01

# Criando nossas próprias diretivas

**Transcrição**

Começando deste ponto? Você pode fazer o DOWNLOAD (https://s3.amazonaws.com/caelum-online-public/angular-1/stages/03-alurapic.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

Progredimos com nossa aplicação, porém a marcação HTML da página `index.html` está um pouco confusa, pelo menos para mim. São muitas divs e classes, tudo porque estamos usando o Bootstrap. Ganhamos de um lado, mas perdemos em termos de legibilidade. Por exemplo, a marcação de um painel é assim:

```
<!-- exemplo, não entra em nenhum lugar -->
<div class="panel panel-default">
    <div class="panel-heading">
        <h3 class="panel-title text-center">Leão</h3>
      </div>
      <div>
          // conteúdo do painel
      </div>
</div>
```

## Confinando a complexidade do painel em uma diretiva

O mundo seria melhor se pudéssemos esconder a complexidade da marcação que vimos utilizando algo como um componente:

```
<!-- exemplo, não entra em nenhum lugar -->
<meu-painel titulo="Leão">
  // conteúdo do Painel
</meu-painel>
```

Mas a tag `meu-painel` é um componente que não existe no mundo HTML. Porém, lembram das diretivas do Angular? Elas nada mais são do que componentes reutilizáveis e que podem existir não apenas como atributos, mas como tags também. Será que podemos criar uma diretiva customizada do Angular que esconda a complexidade do painel do Bootstrap? Sim e é isso que faremos agora para deixar nossa marcação mais elegante, inclusive poderemos aproveitá-la em outros lugares da nossa aplicação.

O primeiro passo é criar um módulo em separado que conterá nossas diretivas. Vamos criá-lo em:

`public/js/directives/minhas-diretivas.js`

```
// public/js/directives/minhas-diretivas.js

angular.module('minhasDiretivas', [])
```

Criamos um novo módulo. Antes de mais nada, precisamos importar seu script em `index.html`, e inclusive adicioná-lo como dependência do módulo principal da nossa aplicação.

```
<!-- public/index.html -->
<!DOCTYPE html>
<html lang="pt-br" ng-app="alurapic">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <title>Alurapic</title>
        <link rel="stylesheet" href="css/bootstrap.min.css">
        <link rel="stylesheet" href="css/bootstrap-theme.min.css">
        <script src="js/lib/angular.min.js"></script>
        <script src="js/main.js"></script>
        <script src="js/controllers/fotos-controller.js"></script>
        <script src="js/directives/minhas-diretivas.js"></script>
    </head>
    <body ng-controller="FotosController">
        <div class="container">
            <div class="jumbotron">
                <h1 class="text-center">Alurapic</h1>
            </div>
            <div class="row">
                <div class="panel panel-default col-md-2" ng-repeat="foto in fotos">
                    <div class="panel-heading">
                        <h3 class="panel-title text-center">{{foto.titulo}}</h3>
                    </div>
                    <div class="panel-body">
                        <img class="img-responsive center-block" src="{{foto.url}}">
                    </div><!-- fim panel-body -->
                </div><!-- fim panel panel-default -->
            </div><!-- fim row -->
        </div><!-- fim container -->
    </body>
</html>


// public/js/main.js

angular.module('alurapic', ['minhasDiretivas']);
```

Pronto, agora criamos nossa primeira diretiva através da função **directive**. Ela recebe como primeiro parâmetro o nome da diretiva em *camelCase* e como segundo uma função que deve retornar um **directive definition object (DDO)**:

```
// public/js/directives/minhas-diretivas.js

angular.module('minhasDiretivas', [])
    .directive('meuPainel', function() {
        var ddo = {};
        return ddo;
    });
```

Veja que estamos retornando um objeto ainda sem qualquer configuração. Vamos começar restringindo a forma de uso da nossa diretiva. Como assim? Uma diretiva em Angular pode ser usada como **E*lemento,* *A**tributo ou **C*omentário***

*(esta última muito incomum). Vamos estipular que nossa diretiva pode ser usada tanto como atributo ou como elemento, adicionando em nosso DDO a propriedade* **restrict** com valor "AE":

```
// public/js/directives/minhas-diretivas.js

angular.module('minhasDiretivas', [])
    .directive('meuPainel', function() {
        var ddo = {};

        ddo.restrict = "AE";

        return ddo;
    });
```

