Angularjs 1.x

Table des matières

Version d'Angular utilisée: 1.5.9

Site, guide, api

1. Création de projet

« Simple »

Avec un **gestionnaire de packages** :

- **Bower**

```
bower i angular -S
```

Autres possibilités :

- **CDN** : cdnjs, Google
- **Téléchargement** depuis le site
- **Packages NuGet** avec Visual Studio (chercher « AngularJS.Core »)
- Génération d'un projet de départ avec Yeoman (pour angular)

Serveur

- **Asp.Net**
  - Créer un projet Web avec le modèle « **Single Page Application** »
  - Utiliser les **packages NuGet** pour installer les dépendances Angular
  - Mettre son **code** Angular dans le dossier « **Scripts\app** »

Avec un router Angular en mode « html5 »

… Dans « **Web.config** »

Réécriture d'url afin que ce soit Angular qui gère le routing et ne pas avoir une erreur 404 côté serveur avec le mode html5 du router.

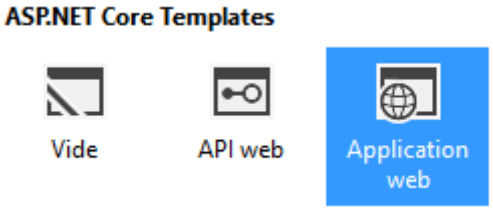
```
<system.webServer>
 <rewrite>
      <rules>
        <rule name="AngularJS" stopProcessing="true">
          <match url=".*" />
          <conditions logicalGrouping="MatchAll">
            <add input="{REQUEST_URI}" negate="true" pattern="^/api/" ignoreCase="true" />
            <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true" />
            <add input="{REQUEST_FILENAME}" matchType="IsDirectory" negate="true" />
          </conditions>
          <action type="Rewrite" url="/" />
        </rule>
      </rules>
    </rewrite>
</system.webServer>
```

Pour tout sauf pour les uris Web Api

… Autre possibilité … Redéfinir la route par défaut (« RouteConfig.cs »)

```
routes.MapRoute(
                name: "Default",
                url: "{*.}",
                defaults: new { controller = "Home", action = "Index" }
            );
```

- **Asp.Net Core**
  - Créer un projet Web .NET Core avec le modèle « **Application web** »



  - Ajouter les dépendances Angular au « **bower.json** » du projet. Depuis une invite de commande ou clic droit sur « dépendances … bower » > « Ouvrir bower.json ». Les dépendances sont installées dans le dossier « wwwroot\lib » à la sauvegarde du « bower.json ».
  - Mettre le **code** Angular dans le dossier « **wwwroot\js** »

Avec un router Angular en mode « html5 »

… Dans « **Startup.cs** »

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env,
ILoggerFactory loggerFactory)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
                app.UseBrowserLink();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
            }

            app.Use(async (context, next) => {
                await next();

                if (context.Response.StatusCode == 404)
                {
                    context.Request.Path = "/";
                    context.Response.StatusCode = 200;
                    await next();
                }
            });


            app.UseApplicationInsightsExceptionTelemetry();

            app.UseStaticFiles();

            app.UseMvc(routes =>
            {
                routes.MapRoute(
                    name: "default",
                    template: "{controller=Home}/{action=Index}/{id?}");
            });
        }
```

Evite la page 404 côté serveur et permet à Angular de gérer le routing à la saisie dans la barre d'adresse

- **PHP**
    - Créer un projet « local » (exemple dans le dossier « www » avec Wamp)
    - « **npm init** » (création de « package.json ») puis « **bower init** » (création de « bower.json ») et ajouter les dépendances Angular au « **bower.json** »

Définir avec « .bowerrc » le répertoire ou les librairies seront installées

```
{
  "directory": "public/lib"
}
```

    - « composer init » (création du « **composer.json** »). Créer une page « index.php ».

Avec un router Angular en mode « hash »

```
<?php
require 'public/index.html';
```

Avec un router Angular en mode « html5 »

… Tout simple « index.php »

```
<?php

require "vendor/autoload.php";

$url = $_GET['url'];
if(preg_match("#^/api/#i", $url) == 1){
   // api
}
else {
   include "index.html";
}
```

« .htaccess »

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^(.*)$ index.php?url=/$1 [QSA,L]
```

… Avec un **router PHP**

Exemple Slim … installé dans le dossier « vendor » avec la commande

```
composer require slim/slim "^3.0"
```

« index.php »

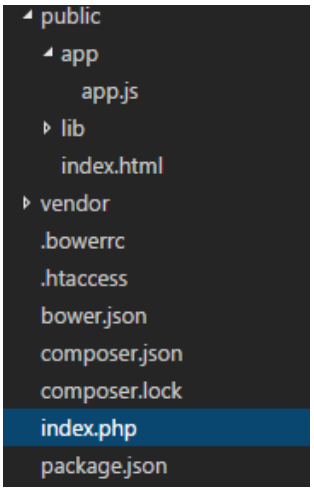
Toutes les urls pointent « index.html » afin que ce soit le router Angular qui gère le routing (et les éventuelles erreurs de routing)

```
<?php
require 'vendor/autoload.php';
```

```
$app = new \Slim\App;

// autres routes (api par exemple)

$app->get('/[{path:.*}]', function () {
    require 'public/index.html';
});
$app->run();
```

« .htacces »

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^ index.php [QSA,L]
```



- **Node.js**
  - Création d'un dossier pour le projet Node
  - « **npm init** » (création de « package.json ») puis « **bower init** » (création de « bower.json ») et ajouter les dépendances Angular au « **bower.json** »

Définir avec « .bowerrc » le répertoire ou les librairies seront installées

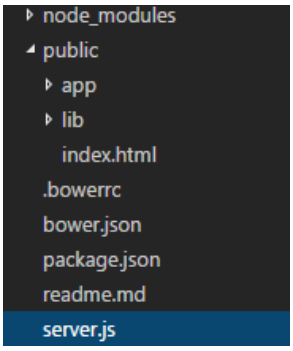
```
{
  "directory": "public/lib"
}
```

« server.js » :

Toutes les urls pointent « index.html » afin que ce soit le router Angular qui gère le routing (et les éventuelles erreurs de routing)

```
var express = require("express"),
   app = express(),
   path = require('path');
app.use(express.static("public"));

