

Data Bind e Angular CLI

APRESENTAR O RECURSOS DATA BINDING DO ANGULAR JS E DO ANGULAR CLI, ALÉM DOS CONCEITOS SOBRE MVC (MODEL, VIEW E CONTROLLER) EM UMA APLICAÇÃO WEB.

AUTOR(A): PROF. EDSON MELO DE SOUZA

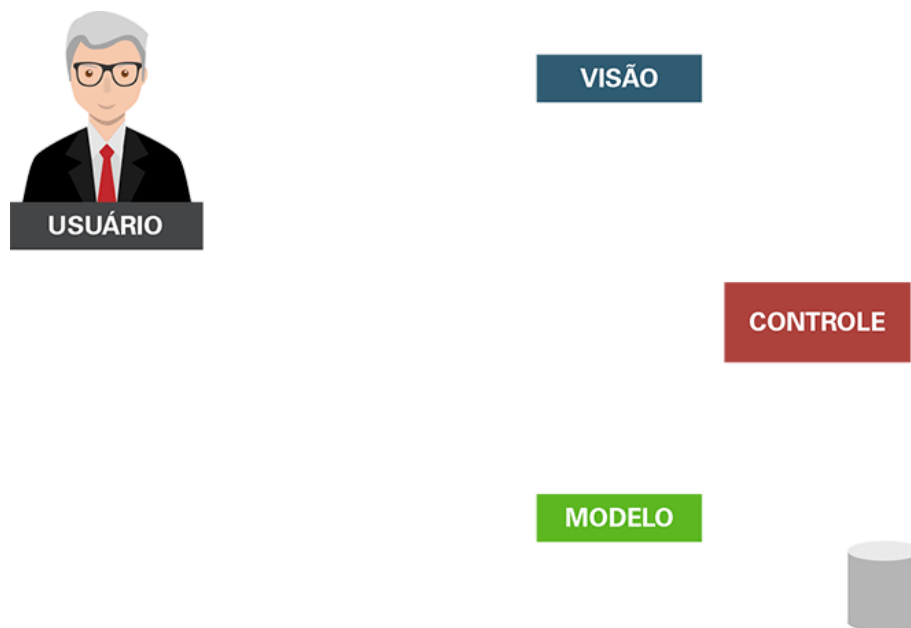
1. Introdução

Neste tópico serão abordados dois recursos do Angular JS que são muito importantes para o desenvolvimento de aplicações web no *Front-End* que são o Data Binding e o Angular CLI, além de uma rápida introdução do conceito do MVC para o desenvolvimento de aplicações web.

2. O MVC (*Model-View-Controller*)

O MVC é um padrão de arquitetura de software que visa realizar a separação dos elementos de um *software* em *FrontEnd* (visão do usuário – tela) e do *BackEnd* (processamento do núcleo ou motor da aplicação).

Este modelo facilita tanto o desenvolvimento quanto a manutenção do *software*, pois separa a regra do negócio da interface com o usuário. Na figura a seguir é mostrada a divisão e o funcionamento deste modelo.



Legenda: REPRESENTAÇÃO DO MODELO MVC

As siglas do MVC representam o Modelo, Visão e Controle, os quais são descritas a seguir:

Modelo (Model) – é a camada de *software* que representa os dados, de forma que permita ao sistema realizar o acesso (escrita e leitura) em um banco de dados, aos referidos dados. Nele são definidas as regras de validação e como o acesso aos dados são realizados, tornando transparente ao usuário como este é realizado.

Controle (Controller) – é a parte do *software* que realiza as ações de transações no sistema, entretanto, ele não faz acesso aos dados (função do *Model*), nem mesmo realiza a apresentação destes para o usuário. Ou seja, a função do controle é gerenciar o “de onde vem” e “para onde vão” os dados.

