UNINOVE

Diretivas, Serviços, Filtros e Rotas

APRESENTAR OS CONCEITOS SOBRE OS RECURSOS DE DIRETIVAS, SERVIÇOS, FILTROS E ROTAS DO ANGULAR JS, POR MEIO DE EXEMPLOS E EXPLICAÇÃO DOS CONCEITOS RELATIVOS A CADA RECURSO.

AUTOR(A): PROF. EDSON MELO DE SOUZA

1. Introdução

Neste tópico serão abordados mais recursos do *framework* Angular JS que possibilitarão a criação de aplicações simples para Web, explicando o conceito de cada um deles, além de mostrar exemplos de aplicação.

2. Diretivas (*Directives*)

O Angular trabalha com o conceito de diretivas, que são instruções ou orientações que o interpretador JavaScript deve executar quando as localiza dentro de um arquivo HTML, através da especificação de *tags* personalizadas que podem ser reutilizadas. As diretivas do Angular são iniciadas com as letras "ng", seguidas da sua funcionalidade.

A utilização de diretivas elimina a necessidade de escrever muito JavaScript, pois o *framework* se encarrega disso, permitindo incorporar a lógica da aplicação diretamente no HTML. A seguir estão listadas as principais diretivas do Angular (W3SCHOOLS, 2017):

- ng-app - define que se trata de uma aplicação Angular, configurando um elemento HTML como *root* ou raiz da aplicação, ou seja, onde a aplicação será iniciada dentro do arquivo, alterando o comportamento da *tag* que foi utilizada;
- ng-bind - vincula os dados de uma aplicação como uma visualização do HTML, alterando automaticamente o texto de um HTML com os dados originados de um resultado obtido.
- ng-model - é a diretiva que vincula um valor de um controle do HTML como (campos de entrada, caixas de seleção, entre outros) ao dados da aplicação, sendo similar ao ng-bind, entretanto, permite que as mensagens sejam bidirecional (*two-way*);
- ng-class – essa diretiva permite que seja definida dinamicamente classes CSS em um elemento HTML por ligação de dados a uma expressão que representa todas as classes a serem adicionadas.
- ng-click – permite instanciar o evento de click (*onclick*);

- ng-controller – atribui uma classe de controlador (*controller*) para uma visualização (*view*). Este é um aspecto fundamental de como angular suporta os princípios por trás do padrão de projeto *Model-View-Controller*.
- ng-repeat – instancia um item de uma coleção onde cada instância, onde a variável de loop (repetição) informada é definida para o item de coleção atual e $ index é definido como o índice do item ou da chave, similar ao laço de repetição (*for*);
- ng-show e ng-hide – mostra ou exibe uma *tag* HTML com base no resultado de uma expressão booleana (*true* ou *false*);
- ng-switch – usada para trocar condicionalmente a estrutura DOM em seu modelo com base em uma expressão de escopo.;
- ng-view – diretiva que complementa o serviço *$route* incluindo o modelo renderizado da rota atual no arquivo de layout principal (index.html). Toda vez que a rota atual mudar, a visualização incluída muda com ela de acordo com a configuração do *$route service.template*;
- ng-if – permite criar um bloco lógico (*if*), removendo ou recriando uma parte da árvore DOM, com base em uma expressão lógica.

A seguir vamos ver exemplos de diretivas que atuam com eventos, ou seja, uma diretiva que responde à alguma ação. Sabendo agora um pouco mais sobre as diretivas, podemos avançar e iniciar a construção de exemplos.

2.1 ng-click

No código a seguir vamos ver um simples exemplo para podermos compreender a sintaxe e como as coisas funcionam com o Angular. Neste exemplo veremos como utilizar o ng-click que está relacionado ao uso do *mouse*.

