
);
```

```
echo $this->Html->link(  
    'Delete',  
    ['controller' => 'Recipes', 'action' => 'delete', 6],  
    ['confirm' => 'Are you sure you wish to delete this recipe?'],  
);
```

```
echo $this->Html->link('View image', [  
    'controller' => 'Images',  
    'action' => 'view',  
    1,  
    '?' => ['height' => 400, 'width' => 500]  
]);
```

## Linkando Vídeos

```
<?= $this->Html->media('audio.mp3') ?>
// Output
<audio src="/files/audio.mp3"></audio>
```

```
<?= $this->Html->media('video.mp4', [
    'fullBase' => true,
    'text' => 'Fallback text'
]) ?>
```

```
<?= $this->Html->media(
    ['video.mp4', ['src' => 'video.ogg', 'type' => "video/ogg; codecs='theora, vorbis'"]],
    ['autoplay']
) ?>
```

## Link para Imagens

```
Cake\View\Helper\HtmlHelper::image(string $path, array $options = [])
```

Creates a formatted image tag. The path supplied should be relative to webroot/img/.

```
echo $this->Html->image('cake_logo.png', ['alt' => 'CakePHP']);
```

Irá mostrar:

```

```

## CSS CakePHP

Customizar CSS do bootstrap numa view

```
echo $this->Form->input('grupo',['style'=>'width: 200px']);
```

## Tipos de Campos

```
input[type="text"],
input[type="password"],
input[type="email"],
input[type="tel"],
input[type="select"] {
    max-width: 280px;
}
```

## Usando num template login.ctp

```
<div class="users form">
<?= $this->Flash->render('auth') ?>
  <?= $this->Form->create() ?>
  <div align="center">
  <fieldset>
    <legend><?= __('<h3><b>Favor entrar seu com login e senha</b></h3>') ?></legend>
    <?= $this->Form->input('username', ['label'=>'Login', 'class'=>'col4']) ?>
    <?= $this->Form->input('password', ['label'=>'Senha', 'class'=>'col4']) ?>
  </fieldset>
  <?= $this->Form->button(__('Acessar')); ?>
  <?= $this->Form->end() ?>
  </div>
</div>
```

No arquivo webroot/css/custom.css

```
/* Para a customização da largura dos campos de form: echo $this->Form->input('grupo',
[class'=>'col2']); */
.col1{
  width:50px;
}

.col2{
  width:100px;
}

.col3{
  width:150px;
}

.col4{
  width:200px;
}

.col5{
  width:250px;
}

.col6{
  width:300px;
}

.col7{
  width:350px;
}

.col8{
```

```

width:400px;
}

.col9{
width:950px;
}

```

## JavaScript em View/Template

```

<div class="actions columns col-lg-2 col-md-3">
  <h3><? = __('Ações') ?></h3>
  <ul class="nav nav-stacked nav-pills">
    <li class="active disabled"><? = $this->Html->link(__('Novo Materiai'), ['action' => 'add'])
?></li>
    <li><? = $this->Html->link(__('Listar Materiais'), ['action' => 'index']) ?></li>
  </ul>
</div>

```

```

<script type="text/javascript">

```

```

function grupoFocus() {
  // conta_grupo receber o valor da combo grupo
  var elt = document.getElementById('grupo');
  var selection=elt.options[elt.selectedIndex].innerHTML;
  document.getElementById('conta').value = selection;
  // Foco na descrição
  document.getElementById('descricao').focus();
}
</script>

```

```

<div class="materiais form col-lg-10 col-md-9 columns">
  <? = $this->Form->create($materiai); ?>
  <fieldset>
    <legend><? = __('Add Materiai') ?></legend>
    <?php
      echo $this->Form->input('codigo', ['autofocus'=>'autofocus']);
      echo $this->Form->input('grupo', ['label'=>__, 'id'=>'grupo', 'readonly' => true, 'options'
=> $grupos, 'empty'=>'Selecione o grupo', 'onchange'=>'grupoFocus()']);
      echo $this->Form->input('conta_grupo',['id'=>'conta', 'readonly'=>true]);
      echo $this->Form->input('descricao',['id'=>'descricao']);
      echo $this->Form->input('uid_inclusao');
      echo $this->Form->input('uid_data_inclusao');
      echo $this->Form->input('uid_alteracao');
      echo $this->Form->input('uid_data_alteracao');
      echo $this->Form->input('status');
    ?>
  </fieldset>
  <? = $this->Form->button(__('Enviar'), ['class' => 'btn-success']) ?>
  <? = $this->Form->end() ?>
</div>

```

**Mais detalhes em:**

<http://book.cakephp.org/3.0/en/views/helpers/html.html>  
<http://book.cakephp.org/3.0/en/views/helpers/paginator.html>  
<http://book.cakephp.org/3.0/en/views/helpers/form.html>  
<http://book.cakephp.org/3.0/en/views.html>  
<http://book.cakephp.org/3.0/en/core-libraries/form.html>  
<http://book.cakephp.org/3.0/en/core-libraries/email.html>  
<http://book.cakephp.org/3.0/en/views/helpers.html>



### 9.3.3 – TimeHelper

*Há dias que você tem que ir para a frente só com o que você tem na mão, não dá para esperar pela motivação.*

#### Data e Hora com CakePHP 3

```
class Cake\I18n\Time
```

Se você precisa das funcionalidades do TimeHelper fora de uma View, então use a classe Time:

```
use Cake\I18n\Time;
```

```
class UsersController extends AppController
{
    public function initialize()
    {
        parent::initialize();
        $this->loadComponent('Auth');
    }

    public function afterLogin()
    {
        $time = new Time($this->Auth->user('date_of_birth'));
        if ($time->isToday()) {
            // Greet user with a happy birthday message
            $this->Flash->success(__('Happy birthday to you...'));
        }
    }
}
```

#### Criando Instâncias de Time

Existem algumas maneiras de criar instâncias de Time:

```
use Cake\I18n\Time;
// Create from a string datetime.
$time = Time::createFromFormat(
    'Y-m-d H:i:s',
    $datetime,
    'America/Fortaleza'
);
// Create from a timestamp
$time = Time::createFromTimestamp($ts);

// Get the current time.
$time = Time::now();
```

```
$time = new Time('2014-01-10 11:11', 'America/Fortaleza');  
$time = new Time('2 hours ago');
```

Mais:

```
$now = new Time('2014-04-12 12:22:30');  
Time::setTestNow($now);
```

```
// Retorna '2014-04-12 12:22:30'  
$now = Time::now();
```

```
// Retorna '2014-04-12 12:22:30'  
$now = Time::parse('now');
```

## **Manipulação**

Depois de criada a instância podemos manipular as instâncias de Time usando métodos setters.

```
$now = Time::now();  
$now->year(2016)  
    ->month(08)  
    ->day(31);
```

Também podemos usar os métodos providos pelo PHP embutidos na classe DateTime:

```
$now->setDate(2016, 08, 31);
```

Datas podem ser modificadas através da subtração e adição de seus componentes:

```
$now = Time::now();  
$now->subDays(5);  
$now->addMonth(1);
```

```
// Usando strtotime strings.  
$now->modify('+5 days');
```

Você pode receber os componentes internos da data acessando suas propriedades:

```
$now = Time::now();  
echo $now->year; // 2014  
echo $now->month; // 5  
echo $now->day; // 10  
echo $now->timezone; // America/Fortaleza
```

Também é permitido assinalar diretamente essas propriedades para modificar a data:

```
$time->year = 2015;  
$time->timezone = 'Europe/Paris';
```



## Formatando

```
static Cake\I18n\Time::setJsonEncodeFormat($format)
```

O método configura o formato default usado quando convertendo um objeto para json:

```
Time::setJsonEncodeFormat('yyyy-MM-dd HH:mm:ss');
```

Este método precisa ser chamado estaticamente.

```
Cake\I18n\Time::i18nFormat($format = null, $timezone = null, $locale = null)
```

Algo bem comum a fazer com instâncias de Time é imprimir as datas formatadas. Veja:

```
$now = Time::parse('2014-10-31');
```

```
// Prints a localized datetime stamp.  
echo $now;
```

```
// Outputs '10/31/14, 12:00 AM' for the en-US locale  
$now->i18nFormat();
```

```
// Use the full date and time format  
$now->i18nFormat(\IntlDateFormatter::FULL);
```

```
// Use full date but short time format  
$now->i18nFormat([\IntlDateFormatter::FULL, \IntlDateFormatter::SHORT]);
```

```
// Outputs '2014-10-31 00:00:00'  
$now->i18nFormat('yyyy-MM-dd HH:mm:ss');  
Configurando o Locale Padrão e o Formato de String
```

```
// The same method exists on Date, FrozenDate, and FrozenTime  
Time::setDefaultLocale('pt-BR');
```

## Datas

A classe Date no CakePHP implementa a mesma API e métodos que o Time\I18n\ . A principal diferença entre hora e data é que Data não acompanha componentes de tempo, e está sempre em UTC. Como um exemplo:

```
use Cake\I18n\Date;  
$date = new Date('2015-06-15');
```

```
$date->modify('+2 hours');  
// Outputs 2015-06-15 00:00:00  
echo $date->format('Y-m-d H:i:s');
```

```
$date->modify('+36 hours');  
// Outputs 2015-06-15 00:00:00
```

```
echo $date->format('Y-m-d H:i:s');
```

## Dicas sobre data e hora

Num controller

```
use Cake\I18n\Time;
```

```
use Cake\I18n\Date;
```

```
public function index()
{
    $time = new Time($this->Auth->user('birthday'));
    if ($time->isToday()) {
        // Greet user with a happy birthday message
        $this->Flash->error(__('Hoje é seu aniversário. Meus parabéns...'));
    }

    $this->paginate = [
        'contain' => ['Groups']
    ];
    $permissions = $this->paginate($this->Permissions);

    $this->set(compact('permissions'));
}
```

Outras formas:

```
$now = Time::now();
```

```
$now->year(2013)
```

```
->month(10)
```

```
->day(31);
```

```
dd($now);
```

```
$now = Time::parse('2014-10-31');
```

```
$now = Time::parse('2014-10-31');
```

```
// Outputs 'Oct 31, 2014 12:00 AM' in en-US
```

```
Time::setDefaultLocale('pt-BR');
```

```
echo $now;
```

```
exit;
```

```
$time = new Time($this->Auth->user('modified'));
```

```
Time::setDefaultLocale('pt-BR'); // ou Date::setDefaultLocale('pt-BR');
```

```
echo $time;
```

```
exit;
```

```
$time = new Time($this->Auth->user('modified'));
```

```
Time::setToStringFormat('dd/MM/yyyy');
```

```
echo $time;  
exit;
```

*src/Template/Cientes/add.ctp ou edit.ctp*

Por padrão o Cake mostra apenas os anos de 2011 até 2021 na combo Ano.

Vamos alterar para que o ano mínimo seja 13 anos antes do atual e máximo seja 100 anos antes do atual, ou seja, como estou em 2016, que mostre de 1916 até 2003, mas isso deve ser pensado para atender ao requisito da tabela/aplicativo. No nosso caso, do DNOCS, devemos usar 18 anos antes, ou mais para o primeiro?

E 100 anos antes do atual ou mais para o segundo.

Precisamos saber da legislação e usar a favor da segurança, deixando uma margem.

```
echo $this->Form->input('nascimento',['label' => 'Nascimento',  
'dateFormat' => 'DMY',  
'minYear' => date('Y') - 100,  
'maxYear' => date('Y') - 13,  
'empty' => [  
'day' => 'Dia',  
'month' => 'Mês',  
'year' => 'Ano'  
]  
]);
```

Mais em:

CakePHP usa Chronos para dar poder ao seu utilitário Timer. Qualquer coisa que você possa fazer com Chronos e DateTime você pode fazer com Time e Date:

<https://github.com/cakephp/chronos>

<http://api.cakephp.org/chronos/1.0/>

Antes da versão 3.2 o CakePHP usava Carbon:

<https://github.com/briannesbitt/Carbon>

<http://book.cakephp.org/3.0/en/views/helpers/time.html>

<https://book.cakephp.org/3.0/en/core-libraries/time.html>

<http://book.cakephp.org/3.0/en/core-libraries/internationalization-and-localization.html#parsing-localized-dates>



## 9.3.4 - flashHelper e Flash Component

*Não sei qual é o segredo do sucesso, mas o segredo do fracasso é tentar agradar a todo mundo.*

Bill Cosby

FlashComponent fornece duas maneiras de definir mensagens flash: seu método mágico `__call()` e seu método `set()`. Para fornecer seu aplicativo com verbosidade, o método mágico `__call()` do FlashComponent permite usar um nome de método que mapeia para um elemento localizado no diretório `src/Template/Element/Flash`. Por convenção, os métodos camelcased serão mapeados para o nome do elemento em minúsculas e sublinhado:

```
// Use em src/Template/Element/Flash/success.ctp
$this->Flash->success('This was successful');
```

```
// Use em src/Template/Element/Flash/great_success.ctp
$this->Flash->greatSuccess('This was greatly successful');
```

### Renderizar em texto plano

```
$this->Flash->set('This is a message');
```

### Renderizar em HTML

```
$this->Flash->info(sprintf('<b>%s</b> %s', h($highlight), h($message)), ['escape' => false]);
```

### Código para mensagens em Flash

```
<div class="message error" onclick="this.classList.add('hidden');">Você não tem permissão para  
acessar esta área</div>
```

### Em action/template

```
<?= $this->Flash->render() ?>
```

```
// em seu Controller
$this->Flash->set('The user has been saved.', [
    'element' => 'success'
]);
```

```
// Em seu arquivo de template: Irá usar great_success.ctp ao invés de success.ctp
<?= $this->Flash->render('flash', [
    'element' => 'great_success'
]);
```

### Detalhes sobre o FlashHelper

<https://book.cakephp.org/3.0/en/views/helpers/flash.html>

<https://book.cakephp.org/3.0/en/controllers/components/flash.html>



## 10 – Detalhes sobre Controllers

*Já experimentou acreditar em você? Tente! Você não faz ideia do que é capaz.*

Os controllers 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. Você vai querer manter seus controllers magros e seus Models gordos. Isso lhe ajudará a reutilizar seu código e testá-los mais facilmente.

Mais comumente, controllers são usados para gerenciar a lógica de um único model. Por exemplo, se você está construindo um site para uma padaria online, você pode ter um RecipesController e um IngredientsController gerenciando suas receitas e seus ingredientes. No CakePHP, controllers são nomeados de acordo com o model que manipulam. É também absolutamente possível ter controllers que usam mais de um model.

Os controllers da sua aplicação são classes que estendem a classe CakePHP ApplicationController, a qual por sua vez estende a classe Controller do CakePHP. A classe ApplicationController pode ser definida em `/app/Controller/AppController.php` e deve conter métodos que são compartilhados entre todos os seus controllers.

Os controllers fornecem uma série de métodos que são chamados de ações. Ações são métodos em um controller que manipulam requisições. Por padrão, todos os métodos públicos em um controller são ações e acessíveis por urls.

Os atributos e métodos criados em ApplicationController vão estar disponíveis para todos os controllers da sua aplicação. Este é o lugar ideal para criar códigos que são comuns para todos os seus controllers. Componentes (que você vai aprender mais tarde) são a melhor alternativa para códigos que são usados por muitos (mas não obrigatoriamente em todos) controllers.

Enquanto regras normais de herança de classes orientadas à objetos são aplicadas, o CakePHP também faz um pequeno trabalho extra quando se trata de atributos especiais do controller. A lista de componentes (components) e helpers usados no controller são tratados diferentemente. Nestes casos, as cadeias de valores do ApplicationController são mescladas com os valores de seus controllers filhos. Os valores dos controllers filhos sempre sobrescreveram os do ApplicationController.

O CakePHP mescla as seguintes variáveis do ApplicationController em controllers da sua aplicação:

- \$components
- \$helpers
- \$uses

Lembre-se de adicionar os helpers Html e Form padrões se você incluiu o atributo \$helpers em seu ApplicationController.

Também lembre de fazer as chamadas de callbacks do ApplicationController nos controllers filhos para obter melhores resultados:

```
function beforeFilter() {
    parent::beforeFilter();
}
```

No CakePHP nós gostamos de manter os nossos actions do controller enxutos, e colocar a maior parte da lógica de negócio de nossa aplicação nos modelos.

### Exemplo de Controller Comentado

```
<?php
// O comando abaixo faz o include da classe AppController que encontra-se na pasta
app/Controller
App::uses('AppController', 'Controller');
// Observe que o controller PostsController estende a classe AppController
class PostsController extends AppController {
```

A propriedade recursive define qual a profundidade que o CakePHP deve ir para procurar modelos associados de dados via métodos find() e read().

Imagine uma aplicação com Groups e cada Group contém muitos Users que por sua vez, cada User tem muitos Articles. Você pode setar \$recursive para vários valores baseado no total de dados que você deseja de volta de uma chamada \$this->Group->find():

- 1 Cake busca somente dados de Group, sem joins.
- 0 Cake busca dados de Group e seu domínio
- 1 Cake busca um Group, seu domínio e seus Users associados
- 2 Cake busca um Group, seu domínio, seus Users associados e os Articles assoaiados aos Users

Não defina \$recursive mais do que você precisa. Quando você tem dados buscados pelo CakePHP você não vai deixar sua aplicação lenta desnecessariamente. Observe também que o nível recursivo padrão é 1.

```
*/
    $this->Post->recursive = 0;
/*
```

O método set() tem como objetivo enviar informações para a View. Com a linha abaixo a view index.ctp poderá chamar o array \$posts, que contém todos os posts paginados.