Nossa diretiva ainda não está pronta, mas ela pode ser usada como elemento, assim:

```
<!-- exemplo, não entra em nenhum lugar -->
<meu-painel></meu-painel>
```

Ou como atributo, neste caso, usamos uma `div` adicionando a diretiva:

```
<!-- exemplo, não entra em nenhum lugar -->
<div meu-painel></div>
```

Uma coisa importante que pode ter passado sem você perceber. O nome da nossa diretiva está em *camelCase*, porém na marcação HTML estamos usando hífen. Este é um padrão do Angular que não podemos deixar de seguir, caso contrário nossa diretiva não funcionará.

Muito bem, qual é o próximo passo? Bem, podemos ter várias diretivas `meu-painel` numa mesma página, mas cada uma com seu próprio título. Algo assim:

```
<!-- exemplo, não entra em nenhum lugar -->
<meu-painel titulo="Leão"></meu-painel>
<meu-painel titulo="Zebra"></meu-painel>
<meu-painel titulo="Girafa"></meu-painel>
```

## Isolando nossa diretiva

Para que cada diretiva tenha seu próprio título, cada uma precisará ter um escopo isolado, que existe independente do contexto na qual está incluída. Além de permitir que cada uma tenha seus próprios dados, podemos reutilizar a diretiva em qualquer lugar sem que ela bagunce o escopo pai no qual está inserida!

Precisamos capturar o título passado pela diretiva para dentro de seu escopo isolado e fazemos isso adicionando em nosso DDO a propriedade **scope**:

```
// public/js/directives/minhas-diretivas.js

angular.module('minhasDiretivas', [])
```

```
    .directive('meuPainel', function() {

        var ddo = {};
        ddo.restrict = "AE";

        ddo.scope = {
            titulo: '@titulo'
        };

        return ddo;
    });
```

Criamos a propriedade `titulo` no escopo isolado da diretiva, porém o valor desta propriedade é curioso. Veja que nele temos `@titulo`, a sintaxe `@` indica que estamos **copiando o valor como string** do atributo `titulo` adicionando na diretiva em nossa marcação. Porém, quando o nome do atributo na diretiva na marcação é igual ao nome da propriedade que guardará o seu valor, podemos deixar apenas `@`:

```
// public/js/directives/minhas-diretivas.js

angular.module('minhasDiretivas', [])
    .directive('meuPainel', function() {

        var ddo = {};
        ddo.restrict = "AE";

        ddo.scope = {
            titulo: '@'
        };

        return ddo;
    });
```

Agora que restringimos a forma de uso da diretiva e criamos seu escopo isolado, precisamos definir a marcação HTML que será utilizada por ela. Fazemos isso através da propriedade **template**:

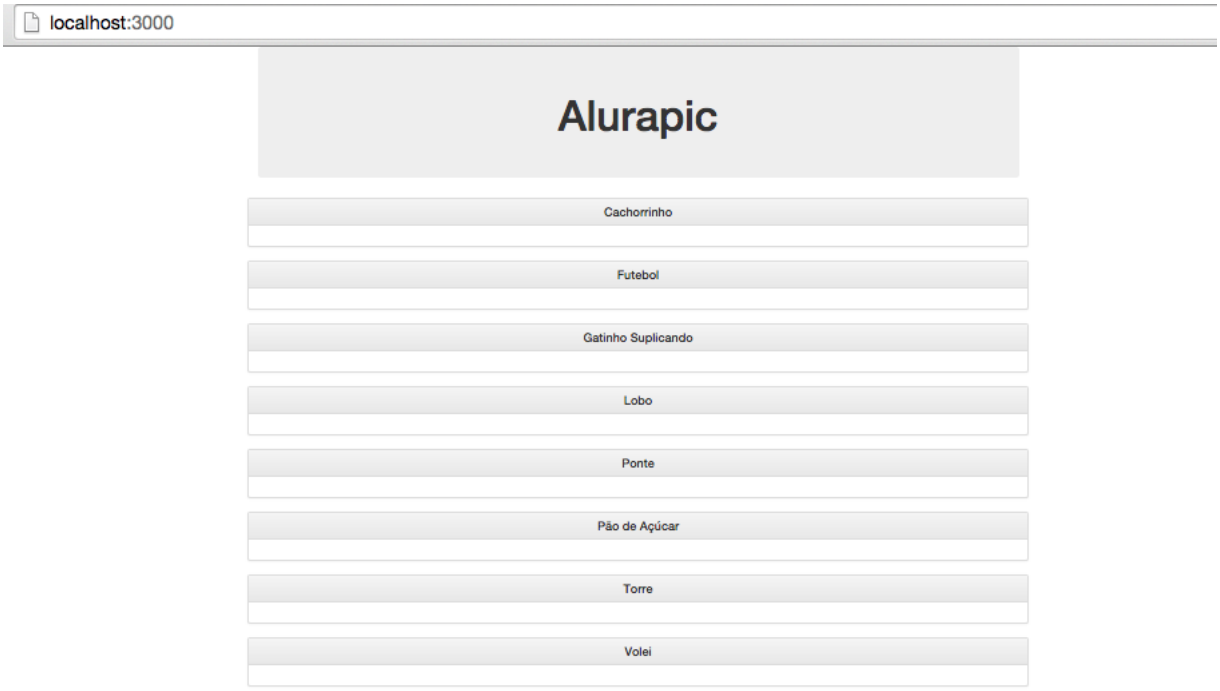
```
// public/js/directives/minhas-diretivas.js

angular.module('minhasDiretivas', [])
    .directive('meuPainel', function() {

        var ddo = {};

        ddo.restrict = "AE";

        ddo.scope = {
            titulo: '@'
        };

        ddo.template =
                '<div class="panel panel-default">'
            +   '   <div class="panel-heading">'
            +   '       <h3 class="panel-title text-center">{{titulo}}</h3> '
            +   '   </div>'
            +   '   <div class="panel-body">'
```

```
            +   '   </div>'
            +   '</div>'

        return ddo;
    });
```

Veja que no próprio template estamos usando a Angular Expression `{{titulo}}` para exibir o titulo passado como parâmetro para a diretiva. Será que funciona? Primeiro, vamos alterar `index.html` para fazer uso da nossa diretiva:

```
<!-- public/index.html -->
<!DOCTYPE html>
<html lang="pt-br" ng-app="alurapic">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <title>Alurapic</title>
        <link rel="stylesheet" href="css/bootstrap.min.css">
        <link rel="stylesheet" href="css/bootstrap-theme.min.css">
        <script src="js/lib/angular.min.js"></script>
        <script src="js/main.js"></script>
        <script src="js/controllers/fotos-controller.js"></script>
        <script src="js/directives/minhas-diretivas.js"></script>
    </head>
    <body ng-controller="FotosController">
        <div class="container">
            <div class="jumbotron">
                <h1 class="text-center">Alurapic</h1>
            </div>
            <div class="row">
                <meu-painel ng-repeat="foto in fotos" titulo="{{foto.titulo}}">
                    <img class="img-responsive center-block" src="{{foto.url}}" alt="{{foto.titu
                </meu-painel>
            </div><!-- fim row -->
        </div><!-- fim container -->
    </body>
</html>
```

Veja que adicionamos em nossa diretiva também a diretiva `ng-repeat`, porque queremos repetir nossa diretiva de acordo com a quantidade de fotos. E o resultado final fica:




