CakePHP3 Treinamento

Introdução

Nosso principal objetivo aqui é o de criar aplicativos de forma simples e prática e também que eles tenham um bom controle de acesso, permitindo apenas aos usuários determinados acessar determinadas partes do aplicativo.

Iremos fazer isso por etapas, de forma organizada.

Usaremos principalmente o Framework CakePHP em sua versão 3. O CakePHP é um framework bem popular entre os demais existentes e ajuda muito na criação de aplicativos. Ele conta com muitos e bons recursos e usa a linguagem PHP, um servidor Web como o Apache ou outros e um SGBD como o MySQL ou outros. Também usa orientação a objetos e alguns padrões de projeto especialmente o MVC e o ORM. O foco do CakePHP é nas convenções. Você segue as convenções que ele determina e ele facilita sua vida.

Pré-requisitos

De conhecimentos
Desejáveis – conhecimento da linguagem PHP, de orientação a objetos e de padrões de projeto.

Obrigatório – boa motivação e vontade de aprender.

De Hardware
Praticamente qualquer computador que esteja funcionando pode ser usado para nosso treinamento.

De Software
Praticamente qualquer sistema operacional pode ser utilizado.
Precisamos basicamente do Apache, PHP e MySQL mais alguns módulos/extensões.

Servidor

O servidor precisa ter Apache, PHP e MySQL no mínimo. Eu prefiro criar este servidor em meu computador desktop, no Linux (preferido) ou no Windows, mas podem ser usados outros SOs.

Podemos usar um servidor online daqueles que já vem com editor e todo o ambiente para programação, como é o caso do Cloud9 (http://c9.io). Este exige conhecimento sobre a linha de comando do Linux, mas mostrarei um guia em seguida. Caso queira experimentar, ele oferece um plano free:
https://c9.io/signup

Podemos usar o Xampp (http://xampp.sf.net), o Wampp, o EasyPHP ou outro no Windows.
No Linux prefiro usar os pacotes da distribuição.

Para começar você precisa já ter um ambiente pronto e configurado com Apache, PHP e MySQL e extensões. Caso não tenha e não saiba como fazer provavelmente este treinamento não é para você Ainda, a não ser que seja daqueles corajosos que tenham uma grande motivação para aprender e queira experimentar. Se for seu caso, procure um tutorial de como instalar o Xampp no Windows ou outro.

Pré-requisitos para Instalar o CakePHP 3.2.9

- Apache2 ou outro servidor web
- PHP 5.5.9 ou superior
- Módulos do Apache: mod_rewrite
- Extensões: mcrypt, php5-mcrypt, intl, mbstring

Instalação do Ambiente

Use o Xampp no Windows e os pacotes no Linux.

Instalação do CakePHP pelo Composer

No Linux
curl -sS https://getcomposer.org/installer | php

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

sudo nano /usr/local/bin/comp

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

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

Criando o aplicativo:
cd /var/www/html/treinamento
comp app1

Acessando
http://localhost/treinamento/app1

No Windows
Download - https://getcomposer.org/Composer-Setup.exe
Instale

Veja detalhes sobre como adicionar o php ao path.

Criar o arquivo bat comp.bat na pasta c:\xampp\htdocs, contendo a linha:

composer create-project --prefer-dist cakephp/app %1
Criando o Aplicativo
Abrir o prompt
cd c:\xampp\htdocs\treinamento
comp app1

Acessando
http://localhost/treinamento/app1

Primeiro Aplicativo

Este primeiro aplicativo apenas instala o CakePHP 3.2.9 através do composer e mostra o aplicativo padrão que já vem com com o Cake. Vem um controller, um action e um template.

Customizando o Terminal/Prompt

Eu gosto de configurar para que ele já abra no diretório onde trabalho.

