Mòduls, Scopes i Controladors

Mòduls

Però, què es un mòdul? Podríem veure-ho com un paquet de Java. Ve a ser una forma d'agrupar funcionalitats d'Angular. Per exemple, podríem crear varies directives relacionades amb un Soci d'un videoclub. Lo normal seria agrupar totes aquestes directives en un unic mòdul i després al crear la nostre aplicació dir que anem a emprar aquest mòdul.

Amb els mòduls, podem realitzar les següents accions:

- Crear un mòdul: Per a crear un mòdul, tenim que invocar el mètode module(). Veiem un exemple emprant la sintaxis correcte:
  angular.module("Soci",[]);

  El corxets com propietat del mètode module(), indiquen la necessitat de crear un nou mòdul amb el nom "Soci". En cas de no indicar els corxets, no el crearia i cercaria si existeix aquest mòdul.

- Obtenir un mòdul: Tal com hem dit, podem obtenir un mòdul existent indicant la sintaxis següent:
  var moduloSoci=angular.module("Soci");

  Ara en la variable de Javascript moduloSoci tenim una referencia al mòdul anomenat Soci, el que ja existia prèviament.

- Crear i obtenir un mòdul: També podem simplificar amb les tasques, resumint la sintaxis en una sola instrucció. A continuació podem veure un exemple:
  var moduloSoci=angular.module("Soci",[]);

Propietats de mòduls

Dins del mètode module(), podem trobar les següents propietats:
El primer argument es el nom del mòdul, que correspon amb el nom de la teva aplicació. En el segon paràmetre pots indicar una sèrie de mòduls addicionals, separats per comes, que serien las teves dependències. Poden ser mòduls propis de AngularJS, de altres autors o que tu mateix hagis decidit organitzar el teu codi en diferents mòduls. El tercer paràmetre es opcional i en ell indicaríem una funció que servis per configurar AngularJS.

Veure Ejercici1.html i Ejercici1.js

Què poden fer els scopes?

Per a poder introduir els controladors tindrem que aturar-nos abans en un concepte que es repeteix fins la sacietat dins de la literatura de AngularJS, el "scope". De fet, tal com anomena la documentació de AngularJS, el objectiu d'un controlador consisteix en realitzar una funció constructora capaç de augmentar el Scope.

El "scope" es la peça mes important del motor de AngularJS i es a on estan les dades que es tenen que manipular dins de la part de presentació.

El scope es un gran contenidor de dades, que transporta i fa visible la informació necessària per a implementar l'aplicació, des de el controlador a la vista i des de la vista al controlador. En termes de codi el

scope no es més que un objecte al que pots assignar propietats noves, amb les dades que necessites, o inclòs amb funcions (mètodes).

Aquestes dades i aquestes funcions estan visibles tant amb el Javascript dels controladors com en el HTML de las vistes, sense que tinguem que realitzar cap codi addicional, doncs Angular ja s'encarrega d'allò automàticament. A més, quan sorgeixen canvis en les dades es propaguen entre els controladors i les visites automàticament. Aquest es realitza per un mecanisme que anomenem "binding", i en AngularJS també "doble binding" (en català seria enllaç), que explicarem amb detall en futurs articles.

Així doncs, des de els controladors anem a ser capaços de treballar amb el scope d'una manera minuciosa, agregant o modificant informació segons ho requereixi la nostra aplicació.

Directives i scopes

Què son las directives

Les directives són noves "ordres" que s'incorporen al HTML i els podem assignar a qualsevol de les etiquetes per medi d'atributs. Són com marques en elements del DOM de la pàgina que li indiquen a AngularJS que tenen que assignar-li un comportament determinat o inclús transformar aquest element del DOM o algun dels seus fills.

Quan s'executa una aplicació que treballa amb Angular, existeix un "HTML Compiler" (Compilador HTML) que s'encarrega de recórrer el document i localitzar les directives que hagués col·locat dins del codi HTML, per a executar aquells comportaments associats a aquestes directives.