// autres routes (api par exemple)

app.get('*', function (req, res) {
   res.sendFile(path.join(__dirname, '/public/index.html'));
```

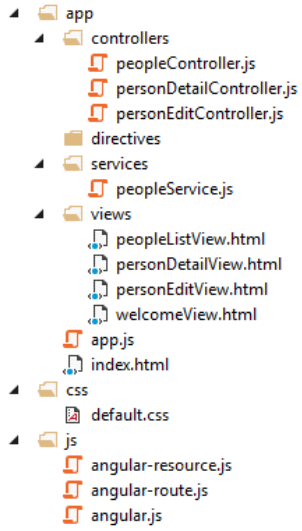
```
});

var server = app.listen(3000, function () {
   console.log('api listening on ', server.address().port);
});
```



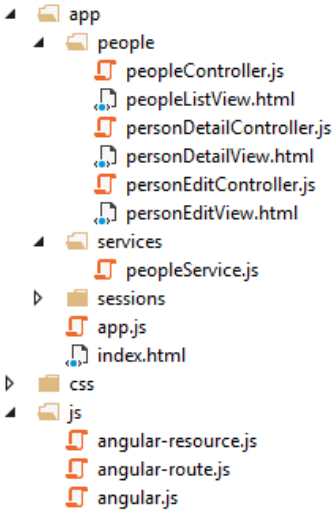
- Autre possibilité : utiliser MEAN.JS

2. Organisation du projet

Par « type »
(petites applications avec un seul module)



Par « feature »



« Mélange des deux »

On peut mélanger les 2 organisations (exemple une feature « people » qui est divisée en dossiers « controllers » , « views », « services »).

3. Conventions de nommage

- **PascalCase** pour les noms de **contrôleurs**
- **camelCase** pour les **directives, services**, etc.
- **Contrôleurs** avec un suffixe « **Ctrl** » ou « **Controller** »

4. Afficher son site

Besoin d'un serveur (à cause du chargement de templates qui requiert des requêtes http qui provoqueraient des cors). Node, Apache,IIS, etc. voir pourquoi pas le serveur d'aperçu de l'éditeur de texte « **Brackets** ».

5. Module

```
(function () {
    'use strict';

    angular.module('app', []);

})();
```

Obtenir un module

- À la création

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

- Par la suite

```
var app = angular.module('app');
```

Projet avec plusieurs modules

Pour **chaque module**, on crée un **fichier de configuration du module**

```
var articles = angular.module('articles',['ngRoute','ngResource']);
```

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

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

… **puis enregistrement des modules** dans le **module principal**

```
var app = angular.module('app',['core','articles','users','ngRoute']);
```

(Référencer tous les modules dans la page d'index)

6. Routing

Soit dans le même fichier que le module, sans dans un fichier à part qu'il faudra référencer dans la page d'index.

a. Avec « Angular route »

Installation

Installation avec un **gestionnaire de packages**

Avec **Bower**

```
bower i angular-route -S
```

Autres possibilités :

- **CDN** : cdnjs
- Packages **NuGet** avec Visual Studio (chercher « AngularJS.Route »)

Ajouter la **dépendance** au module

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

Référencer Angular route dans la page d'index

```
<script src="bower_components/angular-route/angular-route.min.js"></script>
```

Ajouter la directive « **ng-view** » (qui affichera les vues) dans la page d'index

```
<div ng-view></div>
```

Routes

Documentation $routeProvider

Paramètres de la fonction « when » :

- Path (string)
- Route (object)

```
(function () {
    'use strict';

    angular.module('app', ['ngRoute'])
        .config(function ($routeProvider) {
            $routeProvider
                .when('/', {
                    templateUrl: 'app/views/home.html'
                })
                .when('/articles', {
                    templateUrl: 'app/articles/articleListView.html'
                })
                .when('/articles/:id', {
                    templateUrl: 'app/articles/articleDetailView.html'
                })
                .otherwise({ redirectTo: "/" });
        });

})();
```

Route avec contrôleur. On peut indiquer le contrôleur dans la vue ou avec le routing.

Contrôleur avec $scope

- Dans la vue

```
<div ng-controller="ArticleController">
    <h1>{{ title }} </h1>

    <!-- etc. -->
</div>
```

- Dans le routing

```
.when('/articles', {
    templateUrl: 'app/views/articles.html',
    controller:'ArticlesController'
})
```

Contrôleur « As »

- Dans la vue

```
<div ng-controller="ArticleController as vm">
    <!-- etc. -->
</div>
```

- Dans le routing

```
.when('/articles', {
    templateUrl: 'app/views/articles.html',
    controller:'ArticlesController as vm'
})
```

Resolve

```
(function () {
    'use strict';

    var app = angular.module('app', ['ngRoute']);
    app.config(function ($routeProvider) {
        $routeProvider
            .when('/articles/:id', {
                templateUrl: 'app/views/article_view.html',
                controller: 'ArticleDetailController',
                resolve: {
                    article: ['articleService', '$route', function (articleService,
$route) {
                        var id = $route.current.params.id;
                        console.log(id);
                        var article = articleService.getOne(id);
                        console.log(article);
                        return article;
                    }]
                }
            })
            .otherwise({redirectTo: '/'});
    });

})();
```

Ne pas utiliser $routeParams pour récupérer le paramètre qui n'est modifié qu'après le changement de route

… Injection de l'article dans le contrôleur

```
(function () {
    'use strict';

    angular.module('app').controller('ArticleDetailController', ['$scope', 'article',
ArticleDetailController]);

    function ArticleDetailController($scope, article) {

    }
})();
```

Obtenir les paramètres de route et search avec « $routeParams »

Injecter « $routeParams » dans le contrôleur. Pour une route par exemple

```
.when('/posts/:id', {
    templateUrl: 'views/paramsinfo.html',
    controller : 'PostCtrl'
})
```

… On navigue depuis un lien

```
<a href="#/posts/1?query=value">Search</a>
```

```
http://localhost:3000/#/posts/1?query=value
```

On récupère à la fois l'id ainsi que la « recherche »

```
{"query":"value","id":"1"}
```

Events

```
app.run(['$rootScope', '$route', function ($rootScope, $route) {


    $rootScope.$on('$routeChangeStart', function (sender, args) {
        console.log(args);
    });

