


# Routage

Le module **ngRoute** d'angularJS permet d'associer le chargement de contenus à des URLs prédéfinies.

## -- Chargement du module

Le module ngRoute n'est pas chargé par défaut avec AngularJS, il faut le faire explicitement :

```
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/x.x.x/angular-route.min.js"></
script>
```

## -- Intégration de ngRoute à un module

Pour utiliser ngRoute dans un module, il faut déclarer la dépendance au module **ngRoute** dans la déclaration du module :

```
angular.module("sampleApp", ['ngRoute']);
```

## -- Création des vues

### -- Vue principale

La vue principale doit déclarer doit comporter une directive (une seule) **ngView** dans laquelle seront chargées les vues secondaires définies par le routage.

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

La page principale comporte également des liens qui seront définis par la suite dans la configuration du routeur :

```
<a href="#/route1">Route 1</a><br/>
<a href="#/route2">Route 2</a><br/>
```

```
<!DOCTYPE html>
<html ng-app="sampleApp">
<head>
<meta charset="UTF-8">
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.10/angular.min.js"></script
>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.10/angular-route.min.js"></
script>
</head>
<body>
<a href="#/route1">Route 1</a><br/>
<a href="#/route2">Route 2</a><br/>
<div ng-view></div>
</body>
</html>
```

### -- Vues secondaires

Les vues secondaires sont les templates qui seront associés aux URLs /route1 et /route2 :

```
<h1>Route1</h1>
<div ng-bind="content1"></div>
```

## -- Configuration du routage

La configuration du routage se fait par l'appel de la méthode **config** du module, à laquelle on injecte le service **$routeProvider**

### -- Création d'une route

```
angular.module("sampleApp").config(['$routeProvider',
       function($routeProvider) {
           $routeProvider.
               when('/route1', {
                   templateUrl: 'views/route1-template.html',
                   controller: 'RouteController'
               });
       }]);
```

Si l'url **#/route1** est demandée, le template **views/route1-template.html** est chargé, et le contrôleur **RouteController** sollicité.

Ajout du contrôleur gérant le template :

```
angular.module("sampleApp").controller("RouteController",["$scope",function($scope)
{
    $scope.content1="Contenu du template de route1"
}]);
```

**Résultat obtenu :**




### -- Route avec paramètre passé

```
angular.module("sampleApp").config(['$routeProvider',
       function($routeProvider) {
           $routeProvider.
               when('/route1', {
                   templateUrl: 'views/route1-template.html',
                   controller: 'RouteController'
               }).
               when('/route2/:nom', {
                   templateUrl: 'views/route2-template.html',
                   controller: 'RouteController'
               });
       }]);
```

Le paramètre est passé à la l'url **/route2/** par la variable **:nom**

Il faut ensuite modifier le contrôleur **RouteController** et lui injecter le service **$routeParams** pour qu'il puisse récupérer les paramètres passés :

```
angular.module("sampleApp").controller("RouteController",["$scope","$routeParams",f
unction($scope,$routeParams){
    $scope.content1="Contenu du template de route1";
    $scope.params=$routeParams;
}]);
```

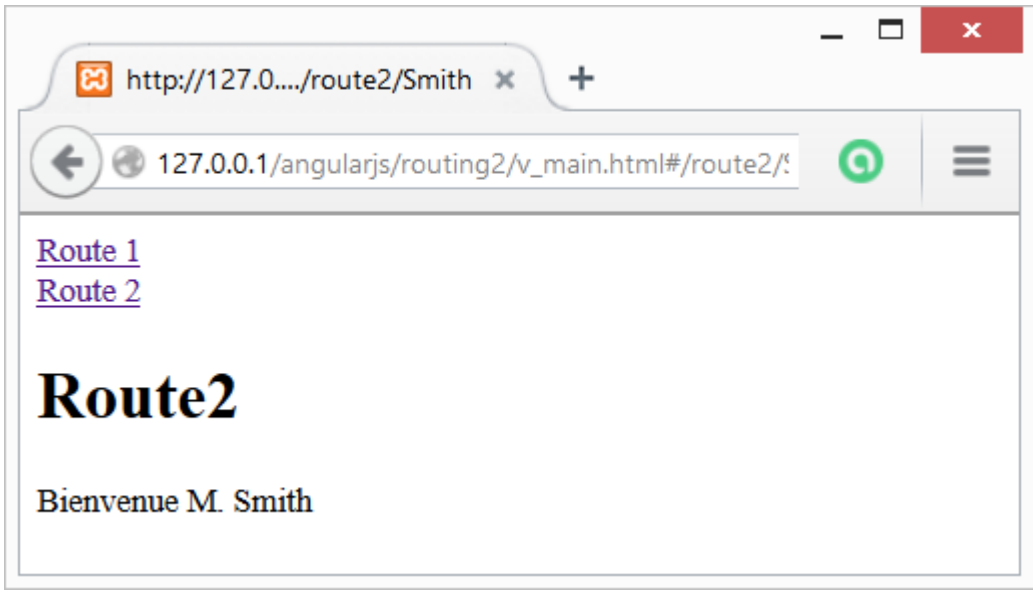
Création du template correspondant à la **route2**, et utilisant la variable **params** déclarée dans le scope et récupérant le paramètre passé :

```
<h1>Route2</h1>
<div>Bienvenue M. {{params.nom}}</div>
```

Modification du lien dans la vue principale :

```
...
<body>
<a href="#/route1">Route 1</a><br/>
<a href="#/route2/Smith">Route 2</a><br/>
<div ng-view></div>
</body>
...
```

