

# **Desenvolva rapidamente utilizando o framework CakePHP**

**Elton Luís Minetto**

# Agenda

- **Ambiente Web**
- **PHP**
- **Problemas**
- **Frameworks**
- **CakePHP**
- **Demonstração**

# Ambiente Web

**É o ambiente formado por algumas tecnologias:**

- **Servidor Web**
- **Protocolo HTTP e HTTPS**
- **Navegadores**
- **HTML, PHP, JSP, CSS, AJAX**
- **Web sites e sistemas**



# PHP

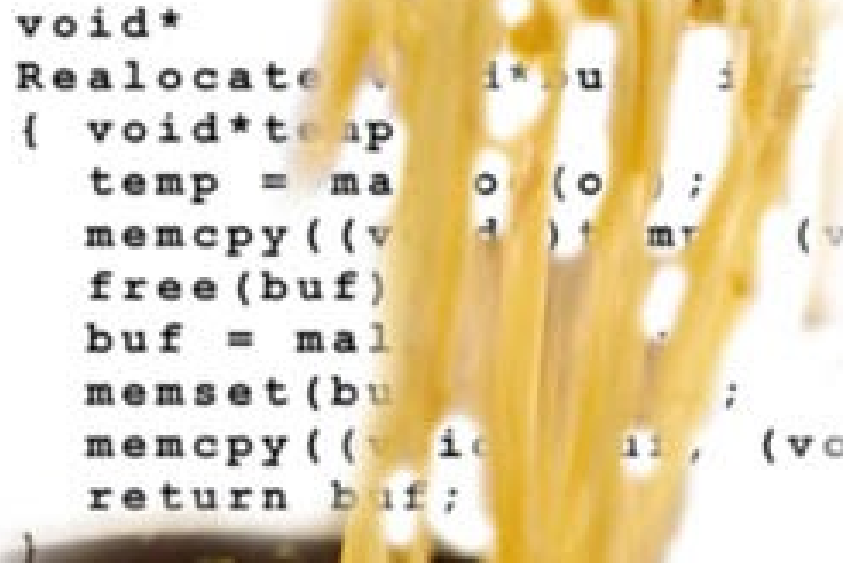
- **Livre**
- **Fácil de usar**
- **Grande documentação**
- **Grande biblioteca de funções**
- **Orientação a Objeto**

# Problemas Ambiente Web/PHP

- Repetição de tarefas
- Dificuldade de debug
- Várias formas de fazer a mesma coisa
- Existem poucos padrões e os que existem não são cumpridos (W3C x Microsoft)
- Os design patterns foram criados para aplicações cliente-servidor, somente agora estão sendo portados para web
- Clientes querem alterar tudo muito rápido
- Desenvolvimento lento
- Pouco reaproveitamento de código
- Não existem ferramentas RAD como Delphi
- Dificuldade de fazer relatórios
- Navegadores não renderizam corretamente algumas coisas
- Novas tecnologias surgindo a todo minuto
- Mistura de layout e lógica

# Problemas Ambiente Web/PHP

```
void*  
Reallocate(void* buf, int size)  
{ void* temp;  
  temp = malloc(size);  
  memcpy((void*)temp, (void*)buf, size);  
  free(buf);  
  buf = temp;  
  memset(buf, 0, size);  
  memcpy((void*)buf, (void*)temp, size);  
  return buf;  
}
```

A photograph of a bowl of spaghetti with a fork lifting a large portion, creating a tall, tangled stack of pasta. The image is overlaid with a semi-transparent box containing C code. The code is a function named 'Reallocate' that takes a void pointer 'buf' and an integer 'size' as arguments. It declares a 'temp' pointer, allocates memory for it using 'malloc', copies the contents of 'buf' to 'temp' using 'memcpy', frees 'buf', and then returns 'temp'. The code is written in a monospaced font and is partially obscured by the spaghetti.

# Frameworks

É uma “base” de onde pode-se desenvolver algo maior ou mais específico. É uma coleção de códigos, classes, funções, técnicas e metodologias que facilitam o desenvolvimento de novos softwares.

