Múltiples vistes i routing

Instal·lació

En la pàgina principal del fitxer HTML tindrem que carregar el fitxer JavaScript mitjançant:

```
1    <script src='//ajax.googleapis.com/ajax/libs/angularjs/1.2.19/angular-route.js'></script>
```

I al crear el mòdul de la nostre aplicació tindrem que fer que depengui de ell:

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

Dit això, anem a incloure **$routeProvider en el nostre codi, actualitzant el nostre script.js** de la següent manera:

```
// script.js



    // afegim ngRoute per a totes les necessitats de enrutament
    var demoApp = angular.module('demoApp', ['ngRoute']);


    // configurem les nostres rutes
    demoApp.config(function($routeProvider) {
  $routeProvider


            // ruta per la pagina principal
            .when('/', {
                templateUrl : 'pages/home.html',
                controller  : 'mainCtrl'
            })
```

```
            // ruta per la pagina about
            .when('/about', {
                templateUrl : 'pages/about.html',
                controller  : 'aboutCtrl'
            })

            // ruta per la pagina de contact
            .when('/contact', {
                templateUrl : 'pages/contact.html',
                controller  : 'contactCtrl'
            });
    });

    // creem el controlador I injectem el Angular $scope
demoApp.controller('mainCtrl', function($scope) {
        // crearem el missatge per mostrar-lo en la vista
        $scope.message = 'Hola món, aquesta es la pagina home!';
    });

    demoApp.controller('aboutCtrl', function($scope) {
  $scope.message = 'Hola! Aquesta es la pagina about.';
    });

    demoApp.controller('contactCtrl', function($scope) {
  $scope.message = 'Vols contactar amb nosaltres?';
    });
```

Expliquem el codi:

- **demoApp.config(function($routeProvider) {…}**: Com ja s'ha comentat, els providers tenen que injectar-se com a funcions **config** del nostre mòdul.
- **$routeProvider.when('url', {…})**: El mètode when de $routeProvider ens permet dir-li a angular quin template carregar (templateUrl) i quin controlador utilitzar (controller) al seleccionar un enllaç a una certa url.

Plantilles de vistes

Angular ens proporciona la directiva **ng-view**, que permet injectar contingut des de templates situats en una ruta determinada (en aquest cas "/home", "/about" i "/contact") en el nostre layout.

¿Com funciona? Tenim que afegir la etiqueta allí a on volem que es renderitzin les nostres pagines, es a dir, obrim **index.html** i en el main, afegim:

```
    <div id="main">


        <!-- angular templating -->
        <!—aquí es a on el contingut serà injectat -->
        <div ng-view></div>


    </div>
```

Creant els nostres templates:

Crearem el directori **pages**, i els arxius home.html, about.html i contact.html. Els donem el següent contingut:

home.html:

```
<!-- home.html -->
    <div class="jumbotron text-center">
        <h1>Home Page</h1>
        <p>{{ message }}</p>
    </div>
```

about.html:

```
<!-- about.html -->
    <div class="jumbotron text-center">
        <h1>About Page</h1>
        <p>{{ message }}</p>
    </div>
```

contact.html:

```
<!-- contact.html -->
    <div class="jumbotron text-center">
        <h1>Contact Page</h1>
        <p>{{ message }}</p>
    </div>
```

Rutes

Ara anem a explicar què son les rutes però per això anem a canviar una mica el que són els nostres conceptes sobre que es una pàgina, una URL ,etc. Els anem a redefinir per adequar-lo a com funciona una pagina com gmail i per suposat com es treballa en AngularJS.

Realment l'aplicació de gmail te moltes *pàgines* encara que en la barra del navegador sempre estiguem en la mateixa pàgina. I els fragments de la URL realment ens indiquen aquestes noves *pàgines* que voldrem mostrar. Llavors ¿què es una ruta?, una ruta es simplement el nom d'un fragment però que ara ja no va a fer referencia a una part dins de la pàgina, sinó que ara al modificar la ruta (el fragment), li indica una nova pàgina que carregar dins de la pàgina *real* que estem veient.
Ara podem tornar a mostrar la taula anterior de la següent forma:

Pàgina	Ruta
Recibidos	**inbox**
Importantes	**imp**
Enviados	**sent**

Ara tenim una pàgina real que es https://mail.google.com/mail/u/0/ i dins d'aquesta carreguem noves pàgines depenent de la ruta (fragment) que posem. A aquesta forma de treballar s'anomena:

- Single-Page Application (SPA)
- Single-Page Interface (SPI)

¿Quines avantatges te aquesta forma de treballar?

