02

Validando nosso form

Transcrição

Quando queremos que o Angular tome conta da validação do formulário para nós, precisamos abdicar do sistema de validação do HTML5. Apesar de extremamente funcional, ele não se integra perfeitamente com o Angular e não é tão flexível quanto este último.

Para desabilitar a validação do HTML5, adicionamos o atributo **novalidate** na tag `form`:

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

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

<!-- código posterior omitido -->
```

Pronto, agora vamos tornar todos os campos do nosso formulário obrigatórios, exceto o campo de `descrição`. Fazemos isso adicionando o mesmo atributo que é usado no HTML5, o atributo **required**:

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

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

<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>
        </div>
        <div class="form-group">
            <label>URL</label>
            <input name="url" class="form-control"
            ng-model="foto.url" required>
        </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>
```

Desligamos a validação do HTML, mas se deixarmos o título em branco e clicarmos em salvar, já teríamos que receber uma mensagem e não recebemos. Diferente do HTML, que já existe uma mensagem por padrão, o Angular precisa que você defina essa mensagem. A vantagem é que temos a flexibilidade de exibir mensagens de validação da forma que desejarmos.

Vamos adicionar, imediatamente após o campo título do nosso formulário, uma tag `span` com as classes `form-control alert-danger`

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

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

<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>

            <!-- novidade -->

            <span class="form-control alert-danger">
                Título obrigatório
            </span>
        </div>
        <div class="form-group">
            <label>URL</label>
            <input name="url" class="form-control"
            ng-model="foto.url" required>
        </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>
```

Ainda não funciona conforme esperado, porque se recarregarmos a página a mensagem de erro será exibida. Para que funcione, sua exibição deve ser condicional. Algo do tipo `se o campo titulo é inválido, exiba a tag span`. O Angular possui a diretiva `ng-show`, que permite a exibição condicional de elementos da tela. Quando seu valor é `true`, o elemento no qual a diretiva está aplicada é exibida, caso contrário não é exibido.

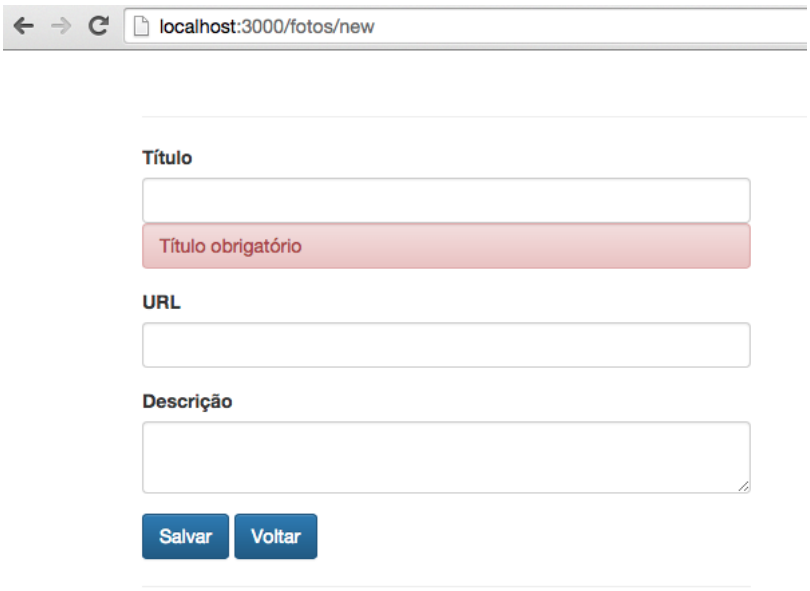
A questão toda é: quem fornecerá o valor da diretiva `ng-show`? A resposta mora em um objeto criado implicitamente que representa nosso formulário. Qual o nome deste objeto? Seu nome é o valor do atributo `name` do formulário, em nosso caso, **formulario**. É através dele que temos acesso a todos os campos do formulário, contanto que cada um deles também tenham definido um valor para o atributo `name`. Sendo assim, podemos fazer para o campo título:

```
// apenas exemplo, não entra em nenhum lugar por enquanto

ng-show = "formulario.titulo.$error.required"
```

Acessamos `formulario.titulo.$error`, que nos dá acesso à interface de erros do Angular. Como queremos saber o status da validação `required`, usamos `formulario.titulo.$error.required`:

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

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

<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>

            <!-- novidade -->

            <span ng-show = "formulario.titulo.$error.required"
                class="form-control alert-danger">
                Título obrigatório
            </span>
        </div>
        <div class="form-group">
            <label>URL</label>
            <input name="url" class="form-control"
            ng-model="foto.url" required>
        </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>
```