## Vantagens

- Todos programam parecido graças a padronizações. Fácil manutenção
- Automatiza tarefas repetitivas. DRY
- Separação de layout e lógica. *no more spaghetti!*
- *Reaproveitamento de códigos*

# Padrões

**Os frameworks são baseados em uma série de padrões de desenvolvimento já consolidadas:**

## **MVC**

**MVC é um acrônimo para Model, View, Controller (Modelo, Visão e Controlador). A idéia é separar todo o desenvolvimento de uma aplicação nestas três partes, ou camadas:**

- Model – gerencia o comportamento dos dados da aplicação.**
- View – gerencia a saída gráfica e textual da parte da aplicação visível ao usuário**
- Controller – interpreta as entradas de mouse e teclado do usuário, comandando a Visão e o Modelo para se alterarem de forma apropriada.**



# Padrões

## ActiveRecord

**Este padrão de design facilita a manipulação de dados contidos em uma base de dados por aplicações desenvolvidas com o paradigma orientados a objetos. Uma tabela de uma base de dados é vista na forma de uma classe, enquanto que cada linha da tabela é considerada um objeto desta classe. Quando um objeto é criado, alterado ou excluído esta ação é automaticamente refletida na base de dados. Desta forma não é necessário que o desenvolvedor conheça uma linguagem de manipulação de dados como SQL, além de manter toda a aplicação desenvolvida no paradigma orientado a objetos.**

# CakePHP

**Cake é um framework para PHP que usa padrões de desenvolvimento conhecidos como ActiveRecord e MVC. A idéia principal é ser um framework estruturado que permita a usuários PHP de todos os níveis desenvolver aplicações web robustas sem perda da flexibilidade.**



# CakePHP - Vantagens

- **Licença flexível (MIT)**
- **Compatibilidade com PHP4 e PHP5**
- **Geração de CRUD para interação com BD**
- **Arquitetura MVC**
- **Validações**
- **Templates**
- **Ferramentas que auxiliam gerar Javascript, AJAX, forms HTML, etc**
- **Comunidade ativa**
- **Funciona em qualquer subdiretório web, com pouca configuração do Apache.**

# Conceitos - Diretórios

/app - aplicação

  /config - arquivos de configuração, DB, etc

  /controllers - controladores da aplicação

  /index.php - página inicial da aplicação

  /models - modelos da aplicação

  /plugins

  /tmp - usado para cache e logs

  /vendors - bibliotecas de terceiros

  /views - visões da aplicação

    /elements - elementos, pedaços da visão

    /errors - páginas de erros customizadas

    /helpers - ajudantes para gerar código

    /layouts - arquivos de layout

    /pages - visões estáticas

/webroot - DocumentRoot para aplicação

  /css - arquivos css

  /files - arquivos comuns

  /img - imagens

  /js - Javascripts

/cake - código fonte do cake

# Requisitos

**Servidor web com as funcionalidades: sessions, mod\_rewrite (não obrigatório mas aconselhável)**

**PHP 4.3.2 ou superior**

**Uma base de dados. Atualmente é suportado MySQL, PostgreSQL, SQLite, ODBC e AdoDB. Futuramente Oracle (CakePHP 1.2)**

# Instalação

**Download do arquivo em  
<http://cakephp.org/downloads>**

**Descompactar e copiar para o diretório root  
do servidor Web. No Ubuntu:**

```
tar xfvj cake_1.1.15.5144.tar.bz2  
mv cake_1.1.15.5144 /var/www/cake
```

**Testar no navegador no endereço:  
<http://localhost/cake>**

# Configuração Apache

**É necessário duas configurações: o mod\_rewrite e o AllowOverride precisam estar ativos.**

**Alterar o httpd.conf (apache2.conf no Ubuntu) e descomentar ou adicionar as linhas :**

```
LoadModule rewrite_module modules/mod_rewrite.so  
AddModule mod_rewrite.c
```

