UNIVERSIDADE da MADEIRA

ESCOLA SUPERIOR DE TECNOLOGIAS E GESTÃO

Ano Letivo 2020/2021

Curso Técnico Superior Profissional em: Tecnologias e Programação de Sistemas de Informação

2 º **Ano**/1 º **Semestre**

Unidade Curricular: Sistemas Gestores de Bases de Dados II

Docente: Magno Andrade **Época:** Normal

BREVE EXPLICAÇÃO DO ANGULAR JS

Angular JS - é uma framework em Javascript para criação de *SPAs* (*Single Page Applications*).

1. Implementa o padrão MVC.
2. Não existe manipulação directa do DOM.
3. Suporta a maioria dos navegadores de *internet*, sejam para PC ou dispositivo móvel.

Single Page Applications

1. Aplicações *web* compostas unicamente por uma página.
2. *Views* (fragmentos) são carregados dinamicamente para as páginas.
3. Melhora a experiência do utilizador (sem *refresh* nas páginas).
4. As chamadas para o servidor são assíncronas.

Conceitos principais

Templates

São escritos em HTML que contém elementos e atributos específicos do AngularJS.

Directivas

1. Marcações em elementos do DOM para estender as funcionalidades do HTML.
    a. Adicionando novos comportamentos ao elemento.
    b. Transformar o elemento DOM e os seus "filhos".
2. As directivas podem estar em:
    a. Elementos → <my-dir></my-dir>
    b. Atributos →
    c. Comentários → <!-- directive: my-dir exp -->
    d. Classes →

Formatos de nomenclatura

1. O compilador do AngularJS suporta múltiplos formatos de nomenclatura.
    a. ng-bind → formato recomendado.
    b. data-ng-bind → formato recomendado para a página HTML ser válida.

Directivas Internas

São directivas que já estão criadas previamente e estendem a funcionalidades dos elementos do HTML.

Alguns exemplos:

1. ngApp → Designa o elemento raiz da aplicação e é tipicamente colocado no elemento raiz da página, ex: na *tag* <body> ou <html>
2. ngClick → especificar um comportamento personalizado quando um elemento é clicado.
3. ngController → atribui uma classe controlador (*controller*) à vista (*view*), que tem funções utilizadas nessa vista (*view*).
4. ngModel → "liga/associa" um elemento *input*, *select*, *textarea,* a uma propriedade/variável no objecto ***scope***.
5. ngRepeat → instancia um *template* uma vez por item de uma colecção, cada *template* tem o seu próprio scope, em que uma variável controla o ciclo.
6. ngSubmit → permite ligar expressões AngularJS para enviar eventos.

Expressões

O AngularJS liga/associa dados ao HTML usando Expressões.

As expressões podem ser escritas das seguintes formas:

1. Dentro de chavetas duplas → {{ expression }}
2. Dentro de directivas → ng-bind="expression"
3. Podem conter literais, operadores e variáveis
    a. {{ 5 + 5 }}
    b. {{ firstName + " " + lastName }}

Módulos

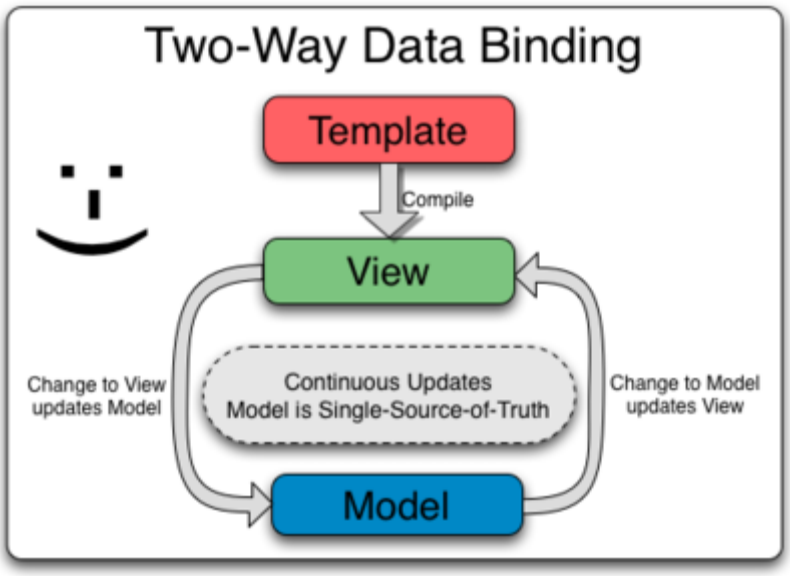
- Um módulo define uma aplicação.
- É um "contentor" para diferentes partes da aplicação.
- É um "contentor" para os controladores (*controllers*) da aplicação.
- Os controladores (*controllers*) pertencem sempre a um módulo.

Como criar um módulo:

```
<div ng-app="myApp">...</div>

<script>

var app = angular.module("myApp", []);

</script>
```

Ligação de dados em dois sentidos (Two-way *Data Binding*)

É a sincronização automática entre o modelo (*model*) e a vista (*view*).