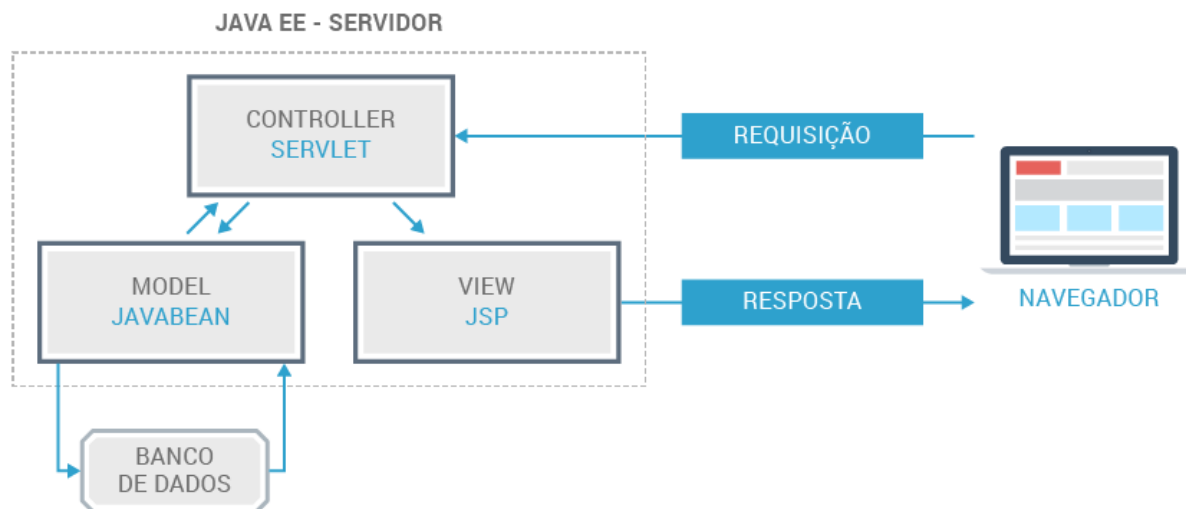
Visão (View) – é a parte do *software* visualizada pelo usuário, na qual ele pode interagir com o sistema por meio de formulários e *links* e é nesta camada que outras tecnologias como o jQuery, Ajax, CSS e HTML são utilizadas.

Quando se trata de aplicações para internet é importante pensar que, de alguma forma ou em algum momento, a aplicação poderá enviar ou receber dados externos, ou seja, ter que realizar a conexão com outras aplicações escritas até mesmo em outras linguagens, como o PHP e o ASP. Portanto, a divisão da aplicação em partes independentes torna isso possível, pois o modelo MVC permite que sejam reaproveitadas partes de uma aplicação em outra aplicação ou até mesmo partes (módulos) independentes, mas que façam uso de algum recurso da aplicação.

2.1 O MVC em Prática

A utilização do MVC facilita o desenvolvimento, pois divide as responsabilidades dentro de uma aplicação e também entre os programadores. Antes de iniciar a programação de uma aplicação é importante pensar na estrutura e definir como os arquivos irão ser armazenados, pois isso faz com que fique mais simples o desenvolvimento.

A seguir é mostrado o funcionamento básico de uma aplicação desenvolvida no modelo MVC utilizando a linguagem JSP (Java Server Pages - <http://docs.oracle.com/javaee/5/tutorial/doc/bnagx.html> (<http://docs.oracle.com/javaee/5/tutorial/doc/bnagx.html>)).



Legenda: FUNCIONAMENTO DO MVC EM UMA APLICAÇÃO JSP (JAVA SERVER PAGES)

3. Data Binding

Em uma definição bem rápida, em Angular JS o Data Binding é o recurso que permite a sincronização de um *model* (modelo) com os componentes de uma *view* (visualização) de forma automática.

Este recurso de vinculação dos dados (*Data Binding*) permite ao desenvolvedor escrever menos código, além de desvincular a manipulação manual de elementos DOM e de seus atributos.

O Angular apresenta um recurso de sincronização conhecido como *Two-Way Data Binding* ou, em tradução livre, Dois Caminhos ou Duas Mãos para Vinculação de Dados. Esse recurso permite que os dados sejam sincronizados nas duas direções, fazendo com que o *model* esteja sempre atualizado, isso porque a *view* é o reflexo do *model* em todos os momentos (MACORATTI, 2017).

No sistema MVC clássico, a maioria dos sistemas vinculam seus *templates* (modelos) como o *model* diretamente em uma *view* e isto não reflete as mudanças da *view* no *model*, necessitando que o programador escreva códigos para realizar a sincronização. Este processo é conhecido como *One-Way Data Binding* ou, em tradução livre, Um Caminho ou Uma Mão para Vinculação de Dados.