AngularJS ens porta una sèrie de directives "de fàbrica" que ens serveixen per fer coses habituals, així nosaltres i altres desenvolupadors poden desenvolupar directives pròpies per enriquir el framework.

Directiva ngApp (ng-app)

Aquesta és la marca que indica l'element arrel de la nostre aplicació. Es col·loca com atribut en la etiqueta que desitgem que sigui la arrel. És una directiva que auto arranca la aplicació web AngularJS. Es llegeix "Enyi ap" i el més comú és posar-ho al principi del documento HTML, en la etiqueta HTML o BODY, però també ho podràs col·locar en un àrea més restringida dins del document en una altre de les etiquetes de la pàgina.

```
<html ng-app>
```

Nota: Com podem comprovar, ng-app és una directiva i s'indica com si fos un atribut el HTML. Però no ho es cap atribut real de HTML. Si s'utilitza un validador de HTML advertirà que aquest atribut és inexistent i s'interpretarà per un error de validació. Per solucionar aquest possible inconvenient, la pràctica aconsellable és col·locar el prefix "data-" a cada directiva.

```
<html data-ng-app>
```

D'aquesta manera el codi validarà perfectament, ja que en HTML5 es posen crear qualsevol tipus d'atributs personalitzats que comencen per "data-" assignar-li qualsevol tipus de dades que vulguem emmagatzemar en la etiqueta.

Per no causar confusions, també podem agregar que a nivell de Javascript les directives les trobaràs nombrades amb notació "camel case", algo com ngApp. En la documentació també les trobem esmentades com camel case, però, com el HTML no és sensible a les majúscules i minúscules no té gaire sentit utilitzar aquesta notació i per això es separen les paraules de les directives per un guió "-".

Opcionalment ngApp pot contenir com a valor un mòdul d'AngularJS a carregar. Això ho veurem més endavant quant es treballi amb mòduls.

Directiva ngModel (ng-model)

La directiva ngModel informa al compilador HTML de AngularJS que s'està declarant una variable del model. Es poden utilitzar dins de camp INPUT, SELECT, TEXTAREA (o controls de formulari personalitzats).

Nota: Cal dir que no forma part del estàndard HTML sinó que és una extensió que tenim gràcies a AngularJS. S'indica amb l'atribut del HTML ng-model, assignant el nom de la variable del model que s'està declarant.

```
<input type="text" ng-model="busqueda">
```

Amb això s'està dient al framework que aquest atent a el que hagui escrit en aquest camp de text, perquè és una variable que es va utilitzar per emmagatzemar algo i perquè és important per a la aplicació.

Tècnicament, el que es fa amb ngModel és crear una propietat dins del "scope" (el model) quin valor tindrà aquell que s'escriu en el camp de text. Gràcies al "binding" quan modifiques aquest valor en el scope per mig de Javascript, també es modificarà el que hagui escrit en el camp de text.

Directiva ngInit

Aquesta directiva ens serveix per inicialitzar dades en la nostre aplicació, per mitja de l'ús d'expressions que s'avaluaran en el context actual a on hagin sigut definides. Dit d'una altre manera, ens permeten carregar contingut en el nostre model, al inicialitzar-se l'aplicació.

Així d'aquesta manera general podem crear variables en el "scope", inicialitzant-les amb valors, etc. Perquè en el moment que les vagis a necessitar estiguin carregades amb les dades que necessitis.

```
<div ng-app ng-init="miArrayDatos = [];">
```

Amb aquest exemple aconseguim que la nostre aplicació inicialitzi en el scope una dada anomenada miArrayDatos com un array buit. Però realment no tenim que prioritzar el situar la directiva ngInit dins de la mateixa etiqueta que inicialitza l'aplicació, doncs podria anar en qualsevol altre etiqueta del codi HTML. Realment, situar-la en aquesta divisió marcada amb ngApp es considerat una mala pràctica. S'ha de tenir en compte el següent al treballar amb ngInit:

El únic cas apropiat a on es tindria que emprar ngInit es en el enllaç de propietats especials de ngRepeat. Si el que desitges es inicialitzar dades en el teu model per a tota l'aplicació, el lloc apropiat seria en el controlador. A continuació veurem un exemple d'us apropiat quan fem ús de la directiva ngRepeat.

