

Angular est un framework écrit en JavaScript par **Miško Hevery** et **Adams Adrons** de chez **Google**. Il permet d’étendre simplement et de manière puissante le langage HTML pour développer rapidement des applications web.

Le HTML est utilisé comme langage de template et le contenu peut réagir en direct en fonction des actions de l’utilisateur.

Le but de ce TD est de créer une web-app de type “Todo list” minimaliste pour bien comprendre le principe de fonctionnement d’Angular.



1. Environnement de travail

Dans un premier temps, créez vous un dossier de travail classique comprenant au minimum :

**js**

└─ main.js

**css**

└─ main.css

index.html

Vous allez ensuite utiliser *bower*  pour installer *bootstrap* (afin d’avoir une jolie webapp).

Commencez par lancer la commande suivante :

bower init .

Cette commande vous aide dans la création d’un fichier bower.json qui représentera la **configuration bower** de votre projet. C’est un fichier très utile notamment pour stocker les noms des librairies que vous installerez avec Bower.

Installez ensuite bootstrap avec la commande suivante :

bower install bootstrap --save .

N’oubliez pas que l’option --save . est importante pour indiquer dans bower.json que votre application utilise la librairie *bootstrap*.

Installez également angular.js :

bower install angular --save .

Puis, faites les déclarations nécessaires dans votre index.html

|  |
| --- |
| <**link** rel="stylesheet" href="CHEMIN\_VERS\_BOOTSTRAP/bootstrap.css"> ... <**script** src="CHEMIN\_VERS\_ANGULAR/angular.js"></**script**> |

2. Mise en place du HTML de l’application

Grâce aux modules de bootstrap, construisez la vue HTML de votre application pour lui donner un aspect visuel épuré :



Pour l’instant, ne vous préoccupez pas du dynamisme, insérez simplement les données de manière statique.

3. Transformation en application Angular

Jusqu’à présent, la page n’est ni plus ni moins que du HTML5 classique. Pour pouvoir utiliser Angular, il va falloir lui indiquer le paramètre ng-app sur l’élément parent (généralement, l’élément <html>)

Placez-donc simplement un attribut ng-app sur votre balise <html> comme ceci :

|  |
| --- |
| <!-- L’attribut “ng-app” permet d’indiquer qu’on commence une application Angular  portant le nom ‘TodoMVC’ --> <**html** ng-app="TodoMVC"> |

Il va également falloir créer un **controlleur angular** qui va englober l’ensemble de notre vue correspondant à l’application.

Généralement, la vue est englobée dans une <div> ou une <section>, à laquelle il faut rajouter un attribut ng-controller="NomDeVotreControlleur as my". Par exemple :

|  |
| --- |
| <**div** class="todo-container" ng-controller="TodoController as my">  <!-- Ici tout le HTML correspondant à la vue de notre module de todolist --> </**div**> |

Le as my est un alias pour accéder aux données.

Il n’est pas obligatoire pour utiliser Angular, mais cela reste une bonne pratique courante :

4. Le controlleur

Il va également falloir initialiser ce controlleur.

Placez vous dans votre fichier .js principal et **écrivez\*** ceci : (\*pas, copiez-collez :)

|  |
| --- |
| **var** app = angular.module('TodoMVC', []);  **var** todolist = [  { taskName : 'Acheter du pain' , done : **true** },  { taskName : 'Réparer le PC' , done : **true** },  { taskName : 'Louer un DVD' , done : **false** },  { taskName : 'Nourrir le chat' , done : **false** },  { taskName : 'Apprendre AngularJS', done : **true** } ];  app.controller('TodoController'**, function**() {   this.todos = todolist;  }); |

Détaillons un peu.

Nous avons d’abord crée un module avec le même nom que notre attribut “ng-app”.

A ce module, nous avons attribué un controlleur “TodoController” (également enregistré avec l’alias “my”) correspondant à la vue HTML.

Le “this” de cette fonction permet déchanger les donnée du controlleur (JS) vers la vue (HTML)

Ainsi, la première chose que nous faisons dans ce controlleur est de créer un nouveau tableau d’objets this.todos qui correspondront à nos tâches (enregistré dans une variable externe qui joue le rôle de modèle). Chaque tâche ayant un nom ‘taskName’ et un état ‘done’ (effectuée ou non effectuée).

La variable “todos” est maintenant accessible dans le HTML **par le biais de l’alias**.

Pour s’en assurer, on peut déjà retourner dans le HTML et écrire :

|  |
| --- |
| <**h3**>  3 restantes sur un total de {{my.todos.length}} </**h3**> |

Les doubles accolades {{ }} permettent d’indiquer qu’on veut écrire du JavaScript via Angular, et ici, je place tout simplement le nombre d’éléments .length de mon tableau todos

Et comme par magie quand on charge la page, le résultat s’affiche :



*(On voit que le 5 correspond bien au nombre d’items dans notre tableau todos )*

Voila qui est fait.