Exemplo de modelo:

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstname = "John";
    $scope.lastname = "Doe";
});
```

Exemplo de vista:

```
<p>First name: {{firstname}}</p>
```

Quando os dados no modelo (*model*) são alterados, a vista (*view*) reflecte esta alteração, e quando os dados da vista alteram-se, o modelo é actualizado também. Isto acontece de forma imediata e automática, certificando-se assim que o modelo e a vista estão em sintonia.

Controladores

As aplicações em AngularJS são controladas por controladores.

A directiva ng-controller define o controlador.

Um controlador está sempre associado a um módulo (*module*).

Exemplo:

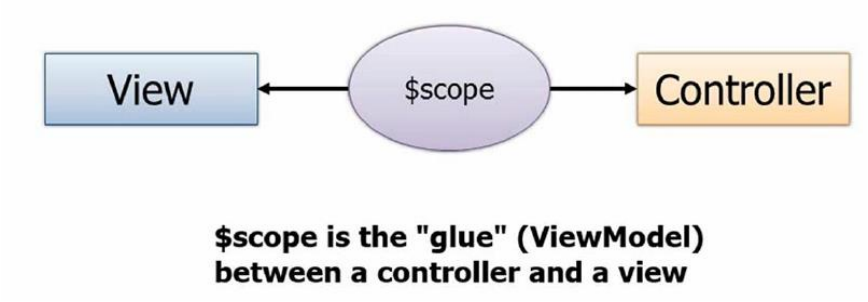
```
<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
});
</script>
```

Scope

O ***scope*** é a parte de ligação entre o HTML (*view*) e o JavaScript (*controller*).

O ***scope*** é um objecto com todas as propriedades e métodos disponíveis.

O ***scope*** está disponível na vista (*view*) e controlador (*controller*).



$scope is the "glue" (ViewModel) between a controller and a view

Exemplo:

```
<div ng-app="myApp" ng-controller="myCtrl">

<h1>{{carname}}</h1>

</div>

<script>
var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {
    $scope.carname = "Volvo";
});
</script>
```

Considerando que uma aplicação AngularJS consiste em:

- Vista (*view*) → que é o HTML.
- Modelo (*model*) → que é os dados actualmente disponíveis na vista actual.
- Controlador (*controller*) → que é as funções de *Javascript* que modificam, removem e controlam os dados.

O ***scope*** é considerado o Modelo (*model*).

Todas as aplicações tem um ***$rootScope*** criado no elemento HTML, que contém a directiva ***ng-app***, e está disponível em toda a aplicação.

Serviços (*Services*)

É possível utilizar serviços já criados ou criar novos.

Em AngularJS, um serviço é uma função, ou objecto disponível e limitado apenas a aplicação AngularJS.

São utilizados nos controladores através da injecção de dependência (*dependency injection*).

O serviço ***$http:***

- É um dos serviços mais usados no AngularJS, utilizado para realizar um pedido a um servidor, e a aplicação trata da resposta do mesmo.

Exemplo:

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
    $http.get("welcome.htm").then(function (response) {
        $scope.myWelcome = response.data;
    });
});
```

Criar o seu próprio serviço:

Para criar um serviço é necessário implementar o seguinte código:

```
app.service('hexafy', function() {
    this.myFunc = function (x) {
        return x.toString(16);
    }
});
```

E depois "injectá-lo" como dependência no controlador:

```
app.controller('myCtrl', function($scope, hexafy) {
    $scope.hex = hexafy.myFunc(255);
});
```

Roteamento (*Routing*)

A maioria das aplicações é composta por mais do que uma vista.

Em aplicações SPA, as vistas são renderizadas na mesma página que contém os elementos em comum (*Layout Template*).

Cada vista (*Partial Template*) é colocada num ficheiro próprio e carregada de forma dinâmica para dentro da *Layout Template*.

Usado para navegar entre diferentes páginas.

Duas maneiras:

- Com a biblioteca oficial do AngularJS – ***ngRoute***

- Com uma biblioteca não oficial - ***UI-Router***

Exemplo com biblioteca não oficial –> ***UI-Router***:

index.html

```
<html>
  <head>
    <script src="lib/angular.js"></script>
    <script src="lib/angular-ui-router.js"></script>
    <script src="helloworld.js"></script>

    <style>.active { color: red; font-weight: bold; }</style>
  </head>

  <body ng-app="helloworld">
    <a ui-sref="hello" ui-sref-active="active">Hello</a>
    <a ui-sref="about" ui-sref-active="active">About</a>

    <ui-view></ui-view>
  </body>
</html>
```

helloworld.js

```
var myApp = angular.module('helloworld', ['ui.router']);

myApp.config(function($stateProvider) {
  var helloState = {
    name: 'hello',
    url: '/hello',
    template: '<h3>hello world!</h3>'
  }

  var aboutState = {
    name: 'about',
    url: '/about',
    template: '<h3>Its the UI-Router hello world app!</h3>'
  }

  $stateProvider.state(helloState);
  $stateProvider.state(aboutState);
});
```