**Adicionar a linha abaixo no DocumentRoot e reiniciar o Apache:**

```
AllowOverride all
```

# talk is cheap...

**...show me the code!**



# Desenvolvendo

## A aplicação

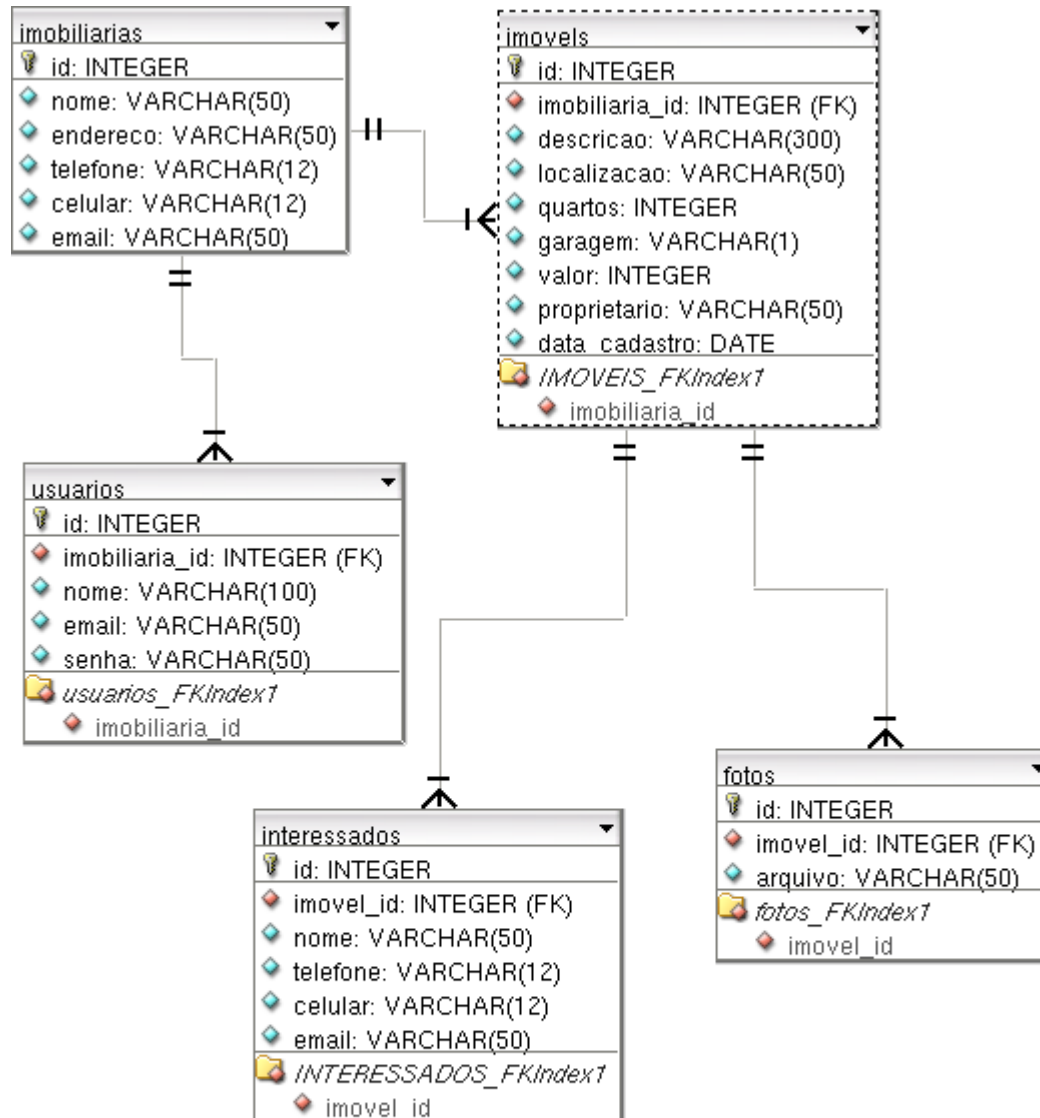
**Foi solicitado a criação de um site para agregar informações de diversas imobiliárias. Os clientes podem acessar e visualizar imóveis a venda e se cadastrar como interessados. Deve haver uma interface de administração para controle das informações cadastradas**

# Desenvolvendo

## **O Cake segue algumas regras de padronização para a base de dados:**

- **Todas as tabelas devem estar no plural.**
- **A chave primária para todas as tabelas deve ser o campo "id".**
- **Para utilizar chave estrangeira, deve possuir nome da tabela no singular acrescido do campo id.**  
**Ex: imovel\_id.**
- **O relacionamento "n-n", deve ser organizado em ordem alfabética e os nomes devem estar no plural. Exemplo: livros\_usuarios**
- **Modelos são no singular e controladores no plural**

# Modelagem



# Usando o CakePHP

## *Iniciando a aplicação*

*O cake possui uma excelente ferramenta para geração da aplicação e de várias partes de código. A ferramenta chama-se bake.*

