Angular

BASES

Introduction

Binding

Liaison entre la logique et l'affichage

Permet d'évaluer une expression et d'afficher son résultat

- Une expression peut être : une variable, une opération, une fonction

{{ *expression* }}

Il existe une directive équivalente aux doubles accolades, il s'agit de ngBind

Exemple

```
<div>
    <!-- Concatène les 2 chaines de caractères et affiche "Hello World" -->
    {{ "Hello " + "World" }}

    <!-- Affiche le contenu de la variable name -->
    {{ name }}

    <!-- Affiche retour de la fonction hello() -->
    {{ hello("John") }}
</div>
```

Directives

Permet de modifier le DOM

Angular propose une liste de directive préconçues

Il est possible de créer des directives personnalisées

Module

Présentation

Permet de découper l’application

Représente une fonctionnalité de l’application

Qu’est ce qu’un module ?

- Contrôleur
- Service
- Animation
- Filtre
- Directive
- …

Gérer un module

Pour enregistrer un nouveau module

angular.module(nom, [dépendances]);

Exemple :

```
var module = angular.module("monModule", []);
```

Premier argument : nom du module

Deuxième arguments : les dépendances du module

Gérer un module

Modifier la directive ngApp

```
<html ng-app="monModule">
```

Créer un scope dédié à l'application

Permet d'utiliser des contrôleurs, filtres etc, appartenant au module

Obtenir un module

```
var module = angular.module("monModule");
```

Contrôleur

Présentation

Contient la logique métier

Va être lié à la vue par une directive

Va fournir un contexte de données à la vue

Créer un contrôleur

Un contrôleur doit appartenir à un module

- Il faut donc récupérer un module ou posséder sa variable

.controller(nom, fonction)

Exemple :

```
var module = angular.module("monModule");

module.controller('HelloController', function ($scope) {

});
```

Chaque contrôleur va posséder son propre scope qui est injecté par angular

ngController

Pour utiliser un contrôleur dans la vue, il faut utiliser la directive ngController

Il faut lui donner le nom du contrôleur

Exemple

```
<div ng-controller="HelloController">

</div>
```

$scope

Cette variable qui est injectée dans le contrôleur représente le contexte de données du contrôleur

Il est possible d'ajouter des variables dans ce contexte qui pourront être retrouvées dans la vue

Une variable dans le contexte peut contenir : une valeur primitives, un objet, un tableau, une fonction, etc

Exemple :

```
module.controller('HelloController', function ($scope) {
    $scope.hello = "Hello World !";
});
```

$scope

Dans l'exemple précédent, une variable « hello » a été ajoutée au contexte

Donc entre les balises qui contienne la directive appelant le contrôleur précédent, il est possible d'exploiter cette variable

Exemple :

```
<div ng-controller="HelloController">
    {{ hello }}
</div>
```

Scope

Un contexte (scope) est un objet qui offre une couche d'abstraction entre la vue et le contrôleur

Un scope peut écouter les changements grâce à un watcher ($watch)

Un scope peut propager des évènements à ses enfants ou son parent ($emit et $broadcast)

Fonctions

Il est possible d'ajouter des fonctions au contexte dans le contrôleur

```
module.controller('HelloController', function ($scope) {

    $scope.sayHello = function (name) {
        return "Hello " + name;
    }
});
```

```
<div ng-controller="HelloController">
    {{ sayHello("John") }}
</div>
```

Héritage des scopes

Bien souvent, un module sera composé de plusieurs contrôleurs

Il n'est pas rare que différents contrôleurs possèdent une variable dans leur contexte ayant le même nom

Qu'arrive t'il si dans la vue, ces contrôleurs s'imbriquent ?

```
module.controller('FirstController', function ($scope) {
    $scope.name = "Code";
});

module.controller('SecondController', function ($scope) {
    $scope.name = "Trainer";
});
```

Héritage des scopes

```
<div ng-controller="FirstController">
    {{ name }}
    <div ng-controller="SecondController">
        {{ name }}
    </div>
</div>
```

Etant donné que chaque contrôleur dispose de son propre contexte, il n'y a pas de collisions entre les variables

Angular met à disposition une variable pour remonter au parent

```
<div ng-controller="SecondController">
    {{ $parent.name }}
</div>
```

$rootScope

Chaque application possède un « scope root »

C’est le parent de tous les scopes

Binding « twoway »

Directive ngModel

Cette directive peut être appliquée à un input, select, textarea

Permet de mettre à jour (ou créer si elle n'existe pas) une propriété dans le scope courant

- En d'autre terme, la vue met à jour le contrôleur

Exemple

```
<div>
    {{ name }}
    <input type="text" ng-model="name" />
</div>
```

Si la variable « name » n'existe pas dans le scope, elle sera créée et aura pour valeur, la valeur qui l'utilisateur tape dans le champ

Directives de structures

If

Permet de supprimer ou reconstruire une partie du DOM en fonction d'une expression

Se fait avec la directive ngIf qui prend une expression à évaluer

```
<div ng-if="age > 18">

</div>
```

Switch

Permet de changer d'affichage en fonction d'une condition

Se fait avec la directive ngSwitch

Chaque cas s'exprime avec la directive ngSwitchWhen

Le cas par défaut s'exprime avec la directive ngSwitchDefault

```
<div ng-switch="name">
    <div ng-switch-when="John">

    </div>
    <div ng-switch-when="Doe">

    </div>
    <div ng-switch-default>

    </div>
</div>
```

Show/hide

Ces directives permettent d'afficher ou cacher une partie de la page.

