



Introduction à AngularJS

Encore un nouveau framework Javascript dans ce qui est maintenant devenu une jungle. Même restreint aux frameworks se basant principalement sur le modèle de conception MVC, il en existe déjà un nombre important : Backbone, Ember, Knockout... Pourtant AngularJS se démarque ostensiblement.

Misko Hevery, son concepteur, préfère ne pas parler de framework mais plus d'un concept de création de web app en JavaScript. AngularJS s'inspire principalement des bonnes pratiques de conception d'application : data-binding, test, couplage-lâche, injection de dépendances, service... C'est ce qui fait toute l'originalité de ce framework. Afin que vous puissiez découvrir par vous-même la puissance et la simplicité d'AngularJS, nous vous proposons de développer une application web de type carnet d'adresses et ainsi parcourir ensemble toutes ses notions clés. Partons du code HTML suivant affichant une liste de 3 contacts :

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="charset=utf-8">
<title>Mon carnet d'adresses</title>
</head>
<body>
<div>
<h2>Mon carnet d'adresses (3 contacts)</h2>
<ul>
<li>
<h3>Robert Hudson</h3>
<p>Groupe : Famille</p>
<p>Mobile : 0645271845</p>
<p>Adresse : 50 rue Victor Hugo 75004 Paris</p>
</li>
<li>
<h3>Pierre Samal</h3>
<p>Groupe : Ami</p>
<p>Mobile : 0642154528</p>
<p>Adresse : 30 avenue des Coquelicots 69001 Lyon</p>
</li>
<li>
<h3>Virginie Dupont</h3>
<p>Groupe : Travail</p>
<p>Mobile : 0655554120</p>
<p>Adresse : 5 rue des Pierres 51100 Reims</p>
</li>
</ul>
</div>
</body>
</html>
```

[Fig.1]

Afficher une simple page HTML avec AngularJS

L'idée maintenant est d'afficher la même page mais en utilisant AngularJS. Commençons par charger le script AngularJS dans la partie <head> de votre page HTML :

```
<script type="text/javascript">
src="http://code.angularjs.org/1.0.1/angular-1.0.1.js"></script>
```

Une fois chargé, le script va alors rechercher sur quelle partie de votre code HTML il doit être actif. Dans AngularJS la notion de directive permet d'enrichir l'arbre DOM HTML pour y insérer des informations explicites liées à l'application.

La directive ng-app permet d'initialiser le compilateur AngularJS, elle doit être placée dans la balise racine de votre application. La plupart du temps ce sera <html> tel que <html ng-app>. AngularJS sera alors actif sur la portion du code délimitée par la balise <html>, définissant ainsi le périmètre de notre module principal. Pour AngularJS, le HTML (donc l'arbre DOM) représente la vue. Le code s'écrit dans la page HTML directement.

```
<body>
<div ng-controller="ContactCtrl">
<h2>Mon carnet d'adresses ({{contacts.length}} contacts)</h2>
<ul>
<li ng-repeat="contact in contacts">
<h3>{{contact.nom}}</h3>
<p>Groupe : {{contact.groupe}}</p>
<p>Mobile : {{contact.mobile}}</p>
<p>Adresse : {{contact.adresse}}</p>
</li>
</ul>
</div>
</body>
```

Le contrôleur se déclare via la directive ng-controller, avec en paramètre la référence à un objet JavaScript. Cet objet matérialisera le lien entre vue et modèle au travers du "scope". Le modèle est ici représenté par la variable contacts.

