CakePHP – um framework Web MVC:

Análise comparativa e implementação da arquitetura Model-View-Controller.

Leonardo Cabral da Rocha Soares[1]

Resumo

Com a crescente demanda por soluções web, inúmeros frameworks web foram desenvolvidos para facilitar o desenvolvimento de soluções. Padrões de projeto surgiram para organizar o desenvolvimento, possibilitando uma maior produtividade e manutenção posterior. Dentro deste novo paradigma de desenvolvimento, o CakePHP surge como um framework web mvc poderoso, logo alcançando fama e status entre os desenvolvedores PHP. Embora reconhecido internacionalmente pela comunidade PHP, o CakePHP não é uma unanimidade, alias, não existe uma unanimidade. Cada desenvolvedor opta por um framework que melhor se encaixe em seu projeto ou com seu perfil. Não há um mecanismo formal para classificarmos um framework frente aos outros, assim, cada framework pode ser melhor na opinião de um desenvolvedor particular. Para nortear a comparação entre frameworks, o artigo *Server-Centric Web Frameworks: An Overview (2007)* reuniu os requisitos mais comumente encontrados na literatura sobre o assunto e será utilizado para analisarmos o CakePHP. O presente artigo mostrará quais requisitos são implementados pelo CakePHP e como isto é feito. Além disto, o presente artigo transcorrerá também sobre o padrão Model-View-Controller (MVC) e sua implementação pelo framework CakePHP.

Palavras-chave: CakePHP. PHP. Framework WEB MVC. Desenvolvimento Web.

---

1 Discente no curso de Especialização em Desenvolvimento de Aplicações Web
Pontifícia Universidade Católica de Minas Gerais – PUC Minas Virtual
Av. Trinta e Um de Março, nº 1020 – Bairro Dom Cabral – Belo Horizonte – MG – CEP: 30535-000
http://www.pucminas.br/virtual/

1. INTRODUÇÃO

O presente artigo dispõe sobre o framework web CakePHP sob a ótica do desenvolvimento segundo o padrão de arquitetura de software MVC (Modelo-Visão-Controlador). Serão apresentadas as características deste framework, sua origem, seus recursos e implantação. Será elaborada uma análise comparativa entre os recursos disponibilizados pelo CakePHP e os citados como requisitos funcionais no artigo *Server-Centric Web Frameworks: An Overview (2007) de Iwan Vosloo e Derrick G. Kourie.* Espera-se que este texto sirva como orientação para futuros desenvolvedores que desejem conhecer mais sobre o framework CakePHP e os recursos essenciais à todos os frameworks que este implementa.

2. CakePHP

O CakePHP é um framework web desenvolvido para dar mais agilidade e flexibilidade no desenvolvimento de sistemas web. É descrito em sua documentação oficial como:

> CakePHP is a free open-source rapid development framework for PHP. Its a structure of libraries, classes and run-time infrastructure for programmers creating web applications originally inspired by the Ruby on Rails framework. Our primary goal is to enable you to work in a structured and rapid manner - without loss of flexibility. (Em <http://book.cakephp.org/1.1/en/introduction-to-cakephp.html>. Acesso em 6 de abril de 2015).

Traduzindo livremente a citação, podemos dizer que CakePHP é um framework de desenvolvimento rápido grátis e código aberto. É uma estrutura de bibliotecas, classes e infraestrutura em tempo de execução para programadores criarem aplicações web originalmente insipiradas pelo framework Ruby on Rails. Seu objetivo primário é possilitar um desenvolvimento estruturado e rápido – sem perder flexibilidade. Seu projeto inicial foi desenvolvido por Michal Tatarynowicz em 2005. Foi publicado sob a licença MIT e aberto a comunidade de desenvolvedores que o

mantém até hoje.

Embora não haja uma unanimidade entre os desenvolvedores PHP, o CakePHP tem ganhado espaço nos centros de desenvolvimento, sendo já consolidado como uma das principais ferramentas de desenvolvimento da linguagem PHP. A figura1 exibe um quadro comparativo entre alguns frameworks web PHP.

