Cours introduction à Angular JS.

Houssam KOURDACHE.

Ingénieur étude et développement Web et Mobile.

- Créé en : 2009, par : **Miško Hever.**
- C'est un Framework MVC et MVW (**Model view Whatever**).
- Une logique métier englobée dans des contrôleurs et directives.
- une application Angular = collection de contrôleurs et de directives.
- Dans le code Html : on indique à quelle application js on veut relier notre DOM Html ; et cela se fait à l'aide de l'attribut html : « ng-app ».
    - « ng-app » : la déclaration de l'application ang-js à appliquer à la page, ne se déclare qu'une seule fois dans une page html.

Un premier exemple :

```
<html ng-app=''myFistApp''>

</Html>
```

```
Var myJsApp=angular.module ('myFisrtApp'), [])
```

I. Installation Angular JS

a. Installation de node.js :

   Soit en téléchargeant node.js sur http://nodejs.org/download/, soit par ligne de commande par : « npm install node.js »

b. Lien téléchargement Angular JS :

   - On peut soit le télécharger directement sur : https://angularjs.org/
   - Soit en le téléchargeant avec **bower** (http://bower.io/), l'injecteur de dépendances pour le front-end, générant toutes les dépendances possibles faisant parties du registre bower comme par exemple jquery, modernizr …Etc. Pour l'installer, il faut installer tout d'abord npm et node.js et puis l'installer via la commande : « npm install –g bower » .
   - Sur le site angular, vous avez téléchargé angualar.js.
   - Copiez-le et collez-le dans le dossier JS de votre projet web.
   - Puis l'inclure dans vos pages html comme ceci :

     ```
     <script type='text/javascript' src='js/angular.js'></script>
     ```

c. Le téléchargement d'Angular JS peut se faire également via les IDE en téléchargeant le plugin Angular ; comme par exemple l'IDE WebStorm ou encore SublimeText.

d. Dans cet exemple nous allons utiliser WebStorm V8.

e. Installation du plugin Angular sur WebStorm V8 :

   - Créer un nouveau projet (spécifier dans le champ 'project type' l'item empty).

- Aller dans « Settings », puis « Javascript » et « libraries » ; sur la liste de droite, si angular en fait pas partie, cliquer sur « download » puis « TypeScript stubs » ; sélectionnez angularjs et cliquez sur « download & install ».

II. La structure d'un projet Angular :

Il y a certaines règles à respecter facilitant l'utilisation et le repérage aux développeurs :
La structure que je conseille est la suivante que j'ai trouvée sur ce site :
http://scotch.io/tutorials/javascript/angularjs-best-practices-directory-structure

```
app/
----- shared/   // acts as reusable components or partials of our site
---------- sidebar/
--------------- sidebarDirective.js
--------------- sidebarView.html
---------- article/
--------------- articleDirective.js
--------------- articleView.html
----- components/   // each component is treated as a mini Angular app
---------- home/
--------------- homeController.js
--------------- homeService.js
--------------- homeView.html
---------- blog/
--------------- blogController.js
--------------- blogService.js
--------------- blogView.html
----- app.module.js
----- app.routes.js
assets/
----- img/      // Images and icons for your app
----- css/      // All styles and style related files (SCSS or LESS files)
----- js/       // JavaScript files written for your app that are not for angular
----- libs/     // Third-party libraries such as jQuery, Moment, Underscore, etc.
index.html
```

III. Générer la structure de votre projet Angular avec YEOMAN :

a. Utiliser YEOMAN pour générer la structure de votre projet Angular
b. Ajouter des contrôleurs, des routes et autres en ligne de commande
c. Installation de YEOMAN : http://yeoman.io/
   - 1 : npm install –g yo
   - 2 : npm install – g generator-angular
   - 3 : créer un répertoire pour votre projet web angular : mkdir projet
   - 4 : naviguer vers votre projet : cd projet
   - 5 : configurer votre répertoire pour le transformer en projet Angular en générant sa structure, à l'aide de : yo angular
   - 6 : puis importer votre projet sur SublimeText ou WebStorm.

a. **Générer des composants Angular :**
   a. Générer un module : suffit de « yo angular »
   b. Générer une route (tout en créant un contrôleur et la vue qui lui est associée) : yo angular:route myroute.

