SERVICES & ROUTING

Ori calvo @2015

http://oricalvo.com
ori.calvo@gmail.com

Objectives

- Create your own services
  - service
  - factory
  - value
  - provider
- Add routing capabilities

Custom Service

- Angular offers many built-in services
- You already met some
  - $rootScope
  - $injector
- You can create your own services
  - Register them into Angular
  - Inject them into controllers/other services

Service Registration

- Angular can only inject objects it is aware of
- Angular supports different recipes for service registration
  - Values
  - Services
  - Factories
  - Constants
  - Providers
- All are singleton !!!

http://oricalvo.com
ori.calvo@gmail.com

Value

- Allows registration of a pre instantiated object
- Since object is created directly by us we cannot express a dependency list
- In practice could be used to register other libraries namespace objects

```
function Logger() { }

angular.module("myApp").value("Logger", new Logger());
```

http://oricalvo.com
ori.calvo@gmail.com

Service

- Allows for constructor function registration
- Can express dependency
- Lazy initialization
- Instantiated by Angular using “new” syntax
- Is singleton

```
function StorageService(Logger) { }

angular.module("myApp").service("StorageService", StorageService);
```

http://oricalvo.com
ori.calvo@gmail.com

Factory

- Allows for factory function registration
- The factory function should return an object instance
- The object instance may access variables declared at the factory function scope thus simulating private data
- May specify dependencies
- Is singleton

```
angular.module("myApp").factory("StorageService", function () {
     var contacts = [];
     return {
         addContact: function (contact) {
         },
         getAllContacts: function () {
         }
     };});
```

http://oricalvo.com
ori.calvo@gmail.com

Configuring a Service

- Suppose you define a service which can be configured by the application
- For example, the StorageService from previous slide may provide an “enableCaching” feature
- You need to think about the availability of the configuration API
  - Can the application change the configuration at any time ?
- In cases where one time configuration is required you should implement a provider

http://oricalvo.com
ori.calvo@gmail.com

Provider

- A service factory
- Allows the application to configure the service before the service is created
- Must conform to Angular specification
  - $get function
  - Returns the service object
- Usually is defined when implementing 3rd party Angular modules
  - Less common for application

Provider


```
angular.module("myApp").provider("StorageService", function () {
     var cachingEnabled = true;

     this.enableCaching = function (enable) {
         cachingEnabled: enable;
     };

     this.$get = function (Logger) {
         var contacts = [];
         return {
             addContact: function (contact) {
             },
             getAllContacts: function () {
             }
         };
     };
});
```

```
angular.module("myApp", [])
     .config(function (StorageServiceProvider) {
     });
```

http://oricalvo.com
ori.calvo@gmail.com

Provider Notes

- The provider is registered with name X but is requested with the name XProvider
- Provider is only accessible during application configuration phase
  - Only config block can ask for a provider
- Provider is always instantiated
  - Even if the service is not requested by application
  - Is instantiated before config block

Constant

- Even a constant value can be injectable
- Useful for sharing constant data between different providers/services
- Can be requested by a config block

```
angular.module("MyApp", [])
     .constant("MSIE", document.documentMode)
     .config(function (LoggerProvider) {
         LoggerProvider.enableBuffering(10);
});
```

documentMode is an IE only attribute

```
function LoggerProvider(MSIE) {
     console.log("LoggerProvider created");
     console.log("    MSIE: " + MSIE);
     this.$get = function () {
     }
}
```

http://oricalvo.com
ori.calvo@gmail.com

Everything is a Provider

- service/factory/value are just wrappers around the provider function

```
function factory(name, factoryFn, enforce) {
     return provider(name, {
         $get: factoryFn
     });
}

function service(name, constructor) {
     return factory(name, ['$injector', function ($injector) {
         return $injector.instantiate(constructor);
     }]);
}

function value(name, val) { return factory(name, valueFn(val), false); }

function valueFn(value) { return function () { return value; }; }
```

http://oricalvo.com
ori.calvo@gmail.com

Config & Run Blocks

- A module supports two type of initialization steps
- Config phase
  - Get access only to providers/constants
  - Use it to specify some global application configuration
  - For example, $locationProvider.html5Mode(true)
- Run phase
  - Get access only to services/factories/values
  - Usually consists of some application specific initialization
  - For example, read data from server and inject it into $rootScope
  - Resembles a main function