PHP Framework	PHP4	PHP5	MVC	Multiple DB's	ORM	DB Objects	Templates	Caching	Validation	Ajax	Auth Module	Modules	EDP
Akelos	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
ash.MVC	-	✓	✓	-	-	✓	✓	-	✓	-	✓	✓	-
CakePHP	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-
CodeIgniter	✓	✓	✓	✓	-	✓	✓	✓	✓	-	-	-	-
DIY	-	✓	✓	-	✓	✓	✓	✓	-	✓	-	-	-
eZ Components	-	✓	-	✓	-	✓	✓	✓	✓	-	-	-	-
Fusebox	✓	✓	✓	✓	-	-	-	✓	-	✓	-	✓	-
PHP on TRAX	-	✓	✓	✓	✓	✓	-	-	✓	✓	-	✓	-

Figura 1: Comparação entre Frameworks Web PHP

Fonte: PHP Frameworks (Em <http://www.phpframeworks.com/>. Acesso em 06 de abril de 2015)

A documentação oficial[2] do framework lista 15 motivos pelos quais um desenvolvedor web deveria escolher o cakePHP:

1. Comunidade ativa e amigável
2. Licença flexível
3. Compatível com PHP4 e PHP5
4. Integração com banco de dados para geração de CRUD e consultas simples
5. Aplicação Scaffolding
6. Utilização da arquitetura MVC
7. URLs amigáveis
8. Validação Built-in
9. Templates rápidos e flexíveis, podendo ser utilizado helpers
10. Views helpers para ajax, javascript, HTML, formulários e outros
11. Componentes para gerenciamento de sessões e segurança
12. Lista de controle flexível
13. Data sanitization
14. Cache de view flexíveis
15. Pouca configuração no servidor web

2 Em: <http://book.cakephp.org/1.1/en/introduction-to-cakephp.html>. Acesso em 06 de abril de 2015.

3. Análise frente o artigo *Server-Centric Web Frameworks: An Overview*

O artigo *Server-Centric Web Frameworks: An Overview* apresenta uma lista de preocupações e funcionalidades que um framework web deve proporcionar:

> —Presentation. Presentation and the specification there of are mentioned by Helman and Fertalj [2003] and Copeland et al. [2000]. Presentation is described in Section 4.1.
> —Form Handling. Topics related to generating forms and dealing with input received from forms are mentioned in Helman and Fertalj [2003] and Copeland et al. [2000]. These are classified as “form handling” functionality, described in Section 4.2.
> —Validation. Validation-related concerns are mentioned by Rode et al. [2004] and Westk amper [2004]. These are incorporated in Section 4.3.
> —Navigation. Navigation-related concerns are mentioned by Helman and Fertalj ¨ [2003], Copeland et al. [2000], and Westk amper [2004]. In the list of requirements presented here, the closely related low-level concept colloquially termed “page flow” is included in Section 4.4.
> —Session Management. Session management is mentioned in Rode et al. [2004]. It is incorporated in the presented list under the heading of “session state” in Section 4.5.
> —Security. Security-related topics such as authentication and authorization are mentioned in Helman and Fertalj [2003], Rode et al. [2004], and Westk amper [2004]. These are incorporated in the list presented here under the heading of authentication in Section 4.6.
> —Back-End Integration. The issue of integrating with back-end systems is only mentioned by Copeland et al. [2000]. In this article, back-end integration is included and explained in Section 4.7. (Iwan Vosloo e Derrik G. Kourier. *Server-Centric Web Frameworks: An Overview, 2007, p.5)*

Analisaremos como o CakePHP implementa, e se implementa, cada um dos requisitos funcionais apresentados.

3.1 Presentation

Este recurso, refere-se a obrigação do framework web de fornecer uma forma usual de criação de interfaces com o usuário; possibilitando a renderização de páginas na linguagem HTML (ou similar) ou através da criação de interfaces gráficas utilizando diversos componentes de interface. O CakePHP trabalha este requisito com a utilização de templates para as views. Os templates são arquivos em simples na linguagem PHP contendo a interface com o usuário. Em geral, o conteúdo das views são códigos em linguagem HTML, mas também são suportadas XML, texto simples, imagens etc. O CakePHP permita ainda a utilização de *helpers* para a geração de views.

3.2 Form handling

O princípio *Form handling*, trata do envio de informações do usuário para o servidor web utilizando o protocolo HTTP. Em geral, o usuário preenche formulários com campos de texto, caixas de seleção, combinação etc e o envia, através de cliques em botões específicos, para uma URL no servidor.

