01

Melhorando o cadastro

Transcrição

Começando deste ponto? Você pode fazer o DOWNLOAD (https://s3.amazonaws.com/caelum-online-public/angular-1/stages/08-alurapic.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

Já conseguimos adicionar, alterar e remover fotos, mas há uma informação que também precisamos cadastrar: cada foto pode pertencer a 3 grupos: esportes, lugares e animais. Uma maneira de forçarmos o usuário a escolher um desses grupos é usar uma combobox que, ao ser clicado, exibe todas opções possíveis. No mundo HTML usamos a tag `select`, onde cada uma das opções é representada pela tag `option`.

Nossa primeira combobox!

No lugar de deixarmos as opções fixas no HTML, montaremos nossa combobox dinamicamente pelo Angular com os dados retornados pelo endereço `http://localhost:3000/v1/grupos`. Já sabemos obter dados do nosso servidor com o serviço `$http.get`.

Em primeiro lugar, vamos adicionar a marcação da nossa combobox em `foto.html`:

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

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

<p ng-show="mensagem.length" class="alert alert-info">{{mensagem}}</p>

<form novalidate name="formulario" class="row" ng-submit="submeter()">
    <div class="col-md-6">
        <div class="form-group">
            <label>Título</label>
            <input name="titulo" class="form-control"
                ng-model="foto.titulo" required
                ng-maxlength="20">
            <span ng-show = "formulario.$submitted && formulario.titulo.$error.required"
                class="form-control alert-danger">
                Título obrigatório
            </span>
            <span ng-show="formulario.$submitted && formulario.titulo.$error.maxlength" class="
                No máximo 20 caracteres!
            </span>
        </div>
        <div class="form-group">
            <label>URL</label>
            <input name="url" class="form-control"
            ng-model="foto.url" required>

            <span ng-show = "formulario.$submitted && formulario.url.$error.required"
                class="form-control alert-danger">
```

```
                URL obrigatória
            </span>

        </div>
        <div class="form-group">
            <label>Descrição</label>
            <textarea name="descricao" class="form-control" ng-model="foto.descricao">
            </textarea>
        </div>

        <!-- novidade aqui! -->

        <div class="form-group">
            <label>Grupo</label>
            <select name="grupo" class="form-control" required>
                <option value="">Escolha um grupo</option>
            </select>
            <span ng-show="formulario.$submitted && formulario.grupo.$error.required" class="fo
                Grupo obrigatório
            </span>
        </div>

        <button type="submit" class="btn btn-primary" ng-disabled="formulario.$invalid">
            Salvar
        </button>
         <a href="/" class="btn btn-primary">Voltar</a>
        <hr>
    </div>
    <div class="col-md-6">
        <minha-foto url="{{foto.url}}" titulo="{{foto.titulo}}">
        </minha-foto>
    </div>
</form>
```

Agora, vamos criar um controller que fornecerá única e exclusivamente nossa lista de grupos. Vamos criar o arquivo `public/js/controllers/grupos-controller.js` :

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

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

    $scope.grupos = [];

        $http.get('/v1/grupos')
        .success(function(grupos) {
            $scope.grupos = grupos;
        })
        .error(function(erro) {
            console.log(erro);
        });
    });
```

Não podemos deixar de importá-lo em `index.html` :

```
<!-- 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>
        <script src="js/controllers/foto-controller.js"></script>

        <!-- novo aqui! -->
        <script src="js/controllers/grupos-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>
```

Prontinho! Mas como iremos associar `GruposController` com a tag `select`? Simples, através da já conhecida diretiva `ng-controller`. Aproveitaremos também para associar a tag com `$scope.foto.grupo` através da diretiva `ng-model`.

```
<!-- public/partials/foto.html -->
<!-- código anterior omitido -->

        <div class="form-group">
            <label>Grupo</label>
            <select name="grupo"
                    ng-model="foto.grupo" class="form-control" required
                    ng-controller="GruposController">
                <option value="">Escolha um grupo</option>
            </select>
            <span ng-show="formulario.$submitted && formulario.grupo.$error.required" class="foı
                Grupo obrigatório
            </span>
        </div>

<!-- código posterior omitido -->
```

Por mais que `foto.html` seja gerenciada por `FotoController`, definido no cadastro de rotas do Angular, nossa tag `select` será gerenciada por `GruposController`. Fantástico!