Config & Run Blocks


```
angular.module("myApp", [])
     .config(function (StorageServiceProvider) {
         console.log("myApp config");
     })
     .run(function (StorageService) {
         console.log("myApp run");
     });
```

- A module is loaded only if it is requested by other code
  - Specified as a dependency of another module
  - Specified as application module by using ng-app directive

$provide

- Angular built-in service which manages all injectables
- You can use it instead of module level functions
  - Not common
  - Allows for late registration after module was loaded
- Is considered a provider so you can get a reference to it only through a config block

```
angular.module("myApp").config(function ($provide) {
     console.log("Registrating a Logger using $provide");

     $provide.service("Logger", Logger);
});
```

http://oricalvo.com
ori.calvo@gmail.com

$provide.decorator

- Suppose you want to tweak a bit an existing service
- Angular allows you to replace any service with your own by setting a decorator
- The decorator is invoked when the service is instantiated
- The decorator gets a reference to the original service and may return a new one

```
angular.module("myApp").config(function ($provide) {
     $provide.decorator("ExternalService", function ($delegate) {
        var wrapper = Object.create($delegate);
        ...
        return wrapper;
     });
});
```

Strict DI

- New to Angular 1.3
- When enabled, Angular verifies that a function being invoked by $injector supplies manually the name of the parameters

```
<!DOCTYPE html>
<html ng-app="MyApp" ng-strict-di>
 <head>
    <title>Index</title>
 </head>
 <body>
   ...
 </body>
 </html>
```

```
function HomeCtrl($scope) {
     console.log("HomeCtrl created");
}

angular.module("MyApp")
     .controller("HomeCtrl", HomeCtrl);
```

Must specify manually the parameters names

```
angular.module("MyApp")
     .controller("HomeCtrl", ["$scope", HomeCtrl]);
```

http://oricalvo.com
ori.calvo@gmail.com

Navigation

- Traditionally navigation was easy
- User clicks on a link and a new entry was added to browser's history
- User can press back/forward native buttons
- But SPA fetches content from server without changing URL
  - This means that clicking on the browser's back button takes you to a totally different web site
  - Bookmarking and copying/pasting URL is useless

http://oricalvo.com
ori.calvo@gmail.com

Hashbang URL

- A trick to bring back support for URLs in SPA
- Is based on the fact that we can modify parts of the URL without triggering page reload
  - Only the part after the # character
- The new URL is automatically added to the browser's history
- We can monitor window.onhashchange
  - IE8+
- URLs become a bit ugly
  - http://mywebsite/#/admin/login
  - http://mywebsite/#/home/contacts

http://oricalvo.com
ori.calvo@gmail.com

HTML5 History API

- Can change browser's URL without making round trip to the server
- Can push a new URL into browser's history
- Can associate a user defined state object with each new history entry
- Built-in mechanism to observe changes in the history stack
- URLs become nice
  - http://mywebsite/admin/login
  - http://mywebsite/home/contacts
- IE10+

History API - Polyfill

- On modern browsers we would like to support clean URLs
- On older browsers need to fall back to the hashbang trick
- Application should not care about the type of navigation technique
- Many existing polyfills
  - Backbone's router
  - HistoryJS
- Angular offers its own → $location

$location

- Abstract layer over URL details
- Masks the difference between hashbang and HTML5 URL modes
- Provides a consistent API regardless of browser
- Provides clean API to easily access/manipulate different parts of the current URL
- Allows us to observe changes to the URL

http://oricalvo.com
ori.calvo@gmail.com

$location – Hashbang mode

- Assuming the following URL
  http://mypp.com/#/admin/login?active=true#menu

- **$location.url()** - /admin/login?active=true#menu
- **$location.path()** - /admin/login
- **$location.search()** - {active: true}
- **$location.hash()** – menu

$location API

- url, path, search and hash can be used to change the current address
- Under hashbang mode the specified address is appended after the hash
  - http://mypp.com/#/admin/login?active=true#menu
- Under HTML5 mode the specified address is set as is using pushState function
  - http://mypp.com/admin/login?active=true#menu

HTML5 Mode

- By default Angular is configured to only use hashbang mode for URLs
- Use $locationProvider.html5Mode to change to HTML5 URL mode

```
angular.module("myApp", [])
      .config(function ($locationProvider) {
            $locationProvider.html5Mode(true);
      })
      .run(function () {
      });
```

