Formularios

Salvando um contato

Queremos criar input do nosso formulário alimente as propriedades de um objeto contato sem qualquer informação. Para isso usamos a diretiva diretiva ng-modell para cada um dos inputs.

Preparando o terreno para o cadastro

O que precisamos agora é implementar o botão salvar. O que ele deve fazer? Submeter o formulário, para podermos acessar $scope.contato 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.

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:

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

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:

Se testarmos nosso código, nada acontecerá, por quê? O motivo é simples: não associamos ContatoController à view parcial formulario.html. Entao adicionamos uma nova rota no arquivos rotas e adicionamos seu controlador

```
$routeProvider.when('/contatos/new',{
      templateUrl: 'partials/formulario.html',
      controller: 'contatoController'
    });
```

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.

Validando nosso formulário

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:

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

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

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 os campos do nosso formulário, uma tag span com as classes form-control alert-danger

```
<span class="form-control alert-danger"> Título obrigatório </span>
```

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.

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 nome:

```
ng-show = "formulario.$submitted && formulario.nome.$error.required"
```

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

Salvando Dados

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, e como segundo os dados que serão enviados, no caso, $scope.contato. O restante é igual já fizemos:

Metodo Submeter

```
$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');
            })
        };
```

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.

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

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:

```
$scope.submeter = function() {
            if ($scope.formulario.$valid) {
                $http.post('/api/contatos', $scope.contato).success(function() {
                $scope.foto = {};
                                    $scope.mensagem = 'Contato cadastrado';
                                })
                                .error(function(erro) {
                                    console.log(erro);
                                    $scope.mensagem = 'Não foi possível cadastrar';
                })}};
```

Update Contato

A estratégia de edição será a seguinte: quando o usuário clicar no botão editar, iremos para a parcial formulario.html, porém enviaremos o ID do contato como parâmetro. Em ContatoController, capturaremos este ID. Quando o ID for passado, buscaremos a foto através deste ID atualizando o $scope.foto com seus dados.

O primeiro passo é registrarmos uma nova rota no Angular que saiba lidar com o ID da foto:

```
$routeProvider.when('/contatos/:contatoID',{
      templateUrl: 'partials/formulario.html',
      controller: 'contatoController'
    });
```

Excelente, mas ainda precisamos ter acesso ao ID do contato em ContatoController para que possamos buscalo em nosso servidor. Existe um serviço especializado do Angular que nos fornecerá este parâmetro, o $routeParams. Como qualquer serviço no Angular, ele é recebido como parâmetro na função que define nosso controller:

```
angular.module('contatos').controller('contatoController',function
($scope, $http,$routeParams) {
```

Como utilizar o serviço? Basta sabermos o nome do curinga que usamos em nossa rota e utilizá-lo como propriedade de $routeParams:

```
if($routeParams.fotoId) {
            $http.get('/api/contatos' + $routeParams.contatoId)
            .success(function(contato) {
                $scope.contato = contato;
            })
            .error(function(erro) {
                console.log(erro);
                $scope.mensagem = 'Não foi possível obter'
            });
        }
```

```
$scope.submeter = function() {
            if ($scope.formulario.$valid) {
                if($routeParams.fotoId) {
                    $http.put('/api/contatos/' + $scope.contato.id, $scope.contato)
                    .success(function() {
                        $scope.mensagem = 'Contato alterada com sucesso';
                    }).error(function(erro) {
                        console.log(erro);
                        $scope.mensagem = 'Não foi possível alterar';
                    });
                } else {
                    $http.post('/api/contatos', $scope.contato)
                    .success(function() {
                        $scope.contato = {};
                        $scope.mensagem = 'Foto cadastrada com sucesso';
                    })
                    .error(function(erro) {
                        console.log(erro);
                        $scope.mensagem = 'Não foi possível cadastrar a foto';
                    })
                }
            }};
```