```
*/
    $this->set('posts', $this->paginate());
}
/**
 * view method
 *
 * @throws NotFoundException
 * @param string $id
 * @return void
 */
```



```

public function view($id = null) {
    // Verifica com o model Post, se existe o $id recebido
    if (!$this->Post->exists($id)) {
        throw new NotFoundException(__('Invalid post'));
    }
    // Opções para filtrar o model Post que tem o $id como chave primária
    $options = array('conditions' => array('Post.' . $this->Post->primaryKey => $id));
    // Passa para a view a variável $post com o primeiro Post encontrado
    $this->set('post', $this->Post->find('first', $options));
}

public function add() {
    // Verifica se a requisição é do tipo POST
    if ($this->request->is('post')) {
        // Cake PHP recomenda chamar a função create() para re-inicializar um model antes
de salvar
        // um novo registro. Criar - create(). Salvar/Atualizar - save().
        $this->Post->create();
        // O método save() salva o registro e também valida
        if ($this->Post->save($this->request->data)) {
            // setFlash mostra uma mensagem
            $this->Session->setFlash(__('The post has been saved'));
            // redirect redireciona para o action index. Também existem outros parâmetros
            $this->redirect(array('action' => 'index'));
        } else {
            $this->Session->setFlash(__('The post could not be saved. Please, try again.'));
        }
    }
}

public function edit($id = null) {
    // Verifica se existe o $id no model Post
    if (!$this->Post->exists($id)) {
        throw new NotFoundException(__('Invalid post'));
    }
    //echo 'CONTROLLER - Enviando para o Model';
    //exit;
    // Verifica se houve requisição do tipo POST ou PUT
    if ($this->request->is('post') || $this->request->is('put')) {
        if ($this->Post->save($this->request->data)) {
            //echo 'CONTROLLER - Vindo do Model';
            //exit;
            $this->Session->setFlash(__('The post has been saved'));
            $this->redirect(array('action' => 'index'));
        } else {
            $this->Session->setFlash(__('The post could not be saved. Please, try again.'));
        }
    } else {
        $options = array('conditions' => array('Post.' . $this->Post->primaryKey => $id));
        $this->request->data = $this->Post->find('first', $options);
    }
}

```

```

}
public function delete($id = null) {
    // Guarda em $this->Post->id o $id passado via parâmetro
    $this->Post->id = $id;
    if (!$this->Post->exists()) {
        throw new NotFoundException(__('Invalid post'));
    }
    // CakeRequest::onlyAllow($methods) - adicionado no Cake 2.3. Substituiu o método
    // requireDelete() das versões anteriores.
    $this->request->onlyAllow('post', 'delete');
    // Exclui o registro
    if ($this->Post->delete()) {
        $this->Session->setFlash(__('Post deleted'));
        $this->redirect(array('action' => 'index'));
    }
    $this->Session->setFlash(__('Post was not deleted'));
    $this->redirect(array('action' => 'index'));
}
}
}

```

Permitir que registros sejam deletados usando requisições GET é perigoso, pois rastreadores na web podem acidentalmente deletar todo o seu conteúdo.

### **Mais detalhes**

<https://book.cakephp.org/3.0/en/controllers.html>

## 10.1 - Componentes

*Nunca desista porque encontrou um obstáculo! Os desafios são o tempero da vida!*

### O que é um Component segundo a documentação do CakePHP:

Os componentes são pacotes de lógica que são compartilhados entre controllers. O CakePHP vem com um conjunto fantástico de componentes que você pode usar para ajudar em várias tarefas comuns. Você também pode criar seus próprios componentes. Se você copia e cola parte de código entre controllers, você deve considerar criar seu próprio Component para conter essa funcionalidade. A criação de Componentes no CakePHP mantém o código dos controllers mais limpos e permite que você reutilize código entre controllers diferentes.

Documentação: <https://book.cakephp.org/3.0/en/controllers/components.html#components>

Os Components podem ser considerados como formas de criar pedaços reutilizáveis de código relacionado a controllers com uma característica específica ou conceito.

O CakePHP conta com vários componentes nativos:

- [AuthComponent](#)
- [Cookie](#)
- [Cross Site Request Forgery](#)
- [Flash](#)
- [Security](#)
- [Pagination](#)
- [Request Handling](#)

**FlashComponent**, são usados nos controllers para criar mensagens flash para o `$_SESSION`, para serem renderizadas em uma View usando [FlashHelper](#).

Criando uma mensagem flash:

```
// Uses src/Template/Element/Flash/success.ctp
$this->Flash->success('This was successful');
```

Usamos os métodos `success()` e `error()` do `FlashComponent` para definir uma mensagem que será armazenada numa variável de sessão.

### Mensagem em texto plano

Também podemos criar uma mensagem em texto plano no controller através do método `set` que será recebida na view respectiva com:

```
$this->Flash->set('Isto é uma mensagem somente texto');
```

### Criando um Component

Para nosso exemplo, vamos criar um Component para criação de senhas com caracteres aleatórios. Para isso precisamos criar uma classe no diretório

src/Controller/Component/, um nome de component seguindo a convenção do cake seria algo como: MinhaClasseComponent.php que estende da classe Component. Como umas das filosofias do nosso Framework favorito é facilitar o trabalho, podemos usar o PHP CLI(command line interface) ou vulgarmente conhecido como Bake.

Para executar o console do cake, precisamos simplesmente executar o código abaixo.

## Carregar componentes

No método initialize() de um controller

```
$this->loadComponent('Flash');  
$this->loadComponent('Auth');
```

## Criação de um Componente

Criar o arquivo

src/Controller/Component/CalcularComponent.php, contendo:

```
<?php  
namespace App\Controller\Component;  
  
use Cake\Controller\Component;  
  
class CalcularComponent extends Component  
{  
    public function soma($valor1, $valor2)  
    {  
        $s = $valor1 + $valor2;  
        return $s;  
    }  
}
```

Adicionar uma chamada no método initialize() do src/Controller/AppController

**Executar num action**, como o src/Controller/CientesController.php

```
public function index()  
{  
    print 'Componente Soma: '.$this->Calcular->soma(2,3);
```

Estando no AppController ele será visto por todos os demais controllers e seus métodos estarão disponíveis logo que cada controller seja executado.

Caso queira que apenas um controller tenha acesso ao mesmo, insira o método initialize() no início deste controller assim :

```
public function initialize()  
{
```

```
parent::initialize();
$this->loadComponent('Calculador');
}
```

## Exemplos de Componentes Simples

```
<?php
/**
 * Component Calculo
 * For CakePHP 3.x
 * Autor: Ribamar FS
 *
 * This component allow access control to each action from application with web interface.
 *
 * Licenced with The MIT License
 * Redistributions require keep copyright below.
 *
 * @copyright Copyright (c) Ribamar FS (http://ribafs.org)
 * @link http://ribafs.org
 * @license http://www.opensource.org/licenses/mit-license.php MIT License
 */
```

```
namespace App\Controller\Component;
use Cake\Controller\Component;
```

```
class CalculoComponent extends Component{
```

```
    private $sum;
    private $dois;
```

```
    public function soma($sum,$dois){
        return ($sum + $dois);
    }
```

```
    public function subtracao($sum,$dois){
        return ($sum - $dois);
    }
```

```
    public function multiplicacao($sum,$dois){
        return ($sum * $dois);
    }
```

```
    public function divisao($sum,$dois){
        return ($sum / $dois);
    }
```

```
}
```

### **Exemplos Práticos com o CakePHP 3**

<https://www.tutorialspoint.com/cakephp/index.htm>

### **Bom exemplo de criação de plugin**

<https://www.cakephpbrasil.com.br/tutorial/2017/03/07/criando-components-no-cakephp-3/>

### **Padrões para a criação de um bom plugin para o CakePHP 3**

<https://www.cakedc.com/plugin-standard>

### **Mais detalhes**

<https://book.cakephp.org/3.0/en/controllers/components.html>

# 11 – Plugins

*Teste os seus limites, enfrente os seus medos e não deixe que nada impeça você de pelo menos tentar!*

Publicação de Plugins no site do CakePHP:

<https://plugins.cakephp.org/>

No CakePHP 3 o plugin é a forma mais adequada de reutilizar código e compartilhar com a comunidade nossos melhores recursos para o Cake.

Basicamente um plugin pode conter um aplicativo a ser usado dentro do Cake como um mini ou sub aplicativo.

Ele pode conter  
Controllers  
Models  
Views/Templates

E dentro destes:  
Componentes, Helpers, Behaviors, etc

Então é um super recurso do Cake.

Empacotamos nosso código, hospedamos no GitHub gratuitamente e publicamos no Packagist para que possa ser usado por nós e por toda a comunidade com grande facilidade.

Depois podemos atualizar via git ou no próprio GitHub com os ótimos recursos do mesmo.

## Plugin Assets

Os recursos da web de um plugin (mas não arquivos PHP) podem ser atendidos através do plugin no diretório webroot, assim como os assets da aplicação principal:

```
/vendor/meuVendor/NomePlugin/webroot/  
    css/  
    js/  
    img/  
    flash/  
    pdf/
```

## Linkando para Assets em Plugins

Você pode usar o sintaxe plugin ao vincular aos recursos do plugin usando o View\Helper\HtmlHelper script, image ou css methods:

```
// Gera a URL /contact_manager/css/styles.css
echo $this->Html->css('ContactManager.styles');

// Gera a URL /contact_manager/js/widget.js
echo $this->Html->script('ContactManager.widget');

// Gera a URL /contact_manager/img/logo.jpg
echo $this->Html->image('ContactManager.logo');
```

## Chamar Element no layout do plugin

```
/vendor/ribafs/cake-control-br/src/Template/Layout/default.ctp
$this->element('CakeControlBr.topmenu');
```

## Habilitar o plugin

Execute a seguinte linha:

```
bin/cake plugin load ContactManager
```

Isso colocará o plugin `Plugin::load('ContactManager')`; no `src/Application.php`.

## Autoloading Plugin Classes

Ao usar `bake` para criar um plugin ou quando instalar um plugin usando o `Composer`, você normalmente não precisa fazer alterações em seu aplicativo para fazer com que o `CakePHP` reconheça as classes que vivem dentro dele.

Em qualquer outro caso, você precisará modificar o arquivo do `composer.json` do seu aplicativo. Para conter as seguintes informações:

```
"autoload": {
  "psr-4": {
    "CakeAclBr\\": "src"
  }
}
```

## Atualizar plugin após alterar composer.json

Sempre que alterar o `composer.json` execute:

```
composer update
```

## Mais detalhes

<https://book.cakephp.org/authorization/1.1/en/index.html>  
<https://book.cakephp.org/authentication/1.1/en/index.html>  
<https://book.cakephp.org/debugkit/3.x/en/index.html>  
<https://book.cakephp.org/migrations/2.x/en/index.html>  
<https://book.cakephp.org/3.0/en/upgrade-tool.html>



## 11.1 - Criação de Plugin para o CakePHP 3

*A melhor maneira de melhorar o padrão de vida está em melhorar o padrão de pensamento. - U. S. Andersen*

Irei relatar detalhadamente como criei o plugin twbs-cake-css:

<https://github.com/ribafs/twbs-cake-css>

E neste plugin que vou criar, para melhorar o aspecto, adicionei parte de um plugin de terceiros, o twbs-cake-plugin: <https://github.com/elboletaire/twbs-cake-plugin>

Acontece que o twbs-cake-plugin usa less ao invés de css puro e achei lento. Então resolvi editar o mesmo e criar um que ficasse como eu queria, somente com CSS, mais simples e mais rápido. Resolvi usar o Bootstrap, apenas com CSS.

Baixei o Bootstrap:

<http://getbootstrap.com/docs/4.0/getting-started/download/>

Eu ainda usei a versão 3, que vinha com as fontes glyphicons.

Baixei o twbs-cake-plugin

Agora veja como mudei o plugin acima para deixar como eu queria:

Comecei eliminando a pasta config com o arquivo bootstrap.php. Ele chama dois outros plugins que não usarei.

Todo o conteúdo da pasta /src eu usarei como está sem nenhuma modificação, pois ela contém o template do bake para gerar o conteúdo com o Bootstrap, apenas mudarei o layout default.ctp.

### Veja como ficou o layout default.ctp:

```
<?php
$cakeDescription = 'CakePHP: the rapid development php framework';
?>
<!DOCTYPE html>
<html>
<head>
  <? = $this->Html->charset() ?>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>
    <? = $cakeDescription ?>:
    <? = $this->fetch('title') ?>
  </title>
  <? = $this->Html->meta('icon') ?>

  <? = $this->Html->css('base.css') ?>
  <? = $this->Html->css('bootstrap.min.css') ?>
  <? = $this->Html->script('bootstrap.min') ?>
```

```

<?= $this->fetch('meta') ?>
<?= $this->fetch('css') ?>
<?= $this->fetch('script') ?>
</head>
<body>
<style>
/* Remover as duas linhas abaixo, caso o texto do site fique com letras muito grandes */
html { font-size: 14px; }
body { font-size: 14px; }
</style>

<nav class="top-bar expanded" data-topbar role="navigation">
  <ul class="title-area large-3 medium-4 columns">
    <li class="name">
      <h1><a href=""><?= $this->fetch('title') ?></a></h1>
    </li>
  </ul>
  <div class="top-bar-section">
    <ul class="right">
      <li><a target="_blank" href="http://book.cakephp.org/3.0/">Documentation</a></li>
      <li><a target="_blank" href="http://api.cakephp.org/3.0/">API</a></li>
    </ul>
  </div>
</nav>
<?= $this->Flash->render() ?>
<div class="container clearfix"> <!-- Assim <div class="clearfix"> ocupará toda a tela-->
  <?= $this->fetch('content') ?>
</div>
<footer>
</footer>
</body>
</html>

```

Agora vejamos como ficou o conteúdo da pasta /webroot:

```

/css
  bootstrap.min.css
/fonts
  glyphsicons-halflings-regular.ttf
/js
  bootstrap.min.js

```

**Um arquivo muito importante é o composer.json**, que contém informações para a instalação pelo composer. Alterei para ficar assim:

```

{
  "name": "ribafs/twbs-cake-css",

```

```

"authors": [
    {
        "name": "Ribamar FS",
        "email": "ribafs@gmail.com",
        "homepage": "http://racotecnic.com",
        "role": "Developer"
    }
],
"description": "Twitter Bootstrap Plugin for CakePHP 3",
"type": "cakephp-plugin",
"keywords": ["cakephp", "bootstrap", "plugin", "template"],
"license": "MIT",
"support": {
    "issues": "https://github.com/ribafs/twbs-cake-css/issues",
    "source": "https://github.com/ribafs/twbs-cake-css.git"
},
"homepage": "https://github.com/ribafs/twbs-cake-css",
"require": {
    "cakephp/cakephp": "^3.3"
},
"autoload": {
    "psr-4": {
        "Bootstrap\\": "src"
    }
}
}
}

```

Outro arquivo importante, que no GitHub funciona como o index.html dos repositórios, é o README.md. Ele geralmente contém o help ensinando a instalar e outras informações.

## README.md

Simple plugin to implement Bootstrap in CakePHP 3

---

This plugin is a fork of the Twitter Bootstrap Plugin  
<https://github.com/elboletaire/twbs-cake-plugin>

This plugin only use CSS, dont use Less.

It also contains bake templates that will help you starting \*twitter-bootstraped\* CakePHP webapps.

General Features

---

- Bake templates.
- Generic Bootstrap layout.

Installation

-----

### ### Adding the plugin

You can easily install this plugin using composer as follows:

```
``bash
composer require ribafs/twbs-cake-css
``
```

### ### Enabling the plugin

After adding the plugin remember to load it in your `config/bootstrap.php` file:

```
``php
bin/cake plugin load Bootstrap
``
```

This will load the CSS for you.

### ### Add Template to src/Controller/AppController.php

```
``php
public function beforeRender(Event $event)
{
    $this->viewBuilder()->theme('Bootstrap');
}
...
``
```

### ### Baking views

You can bake your views using the twitter bootstrap templates bundled with this plugin. To do so, simply specify the `bootstrap` template when baking your files:

```
``bash
bin/cake bake all amigos --theme Bootstrap
``
```

### License

-----

The MIT License (MIT)

**Faltou apenas a licença, que manteve a mesma: MIT.**

Nos resta agora a pasta `/src/`, que de fato contém o código do plugin que será usado pelo Cake.

Ele contém os seguintes arquivos e pastas:

```
src/Template/Bake/Element/form.ctp
src/Template/Bake/Template/add.ctp
src/Template/Bake/Template/edit.ctp
src/Template/Bake/Template/index.ctp
src/Template/Bake/Template/view.ctp
src/Template/Layout/default.ctp
```

Os cinco primeiro mantive intocados. Apenas o **layout default** eu substitui por um que adaptei do original do Cake, que mostro abaixo:

```
<!DOCTYPE html>
<html>
<head>
  <?=$this->Html->charset() ?>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>
    <?=$this->fetch('title') ?>
  </title>
  <?=$this->Html->meta('icon') ?>

  <?=$this->Html->css('base.css') ?>
  <!-- Abaixo estão as duas linhas do css e js que introduzimos -->
  <?=$this->Html->css('bootstrap.min.css') ?>
  <?=$this->Html->script('bootstrap.min') ?>

  <?=$this->fetch('meta') ?>
  <?=$this->fetch('css') ?>
  <?=$this->fetch('script') ?>
</head>
<body>
<style>
/* Remover estas linhas de style, caso o texto do site fique com letras muito grandes */
html { font-size: 14px; }
body { font-size: 14px; }
</style>

<nav class="top-bar expanded" data-topbar role="navigation">
  <ul class="title-area large-3 medium-4 columns">
    <li class="name">
      <h1><a href=""><?=$this->fetch('title') ?></a></h1>
    </li>
  </ul>
</nav>
<?=$this->Flash->render() ?>
<div class="clearfix"> <!-- Alterei esta linha. Caso queira volte ao original, que era assim:
<div class="container clearfix">-->
  <?=$this->fetch('content') ?>
</div>
```

```
<footer>  
</footer>  
</body>  
</html>
```

Assim concluímos nosso plugin. Como me pareceu que pode ser útil para outros usuários do Cake, abriguei no GitHub e Packagist e está em:

<https://github.com/ribafs/twbs-cake-css>

## 11.2 - Plugin DebugKit

*Aquele que tentou e não conseguiu, é superior àquele que não tentou.*

<https://book.cakephp.org/3.0/pt/debug-kit.html>

DebugKit é um plugin suportado pelo time principal do CakePHP que oferece uma barra de ferramentas para auxiliar na depuração de aplicações do CakePHP.

Já vem instalado por default no CakePHP 3

Para usar clique no botão abaixo e à direita no aplicativo. Este:



Então aparece a barra:

ID	Name	Email	Date	View Edit Delete
10	Felix Bradshaw Mccray	dui@elitCurabiturserd.edu	9/16/13	82071617437
11	Idona Jensen Garrett	sem@Crasvulputate.com	1/8/14	07941004794
12	Wayne Ray Padilla	luctus.felis.purus@	4/3/14	60934465323

Clique numa opção para ver detalhes

Exemplo: Sql log, mostra as consultas que levaram ao que aparece na tela

### Sql Log

default

Total Time: 0 ms — Total Queries: 2 — Total Rows: 21

Query	Rows	Took (ms)
<pre>SELECT   Clientes.id AS `Clientes_id`,   Clientes.nome AS `Clientes_nome`,   Clientes.email AS `Clientes_email`,   Clientes.data_nasc AS `Clientes_data_nasc`,   Clientes.cpf AS `Clientes_cpf` FROM   clientes Clientes LIMIT   20 OFFSET 0</pre>	20	0
<pre>SELECT   (     COUNT(*)   ) AS `count` FROM   clientes Clientes</pre>	1	0

**No CakePHP 3 temos duas opções de plugin, que ficam em:**

aplicativo/plugins

aplicativo/vendor/vendorname/plugin

## Instalando um plugin pelo composer

cd aplicativo  
composer require vendorname/plugin-name

Exemplo:  
composer require ribafs/cake-acl-br

## Criação de Plugins no CakePHP

Execute:

```
bin/cake bake plugin Ola
```

Ele criará toda a estrutura, mas vazia.  
Vamos adicionar conteúdo apenas para mostrar como funciona.

## Habilitar/Carregar o Plugin

```
bin/cake plugin load CakeAclBr
```

## Descarregar plugin

```
bin/cake plugin unload CakeAclBr
```

## Funções para debugar o código

Citando duas delas abaixo:

Para verificar o código, use a função debug, cujo retorno ajuda a debugar:

```
debug($variavel_objeto_array);  
exit;
```

A função dd() é ainda mais prática, pois já tem o exit() embutido e não precisamos digitá-lo:

```
dd($variavel_objeto_array);
```

## Mais detalhes

<https://book.cakephp.org/debugkit/>



## 11.3 - Criação do plugin cake-acl-br

*Não importa se seu sonho vai se realizar hoje ou amanhã, mas sim que você trabalhe para o alcançar todos os dias.*

O plugin cake-acl-br foi criado vindo de outros 3 plugins. Primeiro criei o twbs-cake-css - <https://github.com/ribafs/twbs-cake-css>, depois criei o cake-control, que já removi do GitHub, depois o cake-control-br - <https://github.com/ribafs/cake-control-br> e finalmente este, o cake-acl-br, que já está no release 1.30, ou seja, já o atualizei 30 vezes.

### Seus recursos principais são:

- Menu de topo com o element topmenu
- Uso do framework Bootstrap
- Busca com paginação
- Senhas criptografadas com Bcrypt
- Controle de Acesso tipo ACL com administração web
- Dois Layouts: admin e default com cor de fundo que os diferencia
- Datas formatadas para pt-br (veja em Customers)
- Tradução do template do Bake para pt-br
- Customização do bootstrap\_cli adicionando os campos login e logout na geração do Bake
- Validação via frontend no login com pattern e minlength, para exigir senha forte, com pelo menos 8 dígitos, uma maiúscula, uma minúscula e um símbolo. Também com recomendações para validação semelhante pelo CakePHP no UsersTable.php

Geralmente quando vou criar um novo aplicativo eu uso o cake-acl-br, pois o considero muito útil.

Vou esquecer a criação dos plugins do qual ele deriva e me focar apenas na criação do cake-acl-br – <https://github.com/ribafs/cake-acl-br>.

### Um pouco de como implementei os recursos

- Framework bootstrap juntamente com o template do bake para implementar o Bootstrap no código, especialmente nas views. Estes recursos vieram do plugin <https://github.com/elboletaire/twbs-cake-plugin>

- Busca com Paginação. Este veio do Tayron: <http://www.tayron.com.br/blog/121/criando-um-formulario-de-pesquisa-com-cakephp3>. O código para implantar a busca encontra-se no plugin, mas comentado. Para habilitar basta descomentar no action e na view.

- Depois resolvi traduzir as strings do template do bake para que ele gerasse o código traduzido. Esta tradução encontra-se basicamente nos arquivos da pasta src/Template/Bake

- Também adicionei no docs/bootstrap\_cli.php código para que ao gerar o código com bake já seja gerado o código para login e logout

- Criei um element (topmenu) com um menu de topo e já o incorporei ao plugin em src/Template/Element/topmenu.ctp
- Implementei a criptografia com Bcrypt e incorporei ao plugin em src/Template/Bake/Model/entity.ctp
- Criei um componente para controlar o acesso do aplicativo, que faz o principal trabalho de ACL (ControlComponent) e inseri em src/Controller/Component/ControlComponent.php
- Criei um segundo layout, o primeiro para usuários administradores e o outro para os dois restantes e adicionei em src/Template/Layout
- Adicionei ao código do componente ControlComponent suporte para MySQL e PostgreSQL e já ofereci dois scripts .sql para cada um
- Criei um tratamento de erro para o PDO em docs/pdo\_error.ctp e deve ser copiado para o aplicativo/src/Template/Error
- Criei um pequeno estilo em CSS para ajustes no CSS do Bootstrap, que está em docs/custom.css e deve ser copiado para webroot/css
- A parte mais trabalhosa foi encontrar o código correto para lidar com o componente Auth no AppController. Neste controller tem muito código importante para o plugin.
- Finalmente adicionei validação ao login. No frontend com HTML5 e no backend com validação no model, de forma que exija senhas com pelo menos um caractere maiúsculo, um minúsculo, um número, um símbolo e 8 dígitos.

## **Começando pra valer**

Irei começar do plugin cake-control-br, esquecendo que iniciei com o twbs-cake-plugin.

Fazendo o download do cake-acl-br:  
<https://github.com/ribafs/cake-acl-br>

Não vou mostrar detalhes do php na construção do componente, mas apenas detalhes do CakePHP na criação do plugin, que considero relevantes.

## **Detalhes**

<https://github.com/ribafs/cake-acl-br>

## **Observação importante**

O plugin cake-acl-br foi descontinuado e admin-br continua de onde cake-acl-br parou.

## A estrutura de diretórios ficou assim:

```
/config
  bootstrap.php
/docs
  /dicas
  ApplicationController.php
  bootstrap_cli.php
  cake-acl-br-my.sql
  cake-acl-br-pg.sql
  custom.css
  pdo_error.ctp
/src
  /Controller
    /Component
      ControlComponent.php
  /Template
    /Bake
      /Element
        /Controller
          add.ctp
        delete.ctp
        edit.ctp
        index.ctp
        login.ctp
        logout.ctp
        view.ctp
        form.ctp
      /Model
        entity.ctp
      /Template
        add.ctp
        edit.ctp
        index.ctp
        login.ctp
        view.ctp
    /Element
      topmenu.ctp
  /Layout
    admin.ctp
    default.ctp
/webroot
  /css
    bootstrap.min.css
    bootstrap.min.css.map
  /fonts
    glyphicons-halflings-regular.ttf
  /js
    bootstrap.min.js
    jquery.min.js
```

*composer.json*  
*LICENSE*  
*README.md*

## **Detalhando alguns arquivos**

### **config/bootstrap.php**

Neste arquivo são chamados o BootstrapUI e o código que lida com a formatação das datas:

```
<?php
use Cake\Core\Plugin;

Plugin::load('BootstrapUI');

// Habilita o parseamento de datas localizadas
date_default_timezone_set('America/Fortaleza');

/**
 * Locale Formats
 */
\Cake\I18n\Time::setToStringFormat([IntlDateFormatter::MEDIUM,
IntlDateFormatter::SHORT]);
\Cake\I18n\Time::setToStringFormat('dd/MM/YYYY HH:mm');

\Cake\Database\Type::build('date')
->useLocaleParser()
->setLocaleFormat('dd/MM/yyyy');

\Cake\Database\Type::build('datetime')
->useLocaleParser()
->setLocaleFormat('dd/MM/yyyy HH:mm');

\Cake\Database\Type::build('timestamp')
->useLocaleParser()
->setLocaleFormat('dd/MM/yyyy HH:mm');

\Cake\Database\Type::build('decimal')
->useLocaleParser();

\Cake\Database\Type::build('float')
->useLocaleParser();

ini_set('intl.default_locale', 'pt_BR');

/**
 * Default formats
 */
\Cake\I18n\Time::setToStringFormat('dd/MM/yyyy HH:mm:ss');
```

```
\Cake\I18n\Date::setToStringFormat('dd/MM/yyyy');
\Cake\I18n\FrozenTime::setToStringFormat('dd/MM/yyyy HH:mm:ss');
\Cake\I18n\FrozenDate::setToStringFormat('dd/MM/yyyy');
```

## docs/AppController.php (resumido)

```
<?php
namespace App\Controller;

use Cake\Controller\Controller;
use Cake\Event\Event;

class AppController extends Controller
{
    // Controller default para usuários não admins em
    protected $noAdmins = 'Customers';

    public $helpers = [
        'BootstrapUI.Form',
        'BootstrapUI.Html',
        'BootstrapUI.Flash',
        'BootstrapUI.Paginator'
    ];

    public function initialize()
    {
        parent::initialize();

        $this->loadComponent('RequestHandler');
        $this->loadComponent('Flash');
        $this->loadComponent('CakeAclBr.Control');
        $this->loadComponent('Auth', [
            'loginRedirect' => [
                'controller' => 'Permissions',
                'action' => 'index',
                'base' => false
            ],
            'loginAction' => [
                'plugin' => false,
                'controller' => 'Users',
                'action' => 'login'
            ],
            'logoutRedirect' => [
                'controller' => 'Users',
                'action' => 'login'
            ],
            'unauthorizedRedirect' => [
                'controller' => 'Users',
                'action' => 'login',
            ]
        ]);
    }
}
```

```

    'prefix' => false
  ],
  'authError' => 'Sem permissão para acessar esta área!',
  'flash' => [
    'element' => 'error'
  ]
]);

```

```

$this->set('titulo', 'Título do Aplicativo');

```

```

    $user = $this->request->getSession()->read('Auth.User');
    $loguser = $user['username'];
    $this->set('loguser', $loguser);

```

```

    $group = $user['group_id'];
    $this->set('group', $group);

```

```

    $controller = $this->request->getParam('controller');
    $this->set('controller', $controller);

```

```

    $action = $this->request->getParam('action');
    // Via url: users/login?temp=default
    // $layout = $this->request->query('temp');
    // $this->set('template', $layout);

```

```

    if($loguser == 'user' || $loguser == 'manager'){
        $this->viewBuilder()->setLayout('CakeAclBr.default');
    }else{
        $this->viewBuilder()->setLayout('CakeAclBr.admin');
    }

```

```

// // Descomente para acesso total aos actions abaixo
// $this->Auth->allow(['index', 'add', 'edit']);

```

```

    if(isset($group)){
        // Popula a tabela permissions
        // Após o primeiro login pode comentar as 3 linhas abaixo
        $this->Control->populate(1); // Full permissions in all tables to users from Supers

```

```

group
        $this->Control->populate(2); // Full only in groups, users and permissions tables to
users from Admins group
        $this->Control->populate(3); // All in all tables that are not groups, users and
permissions to all users from group Managers

```

```

    // Enviar o controller atual para o element topmenu.ctp
    $this->set('supers', $this->Control->tables($controller, $action, 1));
    $this->set('admins', $this->Control->tables($controller, $action, 2));
    $this->set('managers', $this->Control->tables($controller, $action, 3));
    $this->set('users', $this->Control->tables($controller, $action, 4));

```

```

}

```

```

        if($action != 'login' && $action != 'logout'){
            if(isset($group) && $this->Control->access($controller,$action,$group)==false){
                $this->Flash->error(__('Como usuário "'. $user['username'].'", você não tem
permissão de acesso a '. $controller.'/'. $action));
                return $this->redirect($this->referer());
            }
        }
    }

    public function isAuthorized($user)
    {
        $requestedUserId=$this->request->pass[0];

        if ($user['group_id']==1){
            return true;
        } else if ($user['group_id']!=1 || $user['group_id']!=2) {
            if (!$this->request->action == 'index') {
                if($userid==$user['id']) {
                    return true;
                }
            }
            return false;
        }
        return parent::isAuthorized($user);
    }

    public function beforeRender(Event $event)
    {
        if (!array_key_exists('_serialize', $this->viewVars) && in_array($this->response->getType(),
['application/json', 'application/xml'])) {
            $this->set('_serialize', true);
        }
    }
}

```

**Um arquivo vital do plugin é o de configuração:**

**composer.json**

```

{
    "name": "ribafs/cake-acl-br",
    "authors": [
        {
            "name": "Ribamar FS",
            "email": "ribafs@gmail.com",
            "homepage": "http://ribafs.org",
            "role": "Developer"
        }
    ]
}

```





### **Remover plugin**

composer remove ribafs/cake-acl-br

### **Instalar a versão em desenvolvimento**

composer require ribafs/cake-acl-br:dev-master

### **Instalar uma certa versão**

composer require ribafs/cake-acl-br:1.30

### **Carga de Plugins (exemplo)**

```
$this->addPlugin('autoload' => true, 'bootstrap' => false, 'routes' => true);
```

### **Recomendações para o README de plugins**

<https://gist.github.com/josegonzalez/903066>



## 12 - Dicas de CakePHP 3

*Não importa se seu sonho vai se realizar hoje ou amanhã, mas sim que você trabalhe para o alcançar todos os dias.*

### Foco com HTML5

Foco automático em um campo

```
echo $this->Form->input('grupo',['style'=>'width: 100px', 'type'=>'text', 'autofocus']);
```

### Passando argumentos pela URL

Adicionar a linha ao routes.php:

```
$routes->connect('tests/:arg1/:arg2', ['controller' => 'Tests', 'action' => 'index'], ['pass' => ['arg1', 'arg2']]);
```

Visite:

<http://localhost/exemplos/tests/value1/value2>

### Customizando a Paginação no Controller Customers

```
class CustomersController extends AppController
{
    public $paginate = [
        'fields' => ['Customers.id', 'Customers.name', 'Customers.birthday', 'Customers.cpf',
'Customers.created', 'Customers.modified'],
        'limit' => 12,
        'order' => [
            'Customers.id' => 'asc'
        ]
    ];

    public function index()
    {
        $customers = $this->paginate($this->Customers);

        $this->set(compact('customers'));
    }
}
```

### action index resumido

```
public function index()
{
```

```
$this->set('customers', $this->paginate($this->Posts));
}
```

ou

```
$this->set('customers', $this->Despesas->find('all'));
```

### **action view**

```
public function view($id = null)
{
    $post = $this->Posts->get($id);
    $this->set('posts', $posts);
}
```

### **action add**

```
function add() {
    if (!empty($this->data)) {
        if ($this->Category->save($this->data)) {
            $this->Session->setFlash('Your category has been saved.');
            $this->redirect(array('action' => 'index'));
        }
    }
}
```

## **Implementação do displayField()**

Tenho duas tabelas relacionadas: users e clientes.

Clientes tem um campo user\_id

Em

src/Model/Table/UsersTable.php

Mude o displayField para mostrar não o id mas o username

```
$this->displayField('username');
```

Aproveite e torne requerido o campo user\_id em clientes

src/Model/Table/ClientesTable.php

Adicione

```
$validator->notBlank('user_id');
```

No Clientes/index.ctp no valor do user\_id:

```
<td><? = $cliente->has('user') ? $this->Html->link($cliente->user->id, ['controller' =>
'Users', 'action' => 'view', $cliente->user->id] : " ?></td>
```

## Atualização de Aplicativo

Após instalar um aplicativo podemos atualizar o cake e algum plugin com:  
Acessar o diretório e executar:

```
composer update
```

## Implementando Busca com paginação em Forms do CakePHP 3

Adaptação de:

<http://www.tayron.com.br/blog/121/criando-um-formulario-de-pesquisa-com-cakephp3>

O exemplo aqui apresentado foi com um aplicativo criado através do bake.

Apenas duas tabelas:

```
CREATE TABLE IF NOT EXISTS `groups` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `name` varchar(100) NOT NULL,  
  `created` datetime DEFAULT NULL,  
  `modified` datetime DEFAULT NULL,  
  PRIMARY KEY (`id`)  
);
```

```
CREATE TABLE IF NOT EXISTS `users` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `username` varchar(255) NOT NULL,  
  `password` char(255) NOT NULL,  
  `group_id` int(11) NOT NULL,  
  `created` datetime DEFAULT NULL,  
  `modified` datetime DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `username` (`username`)  
);
```

Banco - busca

Diretório - busca

Os dados do formulário deve ser enviado via query string (get).

E a action do formulário deve sempre apontas para o método index, exemplo:

[www.site.com.br/users/index](http://www.site.com.br/users/index).

## View Users/index.ctp

Entendidas as duas regras acima vamos criar nosso formulário na nossa view antes da tag <table>:

```
<?php
    echo $this->Form->create(null, ['type' => 'get']);

    echo $this->Form->input('search',
        ['class' => 'form-control', 'label' => false,
        'placeholder' => 'Digite o username',
        'value' => $this->request->query('search')]);

    echo $this->Form->button('Pesquisar');
    echo $this->Form->end();
?>
<hr />
```

Substitua o método index() existente do Controller/UserController.php por:

```
public function index()
{
    $this->paginate = [
        'contain' => ['Groups'],
        'conditions' => ['and' => [
            //'Users.people like' => '%'. $this->request->query('search') . '%',
            'Users.username like' => '%'. $this->request->query('search') . '%',
        ]],
        'order' => ['Users.id' => 'DESC']
    ];

    $this->set('users', $this->paginate($this->Users));
    $this->set('_serialize', ['users']);
}
```

```
<?php
namespace App\Model\Table;

use Cake\ORM\Query;
use Cake\ORM\RulesChecker;
use Cake\ORM\Table;
use Cake\Validation\Validator;

class UsersTable extends Table
{
    public function initialize(array $config)
    {
        parent::initialize($config);
```

```

    $this->table('users'); // Name of the table in the database, if absent convention assumes
lowercase version of file prefix
    $this->displayField('full_name'); // field or virtual field used for default display in associated
models, if absent 'id' is assumed
    $this->primaryKey('id'); // Primary key field(s) in table, if absent convention assumes 'id' field

    $this->addBehavior('Timestamp'); // Allows your model to timestamp records on
creation/modification
}
}

```

## Resumo do CakePHP

Geralmente controller são usados para gerenciar a lógica de um único model.

Controllers são classes que estendem a classe AppController. AppController deve conter métodos que são compartilhados entre todos os controllers de sua aplicação.

Os controllers fornecem uma série de métodos que lidam com requisições. Estes são chamados de actions.

Components - são a melhor alternativa sobre código usado por muitos controllers.

Os controllers e componentes contam com um método initialize() que é invocado ao final do construtor do controller.

Qualquer trecho de código ou componente:

- No AppController fica acessível por todos os controllers.
- Num certo controller fica acessível somente para este

## Deployment/Implantação

<https://book.cakephp.org/3.0/pt/deployment.html>

Antes de implantar um aplicativo em produção faça as seguintes verificações/alterações:

- Atualizar o arquivo config/core.php

```

debug = false;

```

Ao desabilitar o debug altera o seguinte:

Mensagens de depuração criadas com pr() e debug() serão desabilitadas.

O cache interno do CakePHP será descartado após 999 dias ao invés de ser a cada 10 segundos como em desenvolvimento.

Views de erros serão menos informativas, retornando mensagens de erros genéricas.

Erros do PHP não serão mostrados.

O rastreamento de stack traces (conjunto de exceções) será desabilitado.

## - Checar a segurança:

Observe que atualmente (3.7.7) o Cake já traz o componente security no AppController:

No método initialize()

```
// $this->loadComponent('Security');
```

Para ter seus recursos disponíveis no nosso aplicativo basta descomentar e usar o formHelper para criar os formulários. Isso pode prevenir diversos tipos de adulteração de formulários e reduzir a possibilidade de overdose de requisições.

Certifique-se que seus models possuem as regras de validação habilitadas.

Verifique se apenas o seu diretório webroot é visível publicamente, e que seus segredos (como seu app salt, e qualquer chave de segurança) são privados e únicos também.

- Aprimorar a performance do aplicativo

Execute

```
composer dumpautoload -o
```

**Corrigir error no código com composer**

```
composer cs-fix
```



## 13 - Aplicativos de Exemplo

*Às vezes desistir parece a saída mais fácil, mas é a alternativa mais dolorosa de todas as tentativas de superação!*

### 13.1 - CakePHP Hello World

Instalação e Teste de Aplicativo Mínimo no CakePHP 3.1.6

Criar a estrutura básica do aplicativo (de dentro do diretório web)

