01

Adicionando novas fotos

Transcrição

Começando deste ponto? Você pode fazer o DOWNLOAD (https://s3.amazonaws.com/caelum-online-public/angular-1/stages/06-alurapic.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

Não podemos postergar ainda mais, precisamos concluir o cadastro de novas fotos! Sabemos que a rota do Angular `localhost:3000/fotos/new` exibirá a view, mas não é nada elegante pedir que o usuário digite esse endereço toda vez quando for cadastrar uma foto, não?

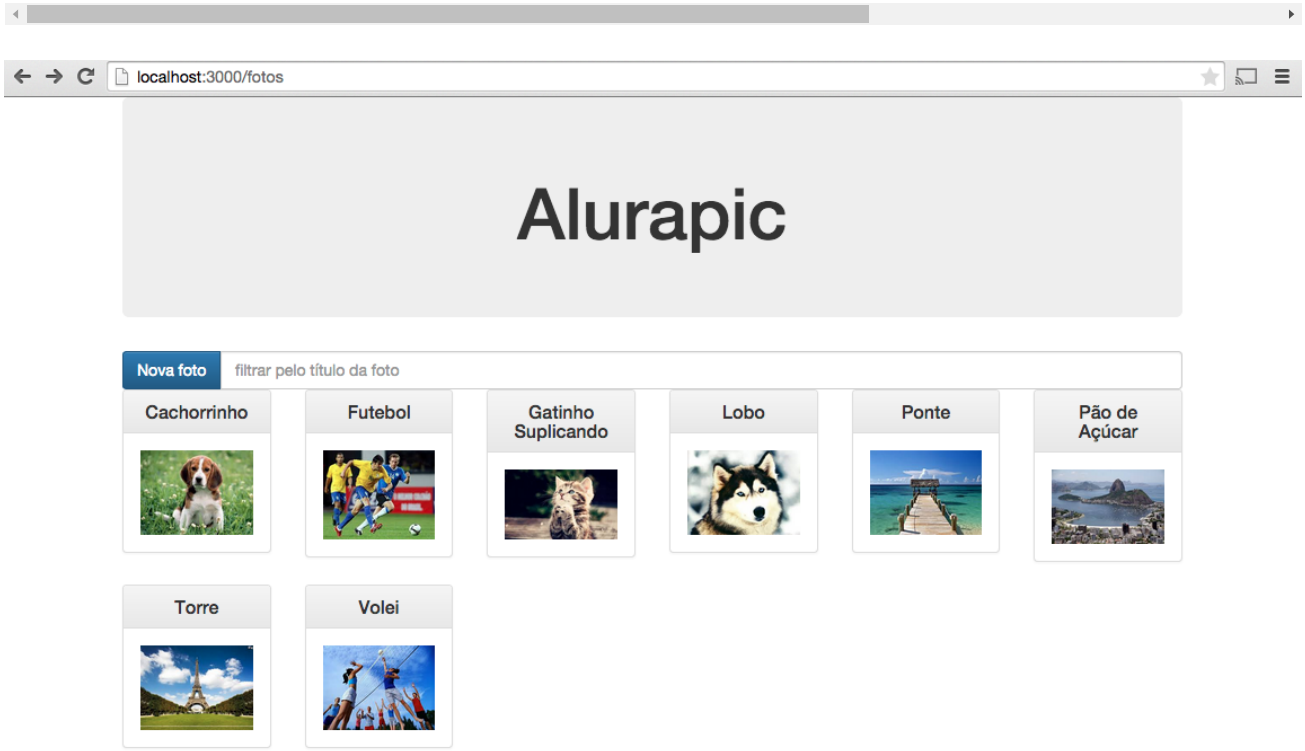
Navegando entre views

Podemos melhorar sua experiência adicionando um link que, ao ser clicado na parcial `principal.html`, chamará nossa rota que exibirá nossa tela de cadastro.

Vamos editar `public/partials/principal.html` para adicionarmos um link que terá um visual de botão graças ao Bootstrap:

```
<!-- public/partials/principal.html -->

<div class="jumbotron">
    <h1 class="text-center">Alurapic</h1>
</div>

<div class="row">
    <div class="col-md-12">
        <form>

            <!-- Novidade! -->

             <div class="input-group">
                <span class="input-group-btn">
                    <a href="/fotos/new" class="btn btn-primary" type="button">
                        Nova foto
                    </a>
                </span>
                <input class="form-control" placeholder="filtrar pelo título da foto" ng-model="

            </div>

            <!-- fim novidade! -->

        </form>
    </div> <!-- fim col-md-12 -->
</div> <!-- fim row -->

<div class="row">
    <meu-painel class="col-md-2 painel-animado" ng-repeat="foto in fotos | filter: filtro" titu
        <minha-foto url="{{foto.url}}" titulo="{{foto.titulo}}">
        </minha-foto>
```

```
    </meu-painel>
</div>
```



Agrupamos nosso botão e nosso campo de pesquisa dentro de um `input-group` do Bootstrap, inclusive nosso botão decora o campo de pesquisa com a classe `input-group-btn` .