```
function ContactListCtrl($scope) {
$scope.contacts = [
{«nom»:»Robert Hudson»,
«groupe»:»Famille»,
«mobile»:»0645271845»,
«adresse»:»50 rue Victor Hugo 75004 Paris»},
{«nom»:»Pierre Samal»,
«groupe»:»Ami»,
«mobile»:»0642154528»,
«adresse»:»30 avenue des Coquelicots 69001 Lyon»},
{«nom»:»Virginie Dupont»,
«groupe»:»Travail»,
«mobile»:»0655554120»,
«adresse»:»5 rue des Pierres 51100 Reims»}]}
```



Détaillons maintenant le contenu de la vue:

- la directive `ng-repeat` reproduit un template (code situé entre l'élément sur lequel la directive est attachée) autant de fois qu'il y a d'éléments dans une collection donnée. Dans notre exemple, nous affichons un élément `` contenant le nom, le groupe, le numéro de mobile et l'adresse pour chaque contact.
- la notation double accolade `{{expression}}` demande à AngularJS d'évaluer une expression et de l'insérer dans le DOM. Cette expression peut-être :
 - une opération : `{{1+1}}` sera évaluée comme 2
 - une propriété du scope : `{{contact.name}}` pourra être évaluée comme "Dupont"

AngularJS ne se contente pas seulement d'une simple insertion dans le DOM puisqu'il va continuellement mettre à jour ce binding à chaque fois que l'expression sera mise à jour elle-même, c'est le two-way binding [Fig.2].

Dans notre cas, par exemple, `{{contacts.length}}` affichera le nombre de contacts de manière dynamique au fur et à mesure que cette liste augmente ou diminue.

Il est important de noter qu'une expression est toujours évaluée par rapport à ce scope. Ce scope est délimité par la directive `ng-controller`.

- La directive `ng-controller="ContactListCtrl"` dans la vue fait référence au contrôleur, qui est un objet JavaScript nommé "ContactListCtrl" et délimite le scope de celui-ci. Le contrôleur définit les comportements et associe le modèle au scope. Dans notre cas, une collection de contacts est affectée à une propriété du scope. La vue pourra donc accéder à cette propriété et l'évaluer grâce à la notation `{{expression}}`.

Il peut y avoir plusieurs contrôleurs sur une même page HTML. Les scopes s'imbriquent de manière hiérarchique et héritent des propriétés de leur scope parent.

> Navigation et routage

Il nous faut maintenant créer une vue pour éditer les contacts existants et en créer de nouveau. Pour ne pas surcharger notre code HTML nous allons extraire les différentes vues.

Le code de chaque vue est placé dans un fichier HTML dédié

- `edit.html` pour l'édition

```
<form name=>editForm</form> ng-submit=>saveContact()</ng>
<label>Nom :</label>
<input type=>text</input> name=>nom</name> ng-model=>contact.nom</ng>

<label>Groupe:</label>
<select ng-model=>contact.groupe</select> name=>groupe</ng>
```

Fig.1

Mon carnet d'adresses (3 contacts)

Robert Hudson	Groupe : Famille
Mobile : 0645271845	Adresse : 50 rue Victor Hugo 75004 Paris
Pierre Samal	Groupe : Ami
Mobile : 0642156321	Adresse : 30 Avenue des Champs-Elysées 75001 Lyon
Virginie Dupont	Groupe : Travail
Mobile : 0699584220	Adresse : 2 rue des Fossés 51100 Reims

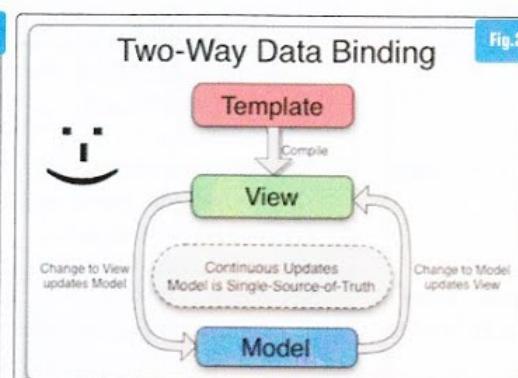


Fig.3

Mon carnet d'adresses (3 contacts)

A screenshot of a web application showing a contact form. The form includes fields for Nom, Groupe, Mobile, and Adresse, each with a placeholder value from Fig.1. Below the form are buttons for Retour and Enregistrer.

```
<option value=>Ami</option>
<option value=>Famille</option>
<option value=>Travail</option>
</select>
```

```
<label>Mobile: </label>
<input type=>text</input> name=>mobile</name> ng-model=>contact.mobile</ng>

<label>Adresse: </label>
<input type=>text</input> name=>adresse</name> ng-model=>contact.adresse</ng>

<a ng-href=>#/list</a> Retour
<input type=>submit</input> ng-click=>saveContact()</input> class=>Enregistrer</a>
</form>
```

- `list.html` pour la liste des contacts

```
<ul>
<li ng-repeat=>contact in contacts</li>
  <h3>{{contact.nom}}</h3>
  <p>Groupe : {{contact.groupe}}</p>
  <p>Mobile : {{contact.mobile}}</p>
  <p>Adresse : {{contact.adresse}}</p>
</li>
</ul>
```

[Fig.3]

- `index.html` pour la vue principale : notez l'arrivée de la directive `ng-view` qui va gérer l'affichage des 2 vues précédentes :

```
<div id=>contacts-panel</div> ng-controller=>ContactCtrl</ng>
<h2>
  Mon carnet d'adresses ({{contacts.length}}) contacts
  <a ng-href=>#/edit</a> Ajouter</a>
</h2>
<div ng-view></div> ...
</div>
</div>
</body>
```

Pour naviguer d'une vue à l'autre il est nécessaire d'utiliser la notion de routes. Principe déjà bien connu dans les frameworks web de tous types, il consiste à utiliser l'URL comme référence à sa page. Avec AngularJS cela se traduira par le code suivant :

```
angular.module('myApp', [])
.config(['$routeProvider', function($routeProvider) {
  $routeProvider.when('/edit/:contactId', {
```