No Windows
Clique com o botão direito – propriedades – Iniciar em c:\xampp\htdocs\treinamento
Aproveita e entra com uma combinação de teclas tipo Ctrl+Alt+C

No Linux
Crie um atalho para o terminal assim:
Configurações do sistema – Teclado – Atalhos de teclado – Atalhos personalizados
Clicar no sinal de +
Nome – Terminal
Comando - gnome-terminal –working-directory=/var/www/html/treinamento
Teclar Enter
Adicionar o atalho – Ctrl+Alt+C

Agora sempre que teclar Ctrl+Alt+C ele abre nosso ambiente já no respectivo diretório.

Alterando o document root do Apache no Linux

Em meu computador desktop no Linux eu gosto de mudar do /var/www/html para /backup/www, backup é minha partiçãod e backup e como ribafs sou o dono dela e assim fica mais confortável para trabalhar.

Criando o aplicativo app1

Ctrl+Alt+C
comp app1

Com este comando, aguardamos um pouco e ele traz o CakePHP atualizado para sua última versão estável juntamente com todas as dependências.

Vejamos. Execute o comando:
cd app1
bin/cake
Mude a barra para \ caso esteja usando o Windows.
Ele mostra algumas informações importantes, inclusive a versão do cake, a do php e alguns diretórios:

```
Welcome to CakePHP v3.2.9 Console
---------------------------------------------------------------
Path: /backup/www/treinamento/app1/src/
PHP : 5.6.11-1ubuntu3.4
```

---
* app: src – neste subdiretório fica todo o código do aplicativo
Abaixo fica todo o código do framework
* core: /backup/www/treinamento/app1/vendor/cakephp/cakephp

Acesse o aplicativo padrão embutido no Cake pelo navegador:

http://localhost/treinamento/app1

Já veremos uma página padrão criada pelo controller padrão e seu template.
Como podemos ver que arquivos são os responsáveis por isso? São muitos, mas os principais são
src/Controller/PagesController.php
src/Template/home.ctp

Este aplicativo não tem um banco de dados, usa apenas página estática.

Convenções do CakePHP 3

Sempre que for encontrando oportunidade irei abordando as convenções necessárias e importantes, mas caso queira se antecipar veja a respectiva seção ou o site oficial:
http://book.cakephp.org/3.0/en/intro/conventions.html

Segundo Aplicativo

Nome de banco de dados – tuto em minúsculas e no singular. Palavras compostas separadas por sublinhado.
Nome de tabela – tudo em minúsculas e no plural.

Para este primeiro aplicativo útil criaremos um banco chamado cadastro contendo apenas uma tabela chamada clientes. Depois de configurar o banco no aplicativo geraremos o código do aplicativo usando o bake, que gera um CRUD completo e com bons recursos.

Banco de Dados

Banco – cadastro
Tabela – clientes

Criar o banco no phpmyadmin e colar 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@justoeuarcu.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');

ALERTA – tome cuidado ao copiar e colar de processadores de texto como o Word ou Writer, pois algumas vezes eles alteram o texto. Um exemplo é este: --, que o writer gosta de mudar para –.

Criando o aplicativo

Ctrl+Alt+C
comp app2

Configurações

Configurar o banco de dados. Editar o arquivo app2/config/app.php

Alterar apenas 3 linhas:

Em
```
    'Datasources' => [
        'default' => [
```

Altere as linhas:

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

Salve e feche

Configure as rotas. Editar o arquivo app2/config/routes.php

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

Colar logo acima e alterar assim:
```
    $routes->connect('/', ['controller' => 'Clientes', 'action' => 'index']);
```

Com isso estamos dizendo que o raiz do nosso aplicativo será respondido pelo action index() do controller Clientes, assim, ao abrir http://localhost/treinamento/app2 ele mostrará a view index.ctp do template clientes, chamada pelo action index() do controller Clientes.

Agora, chamando pelo navegador:
http://localhost/treinamento/app2

Como ele procura o controller Clientes e não encontra, então mostra a mensagem de erro e já ajuda com a mesma. Então vamos criar o código do nosso aplicativo: controllers, models e views.

Gerando o Código do Aplicativo

Abra o terminal, acesse o diretório app2 e execute o comando abaixo:

bin/cake bake all clientes

Observe as mensagens e veja que com isso o bake gerou o código básico de um CRUD, com o controller, o model e a view, criando assim um aplicativo simples mas funcional.