No Angular o processo funciona de modo diferente, pois o *model* (HTML com as marcações e diretivas) sofre a compilação no navegador, produzindo, em tempo real, a *view* e, quaisquer alterações que são realizadas na *view*, são refletidas imediatamente no *model*.

Esse modelo fornece simplicidade no desenvolvimento da aplicação, pois é necessário pensar apenas no modelo, já que a *view* é uma projeção do *model*. Portanto, o *controller* (controlador) é totalmente separado da *view*.

4. Data Model

O Data Model é a coleção ou grupo de dados disponíveis para uma aplicação. E, para trabalhar com ele, é necessário utilizar a diretiva *ng-model*. Essa diretiva irá sincronizar os elementos HTML (*input*, *select*, *textarea*) do *model* com a *view*. Essa diretiva irá permitir então o *Two-Way* entre o modelo e a visão (*model* e *view*).

No código a seguir é mostrada a utilização do *ng-model*, onde criaremos uma aplicação que terá um campo de entrada (*input*) e, ao digitar qualquer conteúdo, o valor será imediatamente escrito no corpo do HTML. Ao carregar a página, ou seja, a aplicação, já será escrito na página um valor que foi configurado no modelo (*model*) dentro do *controller*.

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <meta charset="utf-8">
5.         <meta http-equiv="X-UA-Compatible" content="IE=edge">
6.         <title></title>
7.
8.         <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4
9.
10.        <style type="text/css" media="screen">
11.            body{
12.                margin-left: 200px;
13.                margin-top: 50px;
14.                font-size: 40px;
15.                line-height: 2em;
16.            }
17.
18.            input{
19.                font-size: 40px;
20.                width: 400px;
21.            }
22.
23.            label{
24.                width: 320px;
25.                float: left;
26.            }
27.        </style>
28.
29.        <script>
30.            var app = angular.module('exemploAplicacao', []);
31.            app.controller('meuControle', function ($scope) {
32.                $scope.nome = "";
33.            });
34.        </script>
35.
36.    </head>
37.    <body>
38.
39.        <div ng-app="exemploAplicacao" ng-controller="meuControle">
```

```
40.         <label>Nome: </label> <input ng-model="nome"><br>
41.         <label>Resultado:</label> <strong>{{nome}}</strong><br>
42.     </div>
43.
44. </body>
45. </html>
```

Nas linhas de 29 a 34 é criado o *script* da aplicação.

Na linha 30 é criada uma variável “*var app*” que declara o nome da aplicação. Na linha 31 é criado o *controller* com o nome “meuControle” e recebe uma função que define o escopo da aplicação.

Na linha 32 é declarado o escopo, ou seja, uma variável de escopo “\$scope” denominada “nome”. Essa propriedade armazena um nome que será utilizado posteriormente para exibição na *view*.

Na linha 39 é iniciada a aplicação Angular, informando o seu nome `ng-app="ExemploAplicacao"` e também o *controller* `ng-controller="meuControle"`.

Na linha 40 é utilizada a diretiva `ng-model="nome"`, que, ao ser preenchida, atualiza dinamicamente o *controller* e exibe o valor (linha 41).

Na linha 41 é incluída a expressão `{{nome}}`, que mostrará o valor armazenado na variável “nome”, criada dentro do controle (linha 32).

A seguir você pode visualizar o processo quando valores são inseridos no campo (*input*).



O próximo exemplo utiliza a mesma lógica do anterior, acrescentando mais uma variável para o controle de dois campos. Veja o código a seguir:

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <meta charset="utf-8">
5.         <meta http-equiv="X-UA-Compatible" content="IE=edge">
6.         <title></title>
7.         <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4
8.
9.         <style type="text/css" media="screen">
10.             body{
11.                 margin-left: 200px;
12.                 margin-top: 50px;
13.                 font-size: 40px;
14.                 line-height: 2em;
15.             }
16.
17.             input{
18.                 font-size: 40px;
19.                 width: 400px;
20.             }
21.
22.             label{
23.                 width: 320px;
24.                 float: left;
25.             }
26.         </style>
27.
28.         <script>
29.             var app = angular.module('exemploAplicacao', []);
30.             app.controller('meuControle', function ($scope) {
31.                 $scope.nome = "Edson";
32.                 $scope.sobrenome = "Souza";
33.             });
34.         </script>
35.
36.     </head>
37.     <body>
38.
39.         <div ng-app="exemploAplicacao" ng-controller="meuControle">
```

```
40.         <label>Nome: </label><input type="text" ng-model="nome"><br>
41.         <label>Sobrenome: </label><input type="text" ng-model="sobre
42.
43.         <label>Nome Completo: </label>{{nome + " " + sobrenome}}
44.     </div>
45.
46. </body>
47. </html>
```