Maintenant, il faudrait peut-être aussi indiquer à HTML de lister automatiquement les tâches à l’intérieur de todos de manière automatique. Pour cela, imaginons que jusqu’à présent, vous aviez un code ressemblant à ceci pour lister vos tâches :

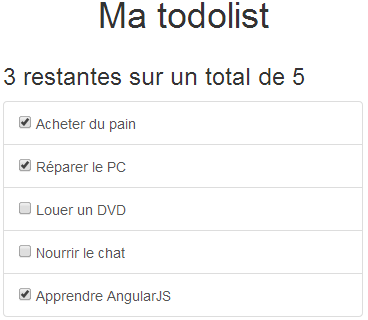
|  |
| --- |
| **<ul>  <li>  <input type="checkbox">  <span>Tâche 1</span>  </li>  <li>  <input type="checkbox">  <span>Tâche 2</span>  </li>  <li>  <input type="checkbox">  <span>Tâche 3</span>  </li> </ul>** |

Ce n’est pas vraiment dynamique. Essayons maintenant ceci :

|  |
| --- |
| <**ul**>  <**li** ng-repeat="todo in my.todos">  <**input** type="checkbox" ng-model="todo.done">  <**span**>{{todo.taskName}}</**span**>  </**li**> </**ul**> |

La directive ng-repeat permet **d’indiquer à HTML de répéter cette balise autant de fois qu’il y a d’éléments dans le tableau** et de les parcourir 1 par 1 (un peu **comme un foreach**).

Et donc s’affiche :



*(On remarque même que les checkboxes sont cochées si la tâche est accomplie. Cela est dû au ng-model qui applique la valeur de l’état de la tâche à l’élément HTML input)*

Nous avons jusqu’ici bien lié notre tableau de tâche avec la vue, mais vous remarquerez que si vous cochez/décochez les checkboxes, le nombre de tâches restantes en haut ne varie pas, car nous l’avions laissé jusqu’à présent la phrase avec la valeur brute : 3 restantes sur un total de {{my.todos.length}}

Il serait intéressant de plutôt écrire quelque chose comme ceci par exemple ...

|  |
| --- |
| <**h3**>  {{my.remaining()}} restantes sur un total de {{my.todos.length}} </**h3**> |

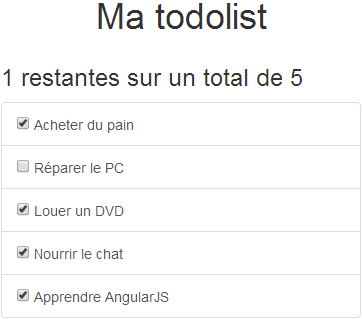
… où my.remaining() serait une fonction que l’on va créer dans le controlleur qui permettra de renvoyer un nombre correspondant aux tâches non accomplies.

Allons-y ! Dans le controlleur, en dessous de la définition du tableau todos de tout à l’heure, écrivez donc :

|  |
| --- |
| **var** app = angular.module('TodoMVC', []);  **var** todolist = [  { taskName : 'Acheter du pain' , done : **true** },  { taskName : 'Réparer le PC' , done : **true** },  { taskName : 'Louer un DVD' , done : **false** },  { taskName : 'Nourrir le chat' , done : **false** },  { taskName : 'Apprendre AngularJS', done : **true** } ];  app.controller('TodoController'**, function**() {   this.todos = todolist;   this.remaining = **function**() {  **var** undone = todolist.length;    todolist.forEach(**function**(el) {  **if** (el.done) {  undone--;  }  });   **return** undone;  };  }); |
|  |

Cette fonction est purement algorithmique. On récupère le nombre totales de tâches qu’on stocke dans la variable undone , puis on parcours chaque item du tableau todolist et on vérifie l’état : si l’état done vaut “true”, alors on décrémente la variable undone , qui lorsqu’on la retournera à la fin, correspondra au nombre de tâches restantes non-effectuées.

Maintenant, essayez de cocher/décocher les checkboxes et regardez le résultat changer en direct sous vos yeux :



*(Il me reste bien 1 seule tâche à effectuer sur les 5 présentes.)*

C’est là qu’est toute la puissance d’angular, il n’y a pas d’évènements à gérer nous-même entre les vues et les controlleurs. Tout changement dans le controlleur est immédiatement répercuté dans la vue correspondante !

