CakePHP

Escola Secundaria Manuel Teixeira Gomes

Programação e Sistemas de Informação – M13

Duarte Manuel Cabrita da Cruz, nº5 12N

DuckIdentity

Conteúdo

Instalação

Primeiro terá de ser criado o "esqueleto" do projeto, para isto teremos de instalar o composer.phar, podemos fazer o download dele através do seguinte link:

`https://getcomposer.org/download/`

Após completar a instalação, na pasta de destino irá ser usado o seguinte código:

```
composer create-project --prefer-dist cakephp/app:4.* trabalhoprojeto
```

Antes de se puder começar a programar, temos de configurar a ligação da DB, para isto termos de aceder a *pasta do projeto/*config/app.php. Dentro deste ficheiro vamos procurar por *Datasources.default*. Assim que se encontre estes ficheiros terão de se fazer algumas modificações:

Nota: Caso esteja a ser utilizada uma versão do cakephp =>4.0 estas modificações terão de ser feitas no ficheiro app_local.php caso exista

```
'default' => [
    'className' => 'Cake\Database\Connection',
    'driver' => 'Cake\Database\Driver\Mysql',
    'persistent' => false,
    'host' => 'localhost',
    'username' => 'cakephp',  //Utilizador ainda têm de ser criado
    'password' => 'bolodephp',
    'database' => 'trabalhoprojeto',      //aqui será colocado o nome
database a ser usada, neste caso esta tabela ainda vai ser criada
    'encoding' => 'utf8mb4',
    'timezone' => 'UTC',
    'cacheMetadata' => true,
],
```

Data Base

Para este trabalho iremos usar uma BD bastante simples com somente 3 tabelas, está tabela será baseada no meu trabalho para a PAP(o qual pode ser consultado no seguinte link(http://www.duckidentity.com/)

Primeiro iremos aceder ao phpmyadmin, caso esteja a usar o wamp como é o meu caso terá de ir ao seguinte link http://localhost/phpmyadmin/.

Se ainda não modificou as credencias padrão o, utilizador padrão é o root com o campo da password em branco.

Antes de criarmos a BD para a segurança da própria é melhor criarmos um utilizador somente para este projeto, para isso vá em Contas de usuário -> Add user Account. No caso deste projeto o utilizador criado foi o cakephp com a password bolodephp.

Agora iremos criar a DB. Para isso clique em SQL e insira o seguinte código.

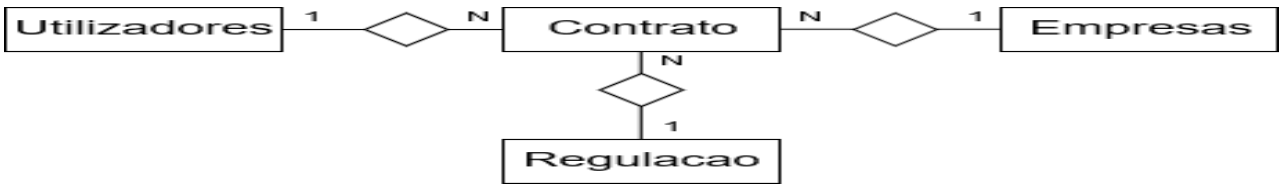
```
CREATE DATABASE trabalhoprojeto //A database podera ter qualquer nome desde
que coincida com o nome colocado no app.php
```

A seguir selecione a data base criada no menu esquerdo. Após ter a BD selecionada vamos voltar a clicar em SQL, para este caso iremos criar 3 tabelas com o seguinte código:

```
CREATE TABLE utilizador(
  id_utilizador int PRIMARY KEY AUTO_INCREMENT,
  nome varchar(90) NOT NULL,
  idade int(2),
  email varchar(250) NOT NULL,
  pass varchar(250) NOT NULL
);
CREATE TABLE empresa(
  id_empresa int PRIMARY KEY AUTO_INCREMENT,
  nome varchar(90) NOT NULL,
  site varchar(150),
  email varchar(250) NOT NULL,
  pass varchar(250) NOT NULL
);
CREATE TABLE regulacao(
    id_regulacao int AUTO_INCREMENT PRIMARY KEY,
    designacao varchar(250) NOT NULL,
    local_efetivo varchar(250) NOT NULL,
    regulamento varchar(250) NOT NULL
);
CREATE TABLE contrato(
    id_contrato int AUTO_INCREMENT PRIMARY KEY,
    id_utilizador int,
    id_empresa int,
    id_regulacao int,
    CONSTRAINT id_utilizador FOREIGN KEY (id_utilizador) REFERENCES
utilizador (id_utilizador),
    CONSTRAINT id_empresa FOREIGN KEY (id_empresa) REFERENCES empresa
(id_empresa),
    CONSTRAINT id_regulacao FOREIGN KEY (id_regulacao) REFERENCES regulacao
(id_regulacao),
    descricao varchar(250) DEFAULT 'Contrato padrão',
    created DATETIME,
    modified DATETIME
);
```