```
cd /backup/www/cake
composer create-project --prefer-dist cakephp/app ola
```

Obs.: dessa forma, com o composer, é mais prático para a criação da estrutura básica. Ainda mais prático que fazendo o download manual e descompactando no diretório web.

Chamar pelo navegador  
http://localhost/cake/ola

#### Classes customizadas:

Controller  
Ola

src/Controller/OlaController.php:

```
$routes->connect('/', ['controller' => 'Ola', 'action' => 'index']);
```

```
<?php
namespace App\Controller;
use App\Controller\AppController;

class OlaController extends AppController
{
    public function index()
    {

    }
}
```

#### View

Template/OlaController/index.ctp:

```
<h1>Olá Mundo do Cake</h1>
```

## Configurações

config/routes.php    Ajustar o controller inicial para:

```
$routes->connect('/', ['controller' => 'OlasController', 'action' => 'index']);
```

Este action diz que quando o usuário chamar o aplicativo pelo raiz o action index do OlasController será chamado.

config/app.php

## Customizando as views

Mudando o CSS

Edite o base.css ou cake.css em webroot/css inserindo no início:

```
#menu-box{
    width:150px;
    float:left;
    border-right:1px solid #CCCCCC;
}
#content-box{
    margin-left:10px;
    width:700px;
    float:left;
    border:1px solid #CCCCCC;
    padding:10px;
    background-color:#F3F3F3;
}
```

Na view troque as classes pelo nosso id, assim:

```
<div id="menu-box">
  <ul class="side-nav">
    <li class="heading"><? = __('Actions') ?></li>
    <li><? = $this->Html->link(__('New Cliente'), ['action' => 'add']) ?></li>
    <li><? = $this->Html->link(__('List Pedidos'), ['controller' => 'Pedidos', 'action' =>
'index']) ?></li>
    <li><? = $this->Html->link(__('New Pedido'), ['controller' => 'Pedidos', 'action' =>
'add']) ?></li>
  </ul>
</div>
```

## 13.2 – Blog

*Só fracassa quem desiste de lutar, quem para de tentar, por isso quando cair lembre-se que reerguer-se já é uma vitória.*

Criando um Simple Blog em CakePHP 3

### Pré-Requisitos

PHP 5.5.9 com pdo\_mysql, intl e mbstring (no terminal digite php -v)

Apache 2

MySQL 5

Caso tenha o cRUL instalado use:

```
curl -s https://getcomposer.org/installer | php
```

```
sudo mv composer.phar /usr/local/bin/composer
```

Instalar no Windows

<http://book.cakephp.org/3.0/en/installation.html>

### Criar uma nova aplicação do Cake:

```
composer create-project --prefer-dist cakephp/app cakeblog
```

O comando acima criou a estrutura de arquivos e diretórios básicos de um aplicativo chamado cakeblog.

### Criar o banco cake\_blog

### Configurar o banco em config/app.php

Configure seu Apache para que tenha suporte ao mod\_rewrite.

### Model

Após criar um model (modelo) no CakePHP, nós teremos a base necessária para interagirmos com o banco de dados e executar operações.

Os arquivos de classes, correspondentes aos models, no CakePHP estão divididos entre os objetos Table e Entity. Objetos Table provêm acesso à coleção de entidades armazenada em uma tabela e são alocados em src/Model/Table. Entity em src/Model/Entity.

### Configurando rotas:

```
config/routes.php
```

```
$routes->connect('/', ['controller' => 'Articles', 'action' => 'index']);
```

## Gerando código com bake:

bin/cake bake all Articles

## Executar para criar um arquivo de migração:

bin/cake bake migration CreateArticles title:string body:text category\_id:integer created modified

## Criar uma tabela de categorias:

bin/cake bake migration CreateCategories parent\_id:integer lft:integer[10] rght:integer[10]  
name:string[100] description:string created modified

## Criar as tabelas

bin/cake migrations migrate

Abra o arquivo src/Model/Table/ArticlesTable.php e adicione o seguinte:

```
// src/Model/Table/ArticlesTable.php
namespace App\Model\Table;
use Cake\ORM\Table;

class ArticlesTable extends Table
{
    public function initialize(array $config)
    {
        $this->addBehavior('Timestamp');
        // ADICIONE AS 3 LINHAS ABAIXO PARA RELACIONAR AS DUAS TABELAS
        $this->belongsTo('Categories', [
            'foreignKey' => 'category_id',
        ]);
    }
}
```

## Para Gerar código

bin/cake bake all Categories

Acesse

<http://localhost/cakeblog>

Adicione alguns registros e veja como funciona.

Acesse

<http://localhost/cakeblog/categories>

## 13.3 – Aplicativo com Bootstrap

*Uma mente ousada está sempre mais preparada para desafiar a sorte e atingir o sucesso!*

### Criação de Layouts

Esta versão cria 3 novos layouts e os aplica respectivamente aos usuários:

super  
admin  
manager

- Baixei os arquivos:

bootstrap.css, jquery.min.js e bootstrap.js e alterei o plugin TwitterBootstrap para que os use offline

<https://github.com/CakePHPBrasil/TwitterBootstrap>

composer require cakephp-brasil/twitter-bootstrap

bin/cake plugin load CakephpBrasil

Em src/View/AppView.php adicionar dentro de initialize(), desta forma:

```
public function initialize()  
{  
    $this->loadHelper('Form', ['className' => 'TwitterBootstrap.Form']);  
}
```

No AppController.php

```
$this->viewBuilder()->theme('TwitterBootstrap');
```

```
$this->set('project_name', 'Título que você quer');
```

### Para alterar o menu superior direito

Crie um arquivo

src/Template/Element/nav-bar-right.ctp

Você pode copiar o modelo dentro de

vendor/cakephp-brasil/twitter-bootstrap/src/Template/Element

### Para alterar o menu superior esquerdo

Crie um arquivo

src/Template/Element/nav-bar-left.ctp

Você pode copiar o modelo dentro de  
vendor/cakephp-brasil/twitter-bootstrap/src/Template/Element

### Gerar o código

```
bin\cake bake all MyModel --theme TwitterBootstrap
```

### Para testar:

```
$ cd path-to-project  
$ bin/cake TwitterBootstrap.publish
```

Ou:

```
$ cd path-to-project  
$ bin/cake TwitterBootstrap.publish all
```

- Copia do arquivo app/webroot/css/bootstrap.css para bootstrap.super.css, bootstrap.admin.css e bootstrap.manager.css (todos no webroot/css)

Editei cada um dos que criei e apenas alterei a cor de fundo do body na linha do background-color abaixo:

```
body {  
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;  
  font-size: 14px;  
  line-height: 1.42857143;  
  color: #333;  
  background-color: #B78999;  
}
```

- Copiei o arquivo

/vendor/cakephp-brasil/twitter-bootstrap/src/Template/Layout/default.ctp  
para super.ctp, admin.ctp e manager.ctp

- Editei cada um dos 3 criados (super.ctp, admin.ctp e manager.ctp) e alterei a linha:

```
<? = $this->Html->css('bootstrap');?>
```

para

```
<? = $this->Html->css('bootstrap.super');?>  
<? = $this->Html->css('bootstrap.admin');?>  
<? = $this->Html->css('bootstrap.manager');?>
```

- Adicionei ao initialize() do ApplicationController.php:

```
$this->set('project_name', 'Título do Aplicativo');  
if($loguser == 'super'){  
  //$this->viewBuilder()->theme('TwitterBootstrap');  
  $this->viewBuilder()->layout('TwitterBootstrap.super');
```

```
}elseif($loguser == 'admin'){  
    $this->viewBuilder()->layout('TwitterBootstrap.admin');  
}elseif($loguser=='manager'){  
    $this->viewBuilder()->layout('TwitterBootstrap.manager');  
}elseif($loguser=='user'){  
    $this->viewBuilder()->layout('TwitterBootstrap.default');  
}
```

Assim, ao logar cada usuário terá seu próprio ambiente.  
Claro que falta fazer um bom acabamento nos layouts, nas cores.

Veja também um tutorial similar que mostra a implementação do Bootstrap no CakePHP 3:  
<https://github.com/ribafs/cakephp-app-bs>





## 13.4 – Aplicativo via Código

*Você não precisa estar um passo à frente dos outros, desde que ca dada dia esteja à frente de você mesmo ontem.*

Criar um aplicativo para exemplo de interação do código do CakePHP entre controllers, models e templates.

cake3\_app\_code

Criar um banco no postgresql com duas tabelas relacionadas

groups e users

Configurar o banco no app.php e a rota para apontar para Users e index

Testar

[http://localhost/cake3\\_app\\_code](http://localhost/cake3_app_code)

### **Criar um novo método com apenas uma mensagem/string**

Criar o src/Model/Table/UsersTable.php com apenas o método abaixo

```
<?php
namespace App\Model\Table;

use Cake\ORM\Query;
use Cake\ORM\RulesChecker;
use Cake\ORM\Table;
use Cake\Validation\Validator;

class UsersTable extends Table
{
    public function olaModel(){
        return "Olá model do CakePHP3";
    }
}
```

No UsersController adicione ao início

```
use Cake\ORM\TableRegistry;
```

E crie o método ola, deixando assim:

```
<?php
namespace App\Controller;
```

```

use App\Controller\AppController;
use Cake\ORM\TableRegistry;

class UsersController extends AppController
{
    public function ola()
    {
        $msg = TableRegistry::get('Users');
        $ola = $msg->olaModel();
        $this->set('ola',$ola);
    }
}

```

No Template Users cria o arquivo ola.ctp contendo:

```

<b>A mensagem abaixo foi criada no método olaModel do Model UsersTable.php<br>
e enviada para a view ola.ctp pelo action ola() do Controller UsersController.php</b>
<br><br>
<?= $ola ?>

```

Chamar pelo navegador assim:

[http://localhost/cake3\\_app\\_code/users/ola](http://localhost/cake3_app_code/users/ola)

Alternativamente criar o action index() no UsersController vazio, assim:

```

public function index(){
}

```

E criar a index.ctp contendo:

```

<li><?= $this->Html->link(__('Olá'), ['controller' => 'Users', 'action' => 'ola']) ?></li>

```

### **Criar um método que pega um registro da tabela users**

No UsersTable.php:

Adicione ao início

```

use Cake\ORM\TableRegistry;

public function umUser()
{
    $users = TableRegistry::get('Users');
    $query = $users->find();
    $user=$query->where(['username' => 'super']);
    return $user;
}

```

No UsersController.php

```
public function umUser()
{
    $user = TableRegistry::get('Users');
    $super = $user->umUser();
    $this->set('super',$super);
}
```

No Template/Users criar um\_user.ctp contendo:

```
<?php
//
?>
<h3>Usuário recebido do model através do controller</h3>
<?php
foreach($super as $campos){
    print ' ID - '.$campos->id.'<br>';
    print ' Grupo - '.$campos->group_id.'<br>';
    print ' Username - '.$campos->username.'<br>';
}
```

### Testar

[http://localhost/cake3\\_app\\_code/users/um\\_user](http://localhost/cake3_app_code/users/um_user)

Adicionar o link abaixo para o action um\_user na index.ctp

```
<li><? = $this->Html->link(__('Um usuário'), ['controller' => 'Users', 'action' => 'um_user']) ?
></li>
```

### Sugestões

Existe muita informação importante na documentação do CakePHP 3 online assim também como podemos aprender muito estudando o código gerado pelo bake.

Em português

<https://book.cakephp.org/3.0/pt/index.html>

Aqui

<https://book.cakephp.org/3.0/en/index.html>



## 13.5 - Aplicativo Finanças Pessoais

*Cada um de nós tem um fogo no coração para alguma coisa. É nossa meta na vida encontrá-lo e mantê-lo aceso.* (Mary Lou Retton)

Este aplicativo tem como objetivo principal efetuar operações entre os elementos do MVC (Controller, Model e View) para um simplificada aplicação de controle de receitas e despesas pessoais .

Procurarei simplificar outras operações, focando apenas na codificação.

### Versão com CakePHP 3.6.1

Criar um Aplicativo com o CakePHP 3.6.1  
Para gerenciamento de finanças pessoais.  
Apenas com as tabelas despesas e receitas

Importante: este código customizado funciona apenas no CakePHP 3.6 ou superior  
Criando behaviors para executar o código da camada de negócios nos models  
Criando components para executar o código da camada de negócios nos controllers

#### - Criar o aplicativo

```
cd /var/www/html  
composer create-project --prefer-dist cakephp/app finanças
```

- Criar o banco de dados

Usaremos como SGBD o MySQL

#### Criar o banco finanças contendo:

```
CREATE TABLE `despesas` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `descricao` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,  
  `valor` int(11) DEFAULT NULL,  
  `mes` char(8) COLLATE utf8_unicode_ci DEFAULT NULL,  
  `created` date DEFAULT NULL,  
  `modified` date DEFAULT NULL,  
  `receita_id` int(11) NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;  
  
INSERT INTO `despesas` (`id`, `descricao`, `valor`, `mes`, `created`, `modified`, `receita_id`)  
VALUES  
(1, 'Mercantil', 500, '05/2017', '2017-05-11', '2017-05-15', 1),  
(2, 'Condomínio', 100, '05/2017', '2017-05-11', '2017-05-15', 1),
```

```
(3, 'Teste', 300, '06/2017', '2017-05-11', '2017-05-11', 2);
```

```
CREATE TABLE `receitas` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `descricao` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,  
  `mes` char(7) COLLATE utf8_unicode_ci DEFAULT NULL,  
  `valor` int(11) DEFAULT NULL,  
  `created` date DEFAULT NULL,  
  `modified` date DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
INSERT INTO `receitas` (`id`, `descricao`, `mes`, `valor`, `created`, `modified`) VALUES  
(1, 'Salário', '05/2017', 3000, '2017-05-11', '2017-05-15'),  
(2, 'Extra', '05/2017', 100, '2017-05-11', '2017-05-15');
```

Observe acima que o mês é tipo texto para que seja inserido algo com mes/ano

- Configurar o banco de dados em `config/app.php`  
e o  
`config/routes.php` para `Despesas/index`

## Gerar o código dos dois CRUDs com o Bake

```
cd /var/www/html/financas
```

```
bin/cake bake all despesas  
bin/cake bake all receitas
```

## Veja pelo navegador

<http://localhost/financas>

Já temos um CRUD básico para as despesas e para as receitas

Mas o importante de um aplicativo deste tipo são o código personalizado, chamado de lógica de negócio, que é por exemplo somar todas as despesas de um mês e subtrair as receitas do mês do total de despesas. Isso o CakePHP não faz e precisamos fazer. Também precisamos criar código nos templates adicionando um formulário

## Agora a parte interessante

O código da camada de negócio

**Observe** que o mês tem um formato próprio, texto sendo dia/ano (dd/aaaa).

**Vamos criar uma função em DespesasTable.php, que retorne a soma das despesas do mês:**

Adicionar ao início:

```
use Cake\ORM\TableRegistry;
```

Adicionar:

```
public function despesasMes($mes){  
  
    $despesas = TableRegistry::get('Despesas')->find();  
    $res = $despesas->select(['total_sum' => $despesas->func()->sum('Despesas.valor')])  
    ->where(['Despesas.mes' => $mes])->first();  
    $total = $res->total_sum;  
  
    return $total;  
}
```

**Criar duas funções no DespesasController.php, que retornem o resultado das funções acima:**

Adicionar ao início do DespesasController

```
use Cake\ORM\TableRegistry;
```

**Mais abaixo**

```
public function despesasMes()  
{  
    $mes = $this->request->data('mes');  
    $despesas = TableRegistry::get('Despesas');  
    $total = $despesas->despesasMes($mes);  
    $this->set('total', $total);  
    $this->set('mes', $mes);  
}  
  
public function saldoMes()  
{  
    $mes = $this->request->data('mes');  
    $this->loadModel('Receitas');  
    $receitas = $this->Receitas->receitaMes($mes);  
  
    $despesas = TableRegistry::get('Despesas');  
    $total = $despesas->despesasMes($mes);  
  
    $saldo = $receitas - $total;  
    $this->set('saldo', $saldo);  
    $this->set('mes', $mes);  
}
```

**Agora criaremos uma função no ReceitasTable.php, que retorne a soma das receitas do mês:**

Adicionar ao início:

```
use Cake\ORM\TableRegistry;
```

Mais abaixo:

```
public function receitaMes($mes){
    $receitas = TableRegistry::get('Receitas')->find();
    $res = $receitas->select(['total_sum' => $receitas->func()->sum('Receitas.valor')])
    ->where(['Receitas.mes' => $mes])
    ->first(); //perform the sum operation
    $total = $res->total_sum;

    return $total;
}
```

Veja que existe grande semelhança entre esta e a das despesas.

**Vamos customizar a src/Template/Despesas e criar duas views novas:**

src/Template/Despesas/despesas\_mes.ctp:

```
<?php ?>
<nav class="large-3 medium-4 columns" id="actions-sidebar">
    <ul class="side-nav">
        <h3><? = __('Despesas') ?></h3>
    <?php
    print '<b>Mês: </b>'. $mes. '<br><b>Total: </b>'. $total;
    ?>
    <br>
    <br>
    <br>
    <li><? = $this->Html->link(__('Voltar'), ['action' => 'index']) ?></li>
</nav>
```

src/Template/Despesas/saldo\_mes.ctp:

```
<?php ?>
<nav class="large-3 medium-4 columns" id="actions-sidebar">
    <ul class="side-nav">
        <h3><? = __('Despesas') ?></h3>
    <?php
    print '<b>Mês: </b>'. $mes. '<br><b>Saldol: </b>'. $saldo;
    ?>
    <br>
    <br>
```



```
<br>
<li><?= $this->Html->link(__('Voltar'), ['action' => 'index']) ?></li>
</nav>
```

## Agora vejamos a customização do index.ctp

Logo abaixo da linha:

```
<li><?= $this->Html->link(__('New Receita'), ['controller' => 'Receitas', 'action' => 'add']) ?></li>
```

Adicione os dois pequenos forms:

```
<b>Despesas de um mês</b>
<?php
    echo $this->Form->create("Despesas",['url'=>'/despesas/despesas_mes']);
    echo $this->Form->input('mes');
    echo $this->Form->button('Submit');
    echo $this->Form->end();
?>
<b>Saldo de um mês</b>
<?php
    echo $this->Form->create("Saldo",['url'=>'/despesas/saldo_mes']);
    echo $this->Form->input('mes');
    echo $this->Form->button('Submit');
    echo $this->Form->end();
?>
```

Depois de pronto podemos efetuar as operações através dos pequenos forms.

- O form chama o action do controller
- O controller chama o model para que devolva o resultado de uma função
- Recebendo o resultado o controller devolve para a view/template respectiva

## Testando

Agora acesse pela web  
<http://localhost/financas>

Pesquise o total das despesas do mês de 05/2017 ou outro que você tenha cadastrado.  
Pesquise qual o saldo de um mês

Isso dá para começar a mexer no código e ir em frente.

## Fluxo das informações

- O usuário acessa o aplicativo pela URL:

<http://localhost/financas>

- Então ele cai em (pela configuração do routes):

<http://localhost/financas/despesas/index>

- Na view Despesas/index.ctp ele consulta a despesa total de um certo mês pelo formulário Despesas de um mês

- Este form o direciona para o action DespesasController/despesasMes() através do comando do form: `$this->Form->create("Despesas",['url'=>'/despesas/despesas_mes'])`

- O controller então registra o DespesasTable e chama através do seu método despesasMes() para saber o total das despesas do referido mês

- O DespesasTable, através do seu método despesasMes() prepara uma consulta e a envia ao banco de dados.

- O banco de dados retorna o resultado para o método despesasMes() do DespesasTable.

- O método do DespesasTable retorna o valor para o DespesasController, em seu método despesasMes()

- O método despesasMes() do DespesasController envia através do método set() o mês e o total de despesas do referido mês para a view com o mesmo nome dele, que é a Despesas/despesas\_mes.ctp

- Finalmente o usuário é redirecionado para a view despesas\_mes.ctp onde recebe o resultado de sua consulta

[http://localhost/financas/despesas/despesas\\_mes](http://localhost/financas/despesas/despesas_mes)

## 13.6 – Aplicativo usando o Plugin admin-br

*Não falta quem faça planos, mas para a história apenas ficam aqueles que os concretizaram.*

Para não me tornar repetitivo e evitar conteúdo desatualizado apenas o remeterei para o projeto admin-br, que está sendo atualizado com frequência:

<https://github.com/ribafs/admin-br>



## 13.7 - Aplicativo com uma área restrita/administrativa

Criar um aplicativo com uma área restrita/administrativa para o CakePHP 3

**Adaptado do vídeo abaixo:**

[https://www.youtube.com/watch?v=eWu6r5aO1Jc&index=6&list=PL83b6tjXNFHe-OVRR0awG3YdIN\\_-Kbc6j](https://www.youtube.com/watch?v=eWu6r5aO1Jc&index=6&list=PL83b6tjXNFHe-OVRR0awG3YdIN_-Kbc6j)

Criar um aplicativo chamado admin\_area.

**Banco admin\_area**

```
CREATE TABLE `customers` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `name` varchar(55) COLLATE utf8mb4_unicode_ci NOT NULL,  
  `birthday` date DEFAULT NULL,  
  `phone` varchar(14) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  
  `observation` text COLLATE utf8mb4_unicode_ci,  
  `created` datetime DEFAULT NULL,  
  `modified` datetime DEFAULT NULL,  
  PRIMARY KEY (`id`)  
);  
  
INSERT INTO `customers` (`id`, `name`, `birthday`, `phone`, `observation`, `created`, `modified`)  
VALUES  
(1, 'Brennan G. Wilcox', '2016-04-15', '(851) 190-1314', 'ante. Maecenas mi felis, adipiscing',  
  NULL, NULL),  
(2, 'Chase Summers', '2016-08-27', '(846) 297-4733', 'Sed molestie. Sed id risus', NULL,  
  NULL),  
(3, 'Sonia L. Mckay', '2015-12-02', '(131) 453-1690', 'fermentum vel, mauris. Integer sem',  
  NULL, NULL),  
(4, 'Isadora L. Bowers', '2015-10-24', '(939) 798-4625', 'consequat, lectus sit amet luctus',  
  NULL, NULL),  
(5, 'Sophia Cochran', '2017-06-15', '(811) 687-0491', 'Aliquam tincidunt, nunc ac mattis',  
  NULL, NULL),  
(6, 'Maxwell T. Burton', '2016-01-12', '(147) 962-3265', 'at arcu. Vestibulum ante ipsum',  
  NULL, NULL),  
(7, 'Desiree Y. Henry', '2017-07-21', '(148) 711-3747', 'vitae dolor. Donec fringilla. Donec',  
  NULL, NULL),  
(8, 'Asher Key', '2015-11-07', '(355) 668-5871', 'a, aliquet vel, vulputate eu', NULL, NULL),  
(9, 'Tyler Castro', '2016-08-31', '(567) 793-5061', 'nec tempus mauris erat eget', NULL,  
  NULL),  
(10, 'Rudyard Weber', '2015-10-26', '(445) 457-4552', 'Morbi vehicula. Pellentesque tincidunt  
tempus', NULL, NULL),  
(11, 'Allen Austin', '2016-04-15', '(758) 867-2179', 'ipsum. Phasellus vitae mauris sit', NULL,  
  NULL),  
(12, 'Octavius Cooper', '2015-10-13', '(101) 625-3985', 'ipsum non arcu. Vivamus sit', NULL,  
  NULL),
```

```

(13, 'Dustin M. Oneill', '2016-04-24', '(276) 722-0976', 'magnis dis parturient montes,
nascetur', NULL, NULL),
(14, 'Giacomo K. Horton', '2016-07-03', '(773) 532-7468', 'neque. Sed eget lacus. Mauris',
NULL, NULL),
(15, 'Signe T. Weaver', '2016-06-17', '(210) 895-3664', 'dui nec urna suscipit nonummy.',
NULL, NULL),
(16, 'Avram O. Delaney', '2016-08-05', '(609) 552-7572', 'Donec luctus aliquet odio. Etiam',
NULL, NULL),
(17, 'Cara Parker', '2016-07-24', '(854) 169-4797', 'ornare lectus justo eu arcu.', NULL,
NULL),
(18, 'Chelsea Mcclain', '2016-08-06', '(363) 636-1560', 'mollis lectus pede et risus.', NULL,
NULL),
(19, 'Wesley Garner', '2016-06-11', '(578) 231-2389', 'Fusce feugiat. Lorem ipsum dolor', NULL,
NULL),
(20, 'Irene P. Arnold', '2016-02-12', '(253) 631-9830', 'accumsan laoreet ipsum. Curabitur
consequat', NULL, NULL),
(21, 'Austin S. Wall', '2016-01-21', '(225) 694-9511', 'Sed eget lacus. Mauris non', NULL,
NULL);

```

```

DROP TABLE IF EXISTS `groups`;
CREATE TABLE `groups` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(100) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created` datetime DEFAULT NULL,
  `modified` datetime DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

```

```

INSERT INTO `groups` (`id`, `name`, `created`, `modified`) VALUES
(1, 'Supers', '2016-08-30 21:15:01', '2016-08-30 21:15:01'),
(2, 'Admins', '2016-08-30 21:15:01', '2016-08-30 21:15:01'),
(3, 'Managers', '2016-08-30 21:15:01', '2016-08-30 21:15:01'),
(4, 'Users', '2016-08-30 21:15:01', '2016-08-30 21:15:01');

```

```

CREATE TABLE `users` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(55) COLLATE utf8mb4_unicode_ci NOT NULL,
  `password` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `group_id` int(11) NOT NULL,
  `created` datetime DEFAULT NULL,
  `modified` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`),
  KEY `group_id` (`group_id`),
  CONSTRAINT `users_ibfk_1` FOREIGN KEY (`group_id`) REFERENCES `groups` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE
);

```

```
INSERT INTO `users` (`id`, `username`, `password`, `group_id`, `created`, `modified`) VALUES
(1, 'super', '$2y$10$zIIAsTubGNqvhn3pS76jN.QeF6msHGYGcTIJ7KgzS757vDsioA3xa', 1,
'2016-09-15 15:57:16', '2019-06-12 19:03:49'),
(2, 'admin', '$2y$10$f26.qAgF5Jnl7b3zIdRIQuKTBkrxM2d1xtrLlIee0EKULfSKgejqm', 2, '2016-
09-15 15:57:16', '2019-06-12 19:03:49'),
(3, 'manager', '$2y$10$fx0/o/XU3WPO5.nnP7cnCeSuFsFjxCMkk72DciLqABzHp50cOFnre', 3,
'2016-09-15 15:57:16', '2019-06-12 19:03:49');
```

### **Três usuários usando o hash bcrypt:**

Login - super  
Senha - abc123S@

Login - admin  
Senha - abc123A@

Login - manager  
Senha - abc123M@

### **Configurar o banco em config/app.php**

**Configurar a rota default** em config/routes.php para Customers/index

### **Gerar o código com o bake**

```
bin/cake bake all customers
```

```
bin/cake bake model Users
bin/cake bake model Groups
```

```
bin/cake bake controller Users --prefix admin
bin/cake bake template Users --prefix admin
```

```
bin/cake bake controller Groups --prefix admin
bin/cake bake template Groups --prefix admin
```

### **Adicionar ao webroot/css/style.css**

```
.emlinha {
    display: inline;
    padding: 10px;
    text-align: center;
}
li a {
    color: #fff;
}
```

**Crie um arquivo src/Template/Element/menu.ctp** contendo:

```
<div>
  <ul>
    <li class="emlinha"><?= $this->Html->link(__('Customers'), '/customers/index') ?></li>
    <li class="emlinha"><?= $this->Html->link(__('Groups'), '/admin/groups/index') ?></li>
    <li class="emlinha"><?= $this->Html->link(__('Users'), '/admin/users/index') ?></li>
    <li class="emlinha"><?= $this->Html->link(__('Login'), '/admin/users/login') ?></li>
    <li class="emlinha"><?= $this->Html->link(__('Sair'), '/admin/users/logout') ?></li>
  </ul>
</div>
```

**Edite o src/Template/Layout/default.ctp**

E sobrescreva o código entre

```
<nav>
e
</nav>
```

Com este abaixo, logo abaixo de <body>:

```
<nav class="top-bar expanded" data-topbar role="navigation">
<?php
  if($loguser){
?>
    <h2 align="center"><?= $titulo ?><?php echo $this->element('menu') ?></h2>
    <div align="center"><?= $this->fetch('title') ?></div>
    <div class="right">Logado como: <strong><?= __($loguser)
?></strong>&nbsp;&nbsp;&nbsp;&nbsp;</div>
<?php
  }else{
    echo '<h3 class="titulo" align="center">Acesso ao Sistema</h3>';
  }
?>
</nav>
```

**Somente customers poderá ser acessado livremente, groups e users serão restritos**

**Criar a rota admin em config/routes.php**

```
Router::prefix('admin', function ($routes) {
    $routes->fallbacks(DashedRoute::class);
});
```

**Obs.:**

Veja que na pasta src/Controller/Admin os controllers tem o namespace para Admin

```
namespace App\Controller\Admin;
```



**Adicionar os actions login e logout** para src/Controller/Admin/UsersController.php:

```
public function login()
{
    if ($this->request->is('post')) {
        $user = $this->Auth->identify();
        if ($user) {
            $this->Auth->setUser($user);
            return $this->redirect($this->Auth->redirectUrl());
        }
        $this->Flash->error(__('Usuário ou senha inválido, tente novamente'));
    }
}

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

**Adicionar o login.ctp** para src/Template/Admin/Users:

```
<div class="users form">
<?=$this->Flash->render('auth') ?>
<?=$this->Form->create() ?>
    <fieldset>
        <legend><?=__('Por favor informe seu usuário e senha') ?></legend>
        <?=$this->Form->input('username', ['autofocus' => true]) ?>
        <?=$this->Form->input('password') ?>
    </fieldset>
    <?=$this->Form->button(__('Login')); ?>
    <?=$this->Form->end() ?>
</div>
```

**Para acessar customers:**

[http://localhost/admin\\_area](http://localhost/admin_area)

Ainda não podemos acessar users e groups e o acesso integral de customers é permitido.

**Implementar a A autenticação**

Para controlar o acesso ao aplicativo

Adicionar suporte para bcrypt ao login:

Adicionar ao início do src/Model/Entity/User.php:

```
use Cake\Auth\DefaultPasswordHasher;
```

Ao final:

```
protected function _setPassword($password)
{
    if (strlen($password) > 0) {
        return (new DefaultPasswordHasher)->hash($password);
    }
}
```

## Implementar a autenticação

Adicione ao initialize() do ApplicationController, logo abaixo do carregamento do componente Flash::

```
$this->loadComponent('Auth', [
    'authorize' => ['Controller'],
    'logoutRedirect' => [
        'controller' => 'Users',
        'action' => 'login',
        'home'
    ],
    //'unauthorizedRedirect' => $this->referer()
    'unauthorizedRedirect' => [
        'controller' => 'Users',
        'action' => 'login',
        'prefix' => false
    ],
    'authError' => 'Você não tem permissão para acessar esta área!',
    'flash' => [
        'element' => 'error'
    ]
]);

$this->set('titulo', 'Aplicativo para área Admin Restrita');
$user = $this->request->getSession()->read('Auth.User');
$loguser = $user['username'];
$this->set('loguser',$loguser);
```

**Restringindo acesso** com o Authorization através do callback isAuthorized().

Agora que os usuários podem se conectar, nós vamos querer limitar os bookmarks que podem ver para aqueles que fizeram. Nós vamos fazer isso usando um adaptador de ‘autorização’. Sendo os nossos requisitos bastante simples, podemos escrever um código em nossa BookmarksController. Mas antes de fazer isso, vamos querer dizer ao AuthComponent como nossa aplicação vai autorizar ações. Em seu ApplicationController adicione o seguinte:

Ao final, abaixo do initialize():

```
public function isAuthorized($user = null)
```

```

{
  // Any registered user can access public functions
  if (!$this->request->getParam('prefix')) {
    return true;
  }

  // Only admins can access admin functions
  if ($this->request->getParam('prefix') === 'admin') {
    return (bool)($user['group_id'] === 1 || $user['group_id'] === 2);
  }

  // Default deny
  return false;
}

```

Observe que os usuários dos grupos 1 (super) e 2 (admin), têm direito a acessar tudo. Aqui podemos customizar mais este comportamento. Mas caso precise de algo mais flexível recomendo o uso de um dos plugins sugeridos ao final.

Abaixo do `isAuthorized()` adicione:

```

public function beforeFilter(Event $event)
{
  $this->Auth->allow(['customers'=>'index', 'customers'=>'view']);
  // Negar acesso para todos os actions do controller Customers para usuários comuns
  //$this->Auth->deny('customers');
}

```

### **Também podemos negar acesso:**

Obs.: Não se aplica para os usuários dos grupos 1 e 2, que podem tudo e já foram previamente autorizados.

### **As regras abaixo aplicam-se somente para os usuários dos grupos 3 em diante.**

```

// Negar acesso a qualquer parte do aplicativo:
$this->Auth->deny();

```

```

// Negar apenas para o action add
$this->Auth->deny('add');

```

```

// Negar para um grupo de actions
$this->Auth->deny(['add', 'edit']);

```

```

// Negar para um controller
$this->Auth->deny('customers');

```

Observe que são públicos apenas os actions index e view do controller Customers. Isto indica a que estamos garantindo acesso completo em todos os controllers:

```

$this->Auth->allow(['index', 'view']);

```

**Podemos controlar o acesso por este método.**

**Para que um controller permita o acesso completo adicione ao seu início:**

```
use Cake\Event\Event;

public function beforeFilter(Event $event)
{
    $this->Auth->allow();
}
```

**Testar o acesso para áreas restritas:**

[http://localhost/admin\\_area/admin/users/add](http://localhost/admin_area/admin/users/add)

ou

[http://localhost/admin\\_area/customers/add](http://localhost/admin_area/customers/add)

**Veja que ele pede o controller Users no src. Então lhe daremos:**

E crie o src/Controller/UsersController.php com apenas:

```
<?php
namespace App\Controller;
use App\Controller\AppController;

class UsersController extends AppController
{
    public function login()
    {
        return $this->redirect('/admin/users/login');
    }
}
```

**Agora experimente acessar**

[http://localhost/admin\\_area/customers/add](http://localhost/admin_area/customers/add)

Será redirecionado para o login.

Acesse com os logins citados no início. Veja que super e admin têm acesso completo e manager tem acesso somente a customers.

**Sugestão de Plugins**

Para um maior controle e mais recursos use um dos plugins abaixo:

Acl da equipe do Cake, via terminal/prompt:

<https://github.com/cakephp/acl>

Um ótimo exemplo de uso

<https://github.com/mattmemmesheimer/cakephp-3-acl-example>

Plugim com administração via interface web

<https://github.com/ribafs/admin-br>

**Mais detalhes em:**

<https://book.cakephp.org/3.0/en/controllers/components/authentication.html>

<https://book.cakephp.org/3.0/en/tutorials-and-examples/cms/authentication.html>

<https://book.cakephp.org/3.0/pt/tutorials-and-examples/blog-auth-example/auth.html>



## 13.8 – Autenticação e Autorização Simples

Criar um aplicativo tipo blog com CakePHP 3 e autenticação simples

Tomando como ponto de partida o exemplo de aplicativo blog do site oficial do CakePHP:

<https://book.cakephp.org/3.0/pt/tutorials-and-examples/blog/blog.html>

Basicamente esta parte:

<https://book.cakephp.org/3.0/pt/tutorials-and-examples/blog-auth-example/auth.html>

### Banco

-- Primeiro, criamos a tabela articles

```
CREATE TABLE articles (  
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
    title VARCHAR(50),  
    body TEXT,  
    created DATETIME DEFAULT NULL,  
    modified DATETIME DEFAULT NULL  
);
```

-- Então inserimos articles para testes

```
INSERT INTO articles (title,body,created)  
    VALUES ('The title', 'This is the article body.', NOW());  
INSERT INTO articles (title,body,created)  
    VALUES ('A title once again', 'And the article body follows.', NOW());  
INSERT INTO articles (title,body,created)  
    VALUES ('Title strikes back', 'This is really exciting! Not.', NOW());
```

```
CREATE TABLE users (  
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
    username VARCHAR(50),  
    password VARCHAR(255),  
    role VARCHAR(20),  
    created DATETIME DEFAULT NULL,  
    modified DATETIME DEFAULT NULL  
);
```

### Criar aplicativo

```
cd var/www/html  
cacomposer.phar create-project --prefer-dist cakephp/app blog
```

### Configurar o banco em config/app.php

Configurar as rotas em config/routes.php para Users/login

## Gerar o código

bin/cake bake all articles

bin/cake bake all users

**Adicionar ao** src/Controller/UsersController.php

```
use Cake\Event\Event;
```

```
public function beforeFilter(Event $event)
{
    parent::beforeFilter($event);
    $this->Auth->allow('add');
}
```

**Alterar o** src/Template/Users/login.ctp, apenas a linha do username para que receba o foco automaticamente ao ser aberta:

```
<?= $this->Form->input('username', ['autofocus' => true]) ?>
```

**Adicione ao** src/Controller/AppController.php, logo ao final do initialize():

```
$this->loadComponent('Auth', [
    'loginRedirect' => [
        'controller' => 'Articles',
        'action' => 'index'
    ],
    'logoutRedirect' => [
        'controller' => 'Pages',
        'action' => 'display',
        'home'
    ]
]);
```

**Ao final da classe:**

```
public function beforeFilter(Event $event)
{
    $this->Auth->allow(['index', 'view', 'display']);
}
```

**Adicione ao início do** src/Controller/UsersController.php

```
public function beforeFilter(Event $event)
{
    parent::beforeFilter($event);
    // Permitir aos usuários se registrarem e efetuar logout.
    // Você não deve adicionar a ação de "login" a lista de permissões.
    // Isto pode causar problemas com o funcionamento normal do AuthComponent.
    $this->Auth->allow(['add', 'logout']);
}
```



```

public function login()
{
    if ($this->request->is('post')) {
        $user = $this->Auth->identify();
        if ($user) {
            $this->Auth->setUser($user);
            return $this->redirect($this->Auth->redirectUrl());
        }
        $this->Flash->error(__('Usuário ou senha inválido, tente novamente'));
    }
}