    $rootScope.$on('$routeChangeSuccess', function (sender, args) {
        console.log(args);
    });

    $rootScope.$on('$routeChangeError', function (sender, args) {
        console.log("Promises rejected " + args.loadedTemplateUrl);
    });

    $rootScope.$on('$routeUpdate', function (sender, args) {
        console.log("Search params updated: " + args.params["query"]);
    });
}]);
```

Avant le changement de route

Après le changement de route

Erreur ou rejet (pour une autorisation)

Observe les changements de valeur pour une « recherche »

Note pour détecter le changement au niveau d'un paramètre de recherche, définir la route avec « reloadOnSearch » à « false »

```
.when('/posts/:id', {
    templateUrl: 'views/paramsinfo.html',
    controller : 'PostCtrl',
    reloadOnSearch: false
})
```

Puis par exemple depuis le contrôleur ajouter une fonction permettant de modifier un paramètre de recherche

```
$scope.search = function () {
    $location.search("query", "new value");
};
```

Lié à un bouton quelconque de la vue

```
<button ng-click="search()" class="btn btn-info">Search</button>
```

Modes « hash » et « html5 »

Par défaut le router est en mode « hash » (routes avec « # » + détection « on hash change »)

On peut passer le router en mode « html5 » avec « $locationprovider » pour avoir des urls plus « propres ».

```
(function () {
    'use strict';

    angular.module('app', ['ngRoute'])
        .config(['$routeProvider', '$locationProvider', function
($routeProvider, $locationProvider) {

            $locationProvider.html5Mode(true);

            $routeProvider
                .when('/', {
                    templateUrl: 'app/home/home.html'
                })
            });

        }]);

})();
```

Ajouter la balise « base » dans la page d'index

```
<base href="/">
```

Liens

Avec un router en mode « hash » on indique « # » puis le chemin + paramètres

```
<a href="#/">Home</a>
<a href="#/articles/">Blog</a>
<a href="#/articles/{{ article.id}}">Lire la suite</a>
```

Avec un router en mode « html5 »

```
<a href="/">Home</a>
<a href="articles/">Blog</a>
<a href="articles/{{ article.id}}">Lire la suite</a>
```

Récupération d'un paramètre passé depuis le contrôleur avec $routeParams

```
(function () {
    'use strict';

    angular.module('app').controller('ArticleDetailController', ['$scope',
'$routeParams', ArticleDetailController]);

    function ArticleDetailController($scope, $routeParams) {

        console.log($routeParams.id);
    }

})();
```

Sécuriser une route

```
(function () {
    'use strict';

    angular.module('app', ['ngRoute'])
        .config(['$routeProvider', '$locationProvider', function
($routeProvider, $locationProvider) {

            $routeProvider
                .when('/articles/new', {
                    template: "<h1>Create new article</h1>",
                    secure: true
                })
        }])
        .run(['$rootScope', '$location','AuthService', function
($rootScope, $location, authService) {

            $rootScope.$on('$routeChangeStart', function (event, to,
from) {
                if (to && to.$$route && to.$$route.secure) {
                    // authorize
                     if (!authService.user.isAuthenticated) {
                         $location.path('/login');
                     }
                }
            });

        }]);

})();
```

Vérification que l'utilisateur est autorisé à accéder à la route, sinon redirection vers la page de login

b. Avec « Angular ui-router »

Installation

Installation avec un **gestionnaire de packages**

Avec **Bower**

```
bower i angular-ui-route -S
```

Autres possibilités :

- **CDN** : cdnjs
- Packages **NuGet** avec Visual Studio

Ajouter la **dépendance** au module

```
var app = angular.module('app', ['ui.router']);
```

Référencer Angular ui-router dans la page d'index

```
<script src="bower_components/angular-ui-router/release/angular-ui-router.min.js"></script>
```

Ajouter la directive « **ui-view** » (qui affichera les vues) dans la page d'index

```
<div ui-view></div>
```

Routes

On peut indiquer le contrôleur dans la vue ou avec le routing comme pour « Angular route »

```
(function () {
    'use strict';

    var app = angular.module('app', ['ui.router']);
    app.config(['$stateProvider',
        '$urlRouterProvider',
        function ($stateProvider, $urlRouterProvider) {
            $urlRouterProvider.otherwise('/');
            $stateProvider
                .state('home', {
                    url: '/',
                    templateUrl: 'app/views/home.html'
                })
                .state('articles', {
                    url: '/articles',
                    templateUrl: 'app/views/articles.html',
                    controller: 'ArticlesController'
                })
                .state('article_detail', {
                    url: '/articles/:id',
                    templateUrl: 'app/views/article_view.html',
                    controller: 'ArticleDetailController'
                })
        }]
    );
})();
```

Autre exemple

```
$stateProvider
        .state('demo', {
            url: '/demo/{myId:[0-9]*}',
            views: {
                'list': {
                    templateUrl: 'views/list.html',
                    controller: ['$scope', '$location', function ($scope, $location) {
                        $scope.title = "Mon titre";
                    }]
                },
                'detail': {
                    templateUrl: 'views/detail.html'
                }
            },
            onEnter: function () {
                console.log("Entered demo state.");
            },
            onExit: function () {
                console.log("Exited demo state.");
            }
        })
```

Regex sur paramètre

Contrôleur défini directement

Plusieurs vues pour un état

Events

Resolve

```
(function () {
    'use strict';
    var app = angular.module('app', ['ui.router']);
    app.config(['$stateProvider',
        '$urlRouterProvider',
        function ($stateProvider, $urlRouterProvider) {
            $urlRouterProvider.otherwise('/');
            $stateProvider
                .state('article_detail', {
                    url: '/articles/:id',
                    templateUrl: 'app/views/article_vie
                    controller: 'ArticleDetailControlle
                    resolve: {
                        article: ['articleService', '$stateParams', function
(articleService, $stateParams) {
                            var article = articleService.getOne($stateParams.id);
                            return article;
                        }]
                    }
                })
        }]
    );
})();
```

On retourne l'article …

… Que l'on injecte dans le contrôleur

```
 (function () {
    'use strict';

    angular.module('app').controller('ArticleDetailController', ['$scope', 'article',
ArticleDetailController]);

    function ArticleDetailController($scope, article)  {

    }

})();
```

Events

```
app.run(['$rootScope', function ($rootScope) {

    $rootScope.$on('$stateChangeStart', function (e, toState, toParams) {

    });

