CitiusTech

Modules

May 2015

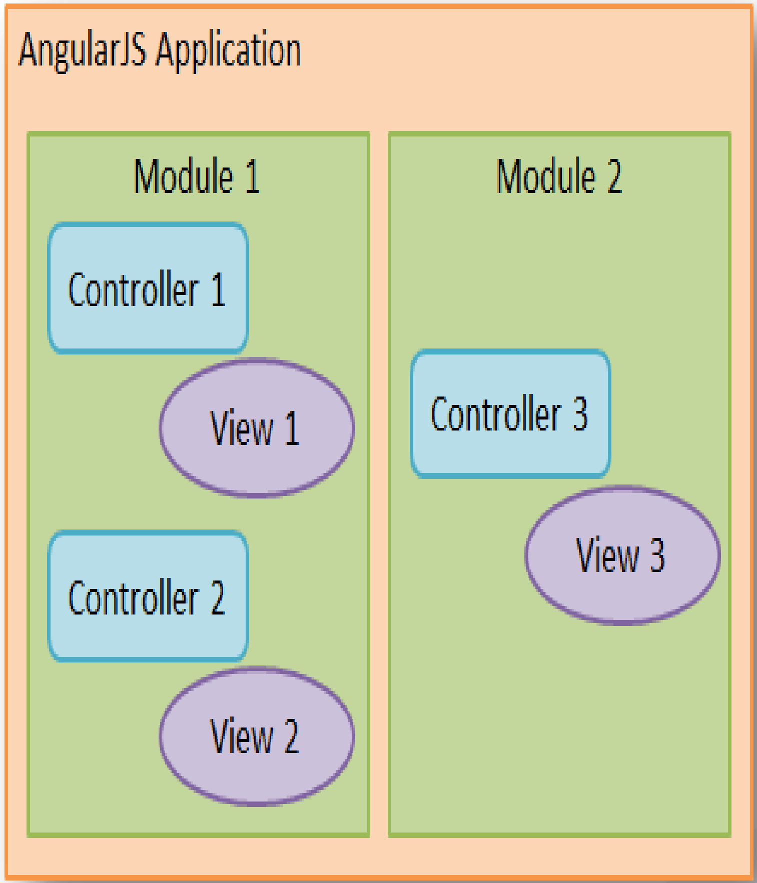
Agenda

- **Modules Introduction**
  - **What is a Module?**
  - **Why Modules?**
- Creating Modules
- Creating Component in a Module
- Inter-Module Code Access
- Module Lifecycle
- Module Loading & Dependencies

Modules Introduction (1/2)

What is a Module in AngularJS?

- A Module is a logical collection/group of application components like – controllers, services, filters, directives and configuration information
- Angular apps don't have a main method
- Instead of main method, Modules provide a mechanism to configure and then bootstrap app components grouped as module
- Reference to myApp module in <div ng-app="myApp"> bootstraps the app using 'myApp' module
- It also provides a way for modules to declare dependencies on other modules