```

templateUrl: 'partials/edit.html',
controller: ContactEditCtrl
});
$routeProvider.when('/list', {
  templateUrl: 'partials/list.html',
  controller: ContactListCtrl
});
$routeProvider.otherwise({ redirectTo: '/list' });
})
);

```

Deux routes sont ici définies :

- `/edit/:contactId` qui pointe vers la vue edit.html et permet de passer un paramètre (`contactId`), associé à un contrôleur `ContactEditCtrl`
- `/list` qui pointe vers list.html associé à un contrôleur `ContactListCtrl`

Pour utiliser ces routes il suffit d'y faire référence via l'attribut `href` ou la directive `ng-href` qui nous permettra de passer un paramètre. Dans notre liste de contact, nous avons ajouté un lien permettant d'édition chaque contact.

```

<ul>
<li ng-repeat="contact in contacts">
  <h3>{{contact.nom}}</h3>
  <p>Groupe : {{contact.groupe}}</p>
  <p>Mobile : {{contact.mobile}}</p>
  <p>Adresse : {{contact.adresse}}</p>
  <a ng-href="#/edit/{{$index}}">Editer</a>
</li>
</ul>

```

NB : Ici `$index` permet de renvoyer la valeur de l'itération du tableau.

```

<option value="Travail">Travail</option>
</select>
</div>
</div>
<div class="control-group ng-class="{error: editForm.mobile.$invalid}">
  <label class="control-label">Mobile: </label>
  <div class="controls">
    <input type="text" name="mobile" ng-model="contact.mobile" ng-maxlength="10" ng-pattern="/^[0-9]+$/">
    <p class="help-inline" ng-show="editForm.mobile.$errormaxlength">
      Ce numéro contient trop de chiffres.
    </p>
    <p class="help-inline" ng-show="editForm.mobile.$error.pattern">
      Le numéro ne doit pas contenir de lettres.
    </p>
  </div>
</div>
<div class="control-group">
  <label class="control-label">Adresse: </label>
  <div class="controls">
    <input type="text" name="adresse" ng-model="contact.adresse">
  </div>
</div>
<div class="form-actions">
  <a class="btn" ng-href="#/list">Retour</a>
  <input type="submit" class="btn btn-success" ng-disabled="editForm.$invalid" value="Enregistrer"/>
</div>
</form>

```

Parcourons ensemble le code :

- La balise `<form>` n'est pas la balise HTML mais une directive AngularJS qui permet de lier le formulaire à un objet JavaScript `formController`. Ce dernier permet de contrôler que tous les champs de saisie du formulaire sont dans un état valide.
- Pour ajouter une validation à un champ de saisie, AngularJS propose quelques directives de validation telles que :
 - `required` : qui invalide le champ de saisie si aucun texte n'est saisi
 - `ng-maxlength="10"` : qui invalide le champ si le nombre de caractère est supérieur 10
 - `ng-pattern="regex"` : qui invalide le champ si le contenu ne correspond pas à l'expression régulière passée en paramètre
- L'attribut `name` de la directive `<form>` est important puisqu'il vous permet d'accéder aux attributs correspondant à l'état du formulaire comme `$invalid` mais également aux champs de saisie qui composent le formulaire. Par exemple, `editForm.mobile` désigne le champ de saisie du numéro de mobile. De même, vous pouvez accéder à l'état de chaque champ de saisie pour savoir s'il est valide ou non. Ainsi, `editForm.mobile.$error.pattern` vaut true si le champ de saisie du mobile contient des lettres.
- Grâce à la directive `ng-show` qui affiche son élément si une condition est remplie et en évaluant l'état de chaque composant, un message s'affiche dans le cas où le champ de saisie est invalide.
- Enfin la directive `ng-disable` désactive le bouton de soumission tant que la condition en paramètre est vrai. En l'occurrence, ici, il s'agit

de `editForm.$invalid` qui est vrai si un des éléments du formulaire est invalide [Fig.4].

> Directives

Pour introduire l'une des fonctionnalités les plus pertinentes, créons un composant contact dont l'objectif sera d'afficher les données d'un contact. Côté du HTML notre vue "list" contient une nouvelle balise `contact-widget` acceptant un paramètre :