Directiva ngRepeat

Aquesta directiva ens servirà per instaurar una repetició (un bucle). Aquesta es utilitzada per repetir un conjunt d'ordres un numero de cops definit. Al implementar la directiva en el codi HTML, s'ha de indicar sobre quina estructura es te que reiterar. ngRepeat s'utilitza de forma molt habitual i ho veurem en molts exemples. De moment podem fer un paral·lelisme amb for-each en el que se reitera sobre cada un dels elements d'una col·lecció.

La etiqueta ón situes el atribut ng-repeat i tot el grup d'etiquetes anidades dins d'aquesta, funcionen con si fossin una plantilla. Al processar el compilador HTML de AngularJS, el codi HTML d'aquesta plantilla es repeteix per a cada element de la col·lecció que s'està reiterant. Dins d'aquesta plantilla tenim un context particular, que es definit en la declaració de la directiva, que equival al element actual en el bucle.

```
<p ng-repeat="llibre in mevaBibilioteca">
Estàs veient: <span>{{llibre}}</span>
```

```
</p>
```

La dada meva Biblioteca seria una dada del teu model, habitualment un array sobre el que pots reiterar, un cop per cada element. Però també podria ser un objecte i en aquest cas la reiteració és realitzarà en cada una de les seves propietats.
En lo relatiu al context propi del bucle, ens hem de fixar que dins de la reiteració podem accedir a la dada "llibre", que compte com a valor, en cada repetició, l'element actual de la col·lecció sobre la que s'està reiterant.
A continuació anem a veure un exemple de com treballa ngRepeat en conjunt amb ngInit.

```
<p ng-repeat=" llibre in mevaBibilioteca " ng-init="paso=$index;">
Elemento con id {{paso}}: <span>{{llibre}}</span>
</p>
```

La directiva ngRepeat administra una sèrie de propietats especials que pots inicialitzar per el context propi de cada repetició. Per a inicialitzar-les utilitzem la directiva ngInit indicant els noms de les variables a on anem a guardar aquestes propietats. En aquest cas estem indicant que dins de la repetició anem a mantenir una variable "paso" que tindrà el valor de $index, que equival al numero index de cada repetició. O sigui en la primera iteració el seu valor serà cero, després valdrà un i així augmentarà una unitat en cada element nou. Igual que disposem de $index, Angular ens proporciona altres propietats útils com $first (que generarà un true en cas de que sigui la primera repetició) o $last (a on indicarà true en l'ultima repetició).

Veure Ejercici2.html

Directiva ngClick

Per acabar explicarem la directiva ngClick. Com es pot veure és utilitzada per especificar un esdeveniment click. En ella posarem el codi (millor dit l'expressió) que s'ha d'executar quan es produeixi el click sobre l'element a on s'ha situat la directiva.
Habitualment al implementar un click s'invoca una funció que administri un esdeveniment, que s'escriu de forma separada al codi HTML.

```
<input type="button" value="Haz Clic" ng-click="procesarClic()">
```

Aquesta funció procesarClic() s'escriu en el controlador, factoria, etc. Seria la forma aconsellable de procedir, encara que també podries escriure expressions simples, amb un subconjunt del codi que podries escriure amb el propi Javascript. Inclòs inclou la possibilitat de escriure diverses expressions si es separa amb un punt i coma.

```
<input type="button" value="haz clic" ng-click="numero=2; otraCosa=dato " />
```

No es diferencia molt a com s'expressen els esdeveniments click en HTML mitjançant el atribut onclick, la diferència aquí és que dins de les expressions podràs accedir a les dades que tinguis en el model.