    $rootScope.$on('$stateChangeSuccess', function (e, toState, toParams)
{

    });

    $rootScope.$on('$stateChangeError', function (e, toState, toParams) {

    });

    $rootScope.$on('$viewContentLoading', function (e, viewConfig) {

    });

    $rootScope.$on('$viewContentLoaded', function (e, viewConfig) {

    });

}]);
```

Liens

Utiliser « **ui-sref** » avec le nom de la route

```
<a ui-sref="home">Home</a>
<a ui-sref="articles">Blog</a>
<a ui-sref="article_detail({ id: article.id })">Lire la suite</a>
```

Redirection

```
$state.go('home');
```

Récupération d'un paramètre passé depuis contrôleur avec $stateParams

```
(function () {
    'use strict';

    angular.module('app').controller('ArticleDetailController', ['$scope', '$stateParams',
ArticleDetailController]);

    function ArticleDetailController($scope, $stateParams) {

        console.log($stateParams.id);
    }
})();
```

c. Obtenir les informations d'url avec « $location »

Exemple

```
http://localhost/angular1/location.html#/mypath/1?query=value#myhash
```

- Protocol : http
- Host : localhost
- Port : 80
- Path : /mypath/1
- Search : ?query=value
- Hash : #myhash
- absUrl : http://localhost/angular1/location.html#/mypath/1?query=value#myhash
- url : /mypath/1?query=value#myhash

Récupérer les informations de location :

```
"absolute-Url": $location.absUrl(),
"hash": $location.hash(),
"host": $location.host(),
"path": $location.path(),
"port": $location.port(),
"protocol": $location.protocol(),
"search": $location.search(),// retour json {"query":"value"}
"url": $location.url()
```

7. Injection de dépendance

« Simple »

Angular aura des problèmes pour résoudre les dépendances après minimisation

```
app.controller('MyCtrl', function ($scope, myService) {

});
```

Ou

```
app.controller('MyCtrl',['$scope', 'myService', function ($scope, myService) {

}]);
```

Ou avec « **$inject** »

```
app.controller('MyCtrl', MyCtrl);
function  MyCtrl($scope, myService) {

}
MyCtrl.$inject = ['$scope', 'myService'];
```

8. Controller

a. Avec $scope

```
(function () {
    'use strict';

    angular.module('app').controller('ArticlesController', ['$scope',
'articleService', ArticlesController]);

    function ArticlesController($scope, articleService) {

        $scope.title = "Articles";
        $scope.articles = [];

        function getAll() {
            $scope.articles = articleService.getAll();
        }
        getAll();
    }

})();
```

Note il est possible de faire très simple. Le problème de ces écritures pourra survenir lors de la minimisation.

```
angular.module('app').controller('ArticlesController', function($scope,
articleService) {

});
// Voir même encore plus simple, si on est moins rigoureux
app.controller('ArticlesController', function($scope, articleService) {

});
```

b. Controller « As »

Pas d'injection de $scope

```
(function () {
    'use strict';

    angular.module('app').controller('ArticlesController', ['articleService',
ArticlesController]);

    function ArticlesController(articleService) {
        var vm = this;

        vm.title = "Articles";
        vm.articles = [];

        function getAll() {
            vm.articles = articleService.getAll();
        }
        getAll();
    }

})();
```

On définit le controller « as vm » avec le routing

```
.when('/articles', {
    templateUrl: 'app/views/articles.html',
    controller: 'ArticlesController as vm'
})
```

```
<h1>{{ vm.title }} </h1>

<div ng-repeat="article in vm.articles">
    <article>
        <header>
            <h2>{{ article.title }}</h2>
        </header>
        <p>{{ article.content }}</p>
    </article>
</div>
```

… ou directement dans la vue

```
<div ng-controller="ArticlesController as vm">

    <h1>{{ vm.title }} </h1>

    <div ng-repeat="article in vm.articles">
        <article>
            <header>
                <h2>{{ article.title }}</h2>
            </header>
            <p>{{ article.content }}</p>
        </article>
    </div>
</div>
```

9. Constant

- string, number, array ou function

Création d'une constante

```
angular.module("app", []).constant("myconstant", "abc");
```

Injection dans un contrôleur

```
app.controller('IocCtrl', IocCtrl);
IocCtrl.$inject = ['$scope','myconstant'];
function IocCtrl ($scope, myconstant) {

}
```

10. Value

- string, number, array ou function
- Ne peut pas être injecté depuis la configuration

```
angular.module("app", []).value("myvalue", 123);
```

Injection dans un contrôleur

```
app.controller('IocCtrl', IocCtrl);
IocCtrl.$inject = ['$scope','myvalue'];
function IocCtrl ($scope, myvalue) {

}
```

11. Factory

```
app.factory("myfactory",['$http', function($http){
    return{
        methodOne: function(){

        },
        methodTwo: function(){

        }
    }
}]);
```

12. Service

Fonction « constructor » créé avec « new »

```
app.service("myservice",['$http', function($http){
    var self = this;
    this.$http = $http;

    this.methodOne = function(){

    };
    this.methodTwo = function(){
```

```
    };
}]);
```

13. Provider

- Peut être configuré

C'est la fonction « $get » qui est appelée lorsqu'un contrôleur par exemple utilise le provider

```
app.provider("logProvider", LogProvider);
function LogProvider() {
    var enabled = true;
    this.disableLogging = function() {
        enabled = false;
    };
    this.$get = function() {
        if (enabled) {
            return {
                log: function(msg) {
                    console.log(msg);
                }
            };
        }
        return {
            log: function () {}
        };
    };
}
```

Configuration Exemple on désactive le logger

```
var app = angular.module("app", []);
app.config('logProvider', function (logProvider) {
    logProvider.disableLogging();
});
```

14. Injector

Documentation

Exemple au lieu de passer le service, on passe « $injector » et on récupère une instance du service

```
app.controller('MyCtrl',['$scope', '$injector', MyCtrl]);
function  MyCtrl($scope, $injector) {

    var myService = $injector.get('myService');
    $scope.message = myService.methodOne();

}
```

Autres fonctions utiles :

- « has » (savoir si une instance du service existe)
- « instantiate » (créer une instance du service)
- « invoke » (invoquer une fonction du service).Exemple :