Ajoutons un peu d’UX : par exemple, ce serait bien de tout simplement barrer les tâches qui ont été effectuées :



Pour ça, nous allons simplement créer une classe CSS :

|  |
| --- |
| .todo-done { **text-decoration**: line-through; **color:** green; } |

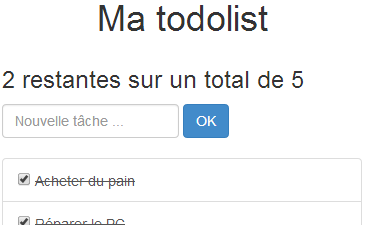
… et reprendre la vue de notre todolist pour ajouter une classe dynamique sur le <span> qui contient l’intitulé de la tâche :

|  |
| --- |
| **<ul>  <li ng-repeat="todo in my.todos">  <input type="checkbox" ng-model="todo.done">  <span ng-class="{ 'todo-done': todo.done }">{{todo.taskName}}</span>  </li> </ul>** |

On dit ici via la directive “ng-class” d’appliquer la classe “todo-isDone” uniquement si la condition “todo.done” renvoie “true”.

Ajouter une nouvelle tâche

Pour terminer, ce serait bien de pouvoir écrire dans une zone de texte le nom d’une nouvelle tâche à ajouter dans la liste, comme-ceci :



Voici un exemple de code minimaliste pour ajouter ce petit formulaire, à adapter selon votre design :

|  |
| --- |
| <**form** ng-submit="my.add()" novalidate>  <**input** type="text" ng-model="my.newTaskName" placeholder="Nouvelle tâche ...">  <**input** type="submit" value="OK"> </**form**> |

Les passages intéressants sont évidemment :

* ng-submit="my.add()" , qui va tout simplement faire appel à la fonction add() du controlleur lorsque le formulaire sera validé.
* novalidate , empêche le rechargement de la page lors de la validation
* ng-model="my.newTaskName" , ce qui nous permet d’avoir un model de disponible pour ce champs que l’on pourra utiliser dans le controlleur.

Et donc, côté controlleur justement, nous allons ajouter la fonction add(), qui va tout simplement insérer un nouvel élément dans le tableau todolist

|  |
| --- |
| **var** app = angular.module('TodoMVC', []);  **var** todolist = [...];  app.controller('TodoController'**, function**() {   // [...]   this.add = **function**() {  todolist.unshift({  taskName : this.newTaskName,  done : **false**  });  this.newTaskName = '';  };  }); |

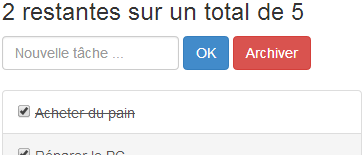
Dès que la fonction add() est appelée, on utilise la fonction unshift() de JavaScript qui fonctionne sur les tableaux et qui permet d’ajouter un élément au début du tableau ([voir la documentation de Array.unshift](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Array/unshift)). Et donc, on ajoute en début de tableau un nouvel objet littéral qui contient le titre de notre tâche dans “taskName” et son état dans “done” que l’on met par défaut à “false”.

Enfin, on efface le contenu du champs <input> en redéfinissant la variable “newTaskName” (qui correspond au modèle du champs vu qu’il est dans le “this”) à une valeur vide.

Améliorations

Normalement à ce stade, vous avez compris comment fonctionne la base d’angular.

Essayez maintenant de penser à plein de petites améliorations que vous pourriez ajouter sur votre webapp, comme par exemple le fait de pouvoir supprimer toutes les tâches qui sont effectuées en cliquant sur un bouton :



Pour cela, je vous donne un indice. Il est possible d’appeler une fonction du controlleur en utilisant l’attribut **ng-click** sur le bouton d’action.

Ensuite, il faudra parcourir le tableau de tâches et créer en parallèle un nouveau tableau dans lequel on ne gardera que les tâches non effectuées.

Autre idée d’amélioration, pourquoi ne pas sauvegarder les tâches de l’utilisateur dans le **localStorage** du navigateur pour qu’il puisse les retrouver plus tard en rechargeant la page.

Cette API JavaScript faisant partie d’ECMAScript 5 est très simple d’utilisation : elle permet de sauvegarder des variables dans la mémoire du navigateur, un peu comme des cookies, mais en plus facile.

[Voir cet d](http://www.alsacreations.com/article/lire/1402-web-storage-localstorage-sessionstorage.html)[’article d’Alsacréations](http://www.google.com/url?q=http%3A%2F%2Fwww.alsacreations.com%2Farticle%2Flire%2F1402-web-storage-localstorage-sessionstorage.html&sa=D&sntz=1&usg=AFQjCNGW2GazQ6jewWKH3qArBY_k_t_hOA) [sur l’utilisation du localStorage](http://www.alsacreations.com/article/lire/1402-web-storage-localstorage-sessionstorage.html).

[Voir aussi la documentation](https://developer.mozilla.org/en-US/docs/Web/Guide/API/DOM/Storage#localStorage).