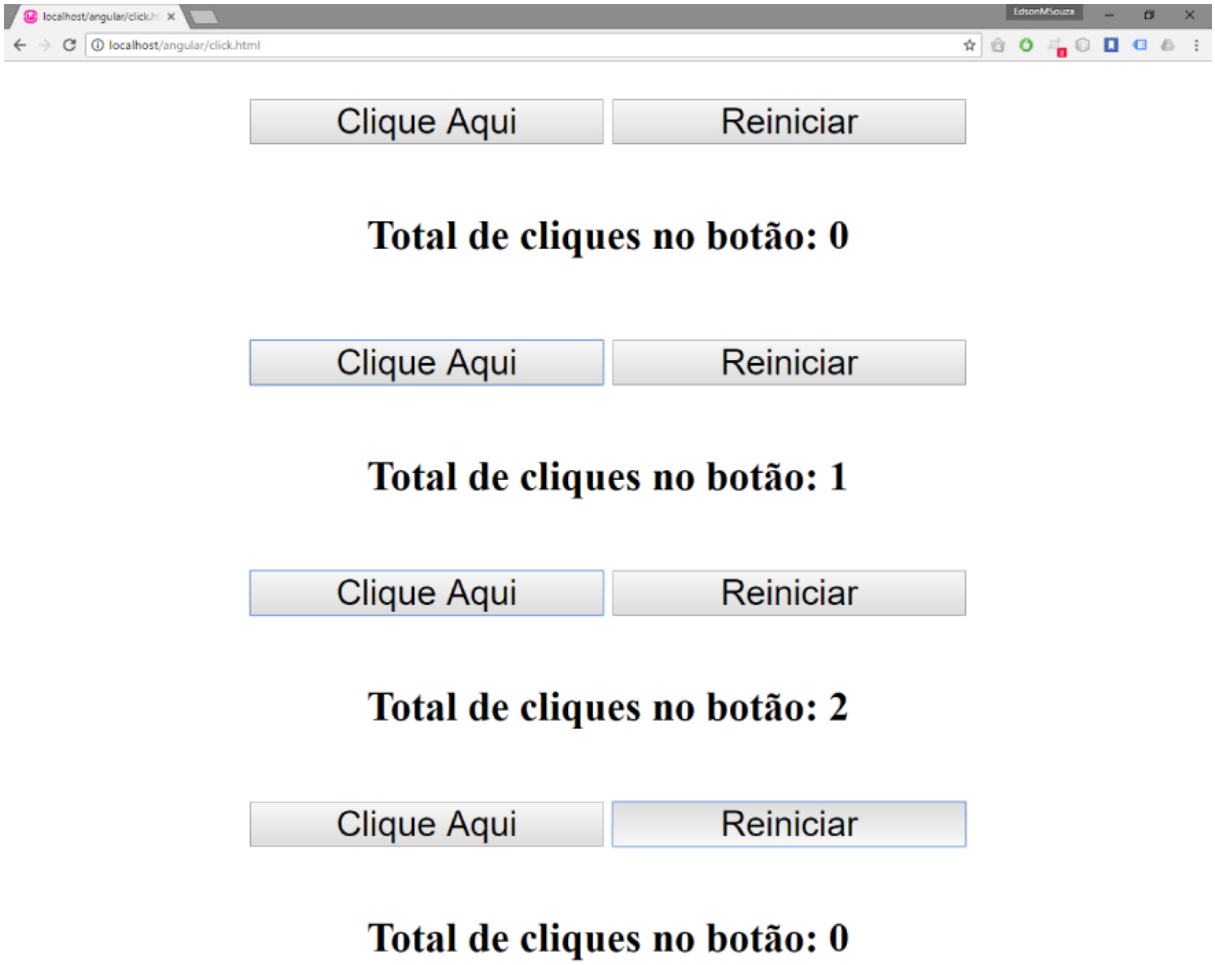
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title></title>
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4

        <style type="text/css" media="screen">
            body{
                margin-top: 100px;
                font-size: 40px;
                line-height: 2em;
                text-align: center;
            }

            button, input{
                font-size: 40px;
                width: 400px;
            }

            label{
                width: 320px;
                float: left;
            }
        </style>

        <script>
            var app = angular.module('aplicacao', []);
            app.controller('controle', function ($scope) {
                $scope.contador = 0;
                $scope.contagem = function () {
                    $scope.contador++;
                }
            });
        </script>
    </head>
    <body>
        <div ng-app="aplicacao" ng-controller="controle">
```

