# **A diretiva ng-repeat**

O Angular permite repetir marcações HTML através da diretiva ng-repeat. Primeiro, precisamos adicioná-la no elemento que desejamos repetir, segundo, precisamos indicar na diretiva a propriedade em $scope do controller que será iterada. Dentro desse contexto, temos alguns exemplos abaixo do uso da diretiva:

1) <p ng-repeat="frase in frases">{{frase}}</p>

2) <p ng-repeat="frases">{{frase}}</p>

3) <div ng-repeat="produto in produtos">{{produtos}}</p>

4) <section ng-repeat="propaganda">{{propaganda}}</p>

Podemos afirmar sobre os exemplos acima que:

* 1 e 3 estão corretas
* 3 e 4 estão erradas
* Todas estão erradas
* **Apenas 1 está correta**

## 

## **Resposta do instrutor**

O código <p ng-repeat="frase in frases">{{frase}}</p> é funcional. Ele itera sobre a lista frases, que existe no escopo de algum controller e para cada elemento da lista fornece um apelido, no caso frase. É através desse apelido que conseguimos acessar cada elemento da lista através da Angular Expression (AE){{frase}}. Para o primeiro elemento da lista, {{frase}} será o primeiro elemento e o parágrafo será criado, para o segundo elemento da lista, {{frase}} será este elemento e um novo parágrafo será criado. Essa repetição acontecerá até que a diretiva ng-repeat chegue ao final da lista.

# **O sistema de injeção de dependências do Angular**

Angular possui seu próprio sistema de injeção de dependências:

1) angular.module('alurapic').controller('a', function($http, $scope) {});

2) angular.module('alurapic').controller('b', $http, $scope) {};

3) angular.module('alurapic').controller('c', function($scope, $http) {});

4) angular.module('alurapic').controller('d', function($scope, http) {});

Levando em consideração o que você aprendeu neste capítulo, qual das alternativas abaixo é verdadeira?

* 1 e 4 injetam corretamente $http e $scope
* **2 e 4 não injetam corretamente $http e $scope**
* 3 e 4 injetam corretamente $http e $scope
* 1 e 3 não injetam corretamente $http e $scope

## 

## **Resposta do instrutor**

Angular foi o pioneiro no suporte à injeção de dependências no lado do cliente (navegador). Uma característica desse sistema é que toda injeção é feita pelo nome do parâmetro, por isso é muito importante passarmos os parâmetros nomeados de acordo com o serviço ou recurso do Angular que desejamos usar.

Ah, as opções 1 e 3 injetam corretamente, mas não eram alternativas de resposta, pelo contrário, a alternativa dizia que elas não injetavam corretamente. Pegadinha! :)

# **O serviço $http**

Angular fornece um serviço exclusivo para realização de requisição Ajax, o $http.Vejamos algumas afirmativas sobre este serviço:

1) Pode ser injetado como qualquer outro artefato do Angular

2) Possui a função .get, que recebe como parâmetro um endereço, inclusive é especializada na leitura de dados.

3) A função .get retorna uma promise.

4) Podemos usar .then e .catch para obter os dados e capturar erros respectivamente ou .success e.error, açúcares sintáticos com a mesma finalidade.

Com base nas afirmações acima, podemos dizer que:

* Todas são verdadeiras, exceto a 2
* Todas são verdadeiras, exceto a 4
* **Todas são verdadeiras**
* Todas são verdadeiras, exceto a 1

## 

## **Resposta do instrutor**

Todas as sentenças são verdadeiras. Existem outras funções em $http responsáveis pelo envio, remoção e atualização de dados quando seguimos o padrão REST(seus verbos).

# **Controller, ng-repeat, $scope, tudo junto!**

Qual das opções abaixo declara corretamente um controller e configura corretamente um elemento da view para usar a diretiva ng-repeat?

a)

// controller  
angular.module('alurapic').controller('NomeDoController', function() {  
 $scope.letras = ['A', 'B', 'C'];  
});

// na view  
<ul ng-controller="NomeDoController">  
 <li ng-repeat="letra in letras">{{letra}}</li>  
</ul>

**b) - X**

// controller  
angular.module('alurapic').controller('NomeDoController', function($scope) {  
 $scope.letras = ['A', 'B', 'C'];  
});

// na view  
<ul ng-controller="NomeDoController">  
 <li ng-repeat="letra in letras">{{letra}}</li>  
</ul>

c)

// controller  
angular.module('alurapic').controller(function($scope) {  
 $scope.letras = ['A', 'B', 'C'];  
});

// na view  
<ul>  
 <li ng-repeat="letras">{{letra}}</li>  
</ul>

## 

## **Resposta do instrutor**

Angular não perdoa! Basta um erro de sintaxe ou omitir um artefato, como $scope, para que nossa funcionalidade deixe de funcionar. A resposta correta obedece à sintaxe do Angular:

// controller  
angular.module('alurapic').controller('NomeDoController', function($scope) {  
 $scope.letras = ['A', 'B', 'C'];  
});

// na view  
  
<ul ng-controller="NomeDoController">  
 <li ng-repeat="letra in letras">{{letra}}</li>  
</ul>

# **$http e suas várias facetas**

Sabemos que $http.get retorna uma promise e que podemos encadear uma chamada à then. Um exemplo:

$http.get('v1/fotos')  
.then(function(retorno) {  
 $scope.fotos = retorno.data;  
})  
.catch(function(erro) {  
 // trata o erro  
});

## 

## **Resposta do instrutor**

O código modificado fica:

$http.get('v1/fotos')  
.success(function(fotos) {  
 $scope.fotos = fotos;   
})  
.error(function(erro) {  
 // trata o erro  
});

Neste exemplo, é importante notar que a função success recebe a lista de fotos diretamente retornada pelo servidor e não um objeto que contém a propriedade .data, quando usamos then. Como temos acesso à lista diretamente, fica mais legível chamarmos o parâmetro de fotos do que simplesmente retorno.