

Concepts de base AngularJS

-- Configuration de l'IDE

Avec Eclipse : Installer AngularJS Eclipse Plugin (AngularJS Eclipse)

Avec PhpStorm ou WebStorm: voir PhpStorm AngularJS

-- Téléchargement

Angular JS download

-- Intégration

Si le fichier est stocké en local :

```
<script src="js/angular.min.js"></script>
</head>
```

ou

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.6.1/angular.min.js"></scri
pt>
</head>
```

Quoi qu'en disent certains, il est préférable d'intégrer AngularJS le plus tôt possible dans la page, pour ne pas voir apparaître la page avant l'exécution des scripts Angular présents (même avec la directive ng-cloack).

-- Premiers pas

Traditionnel Hello world :

```
<!DOCTYPE html>
```

```
<html lang="en" ng-app="HelloApp">
<head>
    <meta charset="UTF-8">
    <title>Hello</title>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.6.1/angular.min.js"></scri
pt>
</head>
<body>
    <div ng-controller="HelloController as hello">
        <br> Message : {{hello.message}}
    </div>

    <script>
angular.module("HelloApp",[]).controller("HelloController",function(){this.message=
"Hello world";});
    </script>
</body>
</html>
```

- **ng-app** ou **data-ng-app** définit l'application (HelloApp)
- **ng-controller** ou **data-ng-controller** définit la zone d'action (la div dans l'exemple) d'un contrôleur
- Les accolades **{{ et }}** définissent une expression et mettent en oeuvre le data-binding entre vue (HTML) et contrôleur (fonction javascript)

Hello world modifié

On ajoute de quoi modifier le message via un champ input, pour illustrer la notion de data-binding (avec la directive **ng-model**)

```
<!DOCTYPE html>
<html lang="en" ng-app="HelloApp">
<head>
    <meta charset="UTF-8">
    <title>Hello</title>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.6.1/angular.min.js"></scri
pt>
</head>
<body>
    <div ng-controller="HelloController as hello">
        <input type="text" ng-model="hello.message">
        <br> Message : {{hello.message}}
    </div>

    <script>
angular.module("HelloApp",[]).controller("HelloController",function(){this.message=
"Hello world";});
    </script>
</body>
</html>
```

Afficher le résultat dans le navigateur, modifiez la valeur du champ input, et constatez les changements...

-- Principaux concepts

-- Directives

Les directives AngularJs permettent de marquer les éléments du DOM (attributs, noms, commentaires, classes CSS...) de façon à indiquer au compilateur AngularJs ($compile) d'associer un comportement à ces éléments DOM ou à les modifier.

La “compilation” consiste à attacher des listeners au éléments HTML pour les rendre interactifs.

```
<div ng-app="" ng-init="prenom='Maurice'">

<p>Prénom : <input type="text" ng-model="prenom"></p>
<p>Vous avez saisi : {{ prenom }}</p>

</div>
```

Liste des directives de l'exemple :

- **ng-app** initialise une application AngularJS.
- **ng-init** initialise les données d'une application.
- **ng-model** associe à la valeur de contrôles HTML (input, select, textarea) les données de l'application (data-binding).

```
<div ng-init="clients=[
{id:1,nom:'Heidi',pays:'Norvège'},
{id:2,nom:'Hans',pays:'Allemagne'},
{id:3,nom:'Maurice',pays:'France'}]">
    <div ng-repeat="client in clients">
        <label><input type="radio" ng-value="client.id" name="client">{{ client.nom
+ ', ' + client.pays }}</label>
    </div>
</div>
```

Liste des directives de l'exemple :

- **ng-init** initialise les données (tableau d'objets).
- **ng-repeat** répète une séquence HTML en exécutant un foreach sur le tableau **clients**.
- **ng-value** définit la valeur de chaque case d'option, à partir de l'id de chaque client

voir https://docs.angularjs.org/guide/directive

-- Expressions

Les expressions permettent de mettre en oeuvre le data-binding, comme la directive ng-model, mais ne se limitent pas aux éléments HTML de saisie. Définies par les balises {{uneExpression}}, elles sont directement insérées dans la page HTML.

Les expressions angular peuvent contenir des variables, des opérateurs, ou des expressions littérales. Elles n'ont pas accès à l'ensemble des variables d'une page (window, document), mais uniquement à une portée locale : le $scope