```
40.             <button ng-click="contagem()">Clique Aqui</button>
41.             <button ng-click="contador = 0">Reiniciar</button>
42.             <h3>Total de cliques no botão: {{ contador}}</h3>
43.         </div>
44.     </body>
45. </html>
```

Nas linhas de 29 a 35 é criado o *script* com a configuração da aplicação. Na linha 30 é criado o *controller*. Na linha 31 é criada uma variável de escopo com o nome de contador. Na linha 32 é criada uma variável chamada contagem que recebe uma função e, na linha 33, a variável contador é incrementada em 1 (++).

Na linha 39 é criada uma "div" que inicializa a aplicação e declara o controle. Na linha 40 um botão é configurado com a diretiva "ng-click" e, cada vez que for clicado nele, a variável contador (dentro do controle) será incrementada em 1 (++). Na linha 41 é configurado um botão para zerar a variável contador. E, por fim, na linha 42, o valor incrementado no clique do botão é mostrado, assim como quando a variável é zerada.

A imagem a seguir mostra o contador em estado inicial e depois uma sequência de dois cliques (incremento) e, por fim, a reinicialização do contador.



2.2 ng-show

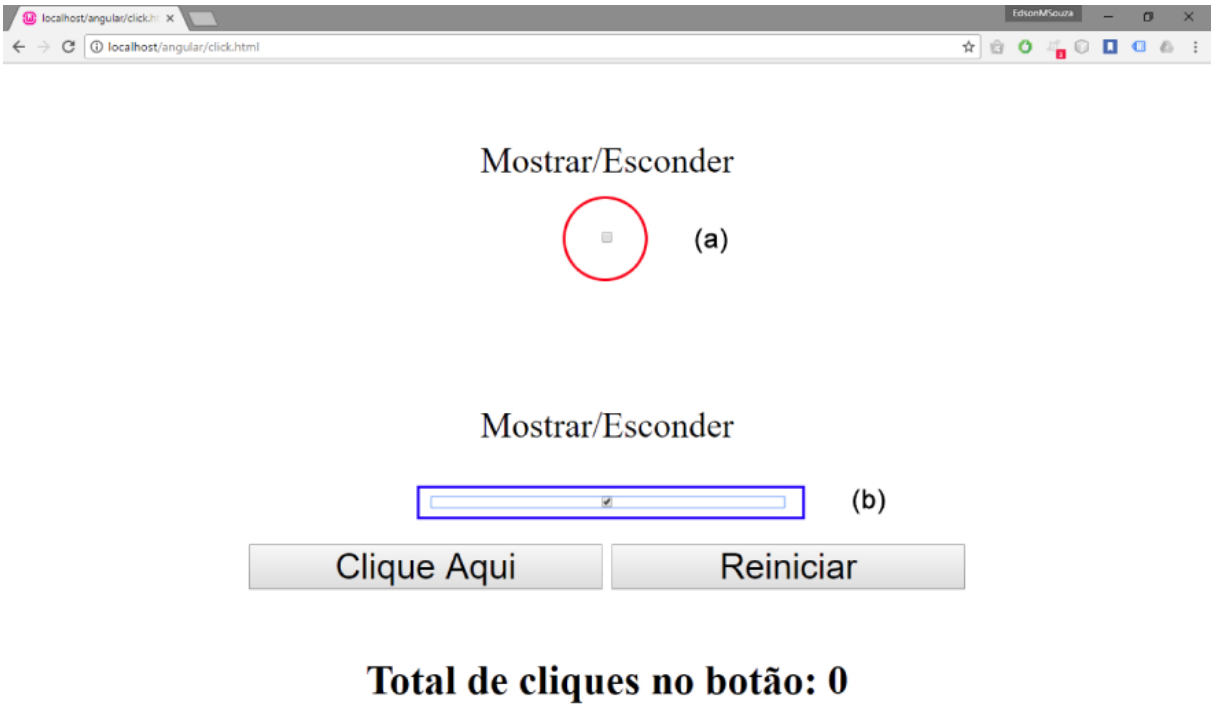
No exemplo a seguir é mostrada a utilização da diretiva ng-show juntamente com o ng-click. Neste exemplo a página é carregada e a "div" "bloco" está oculta. Quando clicado na "check-box", então a diretiva é acionada e mostra o conteúdo oculto, que o contador do exemplo anterior. Se for novamente clicado na check-box, o conteúdo será ocultado. Vamos ao código deste exemplo.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title></title>
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4

        <style type="text/css" media="screen">
            body{
                margin-top: 100px;
                font-size: 40px;
                line-height: 2em;
                text-align: center;
            }

            button, input{
                font-size: 40px;
                width: 400px;
            }

            label{
                width: 320px;
                float: left;
            }
        </style>

        <script>
            var app = angular.module('aplicacao', []);
            app.controller('controle', function ($scope) {
                $scope.contador = 0;
                $scope.contagem = function () {
                    $scope.contador++;
                }
            });
        </script>
    </head>
    <body ng-app="aplicacao">
        Mostrar/Esconder<br>
```

```
40.         <input type="checkbox" ng-model="bloco" />
41.
42.         <div  ng-controller="controle">
43.             <div ng-show="bloco">
44.                 <button ng-click="contagem()">Clique Aqui</button>
45.                 <button ng-click="contador = 0">Reiniciar</button>
46.                 <h3>Total de cliques no botão: {{ contador}}</h3>
47.             </div>
48.         </div>
49.
50.     </body>
51. </html>
```

Nas linhas de 29 a 35 a aplicação é configurada, como no exemplo anterior. Na linha 38 é inicializada a aplicação. Na linha 40 é criada um “check-box” com o modelo “ng-model” denominado de “bloco”. Na linha 42 é inserido o controle e na linha 43 o “ng-show” denominado “bloco”. Nas linhas 44 e 45 estão as *tags* para o contador.