Acesse novamente e veja como ficou. Navegue pelas seções do site.

Veja que já aplicou paginação, CSS e todo um layout adequado para a gente trabalhar.

Experimente adicionar, editar, remover e visualizar registros. Como também mudar para a segunda página e reordenar as colunas de campos clicando no título e bem mais. E mais ainda existe a nossa disposição no framework.

Mais detalhes sobre a geração de código com Bake, veja a seção respectiva ou o site oficial:
http://book.cakephp.org/3.0/en/bake/usage.html

Terceiro Aplicativo

Agora vamos adicionar 3 tabelas: groups, users e privileges. A grande intenção destas tabelas é criar uma área restrita para usuários, ou melhor, criar um controle de acesso, liberando apenas certos usuários para certas seções do aplicativo.

Criar o aplicativo

Ctrl+Alt+C
comp app4

Banco de Dados

Banco – app3
Tabelas: clientes, groups e users.
Crie o banco e importe o script app3.sql

Configurações

app3/config/app.php

Altere:

```
        'username' => 'root',
        'password' => '',
        'database' => 'app3',
```

app3/config/routes.php

Adicione a linha, acima da existente:

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

Gerando o código
cd /backup/www/treinamento

bin/cake bake all clientes
bin/cake bake all groups
bin/cake bake all users

Acesse pelo navegador

http://localhost/treinamento/app3

Experimente acessar
http://localhost/treinamento/app3/users/login
Já estão lá, os actions login() e logout() e a view login.ctp.

Navegue entre as tabelas/controllers assim:
http://localhost/treinamento/app3 – clientes
http://localhost/treinamento/app3/users
http://localhost/treinamento/app3/groups

Mais a frente adicionaremos um menu para tornar isso mais prático. O Cake tem um recurso chamado Element, que pode ser usado para adicionar um menu ao aplicativo, bastando apenas que adicionemos o arquivo e um link ao layout.

Quarto Aplicativo

Agora é a hora de conhecer melhor o CakePHP, conhecer seu código. Vamos começar a criação de um código para controle de acesso. Para isso vamos criar uma função no controller AppController. Este controller fica em:
src/Controller/AppController.php

Depois de pronto o código que iremos implementar, moveremos para um componente. Mais a frente criaremos um plugin contendo o código das tabelas groups, users e privileges e também o componente e algo mais. Este plugin controlará o acesso ao nosso aplicativo de forma bem flexível. Poderemos definir que ations cada usuários poderá acessar, alterar, criar e excluir.

Criar Aplicativo

Acessar o terminal e criar o aplicativo app4:

Ctrl+Alt+C
comp app4

Customizar o bake para facilitar a geração de código do controller Users.

Por padrão o bake gera somente os actions e views default, que são: index, edit, add, delete e view. Mas quando for o caso do controller Users precisamos de mais dois actions: login e logout e de uma view login.

Altere o arquivo app3/config/bootstrap_cli.php para que fique assim:

```
<?php
use Cake\Core\Configure;
use Cake\Core\Exception\MissingPluginException;
use Cake\Core\Plugin;
// Adicionar os 3 abaixo
use Cake\Event\Event;
use Cake\Event\EventManager;
use Cake\Utility\Hash;

// Set logs to different files so they don't have permission conflicts.
Configure::write('Log.debug.file', 'cli-debug');
Configure::write('Log.error.file', 'cli-error');

try {
    Plugin::load('Bake');
} catch (MissingPluginException $e) {
    // Do not halt if the plugin is missing
}
// Adicionar daqui pra frente
```

```
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'
            ];
        }
    }
);
```

Criando modelo de aplicativo

Também podemos criar um aplicativo com o composer e logo que tenha sido criado copiar seu diretório. Exemplo: criar o **app4** e copiar para **modelo**. A partir de agora iremos copiar o aplicativo **modelo** para gerar o **app5** e seguintes.

Com algum tempo nosso aplicativo pode ficar com o CakePHP desatualizado, então executaremos:

```
Ctrl+Alt+C
cd app4
composer update
```

Assim o composer atualiza nosso framework para a última versão estável.