Se fait avec les directives ngShow et ngHide

A la différence de ngIf, les parties ne sont pas supprimées du DOM mais uniquement masquées.

```
<input ng-init="display = true" type="checkbox" ng-model="display" />
<div ng-show="display">
    Affiché
</div>
<div ng-hide="display">
    L'autre div doit être cachée si je suis là !
</div>
```

For

Permet d’itérer sur une liste

Se fait avec la directive ngRepeat

Exemple

```
<ul>
    <li ng-repeat="n in [1,2,3,4,5]">
        {{ n }}
    </li>
</ul>
```

Itérer sur une liste d'objet

```
module.controller('HelloController', function ($scope) {
    $scope.persons = [{
        firstname: "John",
        lastname: "Doe"
    },
    {
        firstname: "Kevin",
        lastname: "Spacey"
    }]
});

<ul>
    <li ng-repeat="p in persons">
    {{ $index + " " + p.firstname + "  " + p.lastname }}
    </li>
</ul>
```

Itérer sur les propriétés d'un objet

```
module.controller('HelloController', function ($scope) {
    $scope.person = {
        firstname: "John",
        lastname: "Doe"
    };
});
```

```
<ul>
    <li ng-repeat="(key, value) in person">
    {{ key + "  " + value }}
    </li>
</ul>
```

Editer une liste

En combinant la directive ngModel, les changements s'appliqueront directement sur l'objet en question

```
<ul>
    <li ng-repeat="p in persons">
        <input type="text" ng-model="p.firstname">
        <input type="text" ng-model="p.lastname">
    </li>
</ul>
```

Filtres

Présentation

Angular propose des fonctionnalités de filtres préconstruites

{{ expression | filtre [: param…] }}

Ils permettent de transformer le rendu des données

Ils se placent dans les vues

Il est possible de créer des filtres personnalisés

Filtre de texte

Changer la casse d’un texte

Mettre en majuscule

```
{{ lastname | uppercase }}
```

Mettre en miniscule

```
{{ firstname | lowercase }}
```

Devise

Permet d'ajouter la devise au texte

- Si aucun paramètre : devise locale
- Premier paramètre : symbole de la devise
- Second paramètre : arrondi

```
{{ 1.337 | currency }}
{{ 1.337 | currency:"€" }}
{{ 1.337 | currency:"€":2 }}
```

Date

Permet de formater une date

L’expression traitée doit être un objet Date, un timestamp ou une string

Voir la documentation pour les formats possibles

```
{{ 1451606461000  | date:'dd-MM-yyyy HH:mm:ss' }}
```

Json

Permet de convertir un objet Javascript en JSON

```
{{ { firstname:'John' } | json }}
```

Limiter

Permet de limiter le nombre de caractère à afficher ou le nombre de résultats à afficher dans une liste

```
<ul>
    <li ng-repeat="p in persons | limitTo: 2">
        {{ $index + " " + p.firstname + "  " + p.lastname }}
    </li>
</ul>
```

Trier

Permet de trier une liste

- Le paramètre peut être une fonction ou une propriété d'un objet
- Le signe – (moins) permet d'inverser le tri ou peut être passé en 2 paramètre sous forme de booléen

orderBy: expression : reverse : comparator

Exemple

```
module.controller('HelloController', function ($scope) {
    $scope.persons = [{
        firstname: "John",
        lastname: "Doe"
    },
    {
        firstname: "Kevin",
        lastname: "Spacey"
    }]
});

<ul>
    <li ng-repeat="p in persons | orderBy: firstname:true">
    {{ p.firstname + "  " + p.lastname }}
    </li>
</ul>
```

Filtrer

Permet de filtrer dans une liste

Pour rechercher partout dans l'objet : filter: « recherche »

```
<input type="text" ng-model="search"  />
<ul>
    <li ng-repeat="p in persons | filter: search">
        {{ $index + " " + p.firstname + "  " + p.lastname }}
    </li>
</ul>
```

Filtrer

Pour filtrer sur une propriété en particulier de l'objet: filter:{propriété: « recherche »}

```
<input type="text" ng-model="search"  />
<ul>
    <li ng-repeat="p in persons | filter:{firstname: search}">
        {{ $index + " " + p.firstname + "  " + p.lastname }}
    </li>
</ul>
```

Il est possible de créer un alias sur la recherche, afin de l'utiliser (pour afficher le nombre de résultats par exemple

```
ng-repeat="p in persons | filter:{firstname: search} as result"
```

Evènements

Présentation

Angular met à disposition des directives permettant d'associer un évènement à un élément

Les directives suivantes sont disponibles :

- ngMousedown
- ngMouseup
- ngMouseenter
- ngMouseleave
- ngMousemove
- ngMouseover
- ngClick
- ngDblclick
- ngBlur
- ngFocus
- ngChange
- ngChecked
- ngCopy
- ngCut
- ngPaste
- ngKeydown
- ngKeypress
- ngKeyup

Exemple

```
<input type="text" ng-model="name" />
<button ng-click="sendClick(name)">Envoyer</button>

module.controller('HelloController', function ($scope) {
    $scope.sendClick = function (name) {
        alert("hello" + name);
    }
});
```

Le code ci-dessus peut être amélioré ! En effet, la directive ngModel va créer une variable dans le scope. Donc il n’est pas nécessaire de la passer en paramètre dans la fonction du contrôleur.

Informations de l’évènement

On peut parfois avoir besoin de plus d’informations sur l’évènement.

Angular met à disposition la variable $event qui doit être passé dans une fonction

```
<button ng-click="sendClick($event)">Envoyer</button>
```

```
module.controller('HelloController', function ($scope) {
    $scope.sendClick = function (event) {
        console.log(event);
    }
});
```