```
app.controller('MyCtrl',['$scope', '$injector', MyCtrl]);
function  MyCtrl($scope, $injector) {
    // invoke
    var result = $injector.invoke(function (myService) {
        return myService.methodOne();
    });
    $scope.message = result;
}
```

15. Decorator

Sert à intercepter, modifier un service / directive / filtre sans modifier le code de base.

Documentation, Exemple

On ajoute la date en début de message de log

```
var app = angular.module("app", []);
app.config(["$provide", function($provide) {
    $provide.decorator("logProvider", ["$delegate", function($delegate) {
        return {
            log: function(msg) {
                var date = new Date();
                $delegate.log(date.toString() + " : " + msg);
            }
        };
    }]);
}]);
```

« $delegate » représente l'instance originale du service

Utilisation du logProvider depuis un contrôleur

```
app.controller("myController", ["$scope", "logProvider", function($scope,
logProvider) {
    $scope.send = function() {
        logProvider.log("My message");
    };
}]);
```

16. Interceptors

Exemples

17. Scopes et events

Events : **$on** pour s'abonner

```
$scope.$on("message", function(e, message) {
    $scope.message = message;
});
```

Puis utiliser :

- **$broadcast** pour publier un event aux **enfants** (scope courant compris)
- **$emit** pour publier un event au scope **parent** (scope courant compris)

```
$scope.$emit("message", "my message");
```

Exemple sur jsfiddle

Exemple

```
<body ng-app="app">

<div  ng-controller="ParentCtrl">
    {{ message }}
    <button ng-click="send()">Send</button>

    <div  ng-controller="ChildrenCtrl">
        {{ message }}
        <button ng-click="send()">Send</button>
    </div>

</div>
```

$broadcast

$emit

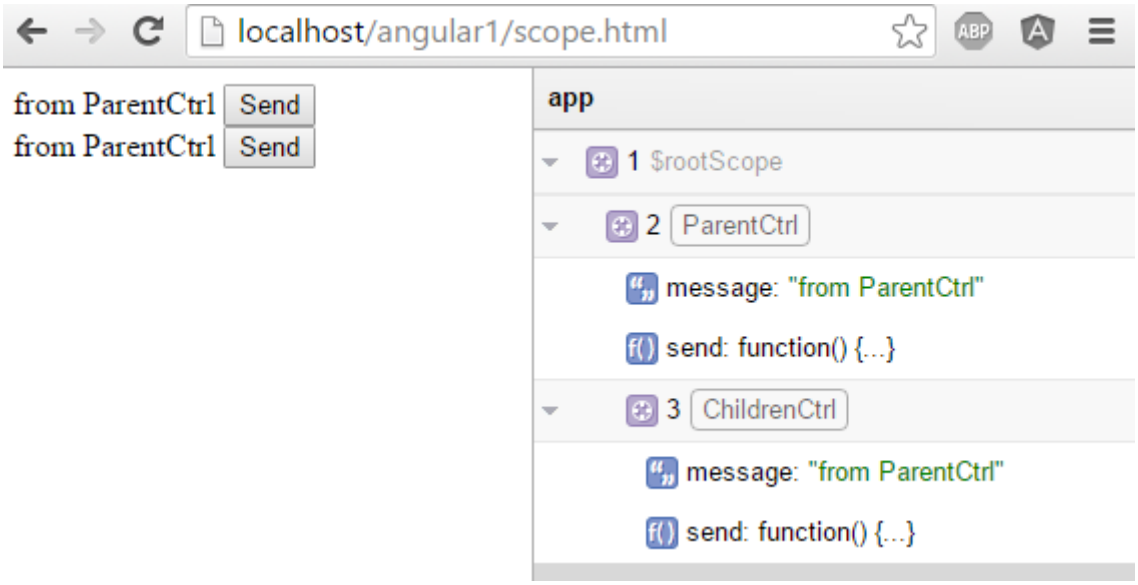
```
var app = angular.module("app", []);
app.controller('ParentCtrl', ParentCtrl);
ParentCtrl.$inject = ['$scope'];
function ParentCtrl ($scope) {
    $scope.$on("message", function(e, message) {
        $scope.message = message;
    });

    $scope.send = function () {
        $scope.$broadcast("message", "from ParentCtrl");
    }
}

app.controller('ChildrenCtrl', ChildrenCtrl);
ChildrenCtrl.$inject = ['$scope'];
function ChildrenCtrl ($scope) {
    $scope.$on("message", function(e, message) {
        $scope.message = message;
    });

    $scope.send = function () {
        $scope.$emit("message", "from ChildrenCtrl");
    }
}
```

- **$watch** pour observer les changements sur une propriété du scope

```
$scope.message = "my message";
$scope.$watch('message',function (newVal,oldVal,scope) {
    console.log(newVal);
});
```

- **$apply** (et $digest) pour valider des changements. Exemple

Extension Chrome pour observer les différents scopes : ng-inspector



18. jqLite

Documentation

La function « element() » est un sélecteur à la « manière » de jQuery

Obtenir le document

```
angular.element(document)
```

Obtenir un élément dans la page

Exemple obtention d'un élément avec la classe CSS « .page-title »

```
var title_el = angular.element(document.querySelector('.page-title'))
```

Obtenir le contrôleur d'un élément

Exemple le contrôleur défini sur le body (depuis la console Chrome)

```
var ctrl = angular.element(document.querySelector('body')).controller()
```

19. « Binding »

Expressions

```
{{ message }}
```

Filtres

On peut ajouter un ou plusieurs filtres à une expression

```
{{ "Bonjour" | uppercase }}
```

Pour filtrer, trier et formater une expression

Expression|filtername:parameter

Principaux filtres :

currency	{{expression|currency : "USD$"}}
date	{{expression|date : "short"}}
filter	Person in people|filter :searchTerm
json	{{expression|json }}
lowercase,uppercase	{{expression|uppercase}}
number	{{expression| number :1}}
orderby	Person in people|orderBy'LastName'
limitTo	Person in people|limitTo :10

Exemples

- Currency
- Date
- Number
- « filter » (string, object ou function)
- OrderBy
- Json
- LimitTo(array)

Custom Simple, avec paramètre, avec dépendance

Built-in directives

Documentation

ng-bind (lecture seule)

On pourrait utiliser une expression {{ title }}