Nota: Caso as ligações entre as tabelas não sejam criadas provavelmente as tabelas não estão na engine certa para isto será preciso altera-la ex: `ALTER TABLE nome_da_tabela ENGINE = INNODB;`
Depois poderá criar as tabelas via código, ou na opção desenhador do phpmyadmin(Selecionar tabela -> Menu superior->Desenhador->Menu que aparecerá a direita das DB's selecione 'Create relationship'-pretende associar)

O DER final desta DB será o seguinte:



Código!

Agora finalmente vamos começar a parte divertida deste projeto, a programação!

Para melhor compreensão do que se está a ser feito irei dividir a parte do código em 6 partes.

1. Criação das Land Pages,
2. Visualizar Registos
3. Novos registos,
4. Modificação de registos,
5. Eliminação de registos

Parte 1 – Criação de Land Pages

A primeira land page que vamos criar será a dos utilizadores. Para isto iremos ter de criar 3 ficheiros dentro da *pasta_do_projeto/src/.*

O primeiro ficheiro será criado em *model/table/* com o nome *UtilizadorTable.php* com o seguinte código:

```php
<?php

namespace App\Model\Table;
use Cake\ORM\Table;

class UtilizadorTable extends Table
{
    public function initialize(array $config): void
    {
        $this->addBehavior('Timestamp');
    }
}
?>
```

De seguida iremos criar o controlador, este será criado em *controller/ com o nome UtilizadorController.php* e o seguinte código:

```php
<?php

namespace App\Controller;

class UtilizadorController extends AppController
{
    public function index()
    {
        $this->loadComponent('Paginator');
        $utilizadores = $this->Paginator->paginate($this->Utilizador->find());
        $this->set(compact('utilizadores'));
    }
}
?>
```

Nota:Este codigo ainda irá sofrer alterações mais para a frente

Para finalizarmos está primeira parte iremos criar a Land Page propriamente dita. Para isto temos de ir em *Template/* dentro desta pasta iremos criar uma subpasta chamada *Utilizadores* sendo que dentro desta subpasta por sua vez iremos criar um ficheiro chamado *index.ctp*. Dentro deste ficheiro iremos colocar o seguinte código.

Nota: Caso esteja a ser utilizada uma versão do cakephp =>4.0 a pasta a ser criada será fora da pasta src, tendo de ir a pasta Template/ que lá estara, e as terminações dos ficeiros não serão .ctp mas sim .php

```
<h1>Utilizadores</h1>
<table>
    <tr>
        <th>Id</th>
        <th>Nome</th>
        <th>Idade</th>
        <th>Email</th>
    </tr>

    <?php foreach ($utilizadores as $utilizador): //Lista todos os
utilizadores existentes na tabela?>
        <tr>
            <td><?= $utilizador->id_utilizador ?></td>
            <td>
                <?= $utilizador->nome ?>
            </td>
            <td>
                <?= $utilizador->idade ?>
            </td>
            <td>
                <?= $utilizador->email ?>
            </td>
        </tr>
    <?php endforeach; ?>
</table>
```

Para as outras tabelas teremos de repetir o mesmo processo, mudando apenas alguns pormenores, devido a isso ser algo tão banal não estará incluído nesta documentação

Considerando que as outras tabelas já estão ligadas vamos criar uma forma de navegar entre elas. Para isso vamos ao ficheiro *src\Template\Layout\default.ctp* e vamos encontrar o seguinte código:

Nota: Caso esteja a ser utilizada uma versão do cakephp =>4.0 o ficheiro encontra-se em \templates\layout\default.php

```
<a target="_blank" href="https://book.cakephp.org/4/">Documentation</a>
<a target="_blank" href="https://api.cakephp.org/4/">API</a>
```

Substitua o código por:

```
<?= $this->Html->link(__('Utilizadores'), ['action' => '../utilizador/']) ?>
<?= $this->Html->link(__('Empresas'), ['action' => '../empresa/']) ?>
<?= $this->Html->link(__('Regulações'), ['action' => '../regulacao/']) ?>
```