Na linha 32 foi inserida uma nova variável ao controle “sobrenome”. Na linha 41 foi incluído mais um “ng-model” para a variável “sobrenome” e, por fim, na linha 43 foram concatenadas as duas variáveis para exibir o resultado.

No vídeo a seguir é mostrado o resultado do processamento no navegador.



5. Controllers

Os exemplos anteriores fizeram uso dos *controllers* para realizar o controle de dois objetos com duas propriedades. Além de variáveis, os controles aceitam métodos (funções internas) para retornar valores sem a necessidade de concatenar os resultados na *view*. No próximo exemplo o programa anterior foi alterado, mostrando a criação de um método que retorne um valor.


```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <meta charset="utf-8">
5.         <meta http-equiv="X-UA-Compatible" content="IE=edge">
6.         <title></title>
7.         <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4
8.
9.         <style type="text/css" media="screen">
10.             body{
11.                 margin-left: 200px;
12.                 margin-top: 50px;
13.                 font-size: 40px;
14.                 line-height: 2em;
15.             }
16.
17.             input{
18.                 font-size: 40px;
19.                 width: 400px;
20.             }
21.
22.             label{
23.                 width: 320px;
24.                 float: left;
25.             }
26.         </style>
27.
28.         <script>
29.             var app = angular.module('exemploAplicacao', []);
30.             app.controller('controleCustomizado', function ($scope) {
31.                 $scope.nome = "";
32.                 $scope.sobrenome = "";
33.
34.                 $scope.nomeCompleto = function () {
35.                     return $scope.nome + " " + $scope.sobrenome;
36.                 };
37.
38.             });
39.         </script>
```

```
40.  
41.     </head>  
42.     <body>  
43.  
44.         <div ng-app="exemploAplicacao" ng-controller="controleCustomizad  
45.             <label>Nome: </label><input type="text" ng-model="nome"><br>  
46.             <label>Sobrenome: </label><input type="text" ng-model="sobre  
47.             <label>Nome Completo: </label> {{nomeCompleto()}}  
48.         </div>  
49.  
50.     </body>  
51. </html>
```

Neste exemplo os valores iniciais das variáveis estão vazios, por esse motivo os campos aparecem também vazios no navegador.

Nas linhas 31 e 32 são criadas as variáveis do escopo. Na linha 34 é criado um método denominado “nomeCompleto” que recebe uma função. A função retorna um valor (linha 35), concatenando as duas variáveis. Na linha 47 o valor retornado pelo método é mostrado por meio da inclusão da expressão “nomeCompleto()”.

No vídeo a seguir é mostrado o resultado do processamento do código anterior.



5.1 Arquivos Externos

Em aplicações mais complexas normalmente são usados arquivos externos para conter a lógica e as regras de negócio. Essa metodologia facilita muito o controle dos códigos, pois a separação minimiza erros e permite a manutenção de forma menos complexa.

No exemplo a seguir o programa usa um arquivo externo para escrever apenas o *controller* da aplicação “controle.js”, que é incluído no arquivo HTML.