```
<span ng-bind="title"></span>
```

ng-model pour l'édition

```
<input ng-model="title" />
```

Note : si la propriété liée à ng-model n'existe pas elle est créée.

ng-repeat (sur tableau)

```
<div ng-repeat="article in articles">
    <article id="{{ article.id }}">
        <header>
            <h2>{{ article.title }}</h2>
        </header>
        <p>{{ article.content }}</p>
        <a href="#/articles/{{ article.id}}">Lire la suite</a>
    </article>
</div>
```

ng-if

Documentation

Le « contenu » ne sera ajouté à la page que si la condition est respectée. C'est la ou cela diffère de ng-show/ng-hide qui se contente de modifier les CSS pour afficher ou non le « contenu »

ng-show et ng-hide

Pour afficher/cacher un élément selon une condition. Les nombres égal à 0, les variables undefined ou null, chaines de caractères vides renvoient false.

Exemple affiche seulement si l'élément est défini

```
<div ng-show="selectedPerson"></div>
```

ng-click

Pour notifier le contrôleur. Pour déclencher une fonction de son contrôleur par exemple

```
<button ng-click="save()"></button>
<button ng-click="selectPerson(person)"></button>
```

ng-src

Pour image.On peut définir une expression pour modifier l'image par exemple

```
<img ng-src="{{person.img}}" />
```

ng-include

Permet d'insérer un template d'un fichier html ou défini dans la page.

- Template d'un **fichier** html inséré

- Template dans la page

```
<div ng-include="'itemTmpl.html'"></div>

<script type="text/ng-template" id="itemTmpl.html">
<!-- code retiré pour la clarté -->
</script>
```

- Template avec switch selon la **condition**

Note il faut ajouter l'extension « .html » malgré que ce soit dans la page.

```
<div ng-include="isEditing ? 'editTmpl.html' : 'itemTmpl.html'"></div>

<script type="text/ng-template" id="itemTmpl.html">
<!-- code retiré pour la clarté -->
</script>
<script type="text/ng-template" id="editTmpl.html">
<!-- code retiré pour la clarté -->
</script>
```

ng-bind-html

Sert à afficher du contenu au format html

Installer « **angular-sanitize** ». Les scripts (XSS) seront automatiquement supprimés.

```
bower install angular-sanitize --save
```

Exemple

```
<body ng-app="app" ng-controller="MyCtrl">

<h1 ng-bind-html="title">

<script src="bower_components/angular/angular.js"></script>
<script src="bower_components/angular-sanitize/angular-
sanitize.js"></script>
<script>

    (function (){

        angular.module("app", [ 'ngSanitize']).controller('MyCtrl',
MyCtrl);
        MyCtrl.$inject = ['$scope'];
        function MyCtrl ($scope) {
            $scope.title = 'Mon <em>titre</em>';
        }

    })();

</script>
```

On ajoute un champ lié au « title » en édition

```
<input ng-model="title">
```

Mon *titre*

Mon titre

Si on modifie la sortie le champ en édition en y ajoutant du JavaScript exemple :

```
Mon <em onmouseover="alert()">titre</em>
```

… le **script est supprimé**

$sce

On peut également utiliser $sce (sans « angular-sanitize ») mais c'est plus dangereux car ici le script ne sera pas supprimé

```
▼<h1 ng-bind-html="getTrustedTitle()" class="ng-binding"> == $0
    "Mon "
    <em onmouseover="alert()">titre</em>
  </h1>
  <input ng-model="title" class="ng-valid ng-not-empty ng-dirty ng-valid-parse ng-touched">
```

```
<body ng-app="app" ng-controller="MyCtrl">

<h1 ng-bind-html="getTrustedTitle()"></h1>
<input ng-model="title">

<script src="bower_components/angular/angular.js"></script>
<script>

    (function (){

        var app = angular.module("app", []);
        app.controller('MyCtrl', MyCtrl);
        MyCtrl.$inject = ['$scope','$sce'];
        function MyCtrl ($scope, $sce) {
            $scope.title = 'Mon <em>titre</em>';
            $scope.getTrustedTitle = function () {
                return $sce.trustAsHtml($scope.title);
            }
        }

    })();

</script>
```

Formulaires

- « ng-show » / « ng-hide »
- « ng-class »
- « ng-click »
- « ng-disabled »
- « required »
- « ng-minlenght » / « ng-maxlength »
- « ng-pattern »
- Sur input
- « ng-model »
- « ng-bind »
- « ng-sanitize »
- « ng-blur » / « ng-focus »

```
<style>
    /* ng-dirty quand le formulaire est modifié */
    form.ng-dirty {
        background-color: lightsalmon;
    }
    /* ng-dirty quand le formulaire n'a pas encore été modifié */
    form.ng-pristine {
        background-color: lightgrey;
    }
    input.ng-valid {
        border-color: lightgreen;
    }
    input.ng-invalid {
        border-color: lightcoral;
    }
    .has-error {
        background: red;
    }
</style>
```

Styles appliqués si un champ est valide/ invalide

Exemple un style appliqué avec ng-class

JavaScript

```
<script src="bower_components/angular/angular.js"></script>
<script>

    (function (){

        angular.module("app", []).controller('MyCtrl', MyCtrl);
        function  MyCtrl($scope) {
            $scope.user = {
                username : '',
                email: "some-address@gmail.com"
            };

            $scope.save = function () {

            };
        }
        MyCtrl.$inject = ['$scope'];

    })();