A imagem a seguir mostra o resultado gerado pelo código anterior, apresentando as duas situações possíveis para este exemplo, ou sej, oculto ou visível. Em (a) a "div" está oculta. Já em (b), após clicar na "check-box", a "div" mostra o conteúdo do seu interior.



2.3 ng-repeat

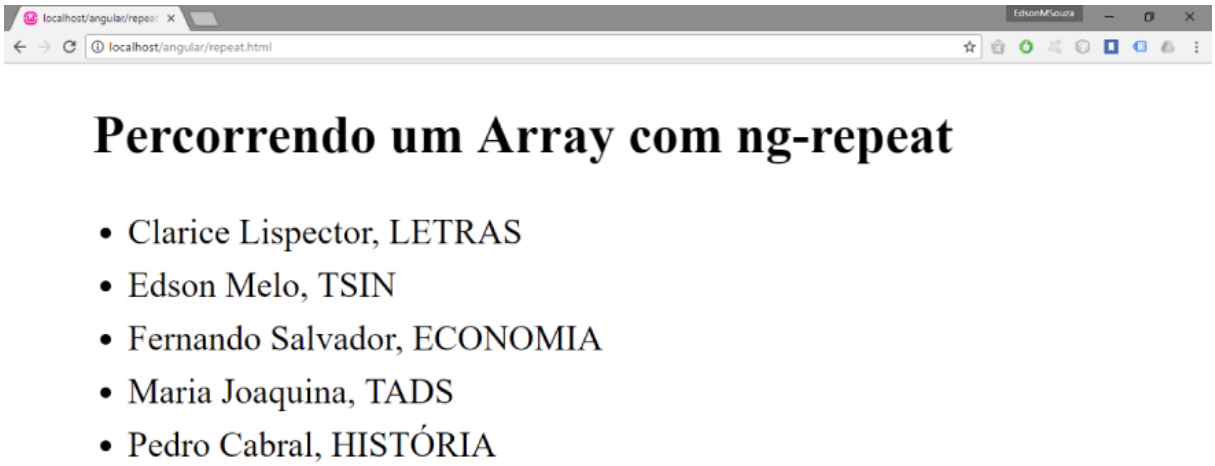
A diretiva "ng-repeat" atua como um laço de repetição, processando variáveis internas, até que um critério seja alcançado. No exemplo a seguir vamos ver como percorrer um "array" utilizando essa diretiva.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title></title>
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4

        <style type="text/css" media="screen">
            body{
                margin-top: 30px;
                margin-left: 100px;
                font-size: 40px;
                line-height: 1.5em;
            }

            button, input{
                font-size: 40px;
                width: 400px;
            }

            label{
                width: 320px;
                float: left;
            }
        </style>

        <script>
            angular.module('aplicacao', []).controller('controle', funct
                $scope.alunos = [
                    {nome: 'Edson Melo', curso: 'TSIN'},
                    {nome: 'Maria Joaquina', curso: 'TADS'},
                    {nome: 'Pedro Cabral', curso: 'HISTÓRIA'},
                    {nome: 'Clarice Lispector', curso: 'LETRAS'},
                    {nome: 'Fernando Salvador', curso: 'ECONOMIA'}
                ];
            });
        </script>
    </head>
```