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <meta charset="utf-8">
5.         <meta http-equiv="X-UA-Compatible" content="IE=edge">
6.         <title></title>
7.         <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4
8.
9.         <style type="text/css" media="screen">
10.             body{
11.                 margin-left: 200px;
12.                 margin-top: 50px;
13.                 font-size: 40px;
14.                 line-height: 2em;
15.             }
16.
17.             input{
18.                 font-size: 40px;
19.                 width: 400px;
20.             }
21.
22.             label{
23.                 width: 320px;
24.                 float: left;
25.             }
26.         </style>
27.
28.         <script src="controle.js"></script>
29.
30.     </head>
31.     <body>
32.
33.         <div ng-app="exemploAplicacao" ng-controller="meuControle">
34.             <label>Nome: </label><input type="text" ng-model="nome"><br>
35.             <label>Sobrenome: </label><input type="text" ng-model="sobre
36.
37.             <label>Nome Completo: </label>{{nome + " " + sobrenome}}
38.         </div>
39.
```

```
40.     </body>
41. </html>
```

Na linha 28 é incluído o controle por meio da linha

```
1. <script src="controle.js"></script>
```

O conteúdo do arquivo “controle.js” é exibido a seguir:

```
1. angular.module('exemploAplicacao', []).controller('meuControle', function() {
2.     $scope.nome = "";
3.     $scope.sobrenome = "";
4.
5.     $scope.nomeCompleto = function () {
6.         return $scope.nome + " " + $scope.sobrenome;
7.     };
8.
9. });
```

Na linha 1 é criado o módulo da aplicação, anexando um controle com as definições do escopo (linhas 2 e 3). Na linha 5 é criado o método que concatena as duas variáveis do escopo, retornando o valor quando o campo no HTML é preenchido.

Finalizamos por aqui o recurso de Data Binding, onde você pode verificar como é fácil realizar o controle de dados vinculados, utilizando o conceito de modelo, visão e controle. Nos links a seguir você poderá aprofundar seus estudos.

SAIBA MAIS!

Como funciona e o que é Data Binding - <https://www.eventials.com/tableless/data-binding-a-nova-moda-nos-frameworks-js/> (https://www.eventials.com/tableless/data-binding-a-nova-moda-nos-frameworks-js/)

Data Binding in AngularJS (inglês) - <https://www.sitepoint.com/two-way-data-binding-angularjs/> (https://www.sitepoint.com/two-way-data-binding-angularjs/)

6. NPM

Segundo o site NodeBR, “NPM é o nome reduzido de Node Package Manager (Gerenciador de Pacotes do Node). A NPM é duas coisas: Primeiro, e mais importante, é um repositório online para publicação de projetos de código aberto para o Node.js; segundo, ele é um utilitário de linha de comando que interage com este repositório online, que ajuda na instalação de pacotes, gerenciamento de versão e gerenciamento de dependência.” (NodeBR, 2016).

Gerenciar dependências significa que se sua aplicação necessita de alguma biblioteca externa, ao invés de você buscar em algum repositório, você executa o “npm” e ele automaticamente faz a inclusão do que é necessário em seu projeto.

A seguir será mostrado como utilizar o Angular CLI, o qual faz uso do “npm” para criar uma aplicação em angular.

7. Angular CLI

O Angular CLI (*Command Line Interface* ou Interface de Linha de Comando) é uma ferramenta utilizada para criar aplicações em Angular JS.


Quando vamos iniciar a criação de uma aplicação grande, é necessária a inclusão de diversos arquivos, bibliotecas, além da criação bem definida quanto a estrutura de pastas (diretórios).

Nesse sentido, o Angular CLI facilita essa criação, pois já executa a verificação de todas as dependências necessárias para o projeto inicial, criando as estruturas de pastas e incluindo os arquivos nos devidos locais.

A seguir é apresentado o passo a passo para a obtenção do Angular CLI e da criação de uma nova aplicação.

Observação: Se você não possui o Node.js instalado em seu computador, acesse esse link (<https://udgwebdev.com/node-js-para-leigos-instalacao-e-configuracao/> (https://udgwebdev.com/node-js-para-leigos-instalacao-e-configuracao/)) ou (<https://tableless.com.br/o-que-nodejs-primeiros-passos-com-node-js/> (https://tableless.com.br/o-que-nodejs-primeiros-passos-com-node-js/)) para realizar a instalação antes de prosseguir.

7.1 Obtendo o angular CLI

Abra o *prompt* de comando do Windows pressionando as teclas Windows+R () e digite o comando cmd. Com o *prompt* aberto, digite o seguinte comando:

```
1. npm install angular-cli -g
```

O processo pode demorar vários minutos, dependendo da sua conexão com a internet e da máquina que você está utilizando. Durante a instalação você deverá ter uma tela parecida com a da figura a seguir:

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows [versão 10.0.14393]
(c) 2016 Microsoft Corporation. Todos os direitos reservados.

C:\Users\souza>npm install angular-cli -g
npm WARN deprecated angular-cli@1.0.0-beta.28.3: angular-cli has been renamed to @angular/cli. Please update your dependencies.
npm WARN deprecated minimatch@2.0.10: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
C:\Users\souza\AppData\Roaming\npm\ng -> C:\Users\souza\AppData\Roaming\npm\node_modules\angular-cli\bin\ng

> node-sass@4.5.3 install C:\Users\souza\AppData\Roaming\npm\node_modules\angular-cli\node_modules\node-sass
> node scripts/install.js

Downloading binary from https://github.com/sass/node-sass/releases/download/v4.5.3/win32-x64-48_binding.node
Download complete
Binary saved to C:\Users\souza\AppData\Roaming\npm\node_modules\angular-cli\node_modules\node-sass\vendor\win32-x64-48_binding.node
Caching binary to C:\Users\souza\AppData\Roaming\npm-cache\node-sass\4.5.3\win32-x64-48_binding.node

> node-sass@4.5.3 postinstall C:\Users\souza\AppData\Roaming\npm\node_modules\angular-cli\node_modules\node-sass
> node scripts/build.js

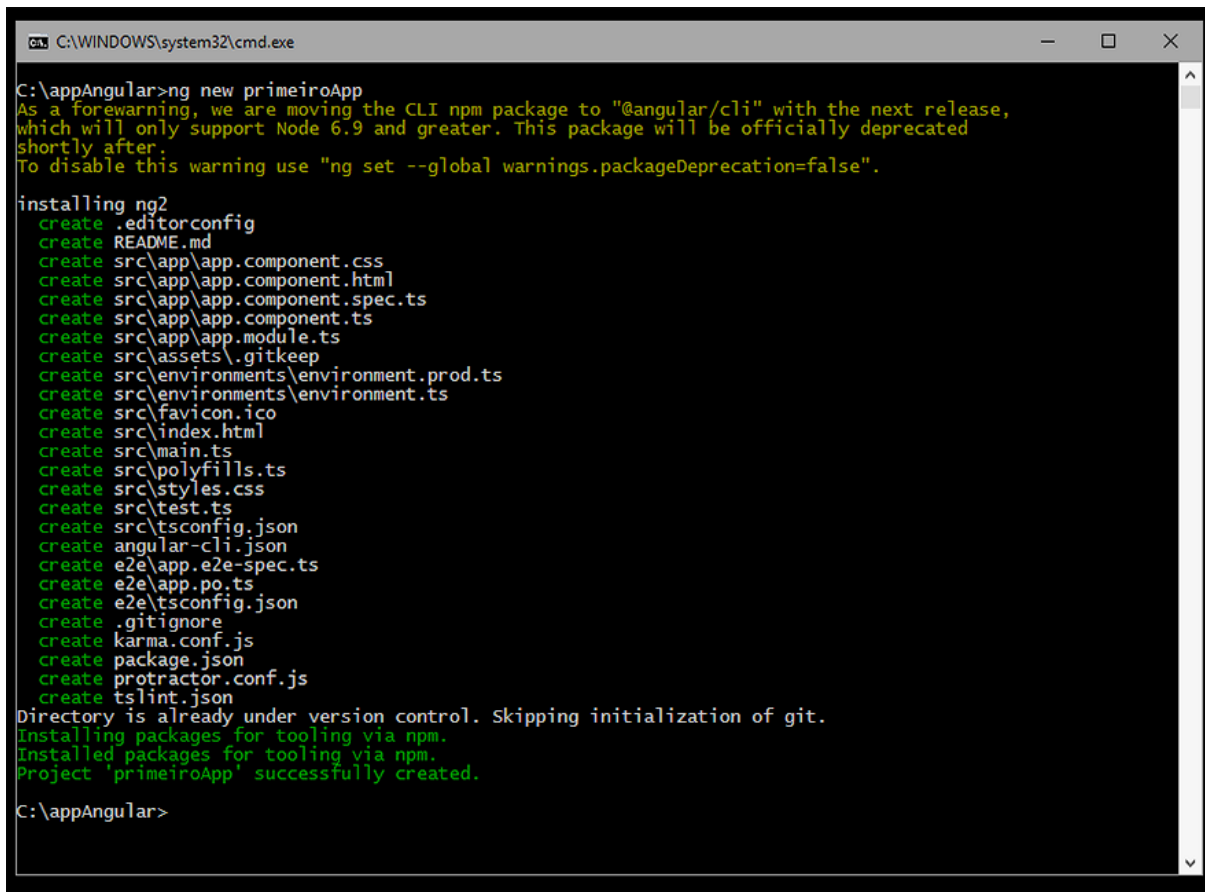
Binary found at C:\Users\souza\AppData\Roaming\npm\node_modules\angular-cli\node_modules\node-sass\vendor\win32-x64-48_binding.node
Testing binary
Binary is fine
C:\Users\souza\AppData\Roaming\npm
-- angular-cli@1.0.0-beta.28.3
+-- @angular/cli@1.0.0-beta.28.3
+-- @angular/tsc-wrapped@0.5.2
+-- tsickle@0.2.6
+-- source-map-support@0.4.15
+-- denodeify@1.2.1
+-- minimist@1.2.0
+-- mkdirp@0.5.1
+-- minimist@0.0.8
+-- rxjs@5.1.0
+-- source-map@0.5.6
+-- source-map-support@0.4.11
+-- symbol-observable@1.0.4

