AngularJS

por Gustavo Silva

O que é

- **AngularJS é um framework Javascript mantido pelo Google que serve de alternativa ao jQuery.**
- **É o que há de mais moderno em Javascript front-end hoje.**
- **Totalmente indispensável para aplicações modernas: rápido, compartimentalizado, fácil de aprender;**
- **Pouca mão de obra qualificada no Brasil.**

Diferenciais

- **Abstrai o máximo de manipulação do DOM e servidor;**
- **Fácil controle de dependências (dependency injection);**
- **Segue o modelo MVC;**
- **Dividido em módulos, melhor organização;**
- **Criação de novas tags HTML;**
- **Ligação entre views e models (two-way data binding);**
- **Declarativo ao invés de imperativo;**
- **Integrada com Node.js e bibliotecas do NPM.**

Comparação

- **<input type="text" id="inputNome" />**
  ** Olá, ! **

  — — —

  $("#inputName").on("change", function(){
  var nome = $(this).val();
  $("#nome").text(nome);
  });

- **<input type="text" ng-model="nome" />**
  ** Olá, {{nome}}! **

Objetos

- **Os objetos do AngularJS são os módulos que compõem a sua aplicação. São de diferentes tipos. Os principais são:**

  - **Controller:** Serve para manusear a view.

  - **Service/Factory:** Cuida da lógica do sistema, funções como conexão com o servidor.

  - **Directive:** Permite criar novas tags HTML, ou novos atributos HTML, abstraindo ainda mais seu DOM;

  - **Filter:** Filtra coisas, mudando o estilo de exibição ou removendo coisas de um array de acordo com uma condição;

Objetos

- **Constant / Value:** Guardam um valor a ser injetado (por exemplo, poderíamos ter o value “Pi”).
- **Decorator:** Faz alterações em outros objetos.
- **Config:** Faz a configuração inicial de alguns objetos e bibliotecas.

Iniciando

- **Criamos nosso aplicativo Angular no arquivo app.js:**
  - *app = angular.module('workshop', []);*
    - Dentro de [], colocamos as bibliotecas que vamos usar.
- **Delimitamos nosso aplicativo no HTML. Pode ser em uma ou no <body>, com a directive "ng-app":**
  - *<body ng-app="workshop">*

Primeira controller

- **No arquivo controllers.js, criamos uma controller no nosso app:**
  - *app = angular.module('workshop');*
    app.controller("HomeCtrl", function(){
    });
    - Todo o conteúdo da controller vai dentro dessa função.
- **Agora, definimos o espaço dela no HTML:**
  - * *
    ...
    * *

Dependency Injection

- **É a grande magia do AngularJS. “Dependency Injection” é o modo como os objetos (services, controllers) se comunicam.**

- *app.service(“somaService”, function(){*
  return {somar: somar};

  function somar(a, b) {
  return a+b;
  }
  });

- *app.controller(“HomeCtrl”, function(somaService){*
  alert(somaService.somar(3, 5));
  });

- **Note como o serviço foi “injetado” na controller para ser usado lá.**

Two-way Data Binding

- **Outra magia do AngularJS. Permite a comunicação entre o HTML e sua controller associada.**
  - Desse modo, objetos atualizados na controller são automaticamente atualizados na view e vice-versa.
- **O responsável pela ponte é o objeto $scope, que deve ser injetado na controller. As propriedades de $scope são os objetos e funções acessados na view através de directives.**
  - *app.controller("HomeCtrl", function($scope){*
    $scope.nome = "Gustavo Silva";
    });
  - * Olá, {{nome}}! *
  - Repare que não foi necessário explicitar o $scope na view.