Hum, nosso formulário já exibe a mensagem de validação assim que é recarregado. Está certo? Depende do que desejamos. Se queremos exibir mensagens de erro sempre que um campo do formulário estiver errado, está certo. Porém, se quisermos validar os campos apenas quando o formulário for submetido, não. Aliás, vamos alterar o valor da diretiva `ng-show` para exibir a mensagem de erro de validação apenas quando o formulário for submetido. Basta adicionarmos mais uma condição, a `formulario.$submitted` que retorna verdadeiro apenas se o formulário foi submetido.

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

  <span ng-show = "formulario.$submitted && formulario.titulo.$error.required"
                class="form-control alert-danger">
                Título obrigatório
  </span>
<!-- código posterior omitido -->
```

Veja que agora `ng-show` só exibirá o elemento se as duas condições forem verdadeiras: o formulário for submetido e o campo inválido. Agora, ao recarregar a página, nosso formulário só será validado quando salvarmos o formulário, ação que disparará sua submissão.

Vamos deixar agora campo URL obrigatório e também preparar a mensagem de erro:

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

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

<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>
            <span ng-show = "formulario.$submitted && formulario.titulo.$error.required"
                class="form-control alert-danger">
                Título obrigatório
            </span>
        </div>
        <div class="form-group">
            <label>URL</label>
            <input name="url" class="form-control"
            ng-model="foto.url" required>

             <!-- novidade -->

            <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>

        <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, mas o Angular permite fazer mais do que simplesmente considerar a obrigatoriedade de um campo. Existem diretivas específicas para validação. Por exemplo, vamos estipular que o campo título não pode passar de 20 caracteres através da diretiva **ng-maxlength**.

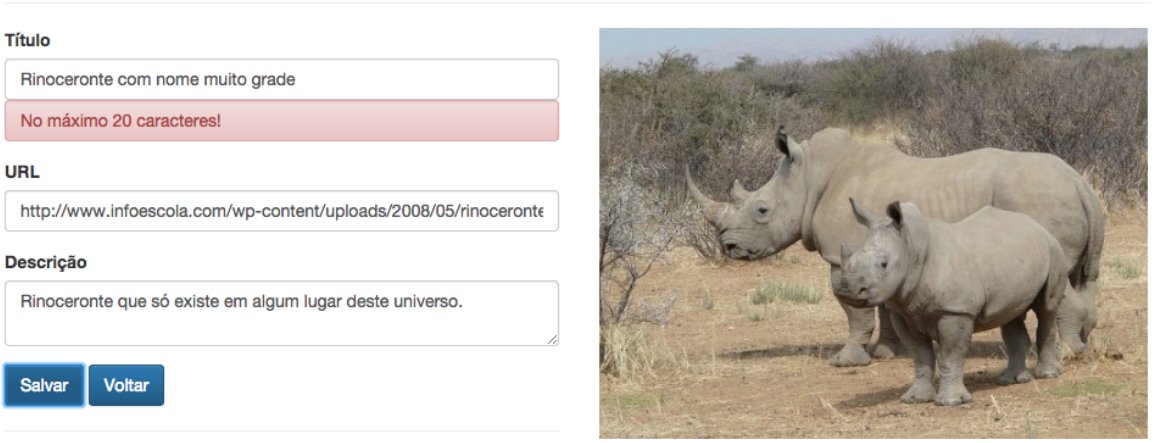
```
<!-- public/partials/foto.html -->

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

<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>

        <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>
```

Vamos aproveitar e fazer o data binding da URL da foto com nossa diretiva `minha-foto` :

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

<div class="col-md-6">
         <minha-foto url="{{foto.url}}" titulo="{{foto.titulo}}">
        </minha-foto>
    </div>
</form>
```