- You must specify a base element inside the HTML

Link Hijacking

- As part of DOM compilation Angular looks for simple links of type <a>
- Angular catches click event and disables page reload
- Modifies browser's URL with the link's href attribute
- Under hashbang mode, normal href (not hashbang) causes page reload
- Therefore, in case you want to support old browser you should always use hashbang URLs

Link Hijacking

- Below link will never cause page reload
- But still a clean url is displayed under HTML5 mode

```
<a href="/#/admin/url?active=true#menu">@("/#/admin/url")</a>
```

- Below link will cause page reload under hashbang mode

```
<a href="/admin/url?active=true#menu">@("/admin/url")</a>
```

$location Events

- $locationChangeStart
  - Is fired before the URL changes
  - Allows you to cancel the change
- $locationChangeSuccess
  - Is fired after URL was changed

```
$rootScope.$on("$locationChangeStart", function (e) {
     if (me.freeze) {
         e.preventDefault();
     }
});
```

http://oricalvo.com
ori.calvo@gmail.com

Routing

- Angular does not support out of the box routing system
- You can use a simple routing system from Angular Team known as angular-route
  - Good enough for mobile application which usually displays one view at a time
- Or, use a more complex routing system from Angular UI team known as UI-Router
  - More appropriate for Desktop/Tablet application which displays multiple views at the same time

angular-route

- Start by download it from http://code.angularjs.org/
- Add a reference to it after angular script
- Add ngRoute dependency to your module
- Use $routeProvider inside module's config method and configure the routing system
- Put div with ng-view inside the main HTML
- Use $route and $routeParams to get data information about the current route

http://oricalvo.com
ori.calvo@gmail.com

angular-route


```
angular.module("myApp", ["ngRoute"])
     .config(function ($routeProvider, $locationProvider) {
         $routeProvider.when("/home", {
             templateUrl: "/views/Main/Home",
             controller: "HomeCtrl",
         })
         $routeProvider.when("/about", {
             templateUrl: "/views/Main/About",
             controller: "AboutCtrl",
         })
         .otherwise({
             redirectTo: "/home“
         });
         $locationProvider.html5Mode(true);
     });
```

```
<body>
     <div ng-view></div>
     ...
     <script src="~/Scripts/angular.js"></script>
     <script src="~/Scripts/angular-route.js"></script>
</body>
```

Default Route

- In case Angular does not find matching route it navigates to the first defined route
- Use otherwise to set the default

```
angular.module("myApp", ["ngRoute"])
     .config(function ($routeProvider, $locationProvider) {
         $routeProvider.when("/home", {
             templateUrl: "/views/Main/Home",
             controller: "HomeCtrl",
         })
         $routeProvider.when("/about", {
             templateUrl: "/views/Main/About",
             controller: "AboutCtrl",
         })
         .otherwise({
             redirectTo: "/about“
         });
     });
```

http://oricalvo.com
ori.calvo@gmail.com

Server Side Implication

- $routeProvider allows us to map URLs to controllers
- Usually the URL is selected to best describe the HTML under action
- The URL has no direct corresponding to server side URL
- This means that clicking refresh on the browser will cause 404 error
- Need to fix server to ignore all “unknown” URLs and return the single page

ASP.NET MVC

- URL which starts with views means a template and must be served as is
- All other should be redirected to the single page HTML

```
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

routes.MapRoute(
     name: "Views",
     url: "views/{controller}/{action}" );

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

Route with Parameter

- A route may contains parameters embedded inside the URL
- Use $routeParams to extract the actual values

```
$routeProvider
     .when("/contact/edit/:id", {
         controller: "EditCtrl as ctrl",
         templateUrl: "/views/Main/Edit",
     });
```

```
angular.module("myApp").controller("EditCtrl", function ($routeParams) {
     var me = this;

     me.contactId = $routeParams.id;
});
```

Template Caching

- $route service downloads the HTML template on demand and caches it for later use
- $templateCache is responsible for managing the cache
- A template can be pre-loaded into the HTML using script tag
- A template can be set manually into $templateCache using plain string
- See next slide

Template Caching