O CakePHP possui um recurso chamado FormHelper. Todo trabalho relativo à criação de formulários pode ser feita por esta classe. A documentação oficial descreve a classe FormHelper

> O FormHelper foca na criação rápida de formulários, possibilitando valdação de dados, atribuição de valores de campos e layout. O FormHelper também é flexível - ele vai fazer quase todo o trabalho automagicamente, ou você pode usar os métodos específicos para ter apenas o que você precise. (Em <http://book.cakephp.org/1.3/pt/The-Manual/Core-Helpers/Form.html>. Acesso 07 de abril de 2015.)

Assim, a implementação desta classe por parte do CakePHP cumpre prontamente os requisitos funcionais propostos pelo item *Form handling.*

3.3 Validation

O *Validation* está diretamente ligado ao Form Handling pois trata da validação dos dados enviados pelo usuário. O framework web precisa fornecer um mecanismo prático para o programador especificar o escopo de valores válidos e como as mensagens de erros serão apresentadas para o usuário.

A classe FormHelper utilizada para a criação de interface de usuário para entrada de dados zela também por este princípio. A classe fornece entradas específicas para cada tipo de dado, permite a definição de valores padrões, a especificação de um limites para os valores informados e ainda permite o controle das mensagens de erro, permitindo não só controlar sua exibição mas até mesmo sobreescrever as mensagens de erro geradas pelos modelos.

Por exemplo, se desejarmos alterar a mensagem de erro padrão do modelo para o erro identificado pela *keyword* tooShort:

```
$this->Form->input('Model.field', array('error' => array('tooShort' => __("Valor inválido', true) )));
```

Para definirmos um valor padrão:

```
<?php
echo $form->input('ingrediente', array('default'=>'Açúcar'));
?>
```

Se desejarmos criar um grupo de opções, onde o usuário deverá selecionar um valor entre 1 e 5:

```
<?php echo $this->Form->input('field', array('options' => array(1,2,3,4,5))); ?>
```

Este último exemplo, iria produzir a seguinte saída em HTML:

```
<div class="input">
    <label for="UserField">Field</label>
    <select name="data[User][field]" id="UserField">
        <option value="0">1</option>
        <option value="1">2</option>
        <option value="2">3</option>
        <option value="3">4</option>
        <option value="4">5</option>
    </select>
</div>
```

Se sairmos um pouco da perspectiva da entrada de dados diretamente pelo usuário, o CakePHP possibilita que os dados sejam validados diretamente nos Modelos:

```
<?php
class User extends AppModel {
    var $name = 'User';
    var $validate = array(
        'login' => array(
            'alphanumeric' => array(
                'rule' => 'alphaNumeric',
                'required' => true,
                'message' => 'Letras e números somente'
                ),
            'between' => array(
                'rule' => array('between', 5, 15),
            'message' => 'Entre 5 e 15 caracteres'
            )
        ),
        'password' => array(
            'rule' => array('minLength', '8'),
            'message' => 'Mínimo de 8 caracteres'
        ),
        'email' => 'email',
        'born' => array(
            'rule' => 'date',
            'message' => 'Insira uma data válida',
            'allowEmpty' => true
        )
    );
}
?>
```

Neste exemplo, foram definidas para o login: ele deve conter apenas letras e números e o tamanho deve ter entre 5 e 15 caracteres. O campo *password* (senha) deve ter no mínimo 8 caracteres. O *email* deve ser um endereço de email válido, e o campo *born* (data de nascimento) deve ser uma data válida. Note também que você pode incluir mensagens personalizadas para que o CakePHP mostre quando as regras definidas falharem.

3.4 Navigation (Page Flow)

Navigation ou *Page Flow* é o termo utilizado para designar as relações entre as diferentes páginas de um web site. Navigation descreve os diferentes caminhos para um usuário atingir um determinado ponto (página) do sistema. Em uma análise mínima, este princípio fala das ferramentas disponibilizadas para ligar as diversas páginas que compõem o sistema baseando na URL (Uniform Resource Locator) das mesmas.

Quando o usuário faz uma requisição qualquer em um sistema desenvolvido com CakePHP, diversos objetos trabalham juntos para completar a requisição solicitada. A figura 2 descreve como este processo ocorre.