```
40.     <body ng-app="aplicacao">
41.         <h2>Percorrendo um Array com <strong>ng-repeat</strong></h2>
42.         <div  ng-controller="controle">
43.             <ul>
44.                 <li ng-repeat="obj in alunos | orderBy:'nome'">
45.                     {{ obj.nome + ', ' + obj.curso}}
46.                 </li>
47.             </ul>
48.         </div>
49.
50.     </body>
51. </html>
```

Nas linhas de 29 a 37 é configurado o módulo da aplicação. Na linha 29 é declarada a aplicação e o controle. Na linha 30 é criada uma variável de escopo que recebe um "array" bidimensional de nomes e cursos. Na linha 43 é declarado o controle para um "div". Na linha 44 é utilizada então a diretiva "ng-repeat" que declara uma variável local "obj" para receber os dados do "array". Essa instrução é dada pela palavra "*in*", seguida do nome do "array" que está no controle. A barra "|" significa que será incluída uma expressão, que, neste caso, tem a função de ordenar os valores pelo "nome". Por fim, na linha 45 são concatenados os valores recebidos por "obj" e então exibidos no navegador.

A imagem a seguir mostra o resultado após o processamento do código anterior.

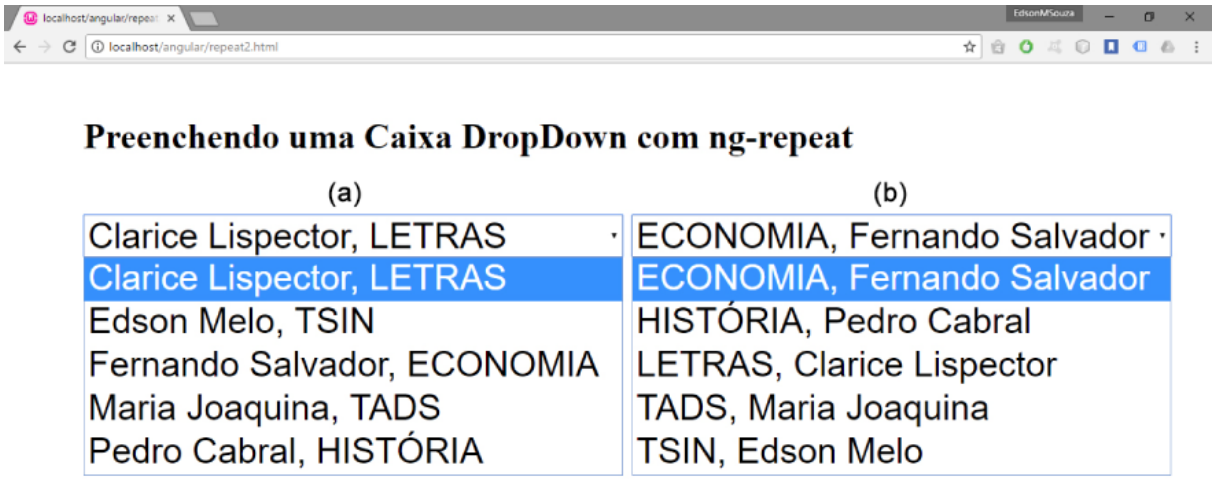


No próximo exemplo é mostrado um código adapatado do anterior que mostra duas caixas DropDown contendo nomes e cursos. No item (a) os valores estão ordenados por nome de forma crescente. Já no item (b), estão ordenados pelo nome do curso.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title></title>
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4

        <style type="text/css" media="screen">
            body{
                margin-top: 30px;
                margin-left: 90px;
                font-size: 40px;
                line-height: 1.5em;
            }

            button, input, select{
                font-size: 40px;
                width: 610px;
            }

            label{
                width: 320px;
                float: left;
            }
        </style>

        <script>
            angular.module('aplicacao', []).controller('controle', funct
                $scope.alunos = [
                    {nome: 'Edson Melo', curso: 'TSIN'},
                    {nome: 'Maria Joaquina', curso: 'TADS'},
                    {nome: 'Pedro Cabral', curso: 'HISTÓRIA'},
                    {nome: 'Clarice Lispector', curso: 'LETRAS'},
                    {nome: 'Fernando Salvador', curso: 'ECONOMIA'}
                ];
            });
        </script>

```

```
40.     </head>
41.     <body ng-app="aplicacao">
42.         <h4>Preenchendo uma Caixa DropDown com <strong>ng-repeat</strong
43.         <div  ng-controller="controle">
44.             <select name="alunos">
45.                 <option ng-repeat="obj in alunos| orderBy:'nome'">
46.                     {{ obj.nome + ', ' + obj.curso}}
47.                 </option>
48.             </select>
49.
50.             <select name="alunos">
51.                 <option ng-repeat="obj in alunos| orderBy:'curso'">
52.                     {{ obj.curso + ', ' + obj.nome}}
53.                 </option>
54.             </select>
55.         </div>
56.
57.     </body>
58. </html>
```

Perceba que só há alteração nas linhas que foram incluídas para criação da DropDown e nas linhas 45 e 51 que fazem a ordenação dos valores " | order by". A imagem a seguir mostra as duas caixas DropDown abertas e mostrando os valores com ordenação diferenciada.



3. Serviços (*Services*)

Um serviço ou *service* é um método no nosso módulo principal (app), que recebe um nome e uma função que o definem (ASSEMANY, 2016). Os *services* também trabalham como um objeto, compartilhando as regras de negócio da aplicação e também compartilhando estados dos objetos.

O *service* é o objeto usado para organizar e/ou compartilhar estados de objetos e as regras de negócio da aplicação, disponibilizando apenas uma instância durante a vida útil da aplicação (FEITOSA, 2015).

Os *services* trabalham com o conceito de dependência, ou seja, é necessário informar quais recursos dependerão do serviço como controles, diretivas ou filtros e já traz em torno de 30 serviços como o de localização "*$location*", que fornece informações sobre a localização (URL) da página que está sendo acessada.