</script>
```

HTML

```
<body ng-app="app"  ng-controller="MyCtrl">

    <form name="userForm" role="form">
        <div class="form-group" ng-class="{ 'has-error': userForm.$invalid }">
            <label for="username">Username:</label>
            <input type="text" name="username" id="username" ng-minlength="3" ng-
maxlength="10" class="form-control" ng-model="user.username" />
            <p class="help-block" ng-show="userForm.username.$error.maxlength">
                Username cannot be greater than 10 characters.
            </p>
            <p class="help-block" ng-show="userForm.username.$error.minlength">
                Username cannot be less than 3 characters.
            </p>
        </div>
        <div class="form-group" ng-class="{ 'has-error': userForm.email.$invalid }">
            <label for="email">Email:</label>
            <input type="email" name="email" id="email" required class="form-control"
placeholder="Enter your email address." ng-model="user.email" />
            <p class="help-block" ng-show="userForm.email.$error.required">
                Email is Required.
            </p>
            <p class="help-block" ng-show="userForm.email.$error.email">
                Email is invalid.
            </p>
        </div>
        <div class="form-group">
            <button type="button" class="btn">Cancel</button>
            <button type="submit" class="btn" ng-disabled="!userForm.$dirty ||
userForm.$invalid" ng-click="save()">
                Save
            </button>
        </div>
    </form>
```

ng-minlength et ng-maxlenght + messages erreurs affichés

ng-class

ng-model

ng-show affichage des messages d'erreur

ng-click

ng-disabled pour rendre le bouton cliquable que si le formulaire n'a pas d'erreur

Exemple « **ng-pattern** »

```
<input type="text" name="textValue" id="textValue" ng-
pattern="/^\(\d{3}\)\s*\d{3}\-\d{4}$/" class="form-control" ng-
model="data.textValue" />
```

Exemple « **ng-blur** » , « **ng-focus** »

```
<input type="text" name="inputOne" ng-blur="onBlur('inputOne')" ng-
focus="ctrl.onFocus('inputOne')">
```

Formulaire avec « **ng-submit** »

```
<form name="userForm" ng-submit="userForm.$dirty && userForm.$valid &&
save()">
```

ng-message affiché avec la condition « when »

Documentation

Vérifier qu'un formulaire est valide en code

```
<form name="loginForm" ng-submit="login.login(loginForm)" novalidate>
```

```
$scope.login = function(form) {
    if (form.$valid) {
        // ...
    }
}
```

Custom directive

Documentation

Exemple

```
var app = angular.module("app", []);
app.directive('myDirective', function () {
            return {
                template :`
                <div>
                    <h1>Hello</h1>
                </div>
                `
            }
})
```

Ecritures possibles

```
<div my-directive></div>
<div my-directive=""></div>
<div data-my-directive></div>
<div my:directive></div>
<div my_directive></div>
```

Restrict

- « A » attribut **<div my-directive> **
- « C » classe css **<div** class="**my-directive**"**> **
- « M » commentaire
- « E » Element **<my-directive> **

```
app.directive('myDirective', function () {

    return {
        restrict : 'ACME',
        template :`
        <div>
            <h1>Hello</h1>
        </div>
       `
    }
})
```

« **template** » ou « **templateUrl** »

```
app.directive('myDirective', function () {
    return {
        restrict : 'ACME',
        templateUrl : 'views/mydirective.html'
    }
})
```

Scope

- « < » one way
- « = » two way
- « @ » attribut
- « & » function

« compile » ou « link » est exécuté pour chaque instance

```
<my-directive></my-directive>
```

```
app.directive('myDirective', function () {
    return {
        restrict : 'E',
        scope : {},
        template : '{{ mymessage }}',
        link : function (scope, element, params) {
            scope.mymessage = "Hello";
        }
    }
})
```

Documentation $compile

```
app.directive('myDirective', function () {
    return {
        restrict : 'E',
        scope : {},
        template : '{{ mymessage }} </br>',
        compile: function compile(tElement, tAttrs, transclude) {
            return {
                pre: function preLink(scope, iElement, iAttrs, controller) {

                },
                post: function postLink(scope, iElement, iAttrs, controller) {
                    count ++;
                    scope.mymessage = "Item " + count;
                }
            }
            // or
            // return function postLink( ... ) { ... }
        }
    }
})
```

20. Services

a. Promises avec « $q »

```
Service.prototype.addContact = function (contact) {
    var deferred = this. $q.defer(),
          _this = this;
    this._$http.post(this._url, contact)
            .then(function (result) {
                deferred.resolve(result.data);
                _this.publish();
            }, function (rejection) {
                deferred.reject(rejection);
            });
    return deferred. promise;
};
```

« success »

« error »

On retourne une « promise »

b. $http

Inecter $http (et $q) dans le service

```
Service.prototype.listContacts = function () {
    this._$http.get(this._url)
            .then(function (result) {
                        return result.data;
            });
};
```

Avec « $q »

```
Service.prototype.listContacts = function () {
    var deferred = this._$q.defer();
    this._$http.get(this._url)
            .then(function (result) {
                        deferred.resolve(result.data);
                    },
                    function (rejection) {
                        deferred.reject(rejection);
                    });
    return deferred.promise;
};
```

Depuis le contrôleur cela pourrait ressembler à

```
service.methodeOne().then(successCallback,errorCallback);
```

c. Cache

Documentation

d. $resource

Installer avec bower angular-resource et ajouter la dépendance « ngResource »

```
bower install angular-resource --save
```

```
function Service($resource, $q) {
    this._url = url;
    this.ContactTemplate = $resource(
            url + ":contactId",
            { contactId: '@id' },
            { 'update': { method: 'PUT' } });
    this._$q = $q;
    this._notify = [];
}

Service.prototype.listContacts = function () {
    var deferred = this._$q.defer(),
            query = this.ContactTemplate.query(function() {
                deferred.resolve(query);
            }, function(rejection) {
                deferred.reject(rejection);
            });
    return deferred.promise;
};

Service.prototype.addContact = function (contact) {
    var deferred = this._$q.defer(), _this = this;
    var newContact = new this.ContactTemplate(contact);
    newContact.$save(function() {
        deferred.resolve(newContact);
        _this.publish();
    }, function(rejection) {
        deferred.reject(rejection);
    });
    return deferred.promise;
};
```

21. Components (Angular >= 1.5)

Documentation

Avec **directive**

```
app.directive('helloWorld', function helloWorld () {
  return {
    restrict        : 'E',
    scope           : {},
    bindToController  : {
      name : '@'
    },
    controller: function helloWorldCtrl () {
      this.logName = function () {
        console.log(this.name);
      }
    },
    controllerAs      : 'hw',
    template        : '<div><span ng-click="hw.logName()">Hello
{{hw.name}}!</span></div>'
  }
});
```

```
<hello-world name="my name"></hello-world>
```

Avec **component**

```
app.component('helloWorld', {
    bindings: {
        name: '@'
    },
    controller : function helloWorldCtrl () {
        this.logName = function () {
            console.log(this.name);
        }
    },
    template : '<div><span ng-click="$ctrl.logName()">Hi
{{$ctrl.name}}!</span></div>'
});
```

a. Création d'un component avec « template »