## Resultado não esperado? Entendendo um pouco mais de diretivas

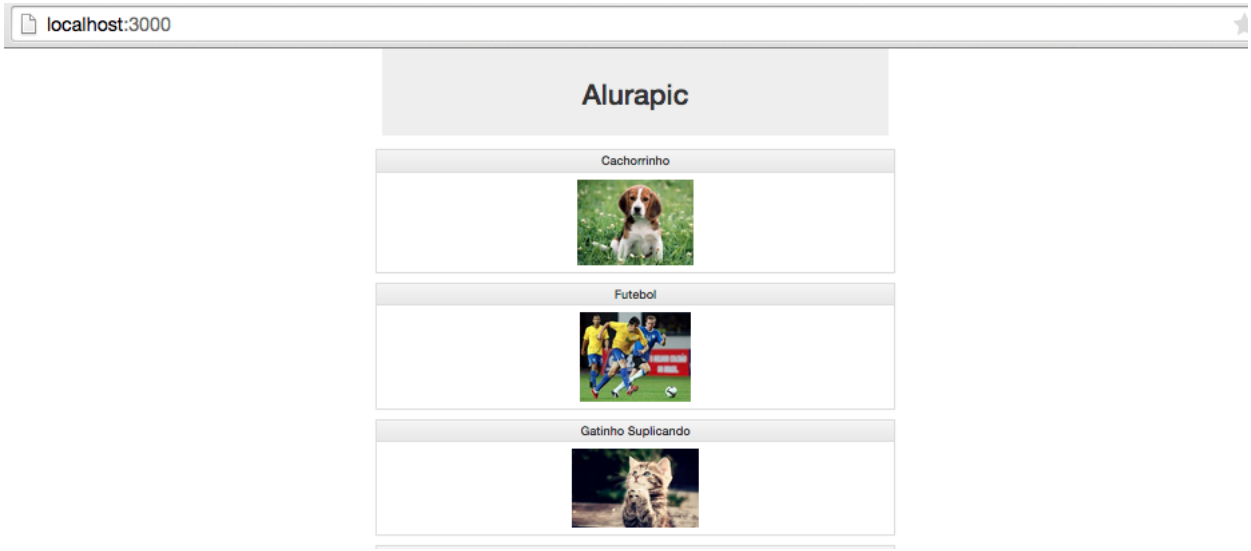
Ops! Funcionou parcialmente! Veja que o painel é montado e o título exibido, mas parece que nossa diretiva ignorou todo o seu conteúdo. Isso acontece porque o Angular altera aquele fragmento do DOM substituindo por um novo que representa a marcação da nossa diretiva, nesse processo, perdemos todos os seus elementos filhos. Para que o Angular preserve o conteúdo original da diretiva, precisamos usar o mecanismo de transclusão.

Apesar do nome intimidador, não se preocupe. Para ativarmos a transclusão basta adicionar no DDO a propriedade `transclude` com valor `true` e na marcação da diretiva adicionar a diretiva `ng-transclude` no elemento que deve preservar seus elementos filhos:

```
// public/js/directives/minhas-diretivas.js

angular.module('minhasDiretivas', [])
    .directive('meuPainel', function() {

        var ddo = {};

        ddo.restrict = "AE";
        ddo.transclude = true;

        ddo.scope = {
            titulo: '@'
        };

        ddo.template =
                '<div class="panel panel-default">'
            +   '   <div class="panel-heading">'
            +   '       <h3 class="panel-title text-center">{{titulo}}</h3> '
            +   '   </div>'
            +   '   <div class="panel-body" ng-transclude>'
            +   '   </div>'
            +   '</div>'
```

```
        return ddo;
    });
```

Agora sim! Vejamos o resultado:




## Separar ainda é a melhor coisa: HTML de um lado e diretiva do outro

Criamos nossa primeira diretiva! Porém, ficar concatenando HTML dentro da propriedade `template` não é algo gostoso de se fazer. Que tal colocarmos a marcação dentro de um arquivo HTML, local onde ele pertence? Para isso, vamos criar o arquivo `public/js/directives/meu-painel.html` com a marcação:

```
<!-- public/js/directives/meu-painel.html -->

<div class="panel panel-default">
    <div class="panel-heading">
        <h3 class="panel-title text-center">{{titulo}}</h3>
    </div>
    <div class="panel-body" ng-transclude>
    </div>
</div>
```

E agora, trocamos em nossa diretiva a propriedade `template` por **templateUrl** apontando para o arquivo HTML com a marcação da diretiva:

```
// public/js/directives/minhas-diretivas.js

angular.module('minhasDiretivas', [])
    .directive('meuPainel', function() {

        var ddo = {};

        ddo.restrict = "AE";
        ddo.transclude = true;

        ddo.scope = {
            titulo: '@'
        };
```

```
        ddo.templateUrl = 'js/directives/meu-painel.html';

        return ddo;
    });
```

O resultado tem que continuar o mesmo, pois alteramos apenas a maneira pela qual organizamos nosso código.

## O que aprendemos neste capítulo?

- diretivas são componentes reutilizáveis
- a criar nossas própria diretivas
- diretivas reutilizáveis devem possuir escopo isolado
- mecanismo de transclusão