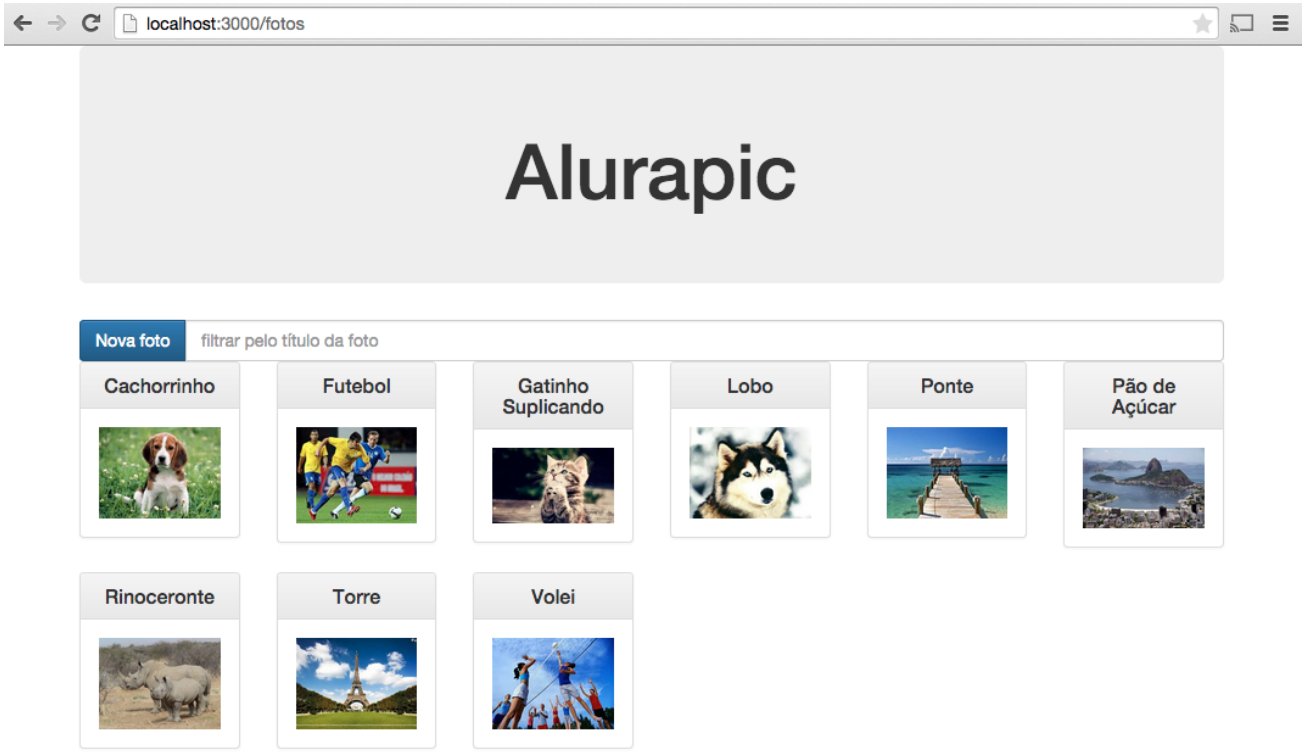
Agora vamos testar nossa validação. Vamos aproveitar e digitar uma URL válida, isso fará com que nossa diretiva `minha-foto` exibe a foto instantaneamente:



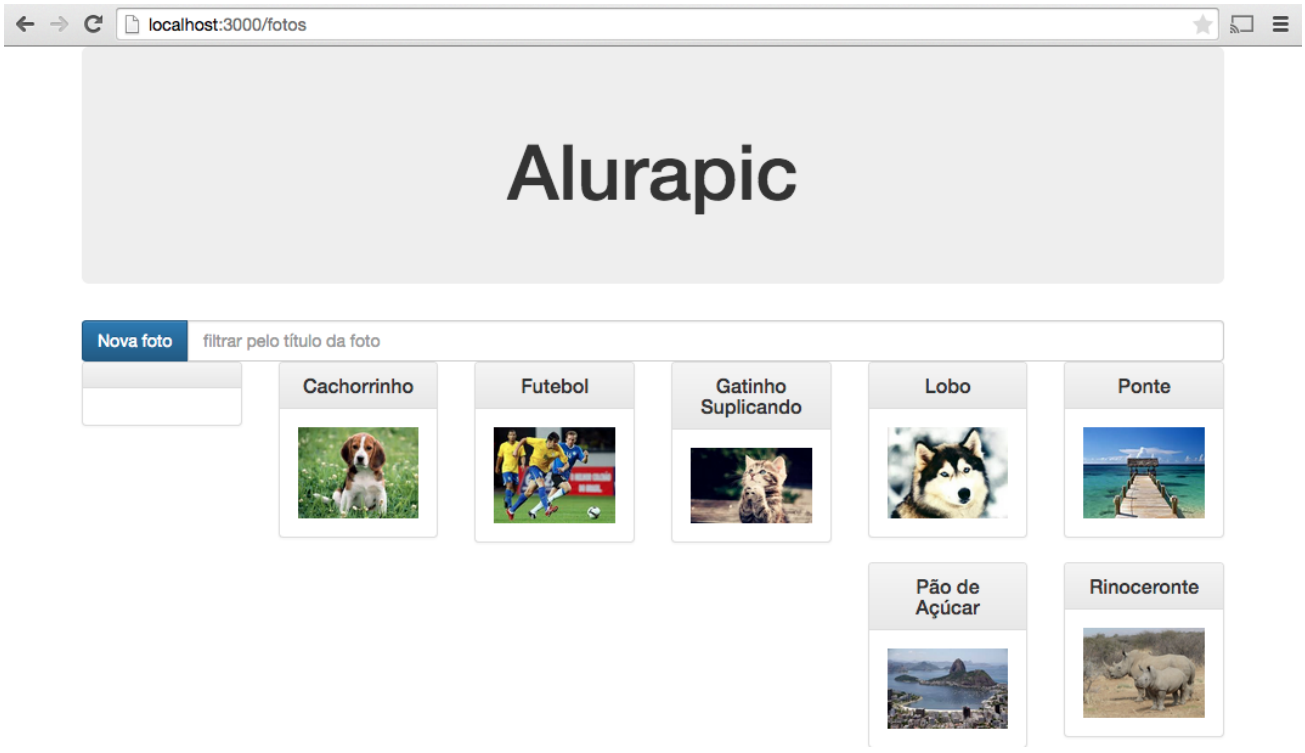
Perfeito! Agora já podemos alterar a função `$scope.submeter` e utilizar o serviço `$http` para gravar nosso produto. Como já dizemos, pedimos `$http` ao serviço de injeção de dependências do Angular. Como queremos enviar os dados, usamos `$http.post`, que recebe dois parâmetros. O primeiro é a URL do nosso server, `/v1/fotos`, e como segundo os dados que serão enviados, no caso, `$scope.fotos`. O restante é igual já fizemos:

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

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

        $scope.foto = {};

        $scope.submeter = function() {

            $http.post('/v1/fotos', $scope.foto)
            .success(function() {
                console.log('Foto adicionada com sucesso');
            })
            .error(function(erro) {
                console.log('Não foi possível cadastrar a foto');
            })
        };

    });
```

Já podemos testar. Quando digitarmos dados válidos e clicamos em salvar, nosso formulário continua preenchido e nem sequer recebemos uma mensagem de sucesso! Bom, atacaremos isso em breve, mas se as informações foram enviadas e salvas, basta clicarmos no botão `voltar` e verificarmos se nossa foto aparece na lista, e realmente aparece!



Muito perfeito! Mas se clicarmos no botão salvar com dados inválidos? Por exemplo, com o tamanho do título sendo maior do que 20 caracteres e deixando o campo URL em branco? O resultado será este:



"Oh my god!", os dados inválidos do nosso formulário foram submetidos e uma foto cadastrada indevidamente. Mas por que o nome ficou em branco? Isso porque dados consideramos inválidos pelo Angular não são aplicados do formulário para o atributo relacionado em `$scope`. Então, no momento do envio dos dados, `$scope.foto.titulo` ficou vazio.

Para resolver isso, basta consultarmos em nosso `$scope` o status do formulário com a sintaxe **$scope.formulario.$valid**. Isso mesmo, através de `$scope` podemos acessar nosso formulário através do seu `name` e perguntarmos se ele é válido consultando a propriedade `$valid`.

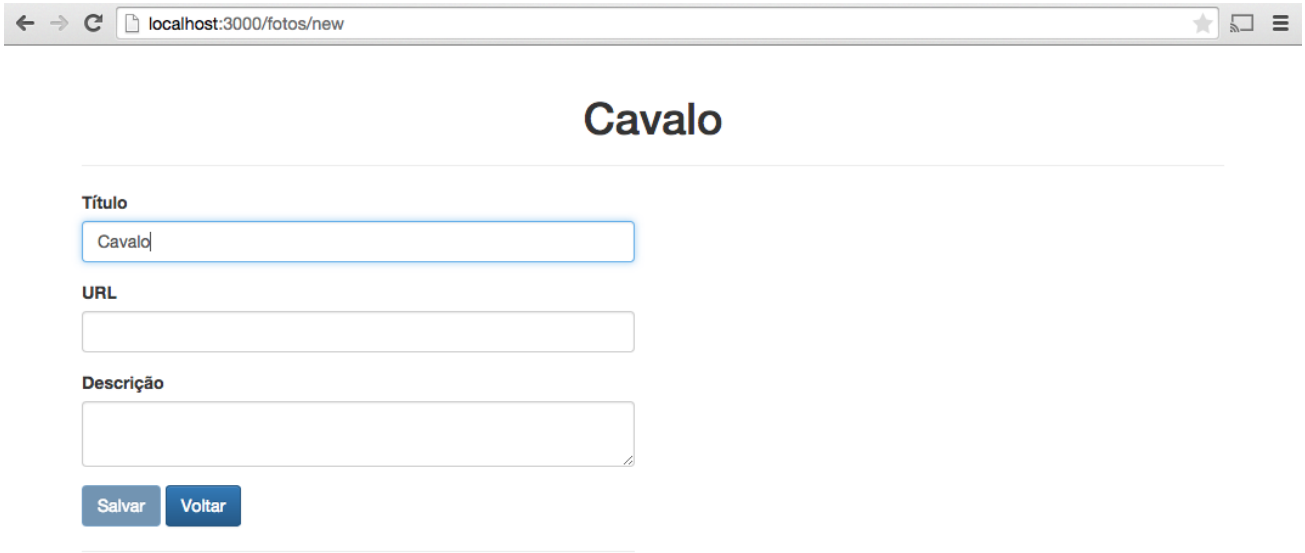
Alterando nosso `FotoController`:

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

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

        $scope.foto = {};

        $scope.submeter = function() {

            if ($scope.formulario.$valid) {

                $http.post('/v1/fotos', $scope.foto)
                .success(function() {
                    console.log('Foto adicionada com sucesso');
                })
                .error(function(erro) {
                    console.log('Não foi possível cadastra a foto');
                })
            }
        };

    });
```