Banco de Dados

Criar o banco app4. Importe o script app4.sql.

Usaremos o mesmo banco do app3, com mais a tabela privileges já com alguns registros cadastrados.

Agora precisamos que cada tabela adicionada, como clientes, funcionarios, produtos ou qualquer outra, contenha o campo user_id por conta do uso do componente Auth, que precisa relacionar as tabelas com users.

Configuração

Configure o app.php para o banco app4 e o routes.php da mesma forma que o app3, sendo Clientes nosso raiz.

Estrutura de diretórios

Para conhecer a estrutura de arquivos e diretórios do CakePHP 3 percorra seu app4.
Para saber detalhes sobre a mesma veja:
http://book.cakephp.org/3.0/en/installation.html

Implementando a criptografia bcrypt para as senhas dos usuários

Adicionando hash bCrypt para a Senha

Isso é feito no Model, especificamente na classe Entity User.php:

Adicione:

```
use Cake\Auth\DefaultPasswordHasher;
```

Ao final da classe adicione a função abaixo:

```
    protected function _setPassword($password)
    {
        return (new DefaultPasswordHasher)->hash($password);
    }
```

Criar Grupos e Usuários

Vamos criar 3 usuários, um para cada grupo:

Groups	**Users**	**Password**
Admins	admin	admin
Managers	manager	manager
Users	user	user

Acesse
http://localhost/treinamento/app4/groups/add

E cadastro os 3 grupos acima.

Acesse
http://localhost/treinamento/app4/users/add

E cadastre os 3 usuários acima com as respectivas senhas.
Após cadastrar os usuários vemos um hash bem grande criado pelo algoritmo do bcrypt.
É interessante editar as views src/Template/Users/index.ctp, view.ctp e edit.ctp removendo o campo senha do form.

Criação de Funções para o Controle de Acesso

Por enquanto, para fazer as coisas mais simples, iremos criar as funções uma de cada vez, para ir testando e depois juntaremos todas em um componente. Aliás, aqui eu já mostro o código todo debugado, com os erros que encontrei corrigidos. A intenção é criar a coisa de forma modular para facilitar.

Depois criaremos o componente e moveremos as 3 funções para ele. Um componente guarda código que colabora com os controllers.

Depois então criaremos um plugin e moveremos o componente, assim como o código do groups, users e privileges para o plugin. Um plugin praticamente é um aplicativo que roda dentro do nosso aplicativo, já trazendo controllers, models, templates, configurações, etc.

Iremos criar a função populate() no src/Controller/PrivilegesController.php que cadastrará todos os actions de um grupo para uma tabela.
Ela tem como objetivo cadastrar os actions de todos os controllers para os grupos das tabelas que sejam diferentes de users, groups e privileges, para clientes e outras que adicionarmos.

Edite o arquivo:
src/Controller/PrivilegesController.php

Adicione ao seu início:

```
use Cake\Datasource\ConnectionManager;
use Cake\ORM\TableRegistry;
```

E adicione o código abaixo:

```
	public function populate($group, $controller){
		// Somente se nenhum action tiver sido cadastrado na tabela privileges para o referido grupo
		$conn = ConnectionManager::get('default');
		//$tables = $conn->query("show tables");

		$conn->execute("insert into privileges (group_name,controller,action) values ('$group', '$controller', 'index')");
		$conn->execute("insert into privileges (group_name,controller,action) values ('$group', '$controller', 'view')");
		$conn->execute("insert into privileges (group_name,controller,action) values ('$group', '$controller', 'add')");
		$conn->execute("insert into privileges (group_name,controller,action) values ('$group', '$controller', 'edit')");
		$conn->execute("insert into privileges (group_name,controller,action) values ('$group', '$controller', 'delete')");
	}
```

Adicionar as linhas abaixo para o método beforeFilter:

```
$this->populate('Admins', 'Clientes');
$this->populate('Managers', 'Clientes');
```

Então chame o aplicativo no navegador apenas uma vez e comente as duas linhas acima. Caso execute novamente o aplicativo sem antes comentar as linhas verá erro de violação de integridade.