```
<body ng-app="app">

<component-one></component-one>
```

Par défaut on utilise $ctrl

```
app.component('componentOne', {
    template : '<h1>{{ $ctrl.title }}</h1>',
    controller : function () {
        var $ctrl = this;
        $ctrl.title = 'Hello';
    }
});
```

b. Renommer l'instance du controller « As »

Exemple en « model »

On utilise « model » dans le template

```
app.component('componentOne', {
    template : '<h1>{{ model.title }}</h1>',
    controllerAs : 'model',
    controller : function () {
        var model = this;
        model.title = 'Hello';
    }
});
```

c. Controller avec dépendance

```
app.component('componentOne', {
    template : '<h1>{{ $ctrl.title }}</h1>',
    controller :['$http', function ($http) {

    }]
});
// ou
function ComponentOneCtrl($http) {

}
app.component('componentOne', {
    template : '<h1>{{ $ctrl.title }}</h1>',
    controller :['$http', ComponentOneCtrl]
});
```

d. $onInit

Chargement de données à l'initialisation du component

```
var app = angular.module("app", []);
app.service('myService',myService);
function myService($http) {
    this.getItems = function () {
        return $http.get('data.json').then(function (response) {
            return response.data;
        })
    };
}
myService.$inject = ['$http'];

app.component('componentOne', {
    templateUrl : 'components/componentone.html',
    controller :['$http', 'myService', function ($http, myService) {
        var $ctrl = this;
        $ctrl.title = 'Hello';

        $ctrl.$onInit = function () {
            myService.getItems().then(function (items) {
                $ctrl.items = items;
            });
        }
    }]
});
```

Un fichier json quelconque

```
[
  {
    "id": 1,
    "title": "Point Break"
  },
  {
    "id": 2,
    "title": "Miami Vice"
  }
]
```

Et le template

```
<h1>{{ $ctrl.title }}</h1>
<ul ng-repeat="item in $ctrl.items">
<li>{{ item.title }}</li>
</ul>
```

Autres events : $onChanges, $onDestroy

e. Bindings

« Equivalent » de « scope » pour les directives

Exemple on récupère l'attribut (@) « message » que l'on réutilise pour sa valeur dans le template

```
<body ng-app="app">

<component-two message="My message"></component-two>
```

```
var app = angular.module("app", []);
app.component('componentTwo', {
    template : '{{ $ctrl.message }}',
    bindings : {
        message : '@'
    }
});
```

f. Sous component

Exemple dans le template d'un « componentOne » on ajoute « componentTwo »

```
<h1>{{ $ctrl.title }}</h1>

<component-two message="My message"></component-two>
```

g. Routing avec components

« Classique » avec « angular-route »

Si on veut indiquer seulement des components pour les routes, remplacer simplement le template des routes par les components

```
<body ng-app="app">

<div ng-view></div>

<script src="bower_components/angular/angular.js"></script>
<script src="bower_components/angular-route/angular-route.js"></script>
<script>

    (function () {

        var app = angular.module("app", [ 'ngRoute']);
        app.config(function ($routeProvider) {
            $routeProvider
                    .when('/',{
                        template: '<a href="#/one">One</a><a href="#/two">Two</a>'
                    })
                    .when('/one', {
                        template: '<component-one></component-one>'
                    })
                    .when('/two', {
                        template: '<component-two message="My message"></component-two>'
                    })
                    .otherwise({redirectTo: "/"});

        });

        app.component('componentOne', {
            template : '<h1>Hello</h1>',
        });
        app.component('componentTwo', {
            template : '{{ $ctrl.message }}',
            bindings : {
                message : '@'
            }
        });

    })();

</script>
```

Avec « angular-component-router »

Documentation

Compatible Angular 2

Installer avec bower

```
bower install angular-component-router --save
```

On définit le routing pour un component, il faut donc avoir un component « de base » contenant les autres components. Exemple « componentApp »

```
<body ng-app="app">

<component-app></component-app>

<script src="../bower_components/angular/angular.js"></script>
<script src="../bower_components/angular-component-
router/angular_1_router.js"></script>
<script>

    (function () {

        var app = angular.module("app", ['ngComponentRouter']);
        app.value("$routerRootComponent", "componentApp");

        app.component("componentApp", {
            templateUrl: 'components/componentapp.html',
            $routeConfig: [
                { path: "/one", component:"componentOne", name: "One" },
                { path: "/two/:id", component:"componentTwo", name: "Two"
},
                { path: "/**", redirectTo: ["One"] }
            ]
        });

        app.component('componentOne', {
            template : '<h1>One</h1>'
        });

        app.component('componentTwo', {
            template : '<h1>Two</h1>{{ $ctrl.message }} ',
            controller : function () {
                var $ctrl = this;
                $ctrl.$routerOnActivate = function (next) {
                    $ctrl.message = next.params.id;
                }
            }
        });

    })();

</script>
```

Dépendance + ajout value sur le « main component »

Routing du component

Récupération du paramètre de route

Template de « componentApp »

```
<a ng-link="['One']">One</a>
<a ng-link="['Two', {id : 10 }]">Two</a>
<ng-outlet></ng-outlet>
```

Liens avec ng-link + nom donné à la route et avec paramètre de route

Conteneur pour les components / navigation

Navigation par programmation depuis un component

```
app.component('componentOne', {
    bindings : {
        '$router' : '<'
    },
    template : '<h1>One</h1><a href ng-click="$ctrl.nav()">Nav</a>',
    controller : function () {
        var $ctrl = this;

        $ctrl.nav = function () {
            $ctrl.$router.navigate(["Two", {id : 20 }, "Two"]);
        };

    }
});
```

h. Components liés

Exemple avec un component accordéon ayant des panels.

```
app.component("accordion", {
    controller: function() {
        var $ctrl = this;
        $ctrl.addPanel = function(panel) {

        };
    },
    // etc.
});
app.component("accordionPanel", {
    require: {
        "parent": "^accordion"
    },
    controller: function() {
        var $ctrl = this;
        model.$onInit = function() {
            model.parent.addPanel(model);
        };
    },
    // etc.
});
```

Permet d'avoir accès aux functions du controller parent