# Usando o CakePHP

## ***Criando um projeto:***

```
cd /var/www/cake/
```

```
php cake/scripts/bake.php -project webimoveis
```

***Após responder y para as perguntas, toda a estrutura de diretórios é criada.***

***Depois de criado o projeto é necessário criar a aplicação.***

# Usando o CakePHP

## ***Criando uma aplicação:***

*php cake/scripts/bake.php -app webimoveis*

*What database driver would you like to use?  
(mysql/mysqli/mssql/sqlite/postgres/odbc) - qual  
database. Neste caso mysql*

*What is the hostname for the database server? -  
localhost*

*What is the database username? - imobiliarias*

*What is the database password? - imobiliarias*

*What is the name of the database you will be  
using? - qual é a base de dado usada. Neste  
exemplo imobiliarias. As tabelas devem ser  
previamente criadas.*

# Usando o CakePHP

*Enter a table prefix? - caso as tabelas tenham algum prefixo no seu nome. Neste caso - n*

*Look okay? (y/n) - confirmar com y caso tudo esteja correto*

**Neste momento foram criadas as configurações de conexão com a base de dados da aplicação. Estas configurações foram salvas em**

*/var/www/cake/webimoveis/config/database.php*

**e podem ser alteradas posteriormente.**

# DICA

***Para gerar os modelos o bake irá trabalhar com os nomes das tabelas no singular. Como o CakePHP foi desenvolvido em inglês existem alguns problemas que ocorrem nessa transformação. Por exemplo, imobiliarias torna-se immobiliareum. Para resolver este problema é necessário alterar o arquivo***

*webmoveis/config/inflections.php*

***e alterar a seguinte linha:***

```
$singularRules = array('/([ti])a$/i' => '\1a');
```



# Usando o CakePHP

***Executando novamente o bake é solicitado a opção de gerar o Modelo, a Visão ou o Controle da aplicação.***

***Gerando o Modelo: O bake conecta na base de dados e encontra as tabelas existentes. Você pode escolher para qual das tabelas irá criar o modelo. Inicialmente vamos escolher a tabela Imobiliarias digitando seu número da lista mostrada.***

***Would you like to supply validation criteria for the fields in your model? (y/n)- se deseja fornecer critérios de validação para os campos da tabela. Neste caso y. É apresentado cada um dos campos e, de acordo com seu tipo na tabela é apresentado algumas validações. Por exemplo o campo id pode ser validado para VALID\_NOT\_EMPTY ou VALID\_NUMBER.***

# Usando o CakePHP

*Would you like to define model associations (hasMany,hasOne, belongsTo, etc.)? (y/n) – se é necessário definir associações entre os modelos. Existem quatro associações:*

***hasOne: possui um***

***hasMany: possui muitos***

***belongsTo: pertence a***

***hasAndBelongsToMany: tem e pertence a muitos***

***Escolhendo y o bake vai identificar as associações e solicitar confirmação para cada uma delas:***

# Usando o CakePHP

*Imobiliaria hasOne Imovel? (y/n)*

*[y] > n*

*Imobiliaria hasOne Usuario? (y/n)*

*[y] > n*

*Imobiliaria hasMany Imovel? (y/n)*

*[y] > y*

*Imobiliaria hasMany Usuario? (y/n)*

*[y] > y*

*Cake test suite not installed. Do you want to  
bake unit test files anyway? (y/n)*

*[y] > n – gerar as classes de teste. Neste caso N*

*0 arquivo models/imobiliaria.php é gerado*

# Usando o CakePHP

*Após criar todos os modelos é preciso criar os Controle. Executando novamente o comando:*

*php cake/scripts/bake.php -app webimoveis*

*Possible Controllers based on your current database – escolher o Imobiliarias*

*Would you like bake to build your controller interactively? - y*

*Would you like to use scaffolding? (y/n) - se queremos criar as tela virtuais de CRUD. N*

*Would you like to include some basic class methods (index(), add(), view(), edit())? (y/n) - se queremos criar as telas de CRUD explicitamente. Neste caso Y*

# Usando o CakePHP

*Would you like to create the methods for admin routing? (y/n) – se queremos criar uma interface diferente para a administração. Y*

*Would you like this controller to use other models besides 'Imobiliaria'? (y/n) – se é necessário usar mais de um modelo. Neste caso n*

*Would you like this controller to use other helpers besides HtmlHelper and FormHelper? (y/n) – se são necessários outros ajudantes além do gerador de Html e Forms. Vamos utilizar o helper de Javascript, então digitamos y e depois Javascript*

# Usando o CakePHP

*Would you like this controller to use any components? (y/n) - se queremos utilizar componentes adicionais. Exemplo: gerador de PDF, Paginação, etc*