Este método/função precisa ser chamado para que seja executado, visto que não é um dos métodos nativos do CakePHP.

O beforeFilter() ficará assim:

```
    public function beforeFilter(\Cake\Event\Event $event) {
            parent::beforeFilter($event);

        $this->populate('Admins', 'Clientes');
        $this->populate('Managers', 'Clientes');
}
```

Caso adicionemos uma nova tabela, produtos por exemplo, então adicionamos a tabela, geramos o código para ela e adicionamos a linha:

```
        $this->populate('Admins', 'Produtos');
```

Para o beforeFilter() do controller Privileges e executamos apenas uma vez para comentar a linha.

Estou refletindo agora e acho que podemos melhorar esta função populate(), de forma que não precisemos ficar comentando e descomentando. Podemos fazer um select inicialmente na tabela privileges de forma que verifique se o action está cadastrado e somente faça o insert se não existir.
Vejamos como ficará:

```
        public function populate($group, $controller){
                $conn = ConnectionManager::get('default');

                $actions = $conn->execute("SELECT action FROM `privileges` WHERE
group_name = '$group' and controller = '$controller' and action='index'")->fetchAll();

                if(!$actions){
                        $conn->execute("insert into privileges (group_name,controller,action)
values ('$group', '$controller', 'index')");
                        $conn->execute("insert into privileges (group_name,controller,action)
values ('$group', '$controller', 'view')");
                        $conn->execute("insert into privileges (group_name,controller,action)
values ('$group', '$controller', 'add')");
                        $conn->execute("insert into privileges (group_name,controller,action)
values ('$group', '$controller', 'edit')");
                        $conn->execute("insert into privileges (group_name,controller,action)
values ('$group', '$controller', 'delete')");
                }
        }
```

Como está esta função somente irá cadastrar os 5 actions, caso ainda não tenhamos cadastrado nenhum action desse controller.

Importante:

Lembre de adicionar a linha abaixo ao UsersController.php:
use Cake\Event\Event;

Função Group

Função que recebe o group_name de privileges, de acordo com o group_id do usuário logado.

Retorna o group_name, se existir
Ou false se não existir

```
	public function group_priv($controller,$action,$group){
		//$controller = $this->request->controller;
		//$action = $this->request->action;

		$conn = ConnectionManager::get('default');
		$group = $conn->execute("select group_name from privileges where 
controller = '$controller' and action ='$action' and group_name='$group'")->fetchAll();

		if($group){
			$grouppriv = $group[0][0];
			return $grouppriv;
		}else{
			return false;
		}
	}
```

Ela será chamada de beforeFilter() por enquanto:

```
		// Chamar a função group_priv()
		if($username == 'admin'){
			$this->group_priv($controller, $action,'Admins');
		}elseif($username == 'manager'){
			$this->group_priv($controller, $action,'Managers');
		}elseif($username == 'user'){
			$this->group_priv($controller, $action,'Users');
		}
```

O beforeFilter() agora deve ficar assim:

```
    public function beforeFilter(\Cake\Event\Event $event) {
            parent::beforeFilter($event);

                $controller = $this->request->controller;
                $action = $this->request->action;
                $user = $this->request->session()->read('Auth.User');
                $username=$user['username'];

                $this->Auth->allow(['view', 'index','edit']);

                if($username == 'admin'){
                        $group='Admins';
                }elseif($username == 'manager'){
                        $group='Managers';
                }elseif($username == 'user'){
                        $group='Users';
                }

                $denied='Acesso Negado!';
                $privilege = 'Privilégio não Cadastrado!';
                if($action != 'login' && $action != 'logout'){
                        if($this->go($controller,$action,$group)==false){
                                $this->Flash->error(__($denied));
                                // Voltar para onde veio, ou seja, nem sair de onde está
                                return $this->redirect($this->referer());
                                // return $this->redirect(['controller' => 'Users','action' =>
'login']);
                                // $this->redirect($this->Auth->logout());

                        }else{
                                if($this->msg == 'priv'){
                                        $this->Flash->set(__($this->privilege));
                                }
                        }
                }
        }
```