Parte 2 – Visualizar registos

Para conseguir visualizar um registo em especifico vamos ter de criar um novo ficheiro view.ctp, e fazer umas pequenas alterações nos ficheiros index.ctp e Controller

Nota: Caso esteja a ser utilizada uma versão do cakephp =>4.0 os ficheiros index e view têm a terninação .ctp

A primeira coisa a fazer é tratar das modificações no Controller, sendo que no caso desta documentação iremos usar o UtilizadorController.php

Logo a seguir a função índex(mas ainda dentro da classe) iremos adicionar o seguinte código que irá criar a ação View:

```
public function view($id) //Visualização de um registo específico
{
    $utilizadores = $this->Utilizador->get($id);
    $this->set(compact('utilizadores'));
}
```

Agora iremos fazer as alterações necessárias no index.ctp

Procure pela linha:

```
<td><?= $utilizador->id_utilizador ?></td>
```

E substitua por:

```
<td>
    <?= $this->Html->link($utilizador->id_utilizador, ['action' => 'view',
$utilizador->id_utilizador) ?>
</td>
```

Agora para podermos ver a "view" falta cria-la para isso dentro da mesma pasta onde esta o index.ctp vamos criar uma chamada view.ctp com o seguinte código:

```
<h1><?= h($utilizadores->nome) ?></h1>
<p>Email: <?= h($utilizadores->email) ?></p>
<p>Idade: <?= h($utilizadores->idade) ?></p>
<p>ID: <?= h($utilizadores->id_utilizador) ?></p>
<?= $this->Html->link(__('Voltar'), ['action' => 'index']) ?>
```

Agora a semelhança do index temos de repetir o processo para cada uma das tabelas.

Parte 3 – Novos registos

Para criar novos registos o processo de criação de ficheiros é bastante parecido ao de visualizar, sendo assim vamos começar por modificar o nosso controlador adicionando o seguinte código.

```
public function add()
{
    $utilizador = $this->Utilizador->newEmptyEntity();
    if ($this->request->is('post')) {
        $utilizador = $this->Utilizador->patchEntity($utilizador, $this-
>request->getData());

        if ($this->Utilizador->save($utilizador)) {
            $this->Flash->success(__('Utilizador criado com sucesso.'));
            return $this->redirect(['action' => 'index']);
        }
        $this->Flash->error(__('Ocorreu um erro ao criar o utilizador.'));
    }
    $this->set('utilizador', $utilizador);
}
```

Vamos modificar agora o nosso index

Depois do código:

```
<h1>Utilizadores</h1>
```

Vamos adicionar:

```
<p><?= $this->Html->link("Adicionar Utilizador", ['action' => 'add']) ?></p>
```

Depois vamo criar a o add.ctp com o seguinte código.

```
<h1>Criar Utilizador</h1>
<?php
echo $this->Form->create($utilizador);
echo $this->Form->control('nome');
echo $this->Form->control('email', ['rows' => '3']);
echo $this->Form->control('idade', ['type' => 'number']);
echo $this->Form->control('pass', ['type' => 'password']);
echo $this->Form->button(__('Salvar Utilizador'));
echo $this->Form->end();
?>
```

Parte 4 – Modificação de registos

Para modificar registos o processo de criação de ficheiros é mais do mesmo das outras duas partes, sendo assim vamos voltar a modificar o controlador. Para isso vamos adicionar o seguinte código:

```
public function edit($id = null) //Edição de um registo específico
{
    $utilizadores = $this->Utilizador->get($id);
    if ($this->request->is(['post', 'put'])) {
        $this->Utilizador->patchEntity($utilizadores, $this->request-
>getData());
        if ($this->Utilizador->save($utilizadores)) {
            $this->Flash->success(__('O Utilizador foi atualizado.'));
            return $this->redirect(['action' => 'index']);
        }
        $this->Flash->error(__(O utilizador não pôde ser atualizado.'));
    }

    $this->set('utilizador', $utilizadores);
}
```

Já no index vamos fazer as seguintes alterações:

No cabeçalho da tabela depois de:

```
<th>Email</th>
```

Irá se adicionar:

```
<th>Editar</th>
```

E depois de:

```
<td><?= $utilizador->email ?> </td>
```

Irá se adicionar:

```
<td><?= $this->Html->link('Editar', ['action' => 'edit', $utilizador-
>id_utilizador]) //Onde iremos editar o registo ?></td>
```

Agora terá de ser criado um ficheiro chamado edit.ctp com o seguinte código

```
<h1><?= h($utilizador->nome) ?></h1>

<?php
echo $this->Form->create($utilizador);
```

```
echo $this->Form->control('email', ['rows' => '3']);
echo $this->Form->control('idade', ['type' => 'number']);
echo $this->Form->control('pass', ['type' => 'password']);
echo $this->Form->button(__('Guardar utilizador'));
echo $this->Form->end();
?>
```

Parte 5 – Eliminação de registos

O código para apagar um registo é relativamente mais simples que os outos, não sendo preciso criar uma parte gráfica.