```

```

public function logout()
{
    return $this->redirect($this->Auth->logout());
}

```

### Adicionar o suporte ao bcrypt para o hash das senhas

Editar src/Model/Entity/User.php e adicionar:

```
use Cake\Auth\DefaultPasswordHasher;
```

```

protected function _setPassword($password)
{
    if (strlen($password) > 0) {
        return (new DefaultPasswordHasher)->hash($password);
    }
}

```

### Criar o src/Template/Users/login.ctp contendo:

```

<div class="users form">
<?=$this->Flash->render('auth') ?>
<?=$this->Form->create() ?>
    <fieldset>
        <legend><?=__('Por favor informe seu usuário e senha') ?></legend>
        <?=$this->Form->input('username') ?>
        <?=$this->Form->input('password') ?>
    </fieldset>
    <?=$this->Form->button(__('Login')); ?>
    <?=$this->Form->end() ?>
</div>

```

### Execute no mysql

```
ALTER TABLE articles ADD COLUMN user_id INT(11);
```

### Edite o ApplicationController.php e mude o Auth para:

```
$this->loadComponent('Auth', [
```

```

'authorize' => ['Controller'], // Adicione está linha
'loginRedirect' => [
    'controller' => 'Articles',
    'action' => 'index'
],
'logoutRedirect' => [
    'controller' => 'Pages',
    'action' => 'display',
    'home'
]
]);

```

### Logo ao final:

```

public function isAuthorized($user)
{
    // Admin pode acessar todas as actions
    if (isset($user['role']) && $user['role'] === 'admin') {
        return true;
    }

    // Bloqueia acesso por padrão
    return false;
}

```

### Adicione ao início do src/Controller/ArticlesController.php

```

public function isAuthorized($user)
{
    // Todos os usuários registrados podem adicionar artigos
    if ($this->request->getParam('action') === 'add') {
        return true;
    }

    // Apenas o proprietário do artigo pode editar e excluir
    if (in_array($this->request->getParam('action'), ['edit', 'delete'])) {
        $articleId = (int)$this->request->getParam('pass.0');
        if ($this->Articles->isOwnedBy($articleId, $user['id'])) {
            return true;
        }
    }

    return parent::isAuthorized($user);
}

```

### Adicione ao src/Model/Table/ArticlesTable.php

```

public function isOwnedBy($articleId, $userId)
{
    return $this->exists(['id' => $articleId, 'user_id' => $userId]);
}

```

## Acesse

[http://localhost/auth\\_blog/users/add](http://localhost/auth_blog/users/add)

## Adicione um usuário

Faça login com ele

[http://localhost/auth\\_blog](http://localhost/auth_blog)

Faça logout

[http://localhost/auth\\_blog/users/logout](http://localhost/auth_blog/users/logout)

## Veja que ele volta para a home do controller Pages.

Vamos mudar isso para que volte para o Users/login

Altere estas linhas no ApplicationController:

```
'logoutRedirect' => [  
    'controller' => 'Users',  
    'action' => 'login',  
    'home'  
]
```

Caso já tenhamos cadastrado todos os usuários do aplicativo devemos remover o add de:

```
public function beforeFilter(Event $event)  
{  
    parent::beforeFilter($event);  
    $this->Auth->allow(['logout']);  
}
```

Pronto. Nosso aplicativo agora somente será acessado por quem autorizarmos.

## Bom exemplo:

<https://lornajane.net/posts/2016/simple-access-control-cakephp3>



## 14 – Trabalhando com o Código do CakePHP 3

*Sorte é o que acontece quando a preparação encontra a oportunidade.*

Fluxo de Informações entre controllers, models e view/templates

**Supondo que temos no aplicativo cliente, com apenas as tabelas users, groups e customers:**

Controller/CustomersController.php

Model/CustomersTable.php

Template/Customers/index.ctp

**Criar em src/Model/CustomersTable.php a função:**

```
public function teste(){
    $query = $this->find('all', [
        'order' => ['Customers.id' => 'ASC']
    ]);
    $row = $query->first(); // Ou ->last()
    print "Model<br>";
    return $row->name;
}
```

**Chamar no index() do src/Controller/CustomersController.php:**

```
public function index()
{
    // Adicionar as 3 linhas abaixo
    print "Controller<br>";

    // Mostrar o primeiro name:
    print $this->Customers->teste();exit;

    $customers = $this->paginate($this->Customers);

    $this->set(compact('customers'));
}
```

**Chamar pela web**

<http://localhost/clientes/users/index>

Mostrará, pois primeiro foi ao controller, depois foi ao model, voltou ao controller e mostrou na view

Controller

Model

Nome de um customer

## Listagem de Users

### View

Veja a ordem:

- 1) O controller recebe a requisição do usuário para mostrar o customer:  
`http://localhost/clientes/customers/index`
- 2) O Controller envia para o Model pedindo o primeiro name
- 3) O Model processa e devolve
- 4) Então o controller envia para a view o primeiro name de customer

**Para receber strings com segurança nos forms, usar a função `h()`:**

```
<!-- File: src/Template/Articles/view.ctp -->
```

```
<h1><?= h($article->title) ?></h1>  
<p><?= h($article->body) ?></p>
```

**Testar se uma requisição é realmente post:**

```
if ($this->request->is('post')) {
```

### Debugar

```
debug($variavel);exit;
```

**Carregar model que não é o default mas associado a este controller**

```
$cliente=$this->loadModel('Clientes');
```

### Recebendo o nome da action atual

```
$action=$this->request->getParam('action');  
print $action;
```

ou

```
use Cake\Routing\Router;  
echo Router::getRequest()->params['action'];
```

### Recebendo nome do Controller atual

```
$controller = $this->request->getParam('controller');  
print $controller;
```

```
echo ucfirst(Router::getRequest()→params['controller']).'.Router::getRequest()→params['action'];
```

**ou**

```
use Cake\Routing\Router;  
print Router::getRequest()→params['controller'];
```

Usando:

```
<?php echo $title_for_layout .' - '. ucfirst(Router::getRequest()-  
>params['controller']).'.Router::getRequest()→params['action']; ?>
```

\$title\_for\_layout é definida no beforeFilter(), como na página 46.

### **Capturar nome do action atual:**

```
$this->request->action
```

### **Capturar nome do controller:**

```
$this->request->controller
```

### **Componente Flash**

Uma forma eficiente de enviar mensagens do controller para as views

```
$this->Flash->msg
```

*msg = success, set, error*

```
public function index()
```

```
{  
$controller = $this->request->getParam('controller');  
$this->Flash->success(__('O nome deste controller é: '.$controller));
```

### **Acesso a Banco de Dados**

O trabalho com bancos de dados no Cake é feito com dois objetos Tables (lida com coleções da dados, tabela, por exemplo) e Entities (lida com apenas um registro).

### **Para trabalhar com Tabelas num controller**

Carregar o objeto Table

```
use Cake\ORM\TableRegistry;
```

Carregar o respectivo objeto

```
$clientes = TableRegistry::get('Clientes');
```

Agora pode trabalhar com seu conteúdo.

```
use Cake\Datasource\ConnectionManager;
```

```
$dsn = 'mysql://root:password@localhost/my_database';  
ConnectionManager::config('default', ['url' => $dsn]);  
$conn = ConnectionManager::get('default');
```

```
$results = $conn->execute('SELECT * FROM articles')->fetchall('assoc');
```

Também podemos usar query builder.

## Redirecionamento para outras páginas

### controller/action

```
$this->redirect(['controller' => 'Clientes', 'action' => 'index'])
```

### URL

```
$this->redirect('http://ribafs.org')
```

### Action do controller atual

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

ou

```
$this->setAction('index')
```

Para o próprio link de onde veio

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

## Datasource

Se você precisar de mais controle sobre suas consultas, você pode fazer uso de instruções preparadas. Isso permite que você se comunique diretamente com o driver de banco de dados e enviar qualquer consulta personalizada que você queira:

```
$db = $this->getDataSource();  
$db->fetchAll(  
    'SELECT * from users where username = ? AND password = ?',  
    array('jhon', '12345')  
);  
$db->fetchAll(  
    'SELECT * from users where username = :username AND password = :password',  
    array('username' => 'jhon', 'password' => '12345')  
);
```



Outro exemplo:

```
$query = "SELECT * FROM user WHERE id=:user_id"
$data = $this->getDataSource()->fetchAll($query, array("usery_id" => $user_id), array("cache"
=> false));
```

### Retornando os nomes de todas as tabelas do banco atual

```
$conn = ConnectionManager::get('default');
$driver = $conn->config()['driver'];

if($driver == 'Cake\Database\Driver\Postgres'){
    $tables = $conn->execute("SELECT relname FROM pg_class WHERE relname !~ '^(\pg_|
sql_)' AND relkind = 'r';")->fetchAll();
}elseif($driver=='Cake\Database\Driver\Mysql'){
    $tables = $conn->execute("SHOW tables")->fetchAll();
}
print $tables;
```

### Identificar SGBD (mysql ou postgres):

#### Add ao início

use Cake\Datasource\ConnectionManager;

```
$conn = ConnectionManager::get('default');
$driver = $conn->config()['driver']; // Outros: database, etc.
if($driver == 'Cake\Database\Driver\Postgres'){
    $this->paginate = [
        'contain' => ['Users'],
        'conditions' => ['or' => [
            'Customers.name ilike' => '%' . $this->request->getQuery('search') . '%',
            'Customers.phone ilike' => '%' . $this->request->getQuery('search') . '%'
        ]],
        'order' => ['Customers.id' => 'DESC' ]
    ];
}elseif($driver=='Cake\Database\Driver\Mysql'){
    $this->paginate = [
        'contain' => ['Users'],
        'conditions' => ['or' => [
            'Customers.name like' => '%' . $this->request->getQuery('search') . '%',
            'Customers.phone like' => '%' . $this->request->getQuery('search') . '%'
        ]],
        'order' => ['Customers.id' => 'DESC' ]
    ];
}else{
    print '<h2>Driver database dont supported!';
    exit;
}
```



## 14.1 - Repassando Informações Entre Controllers

*Um dos mais importantes ingredientes na fórmula do sucesso é saber como lidar com as pessoas.*

Como faço para repassar informações entre dois controllers, exemplo:

Eu tenho no meu banco de dados duas tabelas, uma de usuários e outra de livros, aí eu seleciono um livro que está na tabela livros, pego seu id e tenho que atualizar essa informação na tabela do usuário..

Eu tenho dois controllers, um de usuários e outro de livros, eu tenho que repassar o id do livro para o controller de usuários..

Como passo essa informação entre controllers  
Como seleciono o id do usuário logado no sistema  
Qual o jeito correto de fazer essa atualização

### Autenticação de e-mail

Crédito:

<https://www.youtube.com/watch?v=cEwf9PpbMcQ>

config/app.php

```
...
'EmailTransport' => [
    'default' => [
        'className' => 'Mail',
        // The following keys are used in SMTP transports
        'host' => 'localhost',
        'port' => 25,
        'timeout' => 30,
        'username' => 'user',
        'password' => 'secret',
        'client' => null,
        'tls' => null,
        'url' => env('EMAIL_TRANSPORT_DEFAULT_URL', null),
    ],
    'gmail' => [
        'className' => 'Smtplib',
        'host' => 'ssl://smtp.gmail.com',
        'port' => 465,
        'timeout' => 30,
        'username' => 'ribafs@gmail.com',
        'password' => 'zmxn1029g',
        'client' => null,
        'tls' => null,
        'url' => env('EMAIL_TRANSPORT_DEFAULT_URL', null)
    ],
],
```

## Profile

```
'Email' => [  
    'default' => [  
        'transport' => 'default',  
        'from' => 'you@localhost',  
        //'charset' => 'utf-8',  
        //'headerCharset' => 'utf-8',  
    ],  
    'ribaportal' => [  
        'transport' => 'gmail',  
        'from' => ['ribafs@gmail.com'=>'Portal do RibaFS'],  
    ],  
],
```

## Criar classe mailer usando o bake

bin/cake bake mailer user

## Criar classes no controller e no model

Abrir src/Controller/Mailer/UserMailer.php

Adicionar após a linha  
static ...

```
public function welcome($user)  
{  
    $this->to($user->email)  
    ->profile('ribaportal')  
    ->emailFormat('html')  
    ->template('welcome_email_template')  
    ->layout('default')  
    ->viewVars(['nome' => $user->name])  
    ->subject(sprintf('Bem vindo, %s', $user->name));  
}
```

## Criar o template para o e-mail

src/Template/Email/html/welcome\_email\_template.ctp

```
<h1>Seja bem-vindo(a), <?php echo $nome;?></h1>  
<p>  
    Apenas um teste de envio de e-mail pelo CakePHP.  
</p>
```

Para testar crie um novo usuário e veja se ele enviou o e-mail.

## Recuperação de senha

<https://www.youtube.com/watch?v=cEwf9PpbMcQ>

Capturar numa view dados do user logado:

Capturar o group\_id:

```
$gi=$this->request->session()->read('Auth.User.group_id');
```

```
dd($gi);
```

## Campos Virtuais

São campos que não vêm de tabelas e existem apenas nos formulários. Um bom exemplo é o de confirmação de senha.

Também podemos criar um para calcular a idade de alguém ou para efetuar consultas a bancos de dados no model.

Crédito e Detalhes:

<http://www.naidim.org/cakephp-3-tutorial-11-virtual-fields>



## 14.2 - Conhecendo o CSS default do CakePHP 3

*A lógica pode levar de um ponto A a um ponto B. A imaginação pode levar a qualquer lugar.  
(Albert Einstein)*

A tela principal do aplicativo Customers mostrando a view index.ctp

Id	Name	Birthday	Phone	Created	Modified	Actions
1	Brennan G. Wilcox	4/15/16	(851) 190-1314			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
2	Chase Summers	8/27/16	(846) 297-4733			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
3	Sonia L. Mckay	12/2/15	(131) 453-1690			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
4	Isadora L. Bowers	10/24/15	(939) 798-4625			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
5	Sophia Cochran	6/15/17	(811) 687-0491			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
6	Maxwell T. Burton	1/12/16	(147) 962-3265			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
7	Desiree Y. Henry	7/21/17	(148) 711-3747			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
8	Asher Key	11/7/15	(355) 668-5871			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
9	Tyler Castro	8/31/16	(567) 793-5061			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
10	Rudyard Weber	10/26/15	(445) 457-4552			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
11	Allen Austin	4/15/16	(758) 867-2179			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>

Agora mostrarei por cada região da tela, o respectivo código e mais abaixo as classes usadas no CSS default do CakePHP 3 para estas regiões.

### No layout default.ctp

Quase todas as classes das 4 views defaults criadas pelo bake e do layout.ctp estão abaixo

**layout.ctp** (as imagens abaixo do código):

```
<nav class="top-bar expanded" data-topbar role="navigation">
```



```
<ul class="title-area large-3 medium-4 columns">
  <li class="name">
    <h1><a href=""><?=$this->fetch('title')?</a></h1>
  </li>
</ul>
```



```
<div class="top-bar-section">
  <ul class="right">
```

```

        <li><a target="_blank"
href="https://book.cakephp.org/3.0/">Documentation</a></li>
        <li><a target="_blank" href="https://api.cakephp.org/3.0/">API</a></li>
    </ul>
</div>

```

Aqui embaixo o Cake joga o conteúdo da view/template atual:

```

<div class="container clearfix">
    <?= $this->fetch('content') ?> <!-- Aqui ficará o código da view atual -->
</div>

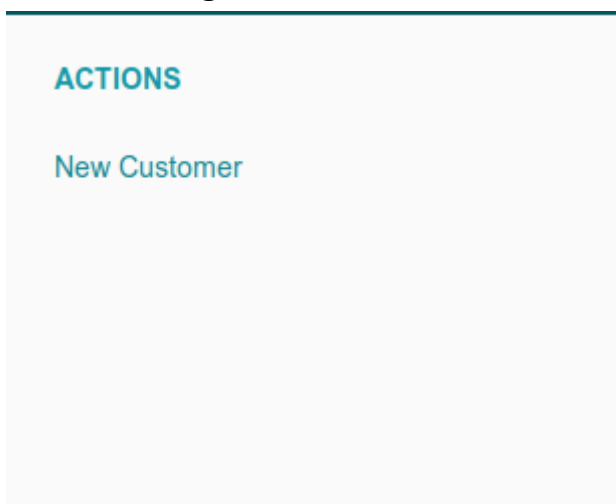
```

**index.ctp:**

```

<nav class="large-3 medium-4 columns" id="actions-sidebar">

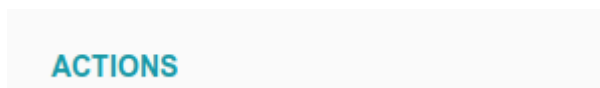
```



```

<ul class="side-nav">
    <li class="heading"><?= __('Actions') ?></li>

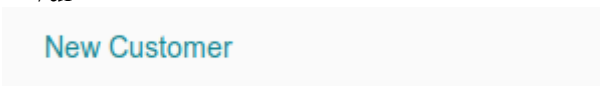
```



```

<li><?= $this->Html->link(__('New Customer'), ['action' => 'add']) ?></li>
</ul>

```



```

</nav>
<div class="customers index large-9 medium-8 columns content">

```



## Customers

Id	Name	Birthday	Phone	Created	Modified	Actions
1	Brennan G. Wilcox	4/15/16	(851) 190-1314			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
2	Chase Summers	8/27/16	(846) 297-4733			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
3	Sonia L. Mckay	12/2/15	(131) 453-1690			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
4	Isadora L. Bowers	10/24/15	(939) 798-4625			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
5	Sophia Cochran	6/15/17	(811) 687-0491			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
6	Maxwell T. Burton	1/12/16	(147) 962-3265			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
7	Desiree Y. Henry	7/21/17	(148) 711-3747			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
8	Asher Key	11/7/15	(355) 668-5871			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>

**As classes em negrito acima estão no style.css ou no base.css citadas abaixo:**

```
.disabled a,  
a.disabled {  
  pointer-events: none;  
}
```

```
.top-bar-section {  
  margin-top: 2.8125rem;  
  z-index: 98  
}
```

```
.right {  
  float: right !important  
}
```

```
.top-bar {  
  background: #333;  
  height: 2.8125rem;  
  line-height: 2.8125rem;  
  margin-bottom: 0;  
  overflow: hidden;  
  position: relative  
}
```

```
top-bar e title-area  
.top-bar.expanded .title-area {  
  background: #01545b;  
}
```

```
.top-bar.expanded, .top-bar, .top-bar-section ul li, .top-bar-section li:not(.has-form) a:not(.button) {
```

```

    background: #116d76;
}

div.message {
    text-align: center;
    cursor: pointer;
    display: block;
    font-weight: normal;
    padding: 0 1.5rem 0 1.5rem;
    transition: height 300ms ease-out 0s;
    background-color: #a0d3e8;
    color: #626262;
    top: 15px;
    right: 15px;
    z-index: 999;
    overflow: hidden;
    height: 50px;
    line-height: 2.5em;
}
div.message.error {
    background-color: #C3232D;
    color: #FFF;
}
.content {
    padding: 2rem;
}

.container {
    overflow: hidden;
    min-height: 92%; /* full height almost always */
}
.column,
.columns {
    position: relative;
    padding-left: 0.9375rem;
    padding-right: 0.9375rem;
    float: left
}
...
.large-3 {
    width: 25%
}
...
.large-9 {
    width: 75%
}
...
#actions-sidebar {
    background: #fafafa;
}

```

```

.side-nav {
  display: block;
  font-family: "Helvetica Neue", Helvetica, Roboto, Arial, sans-serif;
  list-style-position: outside;
  list-style-type: none;
  margin: 0;
  padding: 0.875rem 0
}

.side-nav li {
  font-size: 0.875rem;
  font-weight: normal;
  margin: 0 0 0.4375rem 0
}

.side-nav li.heading {
  color: #008CBA;
  font-size: 0.875rem;
  font-weight: bold;
  text-transform: uppercase
}

```

A finalidade deste trabalho é oferecer informações que ajudem a trocar o template default do Cake para um framework como o Bootstrap, o foundation, o Materialize ou outro e mesmo para um criado com o CSS puro.