Cela va générer un fichier : myroute.js et myroute.html

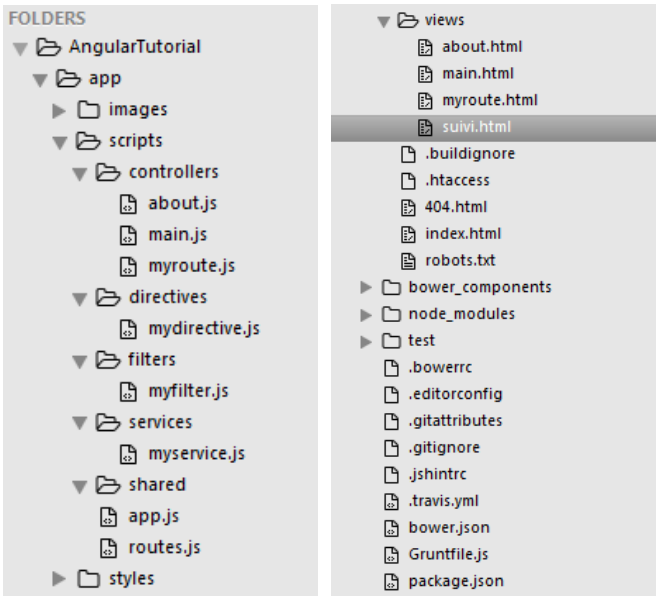
c. Définir une route tout en rajoutant l'uri de la route au fichier app.js : yo angular :route myroute - - uri =my/route
d. Générer un contrôleur : yo angular :controller user. Cela va créer un fichier user.js initiant le contrôleur, dans le répertoire « controllers »
e. Générer une directive : yo angular :directive myDirective. La directive générée sera ajouté au répertoire /directives.
f. Générer un filtre : yo angular :filter myFilter. Le fichier myFilter.js est ajouté dans le dossier /filters.
g. Générer un service : yo angular :service myService. Cela va générer un fichier myService.js dans le répertoire /services.
h. Générer une vue : yo angular :view user. Le fichier user.html généré sera ajouté au répertoire /views.

Pour les autres composants Angular, vous pouvez suivre ce tutoriel :

https://github.com/yeoman/generator-angular

http://yeoman.io/authoring/

Voici ce à quoi ressemble mon projet créé avec YOEMAN (yo angular) :



Avantage :

a. Générer d'un coup les composants des vues et leurs contrôleurs
b. Gagner en productivité
c. Avec WebStorm, vous allez directement vos fichier et composants dans le frame de la console : plus besoin de switcher entre les fenêtres (parfois ça peut déranger).
d. Vous n'aurez plus à créer la structure de votre projet à la main : faites-le en une seule ligne de commande☺.

IV. Chapitres

1. Contrôleur :

Un contrôleur est relié à une application ang-js, permet d'écrire son code et logique. Il est relié à une partie du DOM.

- Pour créer une application, on instancie :

```
Var myJSApp = angular.module ( ' firstAngJSApp ' ,  [] ) ;

On crée un contrôleur :

myJSApp.controller ( 'firstControler' , function ( $Scope ) {

}) ;
```

- Le rattachement du contrôleur au DOM html se fait :

On spécifie la division du code pour laquelle on veut appliquer le contrôleur :

```
<div ng-controller='firstControler'>

</div>
```

2. **$Scope** : la portée, le modèle qui contiendra les données, fonctions et variables qui peuvent être utilisés dans l'application. Quand le scope du contrôleur est modifié, le scope utilisé dans le DOM Html l'est également. Et vice-versa. On parle ici de **binding bi-directionnel.**
Par exemple !

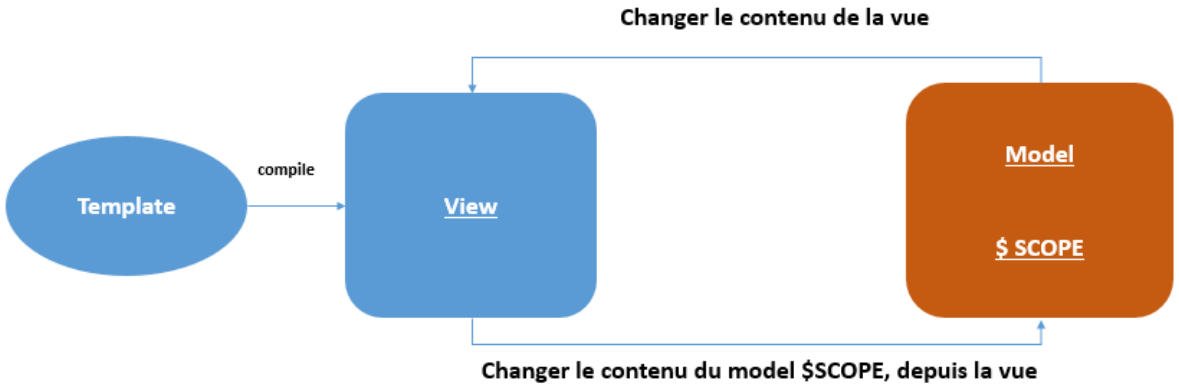
```
Déclarer une variables avec une valeur et l'encapsuler dans un $scope :

myJSApp.controller('firstControler', function($scope){

        $scope.nom='kiriko' ;

}) ;
```