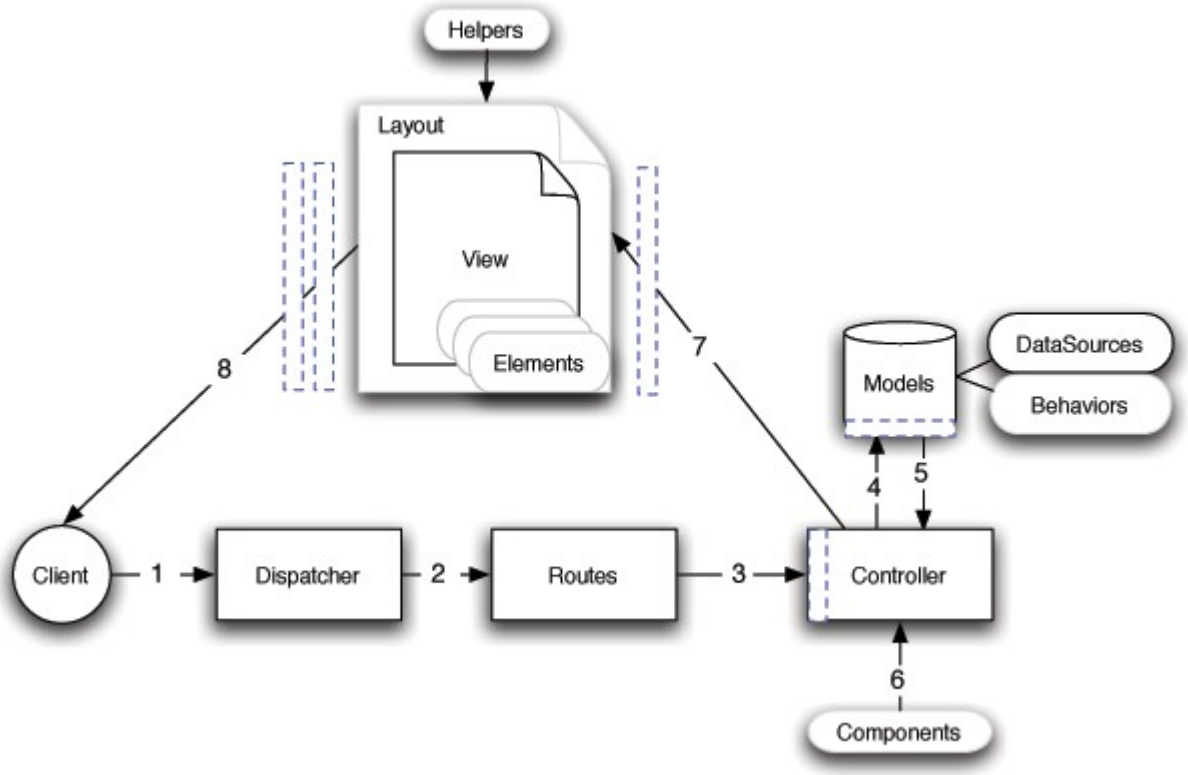

Figura 2: Page Flow

Fonte: Uma requisição típica do CakePHP (Em <http://book.cakephp.org/2.0/pt/getting-started/a-typical-cakephp-request.html>. Acesso em 08 de abril de 2015)

A documentação oficial explica o processo de roteamento e resposta para a requisição. Ao clicar em um link qualquer, o navegador do usuário envia requisição

para o servidor web. O roteador analisa a URL e extrai os parâmetros desta requisição: o controller, a ação, e qualquer outro argumento que afete a lógica do negócio durante esta requisição. Usando rotas, a URL da requisição é mapeada para uma ação de um controller (um método em uma classe controller específica). O controller pode usar modelos para obter acesso aos dados do aplicativo. Após o modelo buscar os dados, estes são retornados para o controller. Callbacks de um Model podem ser aplicados. O controller poderá utilizar componentes para refinar os dados ou executar outras operações (manipular sessão, autenticar ou enviar e-mails são exemplos). Uma vez que o controller usou os modelos e componentes para preparar os dados de forma suficiente, os dados são passados para a view usando o método set() do controller. Callbacks do controller podem ser chamados antes que os dados sejam passados. A view é executada, podendo incluir o uso de elementos e/ou helpers. Por padrão, a view é renderizada dentro de um layout. Adicionalmente, callbacks do controller podem ser aplicados.