- La principal avantatja es la rapidesa. Les aplicacions actuals tenen molt codi CSS, JavaScript , etc. Al tenir una única pàgina real, es carrega tot això al principi un sol cop i quan canviem de *pàgina* tot lo anterior ja esta carregat, i només canviem el HTML de la nova *pàgina*.
- Un altre avantatge es que podem mantenir el nostre estat en variables JavaScript ja que mai sortirem de la pàgina, el que simplifica molt el desenvolupament. ¿Recordeu quan navegar a una altre pàgina podia implicar el passar moltes coses en la URL, per passar el estat de l'aplicació d'una pàgina a una altre?

Modes de rutes

Per utilitzar les rutes en AngularJS el primer que necessitarem serà configurar-lo per poder associar la ruta a la pàgina HTML, que es te que carregar al navegar cap aquesta ruta. Per això tenim el servei $routeProvider de AngularJS.
Aquest servei te 2 mètodes:

- `when` : Associa una ruta amb una pàgina HTML
- `otherwise`: Indica en quina ruta tenim que redirigir-nos quan hi ha una ruta desconeguda

when

El mètode `when` de `$routeProvider` permet associar una ruta a una pàgina HTML, encara que com a mínim també s'indica el controlador que s'utilitzarà en aquesta pàgina.
El mètode `when` accepta dos paràmetres:

- `path`: Es el nom de la ruta, es a dir el que es posarà darrera el símbol "#".
- `route`: Es un objecte que defineix la pàgina HTML a carregar i el controlador que s'utilitzarà en aquesta pàgina HTML. El objecte te que tenir les següents propietats:
  - `templateUrl`: Nom de la pàgina HTML. Te que ser una ruta deferida respecte a la pàgina HTML principal que inclourà aquesta pàgina HTML.
  - `controller`: Es el nom del controlador de AngularJS que s'utilitzarà amb aquesta pàgina.

Veiem un exemple:

```
1  var path='/pagina1';
2
3  var route={
4    templateUrl: "pagina1.html",
5    controller: "Pagina1Controller"
```

```
6   }
7
8   $routeProvider.when(path,route);
```

- Línia 1: Definirem primer el path corresponent a la ruta.
- Línia 3: Definirem el objecte JavaScript que conte les dades de la ruta, essent la pàgina `pagina1.html` i el controlador que utilitzem `Pagina1Controller`
- Línia 8: Ara es quan li diem a AngularJS el path i la definició de la ruta.

En aquest exemple, si el usuari afegeix a la pàgina principal "#/pagina1" es carregarà dins de la pàgina principal la pàgina HTML anomenada `pagina1.html` i com a controlador d'aquesta nova pàgina s'utilitzarà `Pagina1Controller`.

Es podria posar dins de la nova pàgina la directiva `ng-controller` i especificar dins de la pròpia pàgina el controlador en comptes de definir la ruta, encara que mai ho he vist en ningun exemple de AngularJS, i per lo tant no recomano fer-ho i no es la forma habitual de treballar.
Recordeu que encara que el path sigui `/pagina1` i la pàgina s'anomeni `pagina1.html` realment no te perquè coincidir per a res amb els noms, encara que per coherència acostumen a ser similars.
No tindria molt sentit navegar a la ruta `/ventas` i que es carregues la pàgina `compras.html`, encara que no hi ha res que ho impedeixi desenvolupar-ho així.

