# **A diretiva ng-options**

Qual das opções abaixo possui a estrutura sintática correta para criação de uma combobox através da diretiva ng-options?

1. <select   
    ng-model="foto.grupo"  
    ng-controller="GruposController"  
    ng-options="grupo in grupos">  
    <option value="">Escolha um grupo</option>  
    </select>
2. <select   
    ng-model="foto.grupo"  
    ng-controller="GruposController"  
    ng-options="grupo.\_id, grupo.nome for grupo in grupos">  
    <option value="">Escolha um grupo</option>  
    </select>
3. <select   
    ng-model="foto.grupo"  
    ng-controller="GruposController"  
    ng-options="grupo.\_id as grupo.nome for grupo in grupos">  
    <option value="">Escolha um grupo</option>  
    </select>
4. <select   
    ng-model="foto.grupo"  
    ng-controller="GruposController"  
    ng-options="grupo.\_id, grupo.nome in grupos">  
    <option value="">Escolha um grupo</option>  
    </select>

**Resposta: c**

**Explicação**

**<select   
 ng-model="foto.grupo"  
 ng-controller="GruposController"  
 ng-options="grupo.\_id as grupo.nome for grupo in grupos">  
 <option value="">Escolha um grupo</option>  
 </select>**

**A diretiva ng-options 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 título. O restante "for grupo in grupos" percorrerá a lista de grupos disponibilizada no escopo do controller, construindo cada item de nossa lista.**

# **Transformação através de filtro**

Vejamos a seguinte expressão:

<p>{{frase}}</p>

Altere o código acima para colocar {{frase}} em caixa alta, mas usando recursos do Angular

**Explicação**

Angular possui filtros (não confundir com filter, da diretiva ng-repeat). Eles são aplicados através de | seguido do nome do filtro:

// altere aqui  
<p>{{frase | uppercase}}</p>

Mas cuidado: os filtros não funcionam com a diretiva ng-model!

# **Onde está o erro?**

Bootstrap ajuda bastante, mas termos que decorar zilhões de classes pode ser algo um tanto tedioso. Porém, podemos isolar a complexidade de seus componentes, em diretivas. Um exemplo disso é o simples botãoRemover da nossa listagem. Vejamos sua marcação:

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

Criar diretiva não é novidade para nós. Segue o código da diretiva meu-botao-perigo:

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}}</button>';  
 return ddo;  
 });

Nem tudo é perfeito, aliás, nossa diretiva está quase perfeita, a não ser por um erro. Que erro é esse que impedirá o funcionamento da diretiva meuBotaoPerigo?

**Explicação**

Não podemos usar @ para scope.acao. Este modificador sempre passa para diretiva um valor em string . No caso da nossa diretiva, precisamos que scope.acao esteja associada à uma função de um controller, isto é, do escopo pai na qual a diretiva está inserida. Para conseguirmos isso, usamos o modificador &. Sendo assim, para que nossa diretiva funcione, basta uma simples modificação:

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 : '&' // mudei aqui!  
 }  
 ddo.template = '<button class="btn btn-danger btn-block" ng-click="acao()">{{nome}}</button>';  
 return ddo;  
 });