```
<p>Vous avez saisi : {{ prenom }}</p>
```

```
<span>
  1+2={{1+2}}
</span>
```

Voir: https://docs.angularjs.org/guide/expression

-- Filtres

Les filtres AJs permettent de formater/modifier l'affichage d'une expression :

```
<p>Vous avez saisi : {{ prenom | uppercase}}</p>
```

```
<span>
  1+2={{1+2 | currency}}
</span>
```

-- Modules ou application

Un module est un conteneur d'éléments pour une application : services, directives, filtres, contrôleurs. Chaque module d'une application correspond logiquement à une fonctionnalité.

Déclaration d'un module dans un fichier js :

```
var myAppModule = angular.module('myApp', []);
```

Utilisation dans une vue HTML via la directive **ng-app** :

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

voir https://docs.angularjs.org/guide/module

-- Contrôleurs



Les contrôleurs permettent d'associer une logique métier aux vues.
Un contrôleur est défini par un constructeur permettant d'augmenter la portée (le scope) d'AngularJs, il peut utiliser des services, par injection de dépendance.

Exemple de contrôleur dans un module :

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

myApp.controller('CarreController', ['$scope', function($scope) {
    $scope.value=0;
    $scope.carre = function(value) { return value * value; };
}]);
```

L'injection de dépendance est utilisée pour spécifier la dépendance du contrôleur au service Angular **$scope**, qui définit la portée du contrôleur.
La vue associée est la suivante :

```
<fieldset data-ng-app="myApp" data-ng-controller="CarreController">
    <legend>Carré</legend>
    <input type="text" data-ng-model="value">
    <div>Carré : {{carre(value)}}</div>
</fieldset>
```

Elle fait référence à la donnée du modèle **value**, initialisée dans le contrôleur, et doublement référencée dans la vue :

- avec la directive **ng-model** du champ input
- dans l'expression **{{carre(value)}}**

voir https://docs.angularjs.org/guide/controller

ControllerAs et $scope

L'utilisation de controllerAs permet d'éviter l'injection de $scope :

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

myApp.controller('CarreController', [function() {
    this.value=0;
    this.carre = function(value) { return value * value; };
}]);
```

La vue associée est la suivante :

```
<fieldset data-ng-app="myApp" data-ng-controller="CarreController as ctrl">
    <legend>Carré</legend>
    <input type="text" data-ng-model="ctrl.value">
    <div>Carré : {{ctrl.carre(ctrl.value)}}</div>
</fieldset>
```

-- $scope

Le scope définit une portée, notion rattachée aux contrôleurs, ou à certaines directives (ng-repeat par exemple).
Pour en comprendre le fonctionnement, implémenter l'exemple suivant :

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>scope</title>
<style>
.ng-scope {
    border-style: dotted solid;
    border-width: thin;
    border-color: red;
    margin: 1em;
    padding: 1em;
}

label {
    color: red;
}
</style>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.6.1/angular.min.js"></scri
pt>
<script src="app.js"></script>
</head>
<body>

    <div data-ng-app="ScopeModule">
        <label for="root">Scope racine</label> <input type="text" id="root"
            placeholder="Entrez du texte" ng-model="message">
        <div data-ng-controller="ScopeController">
            <label for="ctrl">Scope enfant</label> <input type="text" id="ctrl"
                placeholder="Entrez du texte" ng-model="message">
        </div>
    </div>
</body>
</html>
```

```
angular.module('ScopeModule', []);
angular.module("ScopeModule").controller("ScopeController", [ "$scope",
function($scope) {
} ]);
```

Testez la vue v_scope.html :

1. Entrez une valeur dans la première zone de texte (scope parent), remarquez la valeur de la seconde zone (scope enfant)
2. Modifiez la valeur de la seconde zone, remarquez la valeur de la première
3. Rechargez la page, entrez une valeur dans la seconde zone, entrez une valeur dans la première zone : que remarquez vous ?
4. Expliquez la raison de ce fonctionnement
5. Ajoutez dans le contrôleur le code permettant de rendre les 2 zones toujours dépendantes.

-- Services

Les services peuvent être utilisés pour structurer une application et partager du code entre ses différents modules.
Un service est un objet (singleton) chargé à la demande (lazy loading).