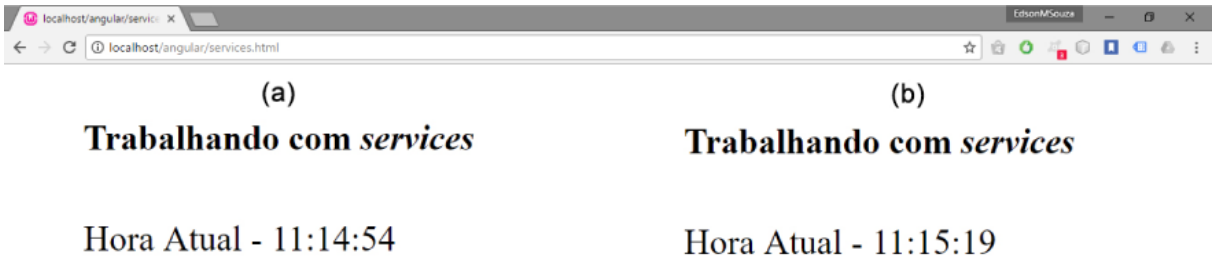
No exemplo a seguir veremos o uso de um *service* para atualizar um valor na *view*, no caso um relógio, que será atualizado a cada segundo.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title></title>
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6

        <style type="text/css" media="screen">
            body{
                margin-top: 30px;
                margin-left: 90px;
                font-size: 40px;
                line-height: 1.5em;
            }

            button, input, select{
                font-size: 40px;
                width: 610px;
            }

            label{
                width: 320px;
                float: left;
            }
        </style>

        <script>
            var app = angular.module('myApp', []);
            app.controller('myCtrl1', function ($scope, $interval) {
                $scope.theTime = new Date().toLocaleTimeString();
                $interval(function () {
                    $scope.theTime = new Date().toLocaleTimeString();
                }, 1000);
            });
        </script>

    </head>
    <body ng-app="myApp">
```

```
40.
41.         <h4>Trabalhando com <em>services</em></h4>
42.         <div ng-controller="myCtrl1">
43.             <p>Hora Atual - {{theTime}}</p>
44.         </div>
45.
46.     </body>
47. </html>
```

Nas linhas de 29 a 35 é criado o *script*. Na linha 30 é criado o escopo do controle, usando como parâmetro um "$interval", que é um serviço (*service*). Na linha31 é criada uma variável "hora" para receber um valor "horário". Na linha 32 é ativado o serviço e, na linha 33 o valor é atribuído novamente de acordo com o intervalo de tempo especificado na linha 34 "1000=1 segundo". Por fim, na linha 43 a variável "hora" é mostrada para o cliente.

O resultado pode sr visualizado na imagem a seguir, onde existem duas imagens (a) e (b), mostrando momentos diferentes de tempo.



4. Filtros (*Filters*)

Os filtros são recursos que fornecem elementos para a formatação de dados. Em aplicações computacionais, normalmente é necessário realizar formatações que serão apresentadas para o cliente (usuário), podendo elas serem do tipo moeda, datas, cep, letras maiúsculas ou minúsculas, entre outros.O Angular já fornece alguns filtros padrão, entretanto, é possível criar seus próprios filtros, de acordo com a necessidade de cada ocasião.

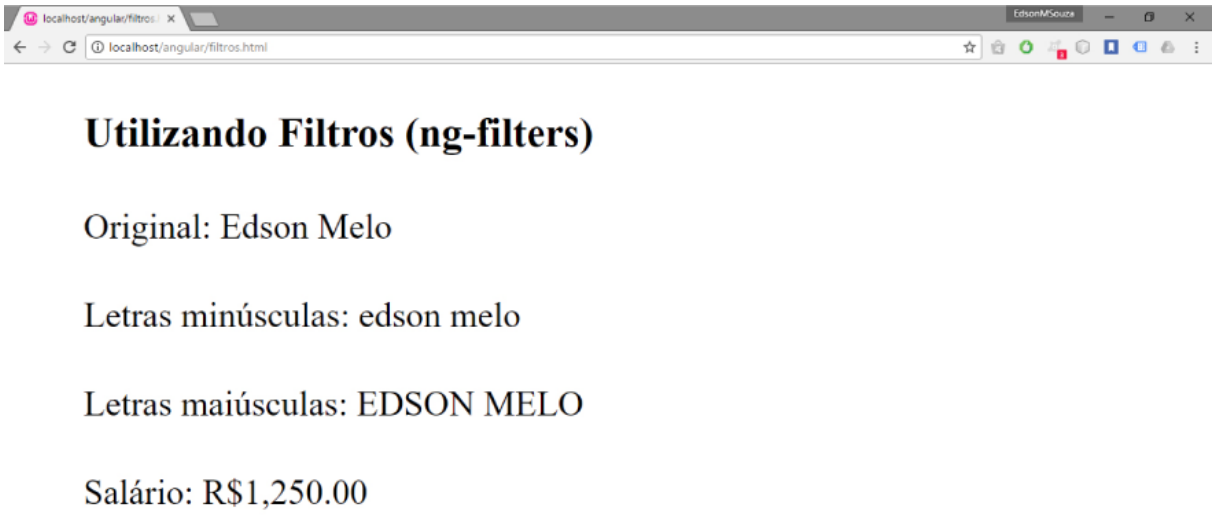
No exemplo a seguir são mostradas algumas opções de formatação para o cliente utilizando os filtros.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title></title>

        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6

        <style type="text/css" media="screen">
            body{
                margin-top: 30px;
                margin-left: 90px;
                font-size: 40px;
                line-height: 1.5em;
            }

            button, input, select{
                font-size: 40px;
                width: 610px;
            }

            label{
                width: 320px;
                float: left;
            }
        </style>

        <script>
            angular.module('aplicacao', []).controller('controle', funct
                $scope.nome = "Edson Melo";
                $scope.salario = 1250
            });
        </script>

    </head>
    <body ng-app="aplicacao">

        <div ng-controller="controle">
```

```
40.             <h3>Utilizando Filtros (ng-filters)</h3>
41.             <p>Original: {{ nome }}</p>
42.             <p>Letras minúsculas: {{ nome | lowercase }}</p>
43.             <p>Letras maiúsculas: {{ nome | uppercase }}</p>
44.             <p>Salário: {{ salario | currency:'R$' }}</p>
45.         </div>
46.
47.     </body>
48. </html>
```