3.5 Session State

O protocolo HTTP é um protocolo sem estado. As informações geradas em uma página web não estão disponíveis para outra página. Entretanto, as aplicações web precisam que esses dados estejam disponíveis entre as diversas páginas acessadas por um usuário em uma determinada conexão (sessão). As informações salvas neste meio de compartilhamento de dados entre URLs de uma mesma conexão são referidas por este requisito, o **Session State**.

A linguagem PHP oferece suporte as sessões através da variável $_SESSION. O CakePHP possui um componente *Session* que encapsula esta variável e fornece diversos métodos relacionados a sessões. Uma descrição completa deste componente e seus diversos métodos pode ser visto em http://book.cakephp.org/1.3/pt/The-Manual/Core-Components/Sessions.html.

3.6 Authentication

Este princípio está diretamente relacionado com o *session state* anterior e diz respeito a necessidade do framewok web de fornecer um mecanismo para controle de usuários, utilizando um nome e uma senha por exemplo. Depois de autenticado, o usuário poderá acessar as funções do sistema disponibilizadas especialmente para ele. Obviamente, este princípio necessidade de uma implementação segura por parte do framework.

O CakePHP possui vários sistemas de autenticação de usuários com diferentes opções. Em síntese, o componente de autenticação irá verificar se o usuário possui uma conta em um site, permitindo ou não o acesso a este. O componente AuthComponent pode ser usado para implementar este tipo de recurso rapidamente. A documentação oficial fornece um tutorial para a criação de um sistema simples de autenticação. O mesmo está disponível em http://book.cakephp.org/1.3/pt/The-Manual/Core-Components/Authentication.html

3.7 Back-end Integration

Normalmente, sistemas web são desenvolvido em várias camadas de arquitetura. Em alguns casos, o sistema web pode precisar acessar alguma operação do servidor, como um servidor de aplicativos, por exemplo. Neste caso, o usuário não deve ter a necessidade de autenticar-se em cada aplicativo que utiliza, uma vez autenticado no sistema web, essa credencial deve ser replicada para os outros aplicativos envolvidos. O framework web deve fornecer uma camada de abstração para que o programador web implemente esses tipos de recursos de baixo nível.

Este recurso ainda não é totalmente suportado pelo CakePHP. Embora algumas integrações de back-end sejam possíveis, como por exemplo, utilizar o Active Directory[3] para autenticar usuários, demais recursos necessários deverão ser

3 O **Active Directory** é uma implementação de serviço de diretório no protocolo LDAP que armazena informações sobre objetos em rede de computadores e disponibiliza essas informações a usuários e administradores desta rede.

implementados pelo programador web utilizando a linguagem PHP.

4. O padrão MVC

Espera-se que toda aplicação escrita em CakePHP siga o padrão de projeto MVC (Model-View-Controller ou Modelo-Visão-Controlador). Este padrão de projeto separa a aplicação em três camadas:

1. O modelo representa os dados;
2. A visão representa a visualização dos dados;
3. O controlador manipula e roteia as requisições dos usuários.

A figura 3, mostra o exemplo de uma requisição MVC em CakePHP

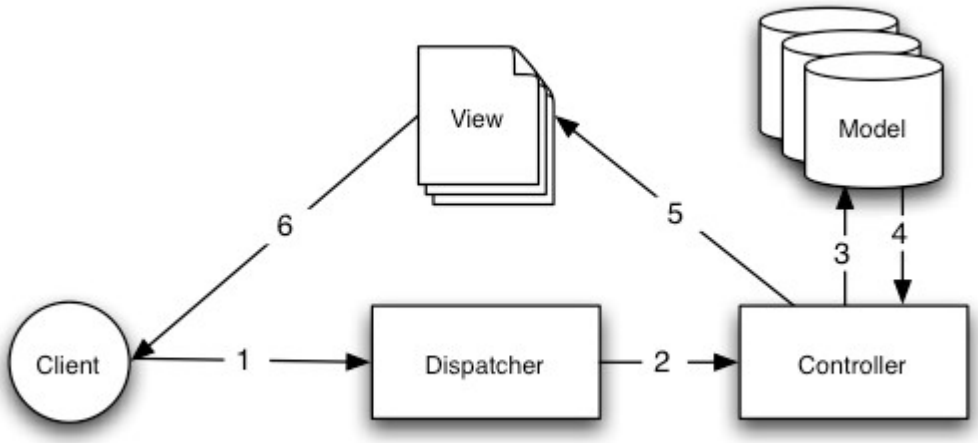

Figura 3: Requisição MVC

Fonte:Entendo o MVC. (Em <http://book.cakephp.org/1.3/pt/The-Manual/Beginning-With-CakePHP/Understanding-Model-View-Controller.html>. Acesso em 09 de abril de 2015.)