## Implementando Bootstrap 3 em aplicativos do CakePHP 3

Uma forma bem simples de implementar o Bootstrap 3 em aplicativos do CakePHP 3 é criar um aplicativo com o CakePHP 3 atual, instalar e habilitar o [plugin admin-br](#) atual e então gerar o código com o bake usando o plugin admin-br.

O código gerado tem o Bootstrap 3 implementado.

Basta então ver o código gerado no src/Template para qualquer das tabelas e adaptar para seu aplicativo.

### Informação recebida do Admad no fórum oficial do CakePHP:

O css default usado no CakePHP é um subconjunto do framework **Foundation v5**.



## 14.3 - Trabalhando com session no CakePHP

*Nossa maior fraqueza está em desistir. A maneira certa de ter sucesso é tentar apenas mais uma vez. (Thomas Edison)*

### Mudando o session de php para database

Primeiro configurar em config/app.php

Ao final:

```
'Session' => [  
    'defaults' => 'php',  
],
```

Mudar para

```
'Session' => [  
    'defaults' => 'database',  
],
```

### Veja que as alternativas são:

- \* - 'php' - Uses settings defined in your php.ini.
- \* - 'cake' - Saves session files in CakePHP's /tmp directory.
- \* - 'database' - Uses CakePHP's database sessions.
- \* - 'cache' - Use the Cache class to save sessions.

### E existem várias opções:

- \* - `cookie` - The name of the cookie to use. Defaults to 'CAKEPHP'. Avoid using `.` in cookie names,
  - \* as PHP will drop sessions from cookies with `.` in the name.
- \* - `cookiePath` - The url path for which session cookie is set. Maps to the `session.cookie\_path` php.ini config. Defaults to base path of app.
- \* - `timeout` - The time in minutes the session should be valid for.
  - \* Pass 0 to disable checking timeout.
  - \* Please note that php.ini's session.gc\_maxlifetime must be equal to or greater than the largest Session['timeout'] in all served websites for it to have the desired effect.
- \* - `defaults` - The default configuration set to use as a basis for your session.
  - \* There are four built-in options: php, cake, cache, database.
- \* - `handler` - Can be used to enable a custom session handler. Expects an array with at least the `engine` key, being the name of the Session engine class to use for managing the session. CakePHP bundles the `CacheSession` and `DatabaseSession` engines.
- \* - `ini` - An associative array of additional ini values to set.

## Exemplo:

```
'Session' => [  
    'defaults' => 'php',  
    'timeout' => 12 * 60, //in minutes  
    'cookie' => 'application-name',  
],
```

Configurar por quanto tempo o usuário ficará conectado no aplicativo. Sempre que me logo, horas depois pede pra mim me logar de novo.

Em config/app.php:

```
'Session' => [  
    'defaults' => 'php',  
    'timeout' => 24*60, //minutos, que dá 24horas  
    'cookie' => 'CAKEPHP',  
],
```

**Sempre que alterar o cache limpe o cache e arquivos temporários.**

**Após efetuar as alterações importe o script de:**

config/schema/session.sql para seu banco de dados

## Algumas funções nativas e globais do CakePHP 3

`__(string $string_id[, $formatArgs])` - Manipula localizações nas aplicações do CakePHP. Usada por padrão nas views geradas pelo bake para mostrar strings. Exemplo:  
<?= \_\_('Ações') ?>

**debug**(mixed \$var, boolean \$showHtml = null, \$showFrom = true)

**dd**(mixed \$var, boolean \$showHtml = null) - semelhante a debug() mas também encerra o processamento ao final

**pr**(mixed \$var) - Encapsula print\_r(), com a adição de tags <pre> ao redor da saída.

**pj**(mixed \$var) - semelhante a pr() mas para saída em json

**h**(string \$text, boolean \$double = true, string \$charset = null) - Encapsulamento da função htmlspecialchars(). Importante nas views. Usar h() quando mostrando informações para prevenir problemas de HTML injection.

## Algumas Constantes do Core

APP - Path para o diretório da aplicação. Retorna /var/www/html/nomeApp/src

CAKE - path do diretório dos arquivos do Cake. No caso este: vendor/cakephp/cakephp/src/

CORE\_PATH - Path para este diretório: vendor/cakephp/cakephp/, com a barra final

DS - Retorna a barra "/"

ROOT - Retorna /var/www/html/nomeApp

TMP - Retorna /var/www/html/nomeApp/tmp/

VENDORS - Retorna /var/www/html/nomeApp/config/

WWW\_ROOT - Retorna /var/www/html/nomeApp/webroot/

TIME\_START - Unix timestamp em microsegundos como um float desde que a aplicação iniciou.

SECOND - 1

MINUTE - 60

HOUR - 3600

DAY

WEEK

MONTH

YEAR - 31536000 (segundos)

### Mais detalhes em:

<https://book.cakephp.org/3.0/en/core-libraries/global-constants-and-functions.html>

Exemplo de uso:

Em um controller, adicione a linha abaixo no action/método index()

```
dd(TIME_START);
```

### Enviar informações do controller para uma view:

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

### Enviar informações da view para o controller

#### Na view search.tcp:

```
$this->Form->create('Model', array('type' => 'get', 'action' => 'search'));  
$this->Form->input('select_tfield_id', array('type' => 'select'));  
$this->Form->input('value');  
$this->Form->end('submit');
```

#### No controller:

```
function search() {  
    $url = $this->params['url'];  
    $id = $url['select_tfield_id'];  
    $value = $url['value'];  
}
```





## 14.4 - Comunicação entre Model e Controller

*Habilidade é o que você é capaz de fazer. Motivação determina o que você faz. Atitude determina o quão bem você faz. (Lou Holtz)*

Criar function abaixo no model ArticlesTable.php:

```
public function teste()  
{  
    return 'Funciona';  
}
```

**Chamar no ArticlesController.php :**

```
use Cake\ORM\TableRegistry;
```

```
$articles = TableRegistry::get('Articles');  
$func=$articles->teste();// Redebe do Model
```

```
//dd($func);  
$this->set('func',$func); // Envia para a view
```

**Na view**

```
print $func  
ou  
debug($func)
```

**Redirecionamento para outras páginas**

**controller/action**

```
$this->redirect(['controller' => 'Clientes', 'action' => 'index'])
```

**URL**

```
$this->redirect('http://ribafs.org')
```

**Action do controller atual**

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

**Para o próprio link de onde veio**

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

**Carregar model que não é o default em Controller**

```
$this->loadModel('Clientes');
```

### **Upload de arquivos com o CakePHP 3**

<http://www.naidim.org/cakephp-3-tutorial-12-file-uploads>

### **Como integrar o login do Facebook com o CakePHP 3**

<https://www.startutorial.com/articles/view/how-to-integrate-facebook-login-with-cakephp-3>

### **Instalando o Wordpress em um aplicativo CakePHP 3**

<https://stackoverflow.com/questions/42891150/installing-wordpress-inside-cakephp-3>

<https://www.startutorial.com/articles/view/installing-wordpress-in-a-cakephp-application>

## 15 - Migrations

*Quem não evita as pequenas faltas, pouco a pouco cai nas grandes.* (T. Kempis)

Migrations é um plugin suportado pela equipe oficial do CakePHP que ajuda você a fazer mudanças no schema do banco de dados utilizando arquivos PHP, que podem ser versionados utilizando um sistema de controle de versão.

Ele permite que você atualize suas tabelas ao longo do tempo. Ao invés de escrever modificações de schema via SQL, este plugin permite que você utilize um conjunto intuitivo de métodos para fazer mudanças no seu banco de dados.

Esse plugin é um wrapper para a biblioteca Phinx(<https://phinx.org/>).

Uma migração é basicamente um arquivo PHP que descreve as mudanças a serem feitas no banco de dados. Um arquivo de migração pode criar ou excluir tabelas, adicionar ou remover colunas, criar índices e até mesmo inserir dados em seu banco de dados.

Uma migração é basicamente um arquivo PHP que descreve as mudanças a serem feitas no banco de dados. Um arquivo de migração pode criar ou excluir tabelas, adicionar ou remover colunas, criar índices e até mesmo inserir dados em seu banco de dados.

Muito útil para quando desejamos distribuir um plugin ou aplicativo do CakePHP, já incluindo os arquivos de migration e/ou seed quando o usuário apenas executa o comando para criar as tabelas e adicionar os registros.



## 15.1 - Migrate

*Não se preocupe com falhas, preocupe-se com as chances que você perde quando não tenta.  
(Jack Canfield)*

Aqui segue um exemplo de migração:

```
<?php
use Migrations\AbstractMigration;

class CreateProducts extends AbstractMigration
{
    /**
     * Change Method.
     *
     * More information on this method is available here:
     * http://docs.phinx.org/en/latest/migrations.html#the-change-method
     * @return void
     */
    public function change()
    {
        $table = $this->table('products');
        $table->addColumn('name', 'string', [
            'default' => null,
            'limit' => 255,
            'null' => false,
        ]);
        $table->addColumn('description', 'text', [
            'default' => null,
            'null' => false,
        ]);
        $table->addColumn('created', 'datetime', [
            'default' => null,
            'null' => false,
        ]);
        $table->addColumn('modified', 'datetime', [
            'default' => null,
            'null' => false,
        ]);
        $table->create();
    }
}
```

### Salvar em

config/Migrations/20190519085224\_Products.php

Veja o formato: YYYYMMDDHHMMSS\_Products.php

## **Também podemos criar a migration usando o bake:**

*bin/cake bake migration CreateProducts name:string description:text created modified*

Você pode criar um arquivo de migração vazio caso deseje ter um controle total do que precisa ser executado. Para isto, apenas omita a definição das colunas:

*bin/cake migrations create MyCustomMigration*

## **Criar a tabela ou as tabelas do(s) Migration(s)**

*bin/cake migrations migrate*

## **Limpar todo o banco, excluindo todas as tabelas**

*bin/cake migrations rollback*

## **Nomes de campos e tipos**

fieldName:fieldType[length]:indexType:indexName

Por exemplo, veja formas válidas de especificar um campo de e-mail:

- email:string:unique
- email:string:unique:EMAIL\_INDEX
- email:string[120]:unique:EMAIL\_INDEX

O parâmetro length para o fieldType é opcional e deve sempre ser escrito entre colchetes

Os campos created e modified serão automaticamente definidos como datetime.

Os tipos de campos são genericamente disponibilizados pela biblioteca Phinx. Eles podem ser:

- string
- text
- integer
- bigint
- float
- decimal
- datetime
- timestamp
- time
- date
- binary
- boolean
- uuid

Há algumas heurísticas para a escolha de tipos de campos que não são especificados ou são definidos com valor inválido. O tipo de campo padrão é string;

- id: integer
- created, modified, updated: datetime

## Adicionando colunas a uma tabela existente

Se o nome da migração na linha de comando estiver na forma "AddXXXXToYYY" e for seguido por uma lista de nomes de colunas e tipos, então o arquivo de migração com o código para criar as colunas será gerado:

```
bin/cake bake migration AddPriceToProducts price:decimal
```

A linha de comando acima irá gerar um arquivo com o seguinte conteúdo:

```
<?php
use Migrations\AbstractMigration;

class AddPriceToProducts extends AbstractMigration
{
    public function change()
    {
        $table = $this->table('products');
        $table->addColumn('price', 'decimal')
            ->update();
    }
}
```

## Especificando o tamanho do campo

Se você precisar especificar o tamanho do campo, você pode fazer isto entre colchetes logo após o tipo do campo, ex.:

```
bin/cake bake migration AddFullDescriptionToProducts full_description:string[60]
```

Executar o comando acima irá gerar:

```
<?php
use Migrations\AbstractMigration;

class AddFullDescriptionToProducts extends AbstractMigration
{
    public function change()
    {
        $table = $this->table('products');
        $table->addColumn('full_description', 'string', [
            'default' => null,
            'limit' => 60,
            'null' => false,
        ])
            ->update();
    }
}
```

Se o tamanho não for especificado, os seguintes padrões serão utilizados:

- string: 255

- integer: 11
- biginteger: 20

### Removendo uma coluna de uma tabela

Da mesma forma, você pode gerar uma migração para remover uma coluna utilizando a linha de comando, se o nome da migração estiver na forma "RemoveXXXFromYYY":

```
bin/cake bake migration RemovePriceFromProducts price
```

Cria o arquivo:

```
<?php
use Migrations\AbstractMigration;

class RemovePriceFromProducts extends AbstractMigration
{
    public function change()
    {
        $table = $this->table('products');
        $table->removeColumn('price');
    }
}
```

### Gerando migrações a partir de uma base de dados existente

Se você está trabalhando com um banco de dados pré-existente e quer começar a usar migrações, ou para versionar o schema inicial da base de dados da sua aplicação, você pode executar o comando `migration_snapshot`:

```
bin/cake bake migration_snapshot Financas
```

Ele gerará um arquivo de migrations com todas as tabelas do banco, mas somente as estruturas, sem os registros:

```
YYYYMMDDHHMMSS_Financas.php
```

**Migrações também podem ser executadas para plugins.** Simplesmente utilize a opção `--plugin` ou `-p`

```
bin/cake migrations migrate -p MyAwesomePlugin
```

### Lista das migrações

```
bin/cake migrations status
```



## 15.2 - Seed - inserindo registros nas tabelas

*Diante de uma dificuldade substitua o eu não consigo pelo vou tentar outra vez.*

*bin/cake bake seed Posts*

Ele gera um esqueleto de classe.

Editar config/Seeds/Posts e adicionar os registros diretamente como abaixo ou usando o Faker

```
<?php
use Phinx\Seed\AbstractSeed;

class PostsSeed extends AbstractSeed
{
    public function run()
    {
        $data = [
            [
                'body' => 'foo',
                'created' => date('Y-m-d H:i:s'),
            ],
            [
                'body' => 'bar',
                'created' => date('Y-m-d H:i:s'),
            ]
        ];

        $posts = $this->table('posts');
        $posts->insert($data)
            ->save();
    }
}
```

### **Integrando seed com a biblioteca faker**

#### **Instalar a Faker**

*composer require fzaninotto/faker*

Detalhes de uso:

<https://github.com/fzaninotto/Faker>

#### **Criar um novo seed**

*bin/cake bake seed Despesas*

Então usar na classe gerada

```
<?php
```

```
use Phinx\Seed\AbstractSeed;
```

```
class DespesasSeed extends AbstractSeed
```

```
{  
    public function run()  
    {  
        $faker = Faker\Factory::create('pt_BR');  
        $data = [];  
        for ($i = 0; $i < 20; $i++) {  
            $data[] = [  
                'descricao' => $faker->userName,  
                'valor'     => $faker->numberBetween($min = 0, $max = 9000),  
                'mes'      => $faker->regexify('0[1-9]\2019|1[1-2]\2019'),  
                'receita_id' => $faker->numberBetween($min = 1, $max = 2),  
                'created'   => date('Y-m-d H:i:s'),  
            ];  
        }  
  
        $table = $this->table('despesas');  
        $table->insert($data)->save();  
    }  
}
```

**Adicionar os registros do Seed no banco**

```
bin/cake migrations seed
```

## 15.3 - Exemplos de uso da Faker

Como tive dificuldade de encontrar alguns exemplos de uso da Faker, então seguem alguns exemplos que colecionei.

```
$faker = Faker\Factory::create('pt_BR');

$cpf = $faker->numberBetween($min = 1000000000, $max = 9999999999);
$nome = addslashes($faker->name);
$credito_liberado = $faker->regexify('[sn]');
$nascimento = $faker->date;
$email = $faker->email;
$user_id = $faker->numberBetween($min = 1, $max = 4);
$quantidade = $faker->randomNumber($nbDigits = NULL, $strict = false);
$preco_venda = $faker->numberBetween($min = 20, $max = 1200);

randomNumber($nbDigits = NULL, $strict = false) // 79907610
randomFloat($nbMaxDecimals = NULL, $min = 0, $max = NULL) // 48.8932
numberBetween($min = 1000, $max = 9000) // 8567

$faker->regexify('[sn]'); // s ou n
$faker->randomElement($array = array ('s','n'));
$faker->randomLetter;
$faker->regexify('[A-Z]+[a-z]{2,5}'); // 2 a 5 letras
$faker->regexify('[A-Z0-9._%+-.]+@[A-Z0-9.-]+\.[A-Z]{2,4}'); // sm0@y8k96a.ej
$faker->randomElement($array = array ('a','b','c')); // 'b'
print $faker->sentence($nbWords = 3, $variableNbWords = true);
$faker->sentence($nbWords = 6, $variableNbWords = true);
$faker->address; // rua, número e cep
$faker->text; // Para grandes quantidades de texto
$faker->sentence($nbWords = 6, $variableNbWords = true);
$faker->text($maxNbChars = 200);
$faker->title($gender = null|'male'|'female'); // 'Ms.'
$faker->name($gender = null|'male'|'female'); // 'Dr. Zane Stroman'
$faker->cityPrefix;
$faker->state;
$faker->stateAbbr;
$faker->buildingNumber;
$faker->city;
$faker->streetName;
$faker->streetAddress;
$faker->postcode;
$faker->country;
$faker->PhoneNumber;
$faker->company;
$faker->date($format = 'Y-m-d', $max = 'now');
$faker->time($format = 'H:i:s', $max = 'now');
$faker->freeEmail;
$faker->password;
```

```
$faker->domainName;  
$faker->url;  
$faker->ipv4;  
$faker->macAddress;  
$faker->creditCardType;  
$faker->creditCardNumber;  
$faker->creditCardExpirationDateString;  
$faker->hexcolor;  
$faker->colorName;  
$faker->fileExtension;  
$faker->mimeType;  
$faker->locale;  
$faker->countryCode;  
$faker->randomHtml(2,3);
```

A partir da versão 1.5.5 do plugin, você pode usar a shell de migrations para popular seu banco de dados. Essa função é oferecida graças ao recurso de seed da biblioteca Phinx. Por padrão, arquivos seed ficarão no diretório config/Seeds de sua aplicação. Por favor, tenha certeza de seguir as instruções do Phinx para construir seus arquivos de seed.