Neste exemplo o recurso de filtro é aplicado diretamente na *view*. Na linha 41 é mostrado o valor original contido no *controller* (linhas de 39 a 45). Na linha 42 é mostrado o valor armazenado em "nome", mas agora em caixa baixa, utilizando a expressão barra "| lowercase". Na linha 43 o valor é convertido para caixa alta "| uppercase". Por fim, na linha 44, um valor numérico (moeda) é convertido e tem o símbolo monetário do real definido.

A imagem a seguir mostra o resultado das conversões realizadas no código anterior.



SAIBA MAIS!

Angular Filters Components - https://docs.angularjs.org/api/ng/filter/ (https://docs.angularjs.org/api/ng/filter/)

Gabriel Feitosa - http://gabrielfeitosa.com/angularjs-filtros/ (http://gabrielfeitosa.com/angularjs-filtros/)

5. Rotas (*Routes*)

As *routes* ou rotas é um recurso que permite que seja feita uma injeção de uma página (*template* HTML) para a visualização em uma (*view*) (ANGULARJS, 2017).

Uma aplicação poderia ser desenvolvida utilizando apenas uma página, ou seja, teríamos um programa dentro de apenas um HTML. Mas, esse tipo de estratégia não é das melhores do mundo, pois, o tamanho do arquivo pode ficar extremamente grande, provocando demora no carregamento e aumento de consumo de recursos no servidor. Portanto, o recurso de rotas do Angular, o ng-route, nos ajuda a administrar melhor essa problemática.

É importante ressaltar que, como as rotas trabalham com requisições HTTP, é necessário que o exemplo seja executado em um Servidor Web como o Apache, que você pode obter nesse link (http://www.wampserver.com/en/ (http://www.wampserver.com/en/)). Nos links em destaque há um tutorial sobre o WAMP.

No exemplo a seguir vamos criar uma página que permitirá a inclusão de outros arquivos HTML (injeção) na nossa aplicação. Ou seja, será possível trocar os valores da páginas sem a necessidade sair da página atual. A aplicação a seguir possui os seguintes arquivos:

- index.htm (Página principal)
- contato.htm
- links.htm
- sobre.htm

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title></title>

        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6

        <style type="text/css" media="screen">
            body{
                margin-top: 30px;
                margin-left: 90px;
                font-size: 30px;
                line-height: 1.5em;
            }

            button, input, select, textArea{
                font-size: 30px;
                width: 610px;
            }

            label{
                width: 320px;
                float: left;
            }
        </style>

        <script>
            var app = angular.module("aplicacao", ["ngRoute"]);
            app.config(function ($routeProvider) {
                $routeProvider
                        .when("/sobre", {
                            templateUrl: "sobre.htm"
                        })
                        .when("/links", {
                            templateUrl: "links.htm"
                        })
```