E nossa função go() assim:

```
        public function go($controller,$action,$groupo){
                $user = $this->request->session()->read('Auth.User');
                $username=$user['username'];
                if($username == 'admin'){
                        $groupo='Admins';
                }elseif($username == 'manager'){
                        $groupo='Managers';
                }elseif($username == 'user'){
                        $groupo='Users';
                }
```

```
                $conn = ConnectionManager::get('default');
                $group = $conn->execute("select group_name from privileges where
controller = '$controller' and action ='$action' and group_name='$groupo'")->fetchAll();

                if($group){
                        $group = $group[0][0];
                }

                if($username == 'admin' && $group=='Admins'){
                        return true;
                }elseif($username == 'manager' && $group == 'Managers' && $controller ==
'Clientes'){
                        return true;
                }elseif($username == 'user' && $group == 'Users' && $controller == 'Clientes'
&& ($action == 'index' || $action == 'view')){
                        return true;
                }else{
                        return false;
                }
        }
```

Controle de Acesso

Precisamos lembrar que devemos cadastrar o acesso no controller Privileges e também ajustar o acesso devido no módulo go(). No caso, o usuário user está com acesso somente ao action index e ao view.

Quinto Aplicativo

Criando um Componente para Controlar o Acesso

Agora iremos criar o esqueleto de um componente usando o bake e mover para ele a função populate() e a função go(), para ficar algo mais profissional.

Criando o Aplicativo

Agora, para criar o aplicativo faremos uma cópia da pasta modelo para app5.

Criação do banco de dados

Criaremos o banco app5 e importaremos o script app5.sql, que contem as tabelas groups, users e privileges, além de clientes.

Configurações

Configurar o banco e as rotas.

Geração do Código do Aplicativo

cd /backup/www/treinamento/app5

bin/cake bake all groups
bin/cake bake all users
bin/cake bake all privileges
bin/cake bake all clientes

Experimente navegar pelo aplicativo

http://localhost/treinamento/app5

Adicionando o Bcrypt

Fazer como fizemos no aplicativo anterior.

Criar Grupos e Usuários

Vamos criar 3 usuários, um para cada grupo:

Groups	Users	Password
Admins	admin	admin
Managers	manager	manager
Users	user	user

Acesse
http://localhost/treinamento/app4/groups/add

E cadastro os 3 grupos acima.

Acesse
http://localhost/treinamento/app4/users/add

E cadastre os 3 usuários acima com as respectivas senhas.
É interessante editar as views src/Template/Users/index.ctp, view.ctp e edit.ctp removendo o campo senha do form.

Cadastro dos Privilégios

Os privilégios de acesso dos usuários aos actions de cada controller é você quem define. Para isso basta cadastrar no controller Privileges, antes de implementar o controle de acesso e fazer um pequneo ajuste no módulo go() se necessário.

Por padrão o controle que já vem na tabela privileges é:

Grupo	**Controller**	**Actions**
Admins	Groups	Todos
Admins	Users	Todos
Admins	Privileges	Todos

Falta Cadastrar o acesso para

Admins	Clientes	Todos
Manager	Clientes	Todos
Users	Clientes	index

Para Admins e Manager usaremos o método populate().
Faremos o cadastro do acesso de Users ao controller Clientes e ao action index pelo navegador, através do controller Privileges, antes de implementar o controle de acesso.