```

Após a finalização da instalação, selecione uma pasta de sua preferência para criar a aplicação. No exemplo foi utilizado o caminho “c:\appAngular”. Após selecionar o local, digite o seguinte comando:

1. ng new primeiroApp

A aplicação começará a ser criada (pode demorar um pouco). Ao final da instalação, você deverá ter uma tela parecida como a da figura a seguir:



```
C:\WINDOWS\system32\cmd.exe
C:\appAngular>ng new primeiroApp
As a forewarning, we are moving the CLI npm package to "@angular/cli" with the next release,
which will only support Node 6.9 and greater. This package will be officially deprecated
shortly after.
To disable this warning use "ng set --global warnings.packageDeprecation=false".

installing ng2
create .editorconfig
create README.md
create src\app\app.component.css
create src\app\app.component.html
create src\app\app.component.spec.ts
create src\app\app.component.ts
create src\app\app.module.ts
create src\assets\gitkeep
create src\environments\environment.prod.ts
create src\environments\environment.ts
create src\favicon.ico
create src\index.html
create src\main.ts
create src\polyfills.ts
create src\styles.css
create src\test.ts
create src\tsconfig.json
create angular-cli.json
create e2e\app.e2e-spec.ts
create e2e\app.po.ts
create e2e\tsconfig.json
create .gitignore
create karma.conf.js
create package.json
create protractor.conf.js
create tslint.json
Directory is already under version control. Skipping initialization of git.
Installing packages for tooling via npm.
Installed packages for tooling via npm.
Project 'primeiroApp' successfully created.

C:\appAngular>
```

O próximo passo é iniciar a aplicação. E, para isso, acessa a pasta da aplicação e execute o seguinte comando:

1. ng serve

Ao executar o comando, você terá uma tela semelhante como a da imagem a seguir:

```

angular-cli
O volume na unidade C não tem nome.
O Número de Série do Volume é AA74-A4DD

Pasta de C:\appAngular
28/05/2017 11:41 <DIR> .
28/05/2017 11:41 <DIR> ..
28/05/2017 11:42 <DIR> primeiroApp
0 arquivo(s) 0 bytes
3 pasta(s) 633.919.102.976 bytes disponíveis

C:\appAngular>cd primeiroApp
C:\appAngular\primeiroApp>ng serve
As a forewarning, we are moving the CLI npm package to "@angular/cli" with the next release,
which will only support Node 6.9 and greater. This package will be officially deprecated
shortly after.

To disable this warning use "ng set --global warnings.packageDeprecation=false".

fallbackLoader option has been deprecated - replace with "fallback"
loader option has been deprecated - replace with "use"
fallbackLoader option has been deprecated - replace with "fallback"
loader option has been deprecated - replace with "use"
fallbackLoader option has been deprecated - replace with "fallback"
loader option has been deprecated - replace with "use"
fallbackLoader option has been deprecated - replace with "fallback"
loader option has been deprecated - replace with "use"
** NG Live Development Server is running on http://localhost:4200. **
Hash: c1cb18b4b51a550464b3
Time: 11845ms
chunk {0} polyfills.bundle.js, polyfills.bundle.map (polyfills) 232 kB {4} [initial] [rendered]
chunk {1} main.bundle.js, main.bundle.map (main) 4.01 kB {3} [initial] [rendered]
chunk {2} styles.bundle.js, styles.bundle.map (styles) 9.71 kB {4} [initial] [rendered]
chunk {3} vendor.bundle.js, vendor.bundle.map (vendor) 2.62 MB [initial] [rendered]
chunk {4} inline.bundle.js, inline.bundle.map (inline) 0 bytes [entry] [rendered]
webpack: Compiled successfully.