Directives Importantes

- **O AngularJS já traz várias directives extremamente úteis e que facilitam muito o desenvolvimento:**
  - **ng-show, ng-hide:** Mostra/Esconde um elemento se o objeto for true ou false.
    - Carregando…
  - **ng-model:** Faz o binding entre o value de um input e um objeto.
    - <input type="text" ng-model="email" />
  - **ng-class:** Adiciona uma classe a um elemento se um objeto for true. Recebe um objeto JS onde a chave é a classe e o conteúdo é o objeto.
    -

Directives Importantes

- **ng-repeat:** Uma das mais legais. É como se fosse um for, repete um bloco HTML para cada item de um array.
  - <div ng-repeat=“pessoa in pessoas”>{{pessoa.nome}}
- **ng-click:** Função a ser chamada ao clicar o elemento.
  - <button ng-click=“fazerLogin()”>Entrar</button>
- **{{ }}:** Apenas exibe o valor do objeto.

Dicas de Directives

- **Directives de true/false podem receber expressões:**
  -
- **Directives como ng-click podem receber atribuições:**
  - Contar
- **PS: Funções no $scope:**
  - *$scope.login = function(user, senha){*
    return "bleh";
    };

    $scope.login("joao", "6666");
    * Entrar *

Value

- *app.value(“Pi”, “3.14”);*

- *app.controller(“HomeCtrl”, function($scope, Pi) {*
  $scope.piMsg = “O valor de Pi é “+Pi;
  });

- **Não pode ser modificado por quem o injetar;**

- **Um Value pode ser um objeto JS; nesse caso seus atributos podem ser modificados pelos utilizadores.**

- **Value X Constant: Iguais, porém Constant não pode ser decorada** (o que é isso?).

Decorator

- *app.decorator("Pi", function($delegate){*
  return $delegate + "159262";
  });

- *app.controller("HomeCtrl", function($scope, Pi) {*
  $scope.piMsg = "O valor de Pi é "+Pi;
  });

- **Decorators modificam outros serviços;**
- **Esta sintaxe só funciona após o Angular 1.4;**
- **O objeto injetado $delegate é o objeto original que modificaremos;**
- **Deve ser declarado depois do objeto alvo!**

Filter

- **Filtros modificam a aparência de valores ou listas sem modificar o objeto inicial.**
- **Podem servir para filtrar resultados, mas também ordenar, modificar a exibição ou outras coisas.**
- **Utilização:**
  - {{ pessoa.nome | uppercase }}
  -
  - $scope.dinheiros = $filter('currency')(15, "R$");

Filters Padrão

- **O Angular já vem com vários filtros prontos:**
  - **uppercase, lowercase:** Modificam strings;
  - **number:** Modifica casas decimais de um número:
    {{ 15 | number:2 }} -> 15.00
  - **currency:** Formata moedas:
    {{ 15 | currency:”R$”:2 }} -> R$15.00
  - **date:** Formata datas. Melhor mexer com momentJS.
  - **limitTo:** Exibe apenas o começo de uma string ou lista;
  - **filter:** Filtra uma lista por um padrão:
    {{ pessoas | filter:{ ‘nome’ : ‘M’} }} -> Maria, Mauro, OMAR
    *PS: **filter:[...]:false** para comparação case-insensitive*
  - ***orderBy:** ordena uma lista:*
    {{ pessoas | orderBy:’+nome’ }} -> ordena por nome da pessoa de A pra Z

Custom Filters

- **Podemos criar nossos próprios filters:**

- *app.filter("filtrador", function() {*
  return function(input, opt1, opt2) {
  var output;
  … filtrar coisas …
  return output;
  });

- **Basta utilizar app.filter() e retornar uma função filtradora.**

- **Ela recebe o input e uma série de parâmetros opcionais, realiza suas funções e retorna um output filtrado.**

Services

- **Realizam ações como processar coisas, conectar-se com servidor, salvar em cache, persistir informação entre controllers, tomar decisões, etc.**

- **Devem ser armazenados em um arquivo próprio, services.js;**

- **Service ou Factory?** Factories retornam um objeto que será instanciado, service é um função construtora de um objeto. Não entendeu?