Criando nosso Componente Control

Usando o bake para isso:

cd /backup/www/treinamento/app5

bin/cake bake component Control

Com isso o bake gera o esqueleto do nosso
src/Controller/Component/ControlComponent.php

Vamos começar adicionando as duas linhas abaixo:
use Cake\Datasource\ConnectionManager;
use Cake\ORM\TableRegistry;

Adicione logo abaixo de:
use Cake\Controller\ComponentRegistry;

Abaixo da propriedade adicione a função populate:

```
	public function populate($group, $controller){
		$conn = ConnectionManager::get('default');

		$actions = $conn->execute("SELECT action FROM `privileges` WHERE group_name = '$group' and controller = '$controller' and action='index'")->fetchAll();

		if(!$actions){
			$conn->execute("insert into privileges (group_name,controller,action) values ('$group', '$controller', 'index')");
			$conn->execute("insert into privileges (group_name,controller,action) values ('$group', '$controller', 'view')");
			$conn->execute("insert into privileges (group_name,controller,action) values ('$group', '$controller', 'add')");
			$conn->execute("insert into privileges (group_name,controller,action) values ('$group', '$controller', 'edit')");
			$conn->execute("insert into privileges (group_name,controller,action) values ('$group', '$controller', 'delete')");
		}
	}
```

Logo abaixo da função populate() adicione a função go():

```
	public function go($controller,$action,$groupo){
		$user = $this->request->session()->read('Auth.User');
		$username=$user['username'];
		if($username == 'admin'){
			$groupo='Admins';
		}elseif($username == 'manager'){
			$groupo='Managers';
		}elseif($username == 'user'){
			$groupo='Users';
		}

		$conn = ConnectionManager::get('default');
		$group = $conn->execute("select group_name from privileges where controller = '$controller' and action ='$action' and group_name='$groupo'")->fetchAll();

		if($group){
			$group = $group[0][0];
		}

		if($username == 'admin' && $group=='Admins'){
			return true;
		}elseif($username == 'manager' && $group == 'Managers' && $controller == 'Clientes'){
			return true;
		}elseif($username == 'user' && $group == 'Users' && $controller == 'Clientes' && ($action == 'index' || $action == 'view')){
			return true;
		}else{
```

```
                return false;
            }
        }
```

Ajustando o src/Controller/AppController.php

Removi os comentários

```
<?php
namespace App\Controller;

use Cake\Controller\Controller;
use Cake\Event\Event;

class AppController extends Controller
{
        public $msg;

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

        $this->loadComponent('RequestHandler');
        $this->loadComponent('Flash');
                $this->loadComponent('Control');

        $this->loadComponent('Auth', [
            'loginRedirect' => [
                'controller' => 'Clientes',
                'action' => 'index'
            ],
            'logoutRedirect' => [
                'controller' => 'Users',
                'action' => 'login'
            ],
                        'unauthorizedRedirect' => [
                                'controller' => 'Users',
                                'action' => 'login',
                                'prefix' => false
                        ],
                    'authError' => 'Faça login antes',
                    'authenticate' => [
                        'Form' => [
                            'fields' => ['username' => 'username']
                        ]
                    ],
                    'storage' => 'Session'
        ]);

    }

    public function beforeFilter(\Cake\Event\Event $event) {
```

```
        parent::beforeFilter($event);

            $controller = $this->request->controller;
            $action = $this->request->action;

            //$this->Control->populate('Admins', 'Clientes');
            //$this->Control->populate('Managers', 'Clientes');

            $user = $this->request->session()->read('Auth.User');
            $username=$user['username'];

            $this->Auth->allow(['view', 'index']);//,'add'

            if(isset($username) && $username == 'admin'){
                $group='Admins';
            }elseif($username == 'manager'){
                $group='Managers';
            }elseif($username == 'user'){
                $group='Users';
            }

            $denied='Acesso Negado!';
            $privilege = 'Privilégio não Cadastrado!';
            if(isset($username) && $action != 'login' && $action != 'logout'){
                if($this->Control->go($controller,$action,$group)==false){
                    $this->Flash->error(__($denied));
                    // Voltar para onde veio, ou seja, nem sair de onde está
                    return $this->redirect($this->referer());
                    //return $this->redirect(['controller' => 'Users','action' => 'login']);
                    // $this->redirect($this->Auth->logout());

                }else{
                    if($this->msg == 'priv'){
                        $this->Flash->set(__($this->privilege));
                    }
                }
            }
        }

    public function beforeRender(Event $event)
    {
        if (!array_key_exists('_serialize', $this->viewVars) &&
            in_array($this->response->type(), ['application/json', 'application/xml'])
        ) {
            $this->set('_serialize', true);
        }
    }
}
```

Antes de começar a testar vamos efetuar os devidos registros para Admins, Managers e Users.
Após executar o aplicativo, o método populate() irá cadastrar Clientes para Admins e Managers.
Falta somente abrir o aplicativo como admin e cadastrar index em Clientes para Users.