*Would you like to use Sessions? (y/n) - se queremos usar Sessões. Sempre é importante para guardarmos informações, principalmente de validação de usuários.*

*Cake test suite not installed. Do you want to bake unit test files anyway? (y/n) - se queremos criar as classes de teste. N*

*0 arquivo controllers/imobiliarias\_controller.php é gerado*

# Usando o CakePHP

*E, finalmente, para gerar a Visão, executar novamente o bake, escolher a opção V e o número correspondente a Imobiliarias.*

*Would you like to create some scaffolded views (index, add, view, edit) for this controller? - ele vai criar a visão para os métodos básicos de inserção, exclusão, etc. Responder y*

*Would you like to create the views for admin routing? (y/n) - como escolhemos no controlador que queremos interface de administração devemos escolher Y nesta opção*

*Um diretório é criado em views/imobiliarias/ com diversos arquivos HTML, um para cada ação do controlador. Exemplo: index.shtml, admin\_edit.shtml, etc.*

# Customizando

*Os passos anteriores devem ser repetidos para todas as tabelas do banco de dados.*

*Após isto, está criado um "esqueleto" da aplicação.*

*Agora podemos customizar os códigos gerados para criarmos a aplicação final.*



# Layout

***Nosso designer nos mandou o layout do site. Agora precisamos colocar este visual nas aplicações geradas pelo CakePHP.***



# Layout

*O layout principal das aplicações encontra-se no arquivo `views/layouts/default.thtml`. Mudando este arquivo podemos transformar automaticamente todas as aplicações geradas.*

*Os arquivos `.css` devem ser salvos no diretório `webroot/css/`*

*Os arquivos `.js` (Javascript) devem ser salvos no diretório `webroot/js/`*

# Página inicial

*Quando acessamos o link*

*<http://localhost/cake/webimoveis/>*

*iremos redirecionar para a página que lista as imobiliárias. Para isso deve-se alterar o arquivo*

*[views/pages/home.thtml](#)*

*Seu novo conteúdo é:*

```
<?php
```

```
header('location:imobiliarias');
```

```
?>
```

# Imobiliárias

- O controlador `imobiliarias_controller.php` gerado pelo `bake` possui diversos métodos. Vamos alterá-lo e deixar somente o método `index()` e os métodos `admin`.*
- O método `index()` é o responsável por mostrar todas as imobiliárias cadastradas e seus imóveis. O código deste método é mostrado no próximo slide*

# Imobiliárias

```
function index($id=null) {  
    //busca as imobiliarias  
    $this->Imobiliaria->recursive = 0; //não recursivo. busca  
        só a imobiliária  
    $this->set('imobiliarias',$this->Imobiliaria->findAll());  
    //busca a imobiliaria atual  
    if($id) { //se já foi selecionado uma, deve buscar todos  
        os dados  
        $this->Imobiliaria->recursive = 2;  
        $this->set('atual', $this->Imobiliaria->  
            >findAllById($id));  
    }  
}
```

# Imobiliárias

***Além disso é preciso alterar o arquivo  
views/imobiliarias/index.shtml***

***para mostrar os dados vindos da tabela. Não é  
possível mostrar todo o arquivo no slide mas está  
disponível junto com esta apresentação.***

***Exemplo de trecho:***

```
<h3><?php echo $atual['0']['Imobiliaria']['nome']?></h3>  
<?php echo $atual['0']['Imobiliaria']['endereco']?><br>  
<b>Telefone: <?php echo  
    $atual['0']['Imobiliaria']['telefone']?></b><br>
```

# Imóveis

***Processo semelhante vamos realizar com o cadastro de imóveis. Primeiro vamos excluir os métodos não necessários do controlador (add, edit, delete), deixando somente o index() e os métodos de administração. A única alteração necessária no método index() é:***

***\$this->Imovel->recursive = 1;***

***para buscar as fotos do imóvel.***

***No arquivo de visão do método index (index.thtml) devemos alterar para mostrar os dados do imóvel e o formulário de cadastro de interessados, conforme mostra a imagem***

# Imóveis

Portal Imobiliário - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost/xxe/webimoveis/Imoveis/Index/6

Gmail SAA netvibes post to del.icio.us my del.icio.us

Gmail - Inbox (5) Portal Imobiliário Nostra Casa

## Portal Imobiliário

Seu portal está aqui (Admin)