Para começar iremos adicionar o seguinte código ao Controlador:

```
public function delete($id) //Eliminaçãp de um registo específico
{
    $this->request->allowMethod(['post', 'delete']);

    $utilizadores = $this->Utilizador->get($id);
    if ($this->Utilizador->delete($utilizadores)) {
        $this->Flash->success(__('O utilizador com id: {0} foi apagado.',
h($id)));
        return $this->redirect(['action' => 'index']);
    }
}
```

E no index a semelhança da modificação vamos criar uma coluna na tabela chamada apagar, para isto colocamos a seguir de:

```
<th>Editar</th>
```

O código:

```
<th>Eliminar</th>
```

E para a ação colocamos o seguinte código(a seguir da ação editar)

```
<td>
    <?= $this->Form->postLink(
        'Apagar',
        ['action' => 'delete', $utilizador->id_utilizador],
        ['confirm' => 'Deseja apagar o utilizador selecionado?'])
    ?>
</td>
```

Visualização de múltiplas tabelas

Para visualizarmos varias tabelas temos de fazer uma pequena modificação ao ficheiro table e ao index. No nosso caso a tabela em questão é a tabela contrato

```
<?php

namespace App\Model\Table;

use Cake\ORM\Table;

class ContratoTable extends Table//Criação da tabela Contrato
{
    public function initialize(array $config): void
    {
        $this->addBehavior('Timestamp');
        $this->setPrimaryKey('id_contrato');
```

```
        $this->belongsTo('Utilizador') //Isto é necessário pois temos de ciar
uma associação entre as duas tabelas
            ->setForeignKey('id_utilizador')
            ->setJoinType('INNER');

        $this->belongsTo('Empresa')
            ->setForeignKey('id_empresa')
            ->setJoinType('INNER');

        $this->belongsTo('Regulacao')
            ->setForeignKey('id_regulacao')
            ->setJoinType('INNER');
    }

}
?>
```

E como estamos a trabalhar com uma View e não com uma tabela só podemos criar o index, para isso usasse o seguinte código:

```
<h1>Contrato</h1>
<p><?= $this->Html->link("Adicionar contrato", ['action' => 'add']) ?></p>
<table>
    <tr>
        <th>Id</th>
        <th>Email utilizador</th>
        <th>Nome empresa</th>
        <th>Id Regulação</th>
        <th>Descrição</th>
        <th>Editar</th>
        <th>Eliminar</th>
    </tr>

    <?php foreach ($contratos as $contrato): //Lista todos os utilizadores
existentes na tabela?>
        <tr>
            <td> <?= $this->Html->link($contrato->id_contrato, ['action' =>
'view', $contrato->id_contrato]) ?></td>
            <td><?= $contrato->has('id_utilizador') ? $this->Html-
>link($contrato->utilizador->email, ['controller' => 'Utilizador', 'action'
=> 'view', $contrato->utilizador->id_utilizador]) : '' ?> </td>
            <td><?= $contrato->has('id_empresa') ? $this->Html-
>link($contrato->empresa->nome, ['controller' => 'Empresa', 'action' =>
'view', $contrato->empresa->id_empresa]) : '' ?></td>
            <td><?= $contrato->id_regulacao ?> td>
            <td><?= $contrato->descricao ?></td>
            <td><?= $this->Html->link('Editar', ['action' => 'edit',
$contrato->id_contrato]) //Onde iremos editar o código?></td>
            <td><?= $this->Form->postLink(
                    'Apagar',
                    ['action' => 'delete', $contrato->id_contrato],
                    ['confirm' => 'Deseja apagar o utilizador selecionado?'])
                ?></td>
        </tr>
    <?php endforeach; ?>
</table>
```

Web Grafia

Composer: https://getcomposer.org/download/

Software para virtualização do SV: http://www.wampserver.com/en/

Ajuda na Sintaxe SQL: https://www.eversql.com/sql-syntax-check-validator/

Desenhos: https://www.draw.io/

Edição de Imagem: https://www.fotor.com/

Fóruns:

https://stackoverflow.com/questions/10039187/changing-table-type-to-innodb#10039256