3. Binding Bi-directionnel :



Donc au sein de notre contrôleur créé précédement, on peut écrire au sein de notre :

```
<div ng-controller='firstControler'>
    <h1>    {{ nom }}    </h1>
</div>
```

Ce code ci-dessus, nous affichera « Kiriko ».
Par conséquent la modification de la variable dans le contrôleur modifiera le contenu affiché.

Supposant maintenant que l'on veuille changer cette valeur à l'aide d'un input (ce qui fait que l'on touche à la partie vue et non pas au JS). Par exemple :

```
<div ng-controller='firstControler'>
    <input type='text' ng-model='nom' />
    <h1>  {{ nom }}  </h1>
</div>
```

L'attribut ng-model, sert à associer la valeur de cette input avec celle contenu dans le model $scope. Ainsi la valeur entrée dans le input modifiera par conséquent la valeur du $scope.nom.

Mapping Bi-directionnel = synchronisation entre le modèle et la vue.

4. **Les Directives :**

Directive : attribut ajouté dans le DOM Html est précédé par un préfixe « ng ».
Cela permet de rendre le code extensible et modulable.
Angular JS a déjà prédéfinies un certain nombre de directives comme par exemple : ng-model employé précédemment, ou encore « ng-show » qui permet d'afficher ou pas un élément selon une condition.

L'utilisation des directives n'handicape pas la récupération des valeurs de ses directives pour les utiliser au sein d'un code Jquery, par exmple.

Les directives est un atout énorme, car permettent de coder proprement des dépendances entre le code et le Html ; de les partager et les réutiliser dans d'autres projets.
On peut les emmagasiner et les réutiliser en piochant dedans.

Donc, une directive est la liaison entre le contrôleur et le DOM, permettant d'identifier un élément html et y appliquer une logique définie dans le DOM.

5. **Les services :**

Sont des singletons injectés dans l'application Angular grâce à l'injection de dépendances.
L'injection de dépendance est un mécanisme permettant d'implémenter l'**inversion de contrôle.**
L'inversion de contrôle est : « ne nous appelez pas, c'est nous qui vous appellerons », ce principe a lieu entre le Framework et l'application. Ce n'est plus l'application qui gère les appels au Framework, mais ce dernier à l'application.

Angular a prédéfini ses services et sont tous préfixé par $.
Ces services peuvent avoir plusieurs utilités comme par exemple, le service « $http ». Ce dernier permet d'exécuter des requêtes AJAX.
Il y a également le service « $compile » qui permet de digérer du code HTML fraichement ajouté et qu'il parse les directives.

Les services sont injectés dans des contrôleurs, dans des directives ou même dans des d'autres services grâce à l'injection de dépendances.

Leur utilité est diverse : un retour de variable initialisé ailleurs, traitement complexe avec des fonctions. Ils permettent un partage de fonctions et de variables entre les contrôleurs, les directives et les services eux-mêmes.

Le contenu d'un service :

Dans le service faut écrire l'essentiel du code métier de l'application sous forme de services. Si l'on excepte les directives, filtres et que l'on limite l'utilisation des contrôleurs aux simples publications dans le scope les variables et fonctions utilisés dans le template.

…… A détailler l'usage des services……

6. **Les routes :**
   - Angular fournit un mécanisme de routage, qui permet de définir pour un certain pattern d'URL, pour une certaine page à charger en AJAX et dont le résultat doit être ajouté dans une portion de DOM en utilisation la directive « ng-view ».
   - On peut également y rattacher un contrôleur :

```
myJSApp.config (
        [
        ' $routeProvider  ',
                Function ( $routeProvider ) {

                        $ routeProvider.when ('/test',  {
                                templateUrl : 'partials/test',
                                controller :  'MyTestController'
                        )} ;
                }
        ]
) ;
```

Dans cet exemple, on modifie la configuration de notre application en ajoutant $routeProvider dans le tableau. Quand l'url est « /test », on récupère le contenu de la page qui se trouve dans « partials/test » et cette page sera rattaché au contrôleur spécifié, i-e MyTestController.

Par la suite spécifier où la page sera affichée en utilisant la directive « ng-view » :

```
<div ng-view>

</div>
```