### CASA no bairro EFAPI-ELIAS GALLON

Localização: Efapi  
Quartos:3  
Garagem:1

R\$ **68000**



#### Estou interessado

Nome:

Telefone:

Celular:

Email:

[Voltar](#)

RSS 2.0 W3C CSS W3C XHTML 1.1



# Interessados

*Quando o visitante preencher o formulário de interessados esta informação será salva na tabela. Para isso iremos utilizar os códigos gerados pelo bake. Quem fará o salvamento na tabela é o interessados\_controller.php, no método add().*

*A única alteração necessária é substituir no método add() a linha*

*`$this->redirect('/interessados/index');`*

*por*

*`$this->redirect('/imobiliarias/index');`*

*Assim, quando salvar os dados o usuário será redirecionado para a página inicial da aplicação.*

# Controle de acesso

***Quando o usuário clicar na opção Admin na página inicial será direcionado a um formulário para fazer o login na aplicação.***

***Para realizar esta tarefa é preciso os seguintes passos:***

- 1) Criar o controlador em controllers/login\_controller.php. Neste arquivo deve-se criar os métodos index(para mostrar o formulário), valida (para fazer a validação), logout(sair do sistema) e menu (para mostrar o menu)***
- 2) Criar o diretório views/login/ com os arquivos index.shtml e menu.shtml***

# Controle de acesso

**3) Verificar se foi realizado o login no momento do acesso da interface de administração. Alterar o arquivo `app_controller.php`. Adicionar o código abaixo:**

```
//funcao que é executada antes de qualquer método dos controladores

function beforeFilter() {

    if(isset($this->params['admin'])) { //se foi acessado a interface de administracao

        if(!$this->Session->check('logado')) {

            $this->Session->setFlash('Acesso negado');
            $this->redirect('/login');

        }

    }

}
```

**Estes arquivos alterados estão em anexo a apresentação**

# Upload das Fotos

*Para realizar o upload das fotos é preciso alterar o controlador fotos\_controller.php e a visão add.thtml. No fotos\_controller.php é preciso adicionar o código abaixo no método admin\_add():*

```
if(isset($this->data['Foto']['arquivo']['tmp_name'])) {  
    //copiando o arquivo para o diretorio final  
    copy($this->  
        data['Foto']['arquivo']['tmp_name'], "/var/www/xxe/web  
        imoveis/webroot/img/".$this->  
        data['Foto']['arquivo']['name']);  
    //substitui o campo arquivo por somente o nome do arquivo  
    $this->data['Foto']['arquivo'] = $this->  
        data['Foto']['arquivo']['name'];  
}  
  
if($this->Foto->save($this->data)) { ....
```

# Upload das Fotos

***E no arquivo views/fotos/admin\_add.thtml é necessário alterar o objeto do formulário para um tipo "file" e modificar a tag <form> :***

...

```
<form action="<?php echo $html->url('/admin/fotos/add'); ?  
    >" method="post" enctype="multipart/form-data">
```

...

```
<?php echo $html->file('Foto/arquivo');?>
```

# Ajax

***O Cake possui uma série de Helpers para auxiliar no desenvolvimento. Um destes Helpers é o de Ajax. Ele usa as bibliotecas JavaScript Prototype e script.aculo.us que devem ser copiadas para o diretório webroot/js/ da aplicação.***

***Vamos fazer um pequeno exemplo. Uma pequena busca de imóveis usando Ajax.***

***Primeiro é preciso criar um novo controlador para a aplicação. No diretório controllers foi criado um novo arquivo, chamado buscas\_controller.php:***

# buscas\_controller.php

```
<?php class BuscasController extends ApplicationController {  
    var $name = 'Buscas'; //nome do controlador  
    var $helpers = array('Html', 'Ajax' ); //uso dos helpers  
    var $uses = array("Imovel"); //indica qual modelo usar  
    function index() { //funcao da pagina inicial  
    }  
    function update () {  
        $consulta = $this->params["form"]["localizacao"]."%";  
        $this->set('imovels',$this->Imovel->findAll("Imovel.localizacao LIKE '$consulta'"));  
        $this->layout = "ajax"; //usar o layout ajax e não html  
    }  
}  
?>
```

# Ajax

***O próximo passo é criar as visões. No diretório views deve-se criar um diretório para as views da aplicação. Neste caso chamamos de buscas. Os arquivos correspondentes aos métodos do controlador foram criados:***