AngularJs propose ses propres services, préfixés du $: $scope, $window... utilisables par injection de dépendance.
voir https://docs.angularjs.org/api/ng/service

Exemple d'utilisation du service $window

Injection du service **$window** dans un contrôleur **GotoController** de notre module **myApp** :
Le contrôleur permet d'accéder à une URL.

```
myApp.controller('GotoController', [ '$scope','$window',function($scope,win) {
    $scope.location="http://www.google.fr";
    $scope.go=function(){win.location=$scope.location;};
} ]);
```

```
<!DOCTYPE html>
<html data-ng-app="myApp">
<head>
<meta charset="UTF-8">
<title></title>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.6.1/angular.min.js"></scri
pt>
    <script src="app/app.js"></script>
</head>
<body data-ng-controller="GotoController">
<input type="text" data-ng-model="location">
<input type="button" data-ng-click="go()" value="Go"/>
```

```
</body>
</html>
```

Utilisation du service ngCookies

Le service **ngCookie** n'est pas intégré par défaut à angular, il est donc nécessaire d'inclure le fichier js correspondant :

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.6.1/angular-cookies.min.js
"></script>
```

Il faut ensuite intégrer ngCookies en tant que dépendance dans l'application :

```
var App = angular.module('appName', ['ngCookies']);
```

nbCookies peut ensuite être injecté dans les contrôleur le nécessitant :

```
app.controller("ctrlName",["$cookies",function($cookies){
    // Retrieving a cookie
  var favoriteCookie = $cookies.get('myFavorite');
  // Setting a cookie
  $cookies.put('myFavorite', 'oatmeal');
}]);
```

Création d'un service

Le service créé : **browser**, va permettre de détecter le navigateur utilisé :

```
myApp.service('browser', [ '$window', function($window) {

    this.get= function() {

        var userAgent = $window.navigator.userAgent;
        var browsers = {
            chrome : /chrome/i,
            safari : /safari/i,
            firefox : /firefox/i,
            ie : /internet explorer|.net/i
        };

        for ( var key in browsers) {
            if (browsers[key].test(userAgent)) {
                return key;
            }
```

```
        };
        return 'Navigateur inconnu';
    };

} ]);
```

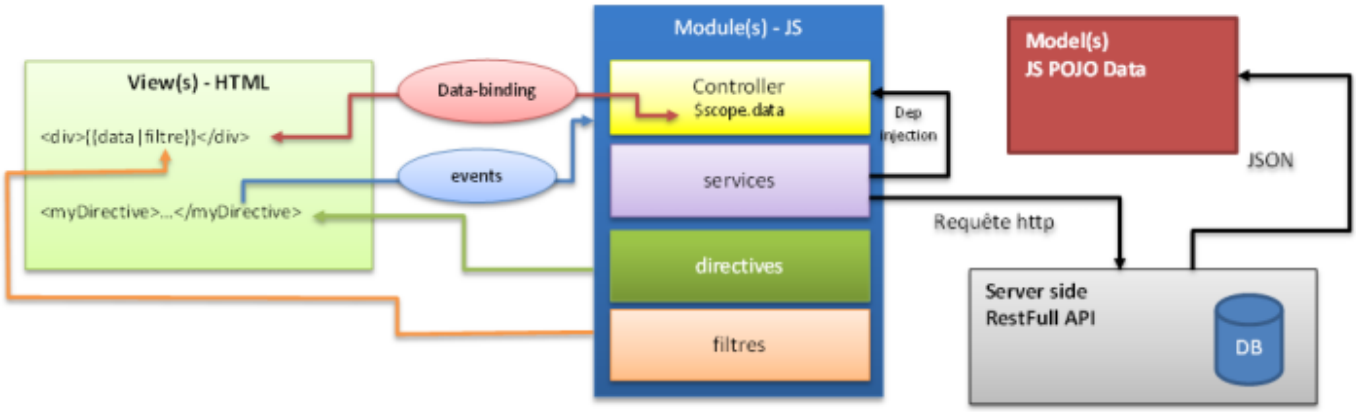
Création d'un contrôleur utilisant le service **browser**, par injection de dépendance :

```
myApp.controller('BrowserController', [ '$scope','browser',function($scope,browser)
{
    $scope.navigateur= browser.get();
} ]);
```

Utilisation du contrôleur dans une vue :

```
<div ng-controller="BrowserController">{{navigateur}}</div>
```

--Schéma général



From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/slam4/richclient/angularjs/bases?rev=1485216557

Last update: **2019/08/31 14:40**