Repare também que nosso botão, na verdade um link, aponta para o endereço `/fotos/new` , justamente a rota que já temos registrada. Agora que já conseguimos navegar entre as parciais `principal.html` e `foto.html` , podemos atacar a marcação desta última.

Nossa primeira view de cadastro

Vamos criar nosso formulário de cadastro, mas ainda sem nos preocuparmos com expressões do Angular, inclusive já vamos adicionar os botões "salvar" e "voltar":

```
<!-- public/partials/foto.html -->

<div class="page-header text-center">
    <h1>TITULO DA FOTO AQUI</h1>
</div>

<form name="formulario" class="row">
    <div class="col-md-6">
        <div class="form-group">
            <label>Título</label>
            <input name="titulo" class="form-control">
        </div>
        <div class="form-group">
            <label>URL</label>
            <input name="url" class="form-control">
        </div>
        <div class="form-group">
            <label>Descrição</label>
```

```
            <textarea name="descricao" class="form-control">
            </textarea>
        </div>

        <button type="submit" class="btn btn-primary">
            Salvar
        </button>
         <a href="/" class="btn btn-primary">Voltar</a>
        <hr>
    </div>
    <div class="col-md-6">
        <minha-foto></minha-foto>
    </div>
</form>
```

Excelente, já podemos verificar o resultado!

Queremos agora que cada input do nosso formulário alimente as propriedades de um objeto `foto` sem qualquer informação. Vamos adicionar a já conhecida diretiva `ng-model` para cada um dos inputs. No título da página, usaremos uma AE:

```
<!-- public/partials/foto.html -->

<div class="page-header text-center">
    <h1>{{foto.titulo}}</h1>
</div>

<form name="formulario" class="row">
    <div class="col-md-6">
        <div class="form-group">
            <label>Título</label>
            <input name="titulo" class="form-control" ng-model="foto.titulo">
        </div>
        <div class="form-group">
            <label>URL</label>
            <input name="url" class="form-control" ng-model="foto.url">
        </div>
        <div class="form-group">
            <label>Descrição</label>
            <textarea name="descricao" class="form-control" ng-model="foto.descricao">
```

```
                </textarea>
            </div>

            <button type="submit" class="btn btn-primary">
                Salvar
            </button>
             <a href="/" class="btn btn-primary">Voltar</a>
            <hr>
        </div>
        <div class="col-md-6">
            <minha-foto></minha-foto>
        </div>
    </form>
```

Quando recarregamos nossa página, o título some e o formulário ainda é exibido. Lembre-se que uma AE não avaliada não resulta em erro, mas apenas na ausência de valor no local onde é utilizada. A mesma coisa acontece com a diretiva `ng-model` .

Agora que já temos o "esqueleto" da nossa tela de cadastro, precisamos de um controller que nos dê suporte para a operação de cadastro. Vamos criar o arquivo `public/js/controllers/foto-controller.js` e definir o controller **FotoController** (no singular), importando-o logo em seguida na view principal da aplicação `public/index.html` :

```
// public/js/controllers/foto-controller.js
angular.module('alurapic')
    .controller('FotoController', function($scope) {

        $scope.foto = {};

    });
```

```
<!-- public/index.html -->
<!DOCTYPE html>
<html lang="pt-br" ng-app="alurapic">
    <head>
        <base href="/">
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <title>Alurapic</title>
        <link rel="stylesheet" href="css/bootstrap.min.css">
        <link rel="stylesheet" href="css/bootstrap-theme.min.css">
        <link rel="stylesheet" href="css/efeitos.css">
        <script src="js/lib/angular.min.js"></script>
        <script src="js/lib/angular-animate.min.js"></script>
        <script src="js/lib/angular-route.min.js"></script>
        <script src="js/main.js"></script>
        <script src="js/controllers/fotos-controller.js"></script>

        <!-- importando o novo controller -->

        <script src="js/controllers/foto-controller.js"></script>
        <script src="js/directives/minhas-diretivas.js"></script>
    </head>
    <body>
        <div class="container">
```

```
            <ng-view></ng-view>
        </div><!-- fim container -->
    </body>
</html>
```

Veja que não importamos o novo script como último script. Por uma questão de organização apenas, ele foi importado imediatamente após o controller já existente.

Vamos voltar nossa atenção para `FotoController`. Veja que `$scope` disponibiliza a propriedade `foto`, um objeto JavaScript, porém sem qualquer propriedade. Não se preocupe, como estamos usando a diretiva `ng-model`, a propriedade indicada será criada automaticamente no objeto, sendo assim, se usamos `ng-model="foto.titulo"`, o Angular criará automaticamente em `$scope.foto` a propriedade `titulo`, inclusive atribuindo o valor digitado pelo usuário.

Preparando o terreno para cadastrarmos novas fotos

O que precisamos agora é implementar o botão `salvar`. O que ele deve fazer? Submeter o formulário, claro, mas precisamos acessar `$scope.foto` no momento da submissão para que possamos enviar os dados assincronamente através do serviço `$http`. JavaScript possui o evento **submit** justamente para isso.