```
<li class="well span6" ng-repeat="contact in contacts">
  <contact-widget contact="contact"></contact-widget>
  <a ng-href="#/edit/{{$index}}>Editer</a>
</li>
```

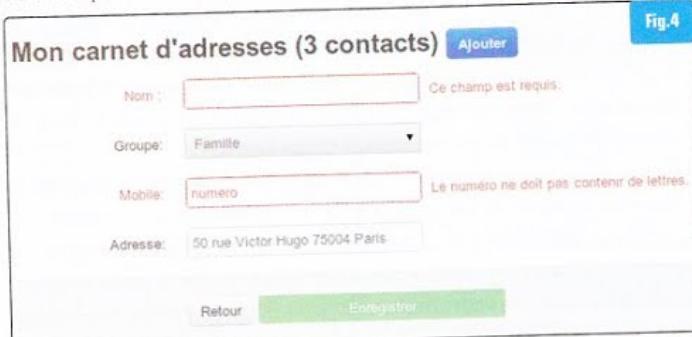
AngularJS se limite à des composants de base et n'a pas vocation à intégrer des composants visuels. Avec les directives, chacun peut créer son propre composant et enrichir le HTML. Même si ce code est aussi facile à écrire en jQuery, il est ici bien plus lisible et explicite. De plus le composant est clairement identifié et son comportement est encapsulé dans un code à part tel que :

```
app.directive('contactWidget', function() {
  return {
    restrict: 'E',           // cette directive est de type élément
    replace: true,          // l'élément doit être remplacé par un
    template ...             // ...
    templateUrl: 'template_contact.html', // ... template dont l'url
    est défini ici
    scope: {
      contact: '='        // l'attribut contact est bindé à l'objet
    contact de la directive
    }
  };
});
```

Ce comportement peut être modifié et surtout testé, ce qui est gage de stabilité et de flexibilité. Enfin c'est aussi une manière de mieux collaborer avec les designers qui pourront aisément intégrer le HTML dans leurs outils. AngularJS propose pour cela différents modes de déclaration et de nommage des directives représentées soit par des balises - `<contact-widget>` - soit par des attributs - `<div contact-widget="contact">` - soit par des styles CSS - `<div class="contact-widget">` - pour que chacun choisisse ce qui lui correspond le mieux selon sa manière de travailler et ses outils.

> Tests

Pour mettre en oeuvre les tests unitaires les frameworks de test JavaScript existants peuvent être utilisés. AngularJS ajoute une



Mon carnet d'adresses (3 contacts) Ajouter

Nom :	<input type="text"/>	Ce champ est requis.
Groupe :	<input type="text"/> Famille	
Mobile :	<input type="text"/> numero	Le numéro ne doit pas contenir de lettres.
Adresse :	50 rue Victor Hugo 75004 Paris	
	Retour	Enregistrer

bibliothèque de tests d'interactions utilisateurs facile à rédiger en utilisant une DSL (Domain Specific Language).

Pour illustrer un test avec notre exemple prenons comme scénario la saisie d'un contact. Le fait de saisir un nom et de cliquer sur "Enregistrer" doit créer un nouveau contact dans la liste :

```
describe('editAndSave', function () {
  beforeEach(function () {
    browser().navigateTo('#/edit/');
  });

  it('doit ajouter le contact dans la liste', function () {
    // saisie d'un nom
    input('contact.nom').enter('Misko Hevery');
    // clic sur Enregistrer
    element(':submit').click();
    // vérification que l'on a basculé sur la vue /list
    expect(browser().window().hash()).toMatch('list');
    // vérification que le contact a bien été ajouté à la liste
    expect(element('[ng-view] ul li:last div h3').text())
      .toMatch('Misko Hevery')
  });
});
```

Le projet annexe angular-seed vous propose un squelette de projet intégrant des scripts pour les tests que vous pouvez intégrer dans votre outil d'intégration continue.

> Conclusion

JavaScript reste encore aujourd'hui un langage "dangereux" pour certains dans le cadre d'application web stratégiques pour les entreprises du fait de sa lisibilité, de l'aspect asynchrone et des différentes implémentations dans les navigateurs. Cependant AngularJS se différencie principalement par l'intégration de concepts importants : évolutivité et maintenabilité. Les tests, le couplage-lâche et la composition font partie intégrante du framework et permettent d'envisager sereinement une application web de grande envergure avec la technologie JavaScript. Rappelons que le Web n'a pas été prévu pour créer des applications au départ. Le nouveau modèle d'application Web JS/HTML/CSS <-> JSON <-> WebAppServer (Java, Python, JavaScript, Ruby ...) n'est pas une mode mais bien un aboutissement dans la mise en oeuvre d'application sur le Web pour répondre à des attentes fortes en terme d'expériences utilisateur. AngularJS répond brillamment à cette attente.

Ressources

Pour en savoir davantage, visitez le site d'AngularJS : <http://angularjs.org/>

Le code de cet article peut être testé en ligne sur le site Plunker : <http://plnkr.co/edit/3Zx40D>

Le code source est disponible sur GitHub : <git@github.com:laureny/AddressBook.git>

Thierry Lau
Développeur SFEIR
twitter : @_aut3rry
Google+ : gplus.to/authierry
blog : devsnote.com

Sébastien Letiélie
Directeur Technique Improve Santé
twitter: @sebmade
Google+ : gplus.to/sebmade
blog : itaware.eu