```
<script type="text/ng-template" id="/views/Main/Home">
     <h1>Home</h1>
     <ul>
         <li ng-repeat="contact in ctrl.contacts">
             <a href="/contact/edit/{{contact.id}}">{{contact.name}}</a>
         </li>
     </ul>
</script>
```

```
angular.module("myApp", ["ngRoute"])
     .config(function ($locationProvider, $routeProvider) {
         ...
     })
     .run(function ($templateCache) {
         $templateCache.put('/views/Main/Edit', 'Bla bla');
     });
```

Reusing Templates with Different Controllers

- When using $routeProvider, the association between the HTML template and the controller is done inside the route configuration
- No need to declare controller name inside the HTML template
- This means that we can use the same template for different controllers
- For example, edit vs. new
  - Same template
  - But slightly different controller behavior

$route Events

- Are broadcasted using $rootScope
  - $routeChangeStart
  - $routeChangeSuccess
  - $routeChangeError – See $route resolve later

```
function HomeCtrl($rootScope) {
     $rootScope.$on("$routeChangeStart", function (e) {
         console.log("$routeChangeStart");
     });
     $rootScope.$on("$routeChangeSuccess", function (e) {
         console.log("$routeChangeSuccess");
     });
}
```

Cancel Route Change

- All previous events are not cancellable !!!
- Use instead $locationChangeStart

```
function HomeCtrl($rootScope) {
    $rootScope.$on("$locationChangeStart", function (e, next, current) {
        console.log("$locationChangeStart");

        if (next.indexOf("/contact/edit")) {
            alert("You are not allowed to edit");

            e.preventDefault();
        }
    });
 }
```

http://oricalvo.com
ori.calvo@gmail.com

Routing & Controllers

- When current route changes
  - A new scope is created based on the new route template
  - A new controller is created
  - The old scope is destroyed
  - Any change done to the DOM is lost
  - The old controller is not used any more and may be collected by the GC
- You are responsible for
  - Freeing any resource previously allocated by the controller
  - Unregister from any relevant event

Controller's Cleanup

- Angular fires a $destroy event on the relevant scope

```
function HomeCtrl($rootScope, $scope, $location, ContactService) {
     var me = this;

     console.log("HomeCtrl ctor");

     me.onLocationChangeStart = $rootScope.$on("$locationChangeStart", function () {
         console.log("$locationChangeStart");
     });

     $scope.$on("$destroy", function () {
    });
}
```

http://oricalvo.com
ori.calvo@gmail.com

$route Limitations

- Only one ng-view can be defined
- $route does not support nested ng-view
- This means that one route corresponds to one rectangle on the screen
- For desktop/tablet application this behavior might be too limiting
- Think about the following URL /**admin/logins**
- We expect admin zone to be loaded into the main ng-view and the logins management view to be loaded into it

UI-Router

- https://github.com/angular-ui/ui-router
- A solution for routing with nested views
- Is built around a state machine
- Each state has a name and corresponding URL, controller and template
  - A state may be defined as a nested of other state
  - When application loads a nested state UI-Router will load the parent too

UI-Router

- Add a module dependency to ui.router
- Use $stateProvider
- Each route entry consists of: name, url, templateUrl and controller
- Name which contains dot implies a nested view
  - For example when switching to a route named "admin.logins" UI-Router will first load the admin view and then the logins view into it
- Use $state.go to change current route
- Use $stateParams to get the values of current route parameter

UI-Router


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

```
<div class="admin-view">
     <h1>Admin</h1>
     <div ui-view></div>
</div>
```

```
angular.module("myApp", ["ui.router"])
     .config(function ($locationProvider, $stateProvider) {
          $stateProvider
               .state("home", {
                    url: "^/",
                    templateUrl: "/views/Main/Home",
                    controller: "HomeCtrl as ctrl",
               })
               .state("admin", {
                    url: "^/admin",
                    templateUrl: "/views/Admin/Home",
               })
               .state("admin.logins", {
                    url: "^/admin/logins",
                    templateUrl: "/views/Admin/Logins",
               });
          $locationProvider.html5Mode(true);
     });
```

http://oricalvo.com
ori.calvo@gmail.com

Additional Services

- $compile
- $exceptionHandler
- $interpolate
- $log
- $parse
- $timeout/$interval
- $window/$document

Summary

- Angular offers many built-in services
- You can register your own
  - And you should …
- Routing is offered through
  - Angular-route module
  - UI-Router