Agora, nossa lógica de envio das informações só será executada caso o formulário seja válido. Antes de testarmos, podemos melhorar ainda mais a experiência do usuário habilitando a exibição do botão `salvar` apenas se o formulário estiver válido.

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

<!-- código anterior omitido -->
 <button type="submit" class="btn btn-primary" ng-disabled="formulario.$invalid">
            Salvar
</button>
<!-- código posterior omitido -->
```

A cada interação do usuário com nosso formulário, a diretiva `ng-disabled` consultará o status do formulário para saber se ele é invalido, caso seja, o botão ficará desabilitado:



Agora, para deixar ainda melhor nosso formulário, vamos exibir uma mensagem de fracasso ou sucesso para indicar o status da operação com o servidor. Inclusive vamos limpar os dados do formulário quando a operação for bem sucedida. Vamos adicionar um parágrafo que consultará `$scope.mensagem` . O parágrafo só será exibido se exibir alguma mensagem:

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

<!-- novidade! Aqui será exibida mensagens para o usuário -->

<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>

        <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, alterando nosso controller para popular a `$scope.mensagem` :

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

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

        $scope.foto = {};
        $scope.mensagem = '';

        $scope.submeter = function() {

            if ($scope.formulario.$valid) {

                $http.post('/v1/fotos', $scope.foto)
                .success(function() {
                    $scope.mensagem = 'Foto cadastrada com sucesso';
                })
                .error(function(erro) {
                    console.log(erro);
                    $scope.mensagem = 'Não foi possível cadastrar a foto';
                })
            }
        };

    });
```

Muito bem, agora basta cadastrarmos uma nova foto e verificar a exibição da nossa mensagem:

Quase lá! Precisamos limpar o formulário quando a mensagem for adicionada com sucesso. Basta atribuirmos um objeto vazio à `$scope.foto` :

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

        $scope.foto = {};
        $scope.mensagem = '';

        $scope.submeter = function() {

            if ($scope.formulario.$valid) {

                $http.post('/v1/fotos', $scope.foto)
                .success(function() {
                    $scope.foto = {};
                    $scope.mensagem = 'Foto cadastrada com sucesso';
                })
                .error(function(erro) {
                    console.log(erro);
                    $scope.mensagem = 'Não foi possível cadastrar a foto';
                })
            }
        };

    });
```

Agora, para cada foto cadastrada com sucesso, o formulário será limpo automaticamente, tudo por causa do data binding do Angular.

O que aprendemos neste capítulo?

- navegação entre views
- a diretiva ng-submit e a interface de eventos do Angular
- executar uma ação no controller através de ações do usuário
- validação de formulário com Angular
- envio de dados para o servidor com $http.post
- mensagens de aviso para o usuário