Mais uma diretiva do angular para nos ajudar! A ng-options

Agora, vamos montar nossas opções através da diretiva *ng-options*:

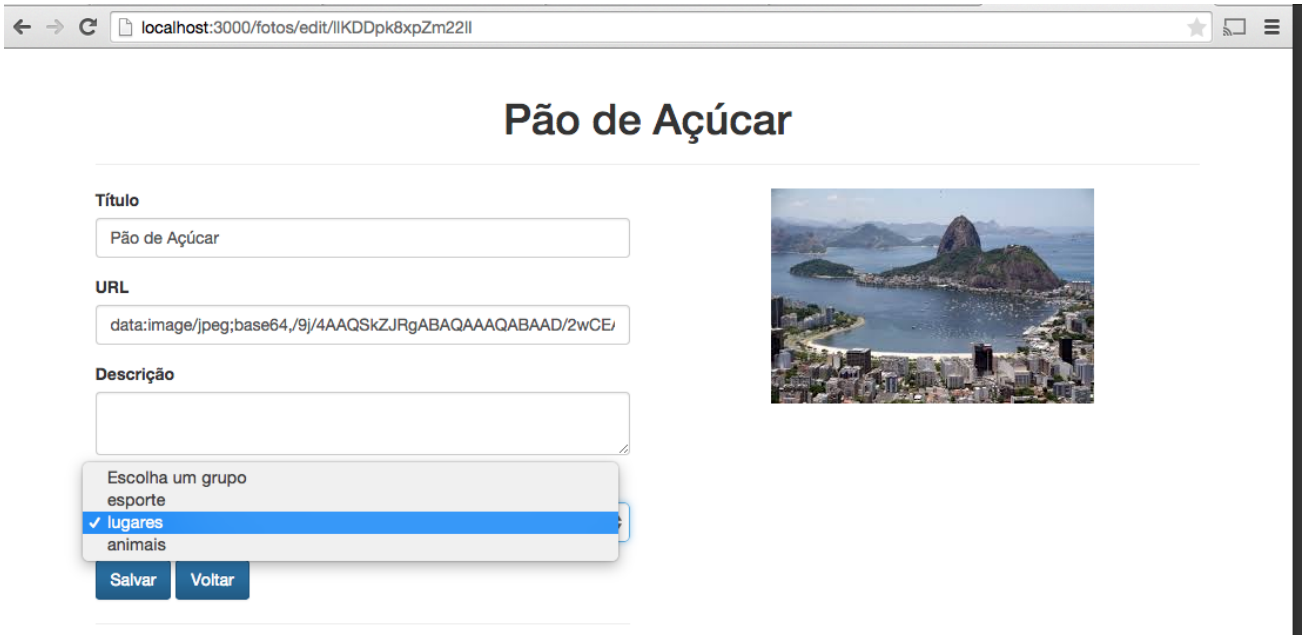
```
<!-- public/partials/foto.html -->
<!-- código anterior omitido -->

        <div class="form-group">
            <label>Grupo</label>
            <select name="grupo"
                    ng-model="foto.grupo" class="form-control" required
                    ng-controller="GruposController"
                    ng-options="grupo._id as grupo.nome for grupo in grupos">
                <option value="">Escolha um grupo</option>
            </select>
            <span ng-show="formulario.$submitted && formulario.grupo.$error.required" class="fo
                Grupo obrigatório
            </span>
        </div>

<!-- código posterior omitido -->
```

Adicionamos a diretiva **ng-options**, que possui comportamento parecido com `ng-repeat` , porém a sintaxe `"grupo._id as grupo.nome"` indica que o valor do elemento será o ID do grupo e o que será exibido para seleção será seu nome. O restante `"for grupo in grupos"` percorrerá a lista de grupos disponibilizada no escopo do controller, construindo cada item de nossa lista.

Já podemos visualizar o resultado:



Melhor que isso só "dois disso"!

Filtros que transformam!

Podemos alterar o grupo, salvar, listar e alterar novamente e verificarmos que o grupo é modificado realmente. Mas podemos ainda melhorar um detalhe: veja que quem cadastrou lá no servidor o nome dos grupos utilizou caixa baixa.