El codi anterior habitualment s'acostuma a compactar-se de la següent forma i es com l'utilitzarem a partir de ara:

```
1   $routeProvider.when('/pagina1', {
2     templateUrl: "pagina1.html",
3     controller: "Pagina1Controller"
4   });
```

otherwise

En el cas de que el usuari indiqui una ruta que no existeix o simplement no posi ninguna ruta, la pàgina principal es podria quedar en blanc ja que no carregaria ninguna pàgina. Per evitar això AngularJS permet, en cas de que no indiquem ninguna ruta o que la ruta no existeix, que es navegui a una ruta predeterminada. Per això utilitzarem el mètode `otherwise`.
El mètode `otherwise` accepta un únic paràmetre amb un objecte, tenint en el objecte la propietat `redirectTo` amb el path de la ruta a la que es navegarà.
Veiem un exemple:

```
1   $routeProvider.otherwise({
2         redirectTo: '/pagina1'
3   });
```

El que fem es dir-li a AngularJS que en cas de que no indiquem ruta o la ruta sigui desconeguda, seria com si la ruta correspongués a `/pagina1`.

ng-view

Ja hem vist com definir les rutes però ens falta una cosa per especificar, ¿en quina part de la pàgina principal es carrega la pàgina HTML de cada ruta? Podríem pensar que la pàgina carregada ocuparà tota la pàgina principal, però, encara que podria fer-se, no es lo normal.

Si mirem les pantalles de gmail, la part esquerra y la superior no canvien en cada ruta per lo que tot aquest HTML tindria que estar en la pàgina principal. Per lo tant, cada pàgina de la ruta només es tindria de carregar en un tag, per exemple `<div>`, que habilitàrem per aquest fi. La forma de dir-li a Angular a on te que carregar la pàgina HTML de la ruta, es simplement afegint la directiva ng-view al element a on desitgem que es carregui.
Veiem un exemple:

```
1   <!DOCTYPE html>
2   <html ng-app="app">
3
4   <head>
5     <script src="//ajax.googleapis.com/ajax/libs/angularjs/1.2.19/angular.min.js"></script>
6     <script src='//ajax.googleapis.com/ajax/libs/angularjs/1.2.19/angular-route.min.js'></sc
7     <script src="//code.angularjs.org/1.2.19/i18n/angular-locale_es-es.js"></script>
8     <script src="script.js"></script>
9   </head>
10
11  <body>
12    <h1>Aques es el títol i no canvia</h1>
13
14    <div ng-view></div>
15
16    <h3>Això es el peu de pagina i no canvia</h3>
17
18  </body>
19
20  </html>
```

- Líneas 5, 6, 7 y 8: Carreguem un sol cop, tot el JavaScript i així estalviem temps al no tenir que tornar a carregar-lo en cada nova ruta a la que naveguem.
- Línia 12: Aquesta es la capçalera de l'aplicació i com no canvia entre las diferents pàgines la posarem en la pàgina principal.
- Línia 14: En aquest `<div>` al incloure la directiva `ng-view` es a on es carregaran les pàgines HTML de les rutes.
- Línia 16: Aquest es el peu de l'aplicació i com no canvia entre les diferents pàgines ho situem en la pàgina principal.

Navegant

Ja tenim tot preparat però ara falta saber cóm es navega a la ruta. Hi ha dos formes:

- Link des de HTML
- Servei $location des de JavaScript

Link des de HTML

La solució es molt fàcil, realment cada ruta es un fragment de la URL actual, així que només tenim que navegar al fragment corresponent a la ruta.

Si tenim definida la ruta:

```
/pagina1
```

Podem posar un link de la següent forma:

```
1   <a href="#/pagina1">Anar a la pagina 1</a>
```

Recorda posar el coixinet "#" abans de les rutes en els enllaços.

Servei $location des de JavaScript

Des de JavaScript AngularJS te un servei que permet navegar a les rutes. Aquest servei s'anomena $location.
No descriurem totes les funcionalitats de $location. Únicament dir que te un mètode anomenat `path` al que li indiquem la ruta i fa que es navegui fins a aquesta ruta.
Si tenim definida la ruta:

```
/pagina1
```

Per exemple, en un controlador podem fer lo següent:

```
1   app.controller("Pagina1Controller", ["$scope","$location",function($scope,$location) {
```

```
2
3       $scope.irPagina1 = function () {
4           $location.path("/pagina1");
5       };
6
7   }]);
```

- Línia 1: Injectem el servei de $location
- Línia 3: Crearem una nova funció que permet navegar a la ruta ”/pagina1”
- Línia 4: Cridem al mètode `$location.path` per navegar a la ruta ”/pagina1”

Ara des de qualsevol part de la pàgina ja podem anomenar al mètode i navegar cap aquesta ruta.

Recorda que al utilitzar el servei de `$location` no s’ha de posar el coixinet ”#”