- **A confusão entre Service e Factory é maior do que a importância dela. Na prática, use um ou outro que dá na mesma.**

Factory

- app.factory(“calculadora”, function(somaService){
    var valorAleatorio = 666;
    return {
        somar: somar,
        substrair: substrair,
        valorAleatorio: valorAleatorio
    };

    function somar(a, b) {
        return somaService.somar(a, b);
    }

    function substrair(a, b) {
        return a-b;
    }
});

Service

- app.service(“calculadora”, function(somaService){
    this.somar(a, b) {
        return somaService.somar(a, b);
    }

    this.subtrair(a, b) {
        return a-b;
    }

    this.valorAleatorio = 666;
});

Custom Directives

- **É possível criar suas próprias directives com o Angular. Por exemplo, se você usa muito este código:**
  - * <span ng-transclude> *
- **Pode abstrair para isso:**
  - *<connection-error>Verifique sua conexão de internet</connection-error>*

Custom Directives

- *app.directive(“connectionError”, function(){*
  return {
  replace: true,
  restrict: ‘E’,
  transclude: true,
  template: “ … ”
  };
  });

- **Replace:** Define se é pra apagar a directive ao colocar o template.
- **Restrict:** O formato da directive. **“E”** é elemento (<connection-error>), **“A”** é atributo (<div connection-error>) e **“C”** é classe (div class=“connection-error”);
- **Escreve-se em camelCase na directive, depois ele converte sozinho pra snake-case.**
- **Custom directives são muito poderosas e complexas, isso é apenas um exemplo bem simples.**

Promises e $q

- **Javascript é uma linguagem assíncrona;**
  - Se uma função chamada for lenta (como uma requisição HTTP), o código depois dela seguirá normalmente enquanto ela é processada.
  - Isso pode ocasionar problemas. Como garantir que um trecho de código só será executado quando outro acabar?
- **Resposta: Promises**
  - Receita: crie sua promise, adicione um trecho de código a ser executado se tudo der certo, um trecho a ser executado se houver erro, amarre a promise numa função lenta e ela irá executar a promise quando terminar de rodar.

Promises e $q

- **Como funciona na prática: biblioteca Q (ou $q no Angular):**

  - *funcaoLerda()**.then**(function(data){*
    alert("uhuul funcionou");
    }, function(error){
    alert("deu xabu "+error);
    });

  - *function funcaoLerda() {*
    var deferred = $q.defer();
    … executando lerdezas …
    ***deferred.resolve();** //acabou!*
    ***deferred.reject();** //deu erro!*
    … mais código …
    //isso vai ser executado antes dos resolves() devido à lerdeza
    ***return deferred.promise;** //retorna a promise para que a função de fora possa*
    //amarrar coisas nela com then()
    }

- ***Não esquecer de injetar $q!***

AJAX

- **AJAX é uma tecnologia de comunicação entre servidor e cliente - e é a principal no Angular.**
- **Serviço responsável: $http (não esqueça de injetar);**
- **Seu método post() retorna uma promise - amarre um .then() nela ou seu código vai ser executado antes que o servidor responda!**

```
$http.post("home.php", {'id': 20}).then(function(data){
    alert("recebi do servidor: "+data);
}, function(error){
    alert("requisição falhou com erro "+error);
});
```

Utilitários

- **$timeout:** equivalente ao setTimeout do JS puro (não usar este!);
- **$resource:** wrapper do $http para requisições no padrão REST;
- **momentJS:** tradicional biblioteca para datas. Funciona normalmente no Angular sem problemas;
- **angular-translate:** excelente para i18n;
- **flex-calendar:** calendário bonito em Angular;
- **angular-local-storage:** armazena dados localmente.

Como saber mais

- Documentação: https://docs.angularjs.org/guide
- Shaping up with AngularJS: curso de graça no CodeSchool. https://www.codeschool.com/courses/shaping-up-with-angular-js
- Tire suas dúvidas e aprenda coisas novas conforme for sendo necessário!

Obrigado!