### **Podemos especificar um plugin**

```
bin/cake bake seed Articles --plugin PluginName
```

As opções --data, --limit e --fields foram adicionadas para exportar dados da sua base de dados. A partir da versão 16.4, o comando bake seed permite que você crie um arquivo de seed com dados exportados da sua base de dados com o uso da flag --data:

### **Exportando a tabela Articles juntamente com seus dados:**

```
bin/cake bake seed --data Articles
```

Por padrão, esse comando exportará todas as linhas encontradas na sua tabela. Você pode limitar o número de linhas a exportar usando a opção --limit:

### **Exportar apenas os 10 primeiros registros encontradas**

```
bin/cake bake seed --data --limit 10 Articles
```

### **Para popular seu banco de dados, você pode usar o subcomando seed:**

```
bin/cake migrations seed
```

### **Para plugins**

```
bin/cake migrations seed -p MeuPlugin
```

**Você pode especificar apenas um seeder para rodar usando a opção --seed:**

```
bin/cake migrations seed --seed ArticlesSeed
```

### **Limpar o cache**

Se você usa o plugin ao fazer o deploy de sua aplicação, garanta que o cache ORM seja limpo para renovar os metadados das colunas de suas tabelas.

```
bin/cake orm_cache clear
```

### **Migrando somente uma migration**

```
bin/cake migrations migrate -p CakeDC/Users
```

```
bin/cake migrations migrate -p Acl
```



## 15.4 - Resumindo

*Você nunca sabe que resultados virão da sua ação. Mas se você não fizer nada, não existirão resultados. (Mahatma Gandhi)*

**Gerando um backup do banco juntamente com os dados em Migrations:**

**Exportar banco de dados existente para uma Migration (apenas a estrutura das tabelas)**

*bin/cake bake migration\_snapshot NomeBanco*

Gera a migration para todo o banco em config/Migrations/20190520132718\_NomeBanco.php

**Exportar uma tabela juntamente com seus registros**

*bin/cake bake seed --data users*

Gera o arquivo config/Seeds/UsersSeed.php

**Limpar todo o banco com rollback**

Para remover todas as tabelas do banco execute:

*bin/cake migrations rollback*

**Restaurar todas as tabelas com seus dados**

Trazer de volta a estrutura das tabelas e os registros, após ter feito a migration e seeds.

*bin/cake migrations migrate*

*bin/cake migrations seed*

**Vantagem Extra**

Obs.: os arquivos de migrations tem mais uma vantagem. São independentes de SGBD. Funcionam em todos os SGBD suportados pelo CakePHP.

**Atualizar aplicativo para a versão atual (3.7.7)**

cd aplicativo

composer require --update-with-dependencies "cakephp/cakephp:3.7.\*"

## **Mudando de SGBD**

Imagina só, você criou seu aplicativo com o MySQL então precisa mudar para o PostgreSQL. Basta exportar as migrations e as seeds no MySQL, depois criar o banco no PostgreSQL e importar as migrations e seeds. Uma beleza!

## **Referências**

<https://book.cakephp.org/3.0/pt/appendices/3-0-migration-guide.html>

<https://book.cakephp.org/3.0/en/upgrade-tool.html>



# 16 – Ambiente de Desenvolvimento

*Sonha como se vivesses para sempre; vive como se fosses morrer hoje.* (James Dean)

A meu ver, o trabalho com programação web, especialmente para o backend (php, mysql e cia), deve ser executado num computador com Linux. Veja meus motivos:

- Como estas ferramentas nasceram no Linux
- Como o linux é menos vulnerável a virus, mawares, etc
- Especialmente como a maioria dos servidores usa Linux
- Como o linux atualmente está muito amigável para uso em desktop (Exemplo maior (para mim): Linux Mint)
- Como a instalação dos pacotes básicos deixa a atualização dos mesmo automática

Acho que é suficiente para dizer que o Linux ou similar é o sistema operacional ideal para ser usado como sistema desktop pelos programadores backend.

## Ferramentas que tornem mais produtivo o trabalho

### Editores de código

- Bem leve: Xed no Linux Mint (já vem na distribuição)
- Bons recursos: Visual Studio Code (<https://code.visualstudio.com>)

### Gerenciador de bancos de Dados

- Adminer (<https://adminer.org>)

### Ambiente de Desenvolvimento com Vagrant

Para o trabalho e mesmo para trabalhar sozinho, a adoção do Vagrant é bastante recomendada, pois podemos criar um ambiente similar ao do servidor com que estamos trabalhando para evitar conflitos. E sempre que mudarmos de projeto podemos criar uma nova box similar ao noso servidor. Cria-se uma box, similar ao servidor, com o mesmo sistema operacional, mesma versão, mesmas extensões e configurações e quando pronto exportamos e passamos uma cópia para cada integrante da equipe.

Inspirado por este livro eu instalei uma box com Ubuntu 18.04 e adicionei todos os softwares que geralmente uso:

- Apache2
- PHP 7.2
- MySQL 5.7
- PostgreSQL 10.8
- Adminer
- Diversas extensões para o PHP
- mod\_rewrite
- composer
- git

Instalei o CakePHP 3.7.7 e criei dois aplicativos, um deles com o plugin admin-br.

Entre outros.

Tudo configurado como eu geralmente faço.

Depois de pronta e testada eu exportei e enviei para a nuvem, no site do vagrant, além de criar um repositório no GitHub e documentar o processo de criação, juntamente com outras informações úteis.

Veja aqui:

<https://github.com/ribafs/cake-vagrant> e <https://ribafs.github.io/cake-vagrant>

# 17 - Referências

*Eu faço da dificuldade a minha motivação. A volta por cima vem na continuação.* Charlie Brown Jr

## Criação de Aplicativos (boa fonte de informações práticas)

<https://book.cakephp.org/3.0/pt/quickstart.html>

<https://book.cakephp.org/3.0/pt/tutorials-and-examples.html>

<https://book.cakephp.org/3.0/en/tutorials-and-examples/cms/installation.html>

## Comunidade brasileira

<https://www.cakephpbrasil.com.br/>

<https://pt-br.facebook.com/groups/cakebrasil/>

<https://pt.stackoverflow.com/questions/tagged/cakephp>

<https://www.meetup.com/pt-BR/topics/cakephp/br/>

<https://groups.google.com/forum/?hl=pt-Pt#!forum/cakephp-pt>

## Internacional

<https://discourse.cakephp.org/>

<https://www.facebook.com/CakePHP/>

<http://cakesf.herokuapp.com/>

<https://stackoverflow.com/questions/tagged/cakephp>

## Outras Boas Referências

[https://www.youtube.com/watch?v=DJh5eHD1dJM&list=PL83b6tjXNFHe-OVRROawG3YdIN\\_-Kbc6j](https://www.youtube.com/watch?v=DJh5eHD1dJM&list=PL83b6tjXNFHe-OVRROawG3YdIN_-Kbc6j) – Canal com 12 aulas

[https://www.youtube.com/playlist?list=PLmY5AEiqDWwBYKoUhTXqoMICoHc6Zd\\_Qe](https://www.youtube.com/playlist?list=PLmY5AEiqDWwBYKoUhTXqoMICoHc6Zd_Qe) – 6 aulas

<http://www.naidim.org/cakephp>

<https://www.startutorial.com/articles>

<https://www.sanisoft.com/blog/>

<https://getawesomeness.herokuapp.com/get/cakephp> (grande referência)

<https://github.com/FriendsOfCake/awesome-cakephp> (outra impressionante referência)

<https://github.com/ziadoz/awesome-php> (esta é de PHP)

<http://www.naidim.org/cakephp>

<https://www.tutorialspoint.com/cakephp/>

<https://www.codexworld.com/cakephp-3-x-tutorial-for-beginners/>

<https://riptutorial.com/cakephp-3-0>

<https://www.startutorial.com/series/view/3>

<https://dev.to/northgoingzax/cakephp-3-bake-by-example-40pp>

<https://www.devsaran.com/blog/10-resources-learn-cakephp>

<https://waltherlalk.com/blog/cakephp-3-tutorial-part-1>

<https://www.dereuromark.de/>

<https://www.toptal.com/cakephp/most-common-cakephp-mistakes>

<https://www.scoop.it/topic/cakephp-reporter>

<https://ribafs.org/> (Seção CakePHP 3)

## Referências sobre Temas

Nice Admin bake theme for CakePHPs Bake plugin

<https://github.com/davidyell/CakePHP-NiceAdminBakeScripts>

CakePHP3: Transparently use Bootstrap

<https://github.com/friendsofcake/bootstrap-ui>

Plugin to Implement ACL in CakePHP 3 applications with web interface, twitter bootstrap and others - <https://github.com/ribafs/cakephp-admin-en-bs>

Implementando Twitter Bootstrap em Aplicativo do CakePHP 3 existente ou novo

<https://github.com/ribafs/cakephp-app-bs>

Plugin para implementar o framework Bootstrap em aplicativos do CakePHP 3 e bake em português

<https://github.com/ribafs/cake-bs-br>

Implementando ACL em aplicativos do CakePHP 3 usando Bootstrap com administração via interface web - <https://github.com/ribafs/admin-br>

Plugin to add twitter bootstrap in applications with CakePHP 3

<https://github.com/ribafs/twbs-cake-css>

CakePHP 3.x helpers for the Bootstrap 3 HTML, CSS & JS framework by @Holt59.

<https://holt59.github.io/cakephp3-bootstrap-helpers/>

CakePHP 3: Bake by example

<https://dev.to/northgoingzax/cakephp-3-bake-by-example-40pp>

Code Generation with Bake

<https://book.cakephp.org/bake/1.x/en/usage.html>

CakePHP 3 tutorial part 3 - Baking a little cake

<https://waltherlalk.com/blog/cakephp-3-tutorial-part-3>

CakePHP AdminLTE Theme

<https://github.com/maiconpinto/cakephp-adminlte-theme>

CakePHP Gentelella Theme

<https://github.com/backstageel/cakephp-gentelella-theme>

<https://colorlib.com/polygon/gentelella/index.html>

cakephp3-bootstrap-theme

<https://github.com/alaxos/cakephp3-bootstrap-theme>

Integrate theme in Cakephp 3.x

<http://findnerd.com/list/view/Integrate-theme-in-Cakephp-3-x/28722/>

Configurar o CakePHP para usar o Twitter Bootstrap automaticamente

<https://www.webdevbr.com.br/configurar-o-cakephp-para-usar-o-twitter-bootstrap-automaticamente>

Theming our CMS

<http://josediazgonzalez.com/2016/12/15/theming-our-cms/>

Free Templates That can Fit with Cakephp

<https://www.cakephpexpert.com/blog/free-cakephp-templates>

Cakeswatch is adaptation of Bootswatch for CakePHP 3.x themes

<https://sherwinrobles.blogspot.com/2016/03/cakeswatch-is-adaptation-of-bootswatch.html>

Using themes in CakePHP 3

<http://tariquesani.net/blog/2014/07/21/using-themes-cakephp-3/>

**Livros sobre CakePHP 2.x**

<https://whatpixel.com/best-cakephp-books/>

<http://josediazgonzalez.com/cakephp-book/>

<https://www.amazon.com.br/Cakephp-Application-Cookbook-James-Watts/dp/1782160086>



## 18 – Algo sobre o novo CakePHP 4

*Quando você quer alguma coisa, todo o universo conspira para que você realize o seu desejo.*

### Algumas novidades na próxima versão do CakePHP:

- Removerá todos os métodos deprecateds
- Removerá todos os métodos não-estáticos de Cake\Database\Type e moverá os métodos de instância comuns para um trait.
- Exigirá pelo menos PHP 7.1
- Usa novas características do PHP 7.1
- Implementa a PSR-16 (<https://github.com/cakephp/cakephp/issues/9507>)
- Suporte para a PSR-15 (<https://github.com/cakephp/cakephp/pull/12907>)
- Atualização da documentação. Remoção de todos os "prior to..."
- Atualização dos plugins do core
- Publicação de documentação de novos plugins
- A documentação sobre plugins será movida para plugin-docs tools
- Não será compatível com aplicativos da versão 3.x. Antes de atualizar para a versão 4, atualize para a 3.8 e resolva os deprecateds warnings

### Exemplo de aviso de código deprecated:

```
deprecationWarning('TableSchema::columnType() is deprecated. Use TableSchema::setColumnType() or TableSchema::getColumnType() instead.');
```

Para desabilitar estes avisos, mudar no config/app.php:

```
'errorLevel' => E_ALL,
```

por

```
'errorLevel' => E_ALL & ~E_USER_DEPRECATED,
```

### Para atualizar para a versão 4.0.x execute:

```
php composer.phar require --update-with-dependencies "cakephp/cakephp:4.0.*"
```

Consulte

<https://book.cakephp.org/4.0/en/appendices/4-0-migration-guide.html>

### Deprecated no 3.7

Cake\Core\Plugin::load() and loadAll() are deprecated. Instead you should use Application::addPlugin().

### Trocar

```
Plugin::load('Bake');
```

### Por

```
$this->addPlugin('Bake');  
$this->addPlugin('Migrations');
```

```
bin/cake plugin load AdminlBr --bootstrap
```

Gera no src/Application.php (surgiu a partir da versão 3.6):

```
$this->addPlugin('AdminBr', ['bootstrap' => true]);
```

**Antes da versão 3.6 usava-se**

```
Plugin::load('NomePlugin');
```

A adicionava-se manualmente no config/bootstrap.php

**A partir da versão 3.6 o comando**

```
bin/cake plugin load NomePlugin
```

Adiciona uma linha não mais em config/bootstrap.php, mas em src/Application.php

Algo como:

```
$this->addPlugin('CakeAclBr', ['bootstrap' => true, 'routes' => true]);
```

```
// Desabilitar rotas para o plugin ContactManager:
```

```
$this->addPlugin(ContactManagerPlugin::class, ['routes' => false]);
```

**Detalhes em**

<https://book.cakephp.org/4.0/pt/index.html>

<https://github.com/cakephp/cakephp/wiki/4.0-Roadmap>

<https://github.com/cakephp/cakephp/wiki/4.1-Roadmap>



## 19 – Conclusão

*Importante: Podemos levar um minuto para aprender algo, mas muitos anos ou a vida inteira para dominar.*

Gostaria de destacar alguns capítulos:

3 – Convenções – Este é muito importante para que se tire o maior proveito do Cake.

5 – Gerando CRUD com o bake e muito mais – O bake é uma ferramenta que facilita muito o trabalho na criação de aplicativos, plugins, componentes, etc

9 – Validações – Muito importantes para melhorar a segurança dos aplicativos

12 – Plugins – Com plugins estendemos muito as funcionalidades do Cake e podemos tornar o trabalho com ele mais agradável e simples. Exemplo: o plugin cake-acl-br

14 – Aplicativos de Exemplo – Vários aplicativos úteis

15 – Codificação no CakePHP 3 – Aqui muitas informações úteis para customizar manualmente o código de aplicativos.

Faço sinceros votos de que este livro venha a tornar seu trabalho com o CakePHP 3 mais produtivo e prazeroso.

Feliz codificação!



## 20 – Sobre min

Sou um engenheiro civil que abandonou a engenharia para trabalhar e se divertir com a informática.

Para mim trabalhar com informática (programação web com PHP, programação mobile, administração de servidores e do SGBD PostgreSQL) causa uma grande satisfação.

Veja detalhes sobre meu trabalho com informática aqui:

<https://ribafs.org>

<https://github.com/ribafs>

Currículo resumido - <https://ribafs.org/portal/curriculo/curriculo.html>

Outros Livros publicados

<https://ribafs.org/portal/curriculo/livros.html>

<https://github.com/ribafs/livros>

Contato pelo site

<https://ribafs.org/portal/curriculo/contato.html>

### Correções e Sugestões

Gosto muito, valorizo e estudo a nossa língua e os conhecimentos básicos deste livro, mas tenho consciência de não dominá-los com perfeição, portanto correções e sugestões serão muito bem vindas e te darei o crédito por elas nas próximas edições.

*Muito agradecido pela leitura.*



## 21 – Apêndice A - Dicas sobre o Composer

### Site oficial

<https://getcomposer.org>

### Documentação

<https://getcomposer.org/doc/>

É um gerenciador de dependências para PHP, com bons recursos para instalar, atualizar e gerenciar módulos em PHP. Ele checa quais as dependências de um pacote e as instala, usando as versões indicadas.

### Pré-requisitos para sua instalação:

- PHP 5.3.2 ou superior
- curl
- php-cli
- php-mbstring
- git
- unzip

Para Windows

<https://getcomposer.org/doc/00-intro.md#using-the-installer>

Para Linux

```
sudo apt install composer
```

### Testando

```
composer list
```

### Instalando uma dependência/pacote em um projeto

```
cd projeto
```

```
composer require vendor_nome/pacote_nome
```

### Instalar um determinado release

```
composer require ribafs/admin-br:1.30
```

### Instalar a versão atual em desenvolvimento

```
composer require ribafs/admin-br:dev-master
```

Ele criará a pasta vendor no diretório atual com as dependências dentro.

### Ajuda

```
composer help require
```

## Arquivos importantes:

composer.json  
composer.lock  
autoload.php

composer.json

<https://getcomposer.org/doc/04-schema.md>

## Criando um pacote com o composer

```
mkdir projeto  
cd projeto
```

```
composer init
```

### Opções:

- name: Name of the package.
- description: Description of the package.
- author: Autor no formato: Nome <autor@email.com>
- type: Type of package.
- homepage: Homepage of the package.
- require: Pacote para o require com uma versão. No formato vendor/pacote:1.0.0.
- require-dev: Requisitos de desenvolvimento. Vide --require.
- stability (-s): Valor para o campo minimum-stability.
- license (-l): Licença do package.
- repository: Provide one (or more) custom repositories. They will be stored in the generated composer.json, and used for auto-completion when prompting for the list of requires. Every repository can be either an HTTP URL pointing to a composer repository or a JSON string which similar to what the repositories key accepts.

## Publicando projeto para ser instalado com composer:

- Hospedar o projeto no GitHub ou similar
- Publicar no Packagist e configurar a atualização automática
- Instalar com o composer

### Dicas:

- Adicione composer.lock no .gitignore em libraries
- Ordene os pacotes no require ou require-dev pelo nome
- Validar o composer.json:  

```
composer validate --no-check-all --strict
```
- Especificando duas versões do PHP:

```
"require": {  
  "php": "7.1.* || 7.2.*"  
},
```
- Configurar autoload-dev para testes:

```
"autoload": {
```

```
"psr-4": {
    "Acme\\": "src/"
}
},
"autoload-dev": {
    "psr-4": {
        "Acme\\": "tests/"
    }
}
},
```

composer install - instalar dependências  
composer update - atualizar dependências  
composer clear-cache - limpar cache

### **Desinstalar plugin**

*composer remove packageauthor/packagename --update-with-dependencies*

### **Após cada alteração no composer.json:**

*composer dump-autoload*

O Packagist.org é o principal repositório de pacotes para o Composer - <https://packagist.org/>