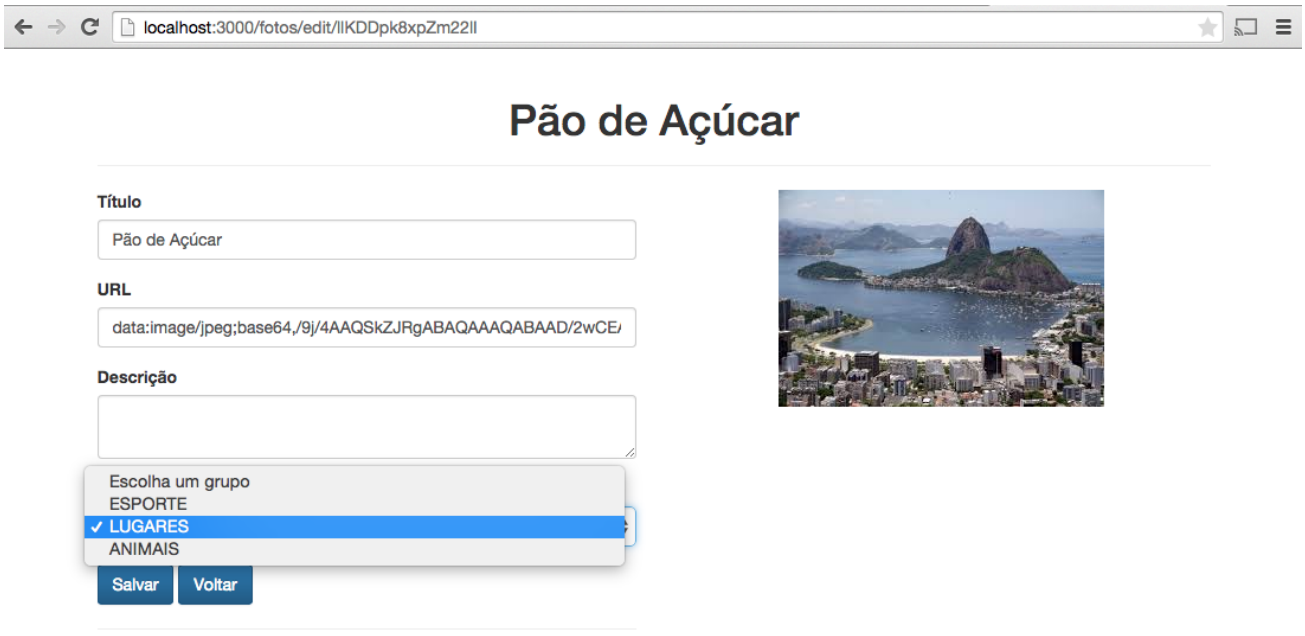
Para melhorar a experiência do usuário, seria interessante colocar o nome dos grupos em caixa alta. Será que precisaremos incomodar o fera do back-end para que retorne os dados em caixa alta? Não, o Angular consegue dar conta disso através de filtros. Utilizamos o filtro `filter` em `ng-repeat`, agora podemos utilizar o filtro `uppercase` para colocar o nome do grupo em caixa alta. Vamos alterar a diretiva `ng-options`:

```
<!-- public/partials/foto.html -->
<!-- código anterior omitido -->

        <div class="form-group">
            <label>Grupo</label>
            <select name="grupo"
                    ng-model="foto.grupo" class="form-control" required
                    ng-controller="GruposController"
                    ng-options="grupo._id as (grupo.nome | uppercase) for grupo in grupos">
                <option value="">Escolha um grupo</option>
            </select>
            <span ng-show="formulario.$submitted && formulario.grupo.$error.required" class="for
                Grupo obrigatório
            </span>
        </div>

<!-- código posterior omitido -->
```

Verificando o resultado:



Terminamos, mas vamos fechar esta aula com mais uma diretiva. Topa?

Facilitando mais uma vez nossa vida com diretivas

Vamos criar a diretiva `meu-botao-perigo`. Na verdade, ela é um atalho para a seguinte estrutura que usamos em `principal.html`:

```
<button class="btn btn-danger btn-block" ng-click="acao()">{{nome}}</button>
```

Fica claro que ela terá dois parâmetros: a função que desejamos executar e o nome do botão. Também vamos restringir seu uso à tag:

```
// public/js/directives/minhas-diretivas.js

angular.module('minhasDiretivas', [])
    .directive('meuPainel', function() {
        // código omitido
    })
    .directive('minhaFoto', function() {
        // código omitido
    })
    .directive('meuBotaoPerigo', function() {
        var ddo = {};
        ddo.restrict = "E";
        ddo.scope = {
            nome: '@',
            acao : '@'
        }
        ddo.template = '<button class="btn btn-danger btn-block" ng-click="acao()">{{nome}}</but

        return ddo;
    });
```

Excelente, até agora nenhuma novidade. Vamos agora substituir o botão `Remover` em `principal.html` pela nossa diretiva:

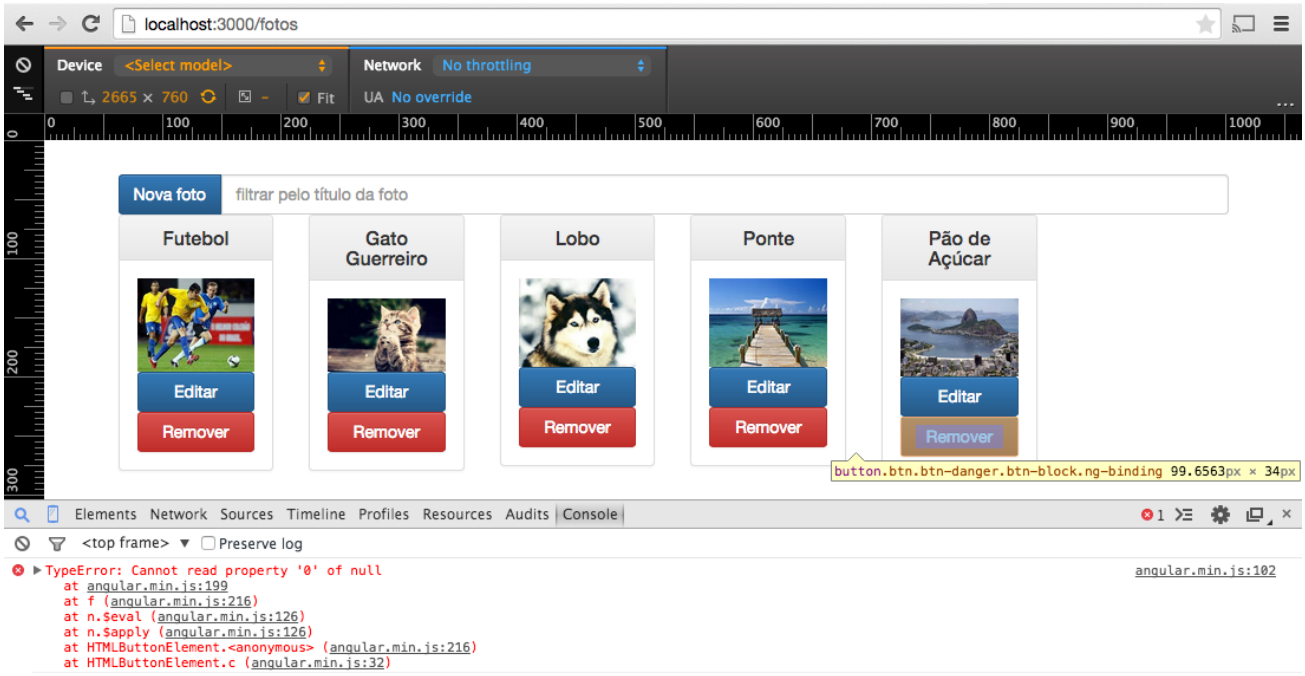
```
<!-- public/partials/principal.html -->
<!-- código anterior omitido -->

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

        <a class="btn btn-primary btn-block" href="/fotos/edit/{{foto._id}}">
            Editar
        </a>
        <meu-botao-perigo nome="Remover" acao="remover(foto)"></meu-botao-perigo>

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

Será que funciona? Recarregando nossa página o botão é exibido, ufa! Porém, quando clicamos nada acontece. Se abrirmos o console do navegador vemos uma mensagem de erro:



Este problema acontece porque no escopo isolado da nossa diretiva, usamos `@` para a propriedade `acao`. Esse modificador realiza uma cópia do valor passado para a diretiva, guardando-a no escopo isolado como string. Isso não funciona porque o valor de `acao` é uma expressão que deve ser avaliada no contexto do controller. A diretiva de nada sabe do seu resultado, mas deve ser capaz de executar a **referência** passada como parâmetro. Nem tudo está perdido, pois Angular possui o modificador **&**, que permite fazer `binding` para uma referência. Alterando nossa diretiva:

```
// public/js/directives/minhas-diretivas.js

angular.module('minhasDiretivas', [])
    .directive('meuPainel', function() {
        // código omitido
    })
    .directive('minhaFoto', function() {
        // código omitido
    })
    .directive('meuBotaoPerigo', function() {
        var ddo = {};
        ddo.restrict = "E";
        ddo.scope = {
            nome: '@',
            acao : '&'
        }
        ddo.template = '<button class="btn btn-danger btn-block" ng-click="acao()">{{nome}}</but

        return ddo;
    });
```

Agora sim! Nossa diretiva funciona como esperado!

O que aprendemos neste capítulo?

- criar uma combobox com Angular
- o papel da diretiva ng-options

- transformação com filtros (ex. uppercase)
- a diferença entre @ e & em diretivas