Os modelos representam os dados e são utilizados pelas aplicações CakePHP para acessar os mesmos. Um modelo, geralmente, representa uma tabela em um banco de dados, mas pode ser utilizado para acessar qualquer arquivo que armazene dados. O modelo implementado por uma aplicação CakePHP estende o application model, AppModel, que estende a classe interna do CakePHP Model. É esta classe Model de núcleo (core Model class) que agracia as funcionalidades para seu modelo.

As visões representam as saídas específicas para cada requisição. Em geral,

as saídas são escritas em HTML, mas podem ser utilizados XML ou JSON. Se necessário, a saída pode não ser renderizada diretamente para o browser do navegador, a camada de visão pode, por exemplo, gerar um arquivo PDF com o resultado da requisição.

Os arquivos de visão são escritos em PHP comum e tem a extensão padrão de .ctp (CakePHP Template). Esses arquivos contêm toda a lógica de apresentação necessária para renderizar a saída para o usuário.

Arquivos de visão são guardados em /app/views/, em um diretório com o nome do controlador que usa os arquivos, e nomeado de acordo com a visão a qual corresponde. Por exemplo, a action (ação) ver() do controlador Produtos será normalmente encontrada em /app/views/produtos/ver.ctp.

Os controllers de aplicação são classes que estendem a classe CakePHP AppController, a qual por sua vez estende a classe núcleo Controller. A classe AppController pode ser definida em app/app_controller.php e deve conter métodos que são compartilhados entre todos os seus controllers. Controllers podem incluir qualquer número de métodos que são geralmente referidos como *actions* (ações). Actions são métodos do controlador usados para mostrar views. Uma action é um único método de um controlador. O despachante do CakePHP chama actions quando uma requisição casa uma URL com uma action do controller.

5. Conclusão

O framework CakePHP implementa os principais recursos citados como requisitos funcionais no artigo *Server-Centric Web Frameworks: An Overview.* Este artigo, reune ampla documentação literária sobre frameworks e é adotado por centro universitários respeitáveis, como a PUC, como referência no estudo de frameworks WEB. Entre os sete requisitos apresentados, *a*penas um o item Back-end Integration ainda não pode ser considerado totalmente implementado. A arquitetura MVC é uma realidade esperada para toda aplicação escrita em CakePHP. Todas as necessidades deste arquitetura são contempladas pelo framework de forma simples e com abrangente documentação disponível. A comunidade CakePHP é amigável e prestativa no esclarecimentos de dúvidas nos diversos fóruns disponíveis. Somando-se todos estes aspectos, apresentados de forma sucinta neste artigo, podemos concluir que o CakePHP é uma excelente escolha para o desenvolvimento de aplicações web utilizando-se a linguagem PHP.

REFERÊNCIAS

Botwe, David A., and Joseph G. Davis. **A Comparative Study of Web Development Technologies Using Open Source and Proprietary Software**, 2015.

Cake Software Foundation, **CakePHP Overview, CakePHP Cookbook Documentation**, 2014

Chowdhary, Vijay. **CakePHP Access Control Lists**. 2012.

Entendendo o Model-View-Controller (MVC). Em <http://book.cakephp.org/1.3/pt/The-Manual/Beginning-With-CakePHP/Understanding-Model-View-Controller.html>. Acesso em 07 de abril de 2015.

Introduction to CakePHP. Em: <http://book.cakephp.org/1.1/en/introduction-to-cakephp.html>. Acesso em 01 de abril de 2015.

Iwan Vosloo and Derrick G. Kourie, ***Server-Centric Web Frameworks: An Overview***. ACM Computing Surveys, 2007.

PHP Frameworks. Em: <http://www.phpframeworks.com/>. Acesso em 05 de abril de 2015.

Watts, James, and Jorge González. ***CakePHP 2 Application Cookbook***. Packt Publishing Ltd, 2014.