O evento `submit` é disparado quando um formulário é submetido e nele podemos adicionar uma função que permite a execução de um código arbitrário que pode cancelar sua submissão caso haja algum problema, como o de um campo que não foi preenchido. Mas estamos usando Angular, e agora? Como interagir com a interface de eventos do JavaScript?

O Angular suporta a interface de eventos do JavaScript através de diretivas. Por exemplo, se quisermos o evento `click`, usamos a diretiva `ng-click`, o evento mouseouver, a diretiva `ng-mouseover` e assim por diante. Sendo assim, para lidarmos com o evento `submit` disparado pelo formulário adicionamos a diretiva **ng-submit** diretamente na tag `form`:

```
<!-- public/partials/foto.html -->

<!-- código anterior omitido -->

<form name="formulario" class="row" ng-submit="submeter()">

<!-- código posterior omitido -->
```

Note que o valor da diretiva `ng-submit` chama uma função que deve ser definida na propriedade `$scope.submeter`. Vamos adicioná-la em nosso controller, porém exibiremos apenas os dados da foto no console do navegador:

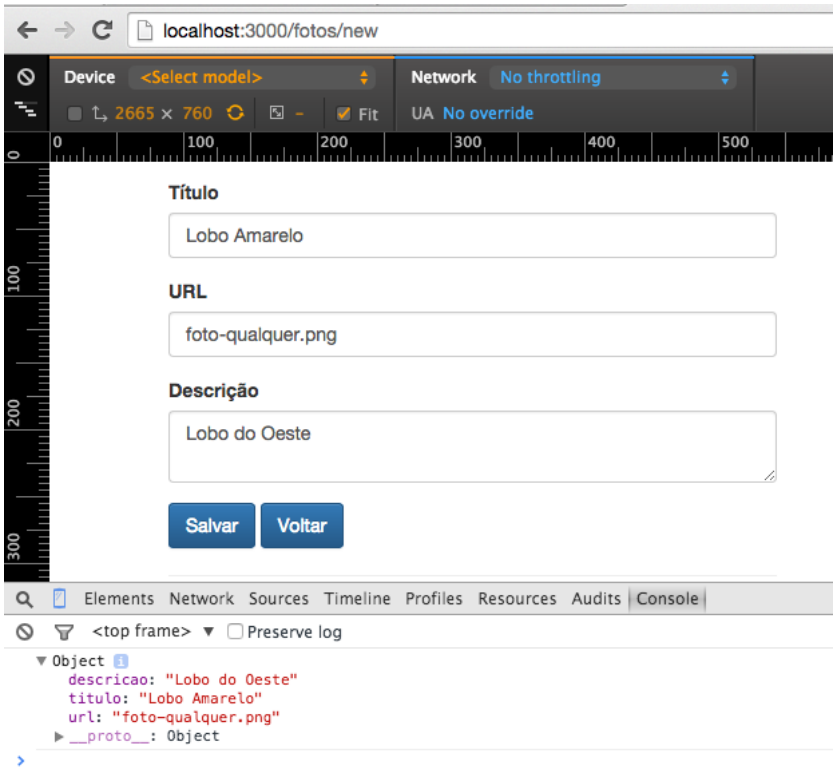
```
// public/js/controllers/foto-controller.js

angular.module('alurapic')
    .controller('FotoController', function($scope) {

        $scope.foto = {};

        $scope.submeter = function() {
            console.log($scope.foto);
        };
```

```
});
```

Se testarmos nosso código, nada acontecerá, por quê? O motivo é simples: não associamos `FotoController` à view parcial `foto.html`. Lembram onde realizamos essa associação? Na configuração de rotas! Vamos editar o arquivo `public/js/main.js` para adicionarmos a chave `controller` que faltava para a foto `/fotos/new`:

```
// public/js/main.js

angular.module('alurapic', ['minhasDiretivas', 'ngAnimate', 'ngRoute'])
    .config(function($routeProvider, $locationProvider) {

        $locationProvider.html5Mode(true);

        $routeProvider.when('/fotos', {
            templateUrl: 'partials/principal.html',
            controller: 'FotosController'
        });

        // adicionando a propriedade controller que faltava.

        $routeProvider.when('/fotos/new', {
            templateUrl: 'partials/foto.html',
            controller: 'FotoController'
        });

        $routeProvider.otherwise({redirectTo: '/fotos'});

    });
```

Agora já podemos recarregar nossa página, clicar no botão `nova foto`, preencher alguma informação e clicar no botão salvar. Para vermos os dados da foto, precisamos abrir o console do navegador com F12 (CMD + SHIFT + C, no MAC):



Funciona! Agora só nos resta enviar os dados capturados para uma rota back-end especializada nesta tarefa, usando o serviço `$http`. Porém, não é incomum validarmos os dados do usuário verificando a obrigatoriedade de algum campo ou aplicando alguma regra mais específica de validação. Em nossa aplicação não será diferente e faremos isso através do Angular.