```
40.                         .when("/contato", {
41.                             templateUrl: "contato.htm"
42.                         });
43.             });
44.         </script>
45.     </head>
46.     <body ng-app="aplicacao">
47.
48.         <p>Clique nos links para navegar!</p>
49.         <a href="#!">Inicial</a> |
50.         <a href="#!sobre">Sobre</a> |
51.         <a href="#!links">Links</a> |
52.         <a href="#!contato">Contato</a>
53.
54.         <div ng-view></div>
55.
56.     </body>
57. </html>
```

Essa aplicação exige que seja incluído outro arquivo do framework (angular-route.js) que está sendo feito na linha 9. Nas linhas de 31 a 43 é criado o módulo da aplicação. Na linha 31 é criada a variável para a aplicação e que recebe como parâmetro o valor “ngRoute”. Na linha 32 é configurada a aplicação com o “app.config”, informando que haverá uma rota “$routeProvider”. Na linha 33 é declara a rota. Nas linhas 34, 37 e 40 são declarados os valores para serem utilizados com parâmetros dos links “rotas”. Nas linhas 35, 38 e 41 são declaradas as páginas relativas as rotas.Nas linhas de 49 a 52 são informados links para as rotas e, por fim, a parte mais importante, na linha 54 é criada uma “div ng-view” que receberá a inclusão das páginas, quando selecionadas.

Nas imagens a seguir são visualizados os acesso e o funcionamento da rota quando um clique é realizado em um link.

A seguir, estão os códigos dos arquivos "htm" necessários. Esses arquivos não possuem as tags HTML normais como "html", "head" ou "body". Eles devem ser criados da forma como estão escritos a seguir:

```
1. <!-- Arquivo: contato.htm -->
2.
3.         <h3>Formulário de Contato</h3>
4.         <form name="contato" method="POST">
5.             <label>Nome: </label><input type="text" name="nome" value=""
6.             <label>Email: </label><input type="text" name="nome" value="
7.             <label>Mensagem: </label> <textarea name="mensagem" rows="4"
8.             <label> </label> <input type="submit" value="Enviar" />
9.         </form>
10.
```

```
1. <!-- Arquivo: links.htm -->
2.
3.         <h2>Links Úteis</h2>
4.         <p><a href="http://www.uninove.br/">Uninove</a></p>
5.         <p><a href="http://www.periodicos.capes.gov.br/">Portal CAPES</a
6.         <p><a href="http://www2.planalto.gov.br/">Planalto DF</a></p>
```

```
1. <!-- Arquivo: sobre.htm -->
2.
3.         <h2>Sobre</h2>
4.         <p>Essa página fala um pouco sobre a empresa ou sua pessoa!</p>
```

Clique nos links para navegar!

Inicial | Sobre | Links | Contato

Clique nos links para navegar!

Inicial | Sobre | Links | Contato

Sobre

Essa página fala um pouco sobre a empresa ou sua pessoa!

Clique nos links para navegar!

Inicial | Sobre | Links | Contato

Links Úteis

Uninove

Portal CAPES

Planalto DF

Você pode perceber nas imagens que a parte superior da tela nunca sobre alteração, pois as demais páginas são inseridas (injetadas) na página principal. Esse recurso é muito utilizado quando estamos trabalhando com o conceito de "*single-page*", ou seja, página única.

Angular 2 e 4 - http://loiane.training/curso/angular-2/ (http://loiane.training/curso/angular-2/)
Angular Routes - https://docs.angularjs.org/api/ngRoute (https://docs.angularjs.org/api/ngRoute)
Aprenda a instalar um servidor Web completo com o WampServer - http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2013/03/aprenda-instalar-um-servidor-web-completo-com-o-wampserver.html (http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2013/03/aprenda-instalar-um-servidor-web-completo-com-o-wampserver.html)
CodeAcademy - https://www.codecademy.com/learn/learn-angularjs (https://www.codecademy.com/learn/learn-angularjs)
Criando uma aplicação Single Page com AngularJS - https://tableless.com.br/criando-uma-aplicacao-single-page-com-angularjs/ (https://tableless.com.br/criando-uma-aplicacao-single-page-com-angularjs/)
Tableless - https://tableless.com.br/criando-uma-aplicacao-single-page-com-angularjs/ (https://tableless.com.br/criando-uma-aplicacao-single-page-com-angularjs/)

Agora que você já teve contato com diversos recursos do Angular JS, é hora de colocar a “mão na massa” e criar as suas aplicações.

Resumo

Nesse tópico foram abordados assuntos como rotas, filtros, serviços e diretivas do Angular JS, os quais são partes integrantes do *framework*, além de apresentar exemplos sobre cada recurso.

Conclusão

Os recursos presentes no Angular JS proporcionam facilidade no desenvolvimento de aplicações web, além de ser um *framework* mundialmente reconhecido e com ampla documentação de fácil acesso. Portanto, investir no aprendizado do Angular JS é um caminho muito interessante a ser seguido e você pode colocar em prática os outros assuntos estudados até agora, mesclando com os recursos do Angular JS.