```

A partir de agora, basta acessar o endereço <http://localhost:4200> (<http://localhost:4200/>) que você visualizará a aplicação em execução, conforme mostra a figura a seguir:



app works!

Agora é só começar a trabalhar na aplicação.

SAIBA MAIS!

Iniciando uma nova aplicação com Angular CLI - <https://medium.com/@rodrigoff/angular2-iniciando-uma-nova-aplica%C3%A7%C3%A3o-com-angular-cli-4550da4dc508>
(<https://medium.com/@rodrigoff/angular2-iniciando-uma-nova-aplica%C3%A7%C3%A3o-com-angular-cli-4550da4dc508>)

Resumo

Neste tópico foram apresentados os conceitos de MVC, Data Binding, NPM e do Angular CLI, mostrando como trabalhar com o Data Binding, *controllers*, *models* e *views*, além da instalação e do angular CLI e da criação de uma aplicação completa em Angular.

Conclusão

Aplicações em Angular JS estão em alta no mercado de tecnologia quando se trata de web. Portanto, é de fundamental importância que você procure saber mais sobre este *framework*, aprofundando seus conhecimentos.

Você pode perceber nos exemplos como o método de desenvolvimento com o Angular é simplificado, e como criar aplicações robustas com pouco código.

ATIVIDADE FINAL

O MVC é um padrão de arquitetura de software que visa realizar

- A. A separação dos elementos de um *software* em *FrontEnd* e *BackEnd*
- B. O aumento do desempenho de uma aplicação.
- C. Melhorar a navegação para o cliente em uma aplicação.
- D. Facilitar a programação.

Sobre o angular CLI é possível afirmar que se trata de

- A. Uma ferramenta para criar aplicações em Angular JS.
- B. Um gerenciador de *Controllers*.
- C. De um repositório de códigos angular JS.
- D. Um auxiliar para criação de *models*.

Em Angular JS, qual o recurso que permite a sincronização entre o model e a view?

- A. Data Binding.
- B. Controller.
- C. Model.
- D. View.

REFERÊNCIA

ANGULAR JS. Angular Docs: Typescript. 2017. Disponível em: <<https://angular.io/docs/ts/latest/>
(<https://angular.io/docs/ts/latest/>)>. Acesso em: 26 maio 2017.

____. Angular Data Binding. 2017. Disponível em: < <https://docs.angularjs.org/guide/databinding>
(<https://docs.angularjs.org/guide/databinding>)>. Acesso em: 26 maio 2017.

BALL, Erin B. Dive into AngularJS. 2015.

MACORATTI, José Carlos. Os fundamentos do AngularJS (revisão). Disponível em:
<http://www.macoratti.net/15/02/angjs_1.htm (http://www.macoratti.net/15/02/angjs_1.htm)>. Acesso em:
27 maio 2017.

RAHMAN, A. Abdur; DEVI, S. Chitra. A framework for ultra-responsive light weight web application using Angularjs. In: Green Engineering and Technologies (IC-GET), 2015 Online International Conference on. IEEE, 2015. p. 1-4.

W3SCHOOLS. AngularJS Tutorial. 2017. Disponível em: <<https://www.w3schools.com/angular/default.asp>
(<https://www.w3schools.com/angular/default.asp>)>. Acesso em: 26 maio 2017.