Agora podemos testar o aplicativo com nosso componente.

http://localhost/treinamento/app5

Faça login como admin, depois logout e login como manager e logout então login como user e navegue pelos vários controllers e actions para ver como reage.

Veja que nenhum action está liberado ao público. Ao acessar o raiz do aplicativo será rdirecionado para o login. Faça o login e experimente.

O aplicativo demo já tem todos estes registros cadastrados e está pronto para ser testado. Apenas crie o banco e configure para testar.

Se tentar logar como usuário **user** terá acesso negado, pois não cadastrou ainda nenhum action para o **user**.
Acesse como admin e vá em privileges e cadastre o controller Clientes e action index para o usuário user. Faça logout e agora login como user.
Agora poderá acessar o index de Clientes.

Tente acessar qualquer outro action que será barrado. Libere agora o action view de Clientes para o user e experimente.

Veja a versão melhorada deste componente no próximo aplicativo (app5a).

Aplicativo app5a

Neste aplicativo apenas melhorarei o componente Control.

Fiz uma cópia do diretório para app5a e copiei também o banco para app5a.

Após copiar diretório e banco e configurar, experimente e veja a diferença. Em termos de funcionalidades para o usuário não mudou tanto mas em termos de código eu procurei otimizar ao máximo e melhorei algumas coisas, confira.

Aplicativo app5b

Aplicativo app5c

Grupos de Usuários

Agora são 4:

Supers - poder total, acessam tudo do aplicativo
Admins -acesso total às tabelas administrativas: groups, users e permissions (agora mudou de privileges para permissions)
Managers - acessam tudo que Admins não acessam: todas as tabelas diferentes de groups, users e permissions
Users - inicialmente tem acesso somente ao login e logout. Para que possa acessar qualquer outro action precisa que seja cadastrado nos actions respectivos. Basta acessar como um Supers ou Admins e cadastrar suas permissões.

Observe que como está os registros na tabela permissions somente serão cadastrados quando for feito o primeiro login no sistema.

Após o primeiro login e execução do sistema, pode comentar as duas linhas respectivas de execução da função populate() no AppController.

Crie o aplicativo fazendo uma cópia do app5b para app5c.

Crie o banco pelo app5c.sql.

Configure o banco, pois as rotas já estão ok.

Abra o aplicativo:

http://localhost/treinamento/app5c

Descomente a linha para permitir add e index no AppController e acesse:
http://localhost/treinamento/app5c/users

Então cadastre 4 usuários, um usuário para cada grupo:

Usuário	Senha	Grupo
super	super	Supers
admin	admin	Admins
manager	manager	Managers
user	user	Users

Então comente novamente a linha que permite add e index.

Faça login no sistema com cada um dos usuários e teste o acesso ao aplicativo.

Sexto Aplicativo

Agora iremos adicionar ao aplicativo um element e um layout. O element criará um menu que facilitará a navegação através dos controllers e o layout deverá diferenciar a área administrativa. Quando o usuário **admin** fizer login ele encontrará um layout diferente dos demais usuários. Podemos criar um layout específico para cada usuário.

Adicionando um Element e um Layout

Devemos customizar o ambiente de trabalho do administrador, usando para isso um element e um layout.

Adicionaremos ao beforeFilter() do AppController:

```
$this->set('title_for_app','Meu Aplicativo');
$this->set('title_for_admin','Administração do Aplicativo');
$this->set('loggedyes','Logado como');
$this->set('loggedno','Não logado!');


$this->set('current_user', $user);
if($user == 'admin'){
        $this->layout = 'admin';
}
```

Criaremos o Elemento topmenu.ctp e o layout_admin.ctp.

Ribamar FS par treinamento dos colegas no DNOCS.