Résultat :




Il est également possible de passer des paramètres de manière plus classique dans l'URL :

```
...
<body>
<a href="#/route1">Route 1</a><br/>
<a href="#/route2/Smith?prenom=John">Route 2</a><br/>
<div ng-view></div>
</body>
...
```

```
<h1>Route2</h1>
```

```
<div>Bienvenue M. {{params.prenom}} {{params.nom}}</div>
```

## -- Route par défaut

Il est possible de définir la route par défaut : celle qui sera utilisée si toutes les autres routes ne trouvent pas leur correspondance :

```
angular.module("sampleApp").config(['$routeProvider',
       function($routeProvider) {
           $routeProvider.
               when('/route1', {
                   templateUrl: 'views/route1-template.html',
                   controller: 'RouteController'
               }).
               when('/route2/:nom', {
                   templateUrl: 'views/route2-template.html',
                   controller: 'RouteController'
               }).otherwise({
                redirectTo: '/route1'
            });
       }]);
```

Ajout d'un lien vers une route inexistante pour utiliser la route par défaut :

```
...
<body>
<a href="#/route1">Route 1</a><br/>
<a href="#/route2/Smith">Route 2</a><br/>
<a href="#/route3">Route inexistante</a><br/>

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

Le lien inexistant vers **#/route3** conduit bien maintenant à la route par défaut **route1**

## -- Masquage du # dans l'URL

### -- Activation du mode HTML5

Pour masquer le # dans les urls utilisées, il est nécessaire d'injecter le service **$locationProvider** dans la configuration du routeur, pour activer le mode html5 :

```
angular.module("sampleApp").config(['$routeProvider','$locationProvider',
       function($routeProvider,$locationProvider) {
           $routeProvider.
```

```
            when('/route1', {
                templateUrl: 'views/route1-template.html',
                controller: 'RouteController'
            }).
            when('/route2/:nom', {
                templateUrl: 'views/route2-template.html',
                controller: 'RouteController'
            }).otherwise({
             redirectTo: '/route1'
         });
        if(window.history && window.history.pushState){
             $locationProvider.html5Mode(true);
           }
    }]);
```

**-- Modification des URLs**

Il est ensuite nécessaire de modifier les liens définis vers les routes dans v_main.html :

```
...
<body>
<a href="route1">Route 1</a><br/>
<a href="route2/Smith?prenom=John">Route 2</a><br/>
<a href="route3">Route inexistante</a><br/>
<div ng-view></div>
</body>
...
```

**-- Correction de l'erreur 404 sur accès direct aux routes**

Le seul problème reste le cas de l'accès direct aux routes par l'URL, ou le rafraîchissement de la page avec la touche F5 du clavier :

La réponse dans ce cas abouti à une **erreur 404**. Pour éviter cette réponse et maintenir un fonctionnement correct, il est nécessaire de modifier la configuration côté server :

1. Activer le **mod_rewrite** d'apache (ou vérifier son activation avec phpinfo)
2. ajouter un fichier .htaccess contenant les lignes suivantes dans le dossier du projet :

```
RewriteEngine on

# Pas de redirection pour les dossiers ou fichiers existants
RewriteCond %{REQUEST_FILENAME} -f [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^ - [L]
# Redirige toutes les autres url vers v_main.html pour permettre le mode HTML5
RewriteRule ^ v_main.html [L]
```

## -- Conservation de variables entre changements de vues

Modifier le template associé à la route2 de façon à avoir une nouvelle variable dans le scope, dont le contenu est modifiable par une zone de texte :

```
<h1>Route2</h1>
<div>Bienvenue M. {{params.nom}} {{params.prenom}}</div>
<div><label>Entrez votre code : <input type="text" ng-model="code"></label></div>
```




La variable code saisie dans la zone de texte est perdue à chaque changement de vue, angularJS créant un nouveau scope à chaque chargement. Si on souhaite conserver la valeur du code saisie dans la vue correspondant à la route 2, il est nécessaire d'utiliser les closures javascript.

AngularJS fournit un outil à cet effet, il s'agit des factories :

### -- Création d'une factory

```
...
angular.module("sampleApp").factory("code", function() {
    return {
        value:"noCode"
    }
})
```

La factory **code** retourne dans notre cas un objet possédant un membre **value** initialisé à “**noCode**”.

## -- Injection de factory

La factory **code** est ensuite injectée en tant que service dans le controller :

```
angular.module("sampleApp").controller("RouteController",["$scope","$routeParams","
code",function($scope,$routeParams,code){
    $scope.content1="Contenu du template de route1";
    $scope.params=$routeParams;
    $scope.code=code;
}]);
```

La variable **code** du $scope correspond à la factory **code**, il ne reste plus qu'à faire référence à **code.value** dans la vue :

```
<h1>Route2</h1>
<div>Bienvenue M. {{params.nom}} {{params.prenom}}</div>
<div><label>Entrez votre code : <input type="text" ng-
model="code.value"></label></div>
```

En cas de changement de vue (passage de route2 à route1 puis retour à route2), les modifications opérées sur la variable code sont conservées (grâce à la closure créée par la factory).

## -- Autres directives

D'autres directives existent, permettant de gérer le contenu et/ou les templates affichés, mais elles n'utilisent pas le routage :

- ng-include
- ng-switch

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

Permanent link:
**http://slamwiki2.kobject.net/slam4/richclient/angularjs/routing?rev=1425371694**

Last update: **2019/08/31 14:39**