***index.thtml***

***update.thtml***



# index.thtml

```
<?php echo $html->charsetTag('UTF-8')?>
<?php echo $javascript->link('prototype')?>
<?php echo $javascript-
    >link('scriptaculous.js?load=effects')?>
<?php echo $ajax->form(array('action' => '/buscas/update'),
    'imovel', array('update' => 'resultados')); ?>
<input type="text" name="localizacao">
<input type="submit" value="Pesquisar">
<div id="resultados"> </div>
```

**As primeiras linhas carregam os arquivos .js necessários. Com o \$ajax->form é criado um formulário cujo action é a função update. Os outros parâmetros são o modelo (imovel) e as opções (neste caso está definindo que a função update vai alterar a div resultados).**

# update.thtml

```
<?php echo $ajax->div('resultados'); ?>
<?php foreach($imovels as $imovel): ?>
<h2><?php echo $imovel['Imovel']['descricao']?> na
    imobiliaria <?php echo
    $imovel['Imobiliaria']['nome']?></h2>
<h3>Detalhes</h3>
Localização:<?php echo $imovel['Imovel']['localizacao']; ?
    ><br>
Quartos:<?php echo $imovel['Imovel']['quartos']; ?><br>
Garagem:<?php echo $imovel['Imovel']['garagem']; ?><br>
Valor:<?php echo $imovel['Imovel']['valor']; ?><br>
<?php endforeach; ?>
<?php echo $ajax->divEnd('resultados'); ?>
```

**Este é o código que irá mostrar os resultados.**

# Conclusões

***Desenvolver usando frameworks facilita a padronização e trabalho em equipe.***

***Acelera o ciclo de desenvolvimento.***

***Cake é fácil de aprender.***

***Cake usa uma estrutura de arquivos prática e simples***

***Tem um nome legal :-)***

# Referências

***<http://www.cakephp.org>***

***<http://www.cakephp.com.br>***

***<http://bakery.cakephp.org/>***

***<http://cakeforge.org/>***

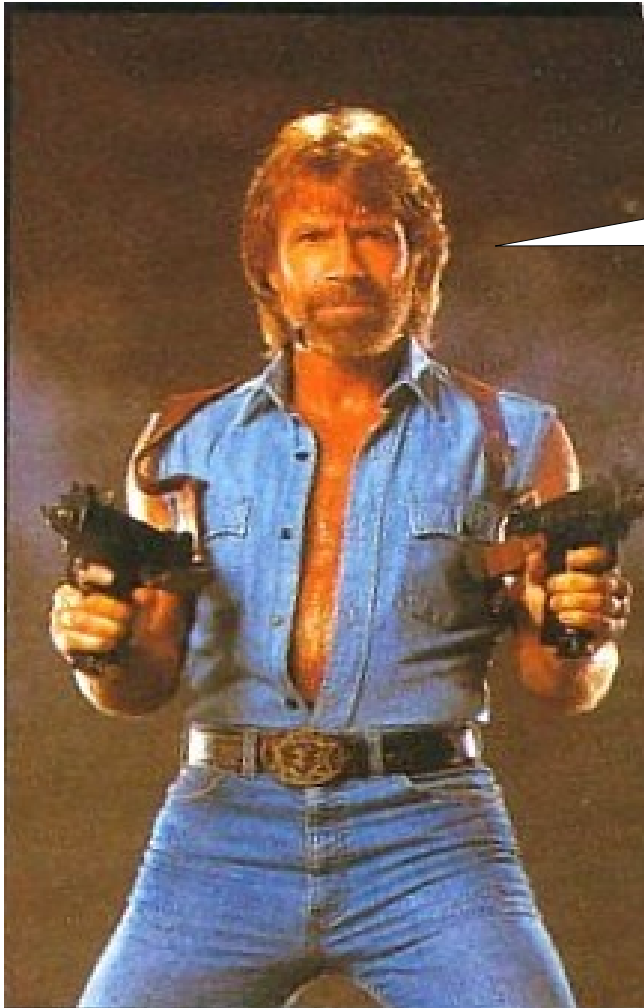
**Atenção!**

***O próximo slide é  
uma prova da  
cara-de-pau do  
palestrante!***

# Propaganda



# Depoimentos



Melhor que um  
roundhouse kick!

# Depoimentos



A força sinto nesse livro!



# Contato

**Elton Luís Minetto**

**eminetto@gmail.com**

**<http://www.eltonminetto.net>**