Important: Encara que tècnicament es pugui escriure expressions directament en el codi HTML, en el valor del atribut HTML, en el valor del atribut ng-click, es té que avaluar amb cura quin tipus de codi es realitza, perquè dins de la filosofia de AngularJS i la del MVC en general, no pots escriure en el teu HTML codi que serveixi per implementar la lògica de la aplicació. El codi que necessites per fer les funcionalitats de la aplicació no s'han de situar a la vista, sinó en el controlador.

Veure Exemple3.html

Exemple d'ús d'aquestes directives en AngularJS

Ara que ja hem conegut les directives, ens falta el posar-ho tot junt en pràctica amb un exercici bàsic amb AngularJS.

En aquesta aplicació tenim un camp de text per escriure un "producte" de la nostre llista de la compra. Al clicar sobre el botó, s'afegirà dins d'un array anomenat "productes" el que s'hagi escrit en el camp de text. A més trobaràs un bucle definit amb ng-repeat que reitera sobre el array "productes" mostrant tots els que s'hagin agregat.

```
<div ng-app ng-init="pensamientos = [];">
    <h1>Altavoz AngularJS</h1>
    <p>
        ¿Què hay de nuevo?
        <br />
        <input type="text" ng-model="nuevoPensamiento" />
        <input type="button" value="Agregar" ng-
click="pensamientos.push(nuevoPensamiento); nuevoPensamiento = '';" />
    </p>
    <h2>Pensamientos que has tenido</h2>
    <p ng-repeat="pensamiento in pensamientos" ng-init="paso = $index">
        Pensaste esto: {{pensamiento}} (Iteración con índice {{paso}})
    </p>
</div>
```

La dada del array "productes" es crea en el scope amb el ng-init de la primera etiqueta. En el camp de text tenim la directiva ng-model per a indicar-li que lo que s'escrigui formarà part del nostre model i s'emmagatzemarà en la variable nouProducte. Com es pot apreciar, en el ng-click se realitza un push d'aquest nou producte dins del array de productes. En ng-repeat es reitera sobre la col·lecció de productes, mostrant-los tots per pantalla, junt amb el index actual que ocupa en el array "productes".

Controladors i jerarquia de controladors

Els controladors en AngularJS son objectes que permeten desenvolupar la lògica de l'aplicació, enllaçar el àmbit, $scope, amb la vista i permet tenir un control total de les dades. Explicant-t'ho d'una altre manera, es l'encarregat de gestionar els esdeveniments.
Els controladors s'enllacen a la vista mitjançant la directiva ng-controller, encara que existeix una excepció en el tema de les rutes.
Els controladors, s'acostumen a instaurar dins dels mòduls que hem definit anteriorment.

Exemple:

```
var app = angular.module('MyApp', []);
app.controller('mainController', function($scope){
   //contenido
});
```

Com veiem, crear un controlador es força senzill, tan sols tenim que assignar-i un nom i injectar-li les dependències.
Tenim que tenir en compte, que tot controlador te un $scope associat, com a conseqüència, veurem que injectem el $scope dins del controlador. Ara be, podem injectar altres components dins del controlador, siguin natius de AngularJS o elaborats per nosaltres mateixos.
Com podem apreciar AngularJS fa ús intens de la injecció de dependències en tots els seus components. Per a injectar noves dependències, només es necessari separar-les amb comes.

```
app.controller('mainController', function($scope, servicio,
fabrica){
   //contenido
});
```

Dins de la vista, utilitzem la directiva “ng-controller” per associar el nostre controlador a la nostre vista, el controlador tenim que afegir-lo per sobre del contingut html sobre el cual desitgem que treballi.

```
<body>
   <div ng-controller="mainController">
      <h1>Hola AngularJS des de @cibernarium</h1>
   <div/>
</body>
```

Dins del àmbit a on s’ha declarat el controlador ng-controller=”mainController” tenim un $scope que permet tenir control total de les dades.

Anem a consultar el Exemple4.html i el ejercici4.js.

Ara per poder practicar una mica més, anem a realitzar un controlador que ens permeti presentar una pregunta amb les seves respostes i amb la possibilitat de que el usuari pugui respondre-la.

```
var app = angular.module('MyApp', []);
app.controller('questionController', function($scope){
   //objeto pregunta
   $scope.pregunta = {
      id: 1,
      premisa: '¿Amb quin llenguatge esta basat el Framework
AngularJS?',
      respostes: [
         {
            id: 1,
            text: 'Javascript',
            active: 'false'
         },
         {
            id: 2,
            text: 'PHP',
            active: 'false'
         },
         {
            id: 3,
            text: 'Java',
            active: 'false'
```

```
            },
            {
                id: 4,
                text: 'NodeJs',
                active: 'false'
            }
        ]
    };
});
```

Aqui ja hem creat un objecte anomenat “pregunta”, que conte la estructura bàsica, la premissa i un array de respostes, aquest objecte el mostrarem en la nostre vista de la següent manera. Veure ejercici5.html i ejercici5.js.

Mostrarem l’objecte pregunta en la vista i per mostrar les claus fem us de la directiva ng-repeat que ens permet reiterar el array d’objectes “respostes”, dins de cada clau mostrarem la variable “text” que conte el text de la resposta.
Ara agregarem funcionalitat per marcar una clau, per lo cual necessitarem la directiva ng-click que l’afegirem dins de cada element de la llista.

```
<li ng-repeat="resposta in pregunta.respostes" ng-click="marcar()">
        {{resposta.text}}
    </li>
```

Ara tenim que definir el mètode dins del nostre controlador, que ens permetrà modificar l’estat de la nostre clau a “true” que identifica quin element ha sigut marcat.

```
$scope.marcar = function(){
    angular.forEach($scope.pregunta.respostes, function(value, key) {
        value.active = false;
    });
    this.resposta.active = true;
};
```

En aquest pas, el primer que hem realitzat, es recórrer el array de claus i hem modificat el estat de tots com a false, després al element escollit li canviem el estat a true, això ens permetrà saber la clau que es troba marcada, alhora de validar les respostes.
Finalment tindrem que afegir la funcionalitat corresponent per el nostre botó “Respondre” que ens permetrà afegir un objecte al vector de respostes, aquest gestionarà el id de la pregunta com la id de la clau marcada.

```
$scope.respuestas = [];
$scope.responder = function(){
    angular.forEach($scope.pregunta.respuestas, function(respuesta,
i) {
        if (respuesta.active)
            $scope.respuestas.push({ id:$scope.id, key:respuesta.id });
    });
};
```

Fins ara, hem afegit el mètode que ens permet recórrer el array de respostes de la nostre pregunta i identificar quina clau va ser seleccionada, per a poder agregar-lo al array “respostes” com un objecte. El que ens restarà ara, es el afegir a la vista el mètode mitjançant la directiva ng-click, en el boto “Respondre”.

Podem veure el exemple complert en els arxius ejercici6.html i ejercici6.js.

Consultar ejercici7.html, ejercici7.js i ejercici7.css com alternativa.

Jerarquia de controladors

A més de tenir en compte la jerarquia de com hem de situar els controladors dins dels mòduls, hem de contemplar també la possibilitat que disposem, de situar controladors per sota d’altres (fent funció de fills i heretant funcionalitats).
Des de la directiva ng-controller, podem crear nous $scope fills, que interactuïn amb els $scopes de qualsevol altre controlador. El $scope de cada controlador te accés a les propietats i mètodes que han sigut definits en altres controladors a més alt nivell. Perquè puguem entendre-ho millor, consulteu el ejercici8.html, ejercici8.js i ejercici8.css.
Since the ng-controllerdirective creates a new child scope, we get a hierarchy of scopes that inherit from each other. The $scope that each Controller receives will have access to properties and methods defined by Controllers higher up the hierarchy. See Understanding Scopes for more information about scope inheritance.