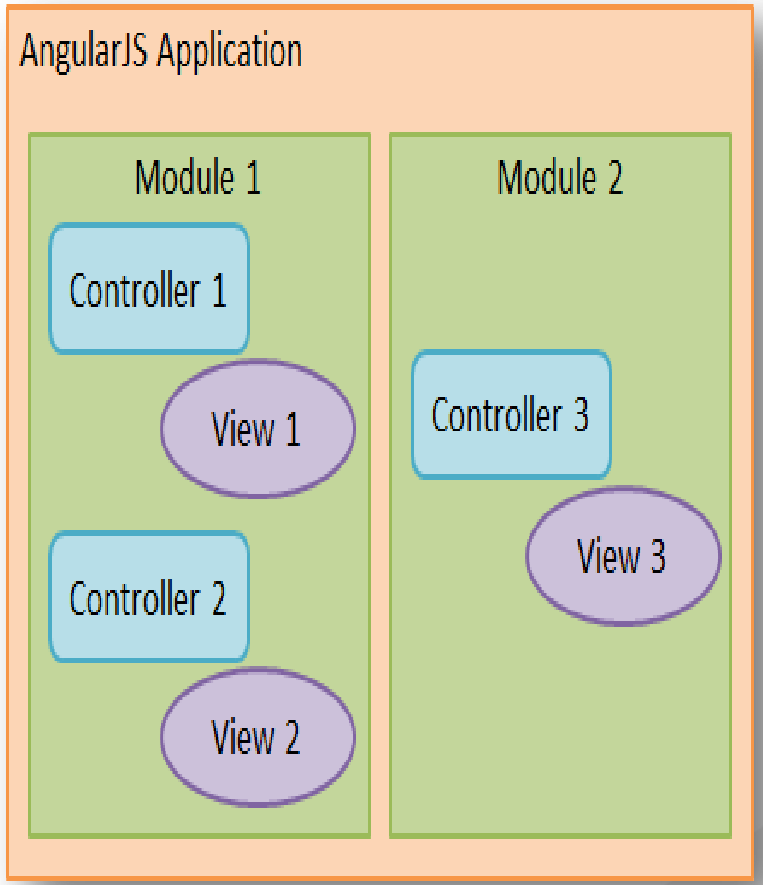


Source : http://viralpatel.net/blogs/angularjs-service-factory-tutorial/

Modules Introduction (2/2)

Why Modules?

- Help to modularize large applications
- Makes application more readable, maintainable
- Keep the global namespace clean
- Helps to package code as reusable modules



Agenda

- Modules Introduction
  - What is a Module?
  - Why Modules?
- **Creating Modules**
- Creating Component in a Module
- Inter-Module Code Access
- Module Lifecycle
- Module Loading & Dependencies

Creating Modules

- angular.module is a global place for creating, registering and retrieving Angular modules

- Modules are created by defining the module name and an array of dependencies on other modules:

```
//create a ‘sampleApp’ module with dependant modules
var sampleApp = angular.module('sampleApp',[ 'appControllers',
'appServices‘  ]);
```

- First argument specifies name of the module to be created and second argument specifies name of the dependant modules to be injected

```
// create a new module without dependant modules
      var myModule = angular.module('myModule', []);
```

Creating Modules

- All modules (angular core or 3rd party) that should be available to an application must be registered using this mechanism

- Once created, modules can be retrieved later on by just passing in a single parameter, the module name:

```
var mod = angular.module("products");
```

- Module using run value:
  http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day4/day4_Demos/2_module_using_run_value.html

Agenda

- Modules Introduction
  - What is a Module?
  - Why Modules?
- Creating Modules
- **Creating Component in a Module**
- Inter-Module Code Access
- Module Lifecycle
- Module Loading & Dependencies

Creating Component in a Module

- All related components should be put together in the same Module
- For e.g.: We can create ‘CustomerController’ as part of ‘custMgmtApp’ module as follows:

```
//create a new module ‘custMgmtApp’
var demoApp = angular.module(‘custMgmtApp', []);


//create a controller in module ‘custMgmtApp’
demoApp.controller(‘CustomerController', function ($scope)
{
   $scope.customers = [
     { name: 'Dave Jones', city: 'Phoenix' },
     { name: 'Jamie Riley', city: 'Atlanta' },
     { name: 'Heedy Wahlin', city: 'Chandler' },
     { name: 'Thomas Winter', city: 'Seattle' }
   ];

});
```

Creating Component in a Module

- Demo Link to create controller in a module: http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day2/day2_Demos/1_CD_Basic.html

Agenda

- Modules Introduction
    - What is a Module?
    - Why Modules?
- Creating Modules
- Creating Component in a Module
- **Inter-Module Code Access**
- Module Lifecycle
- Module Loading & Dependencies

Inter-Module Code Access (1/2)

Module Dependencies

- Modules can list other modules as their dependencies

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

- In above declaration ‘myApp’ has dependent ‘ngRoute’ module

- ‘ngRoute’ will be loaded before ‘myApp’.

```
var app = angular.module('myApp1', [‘myApp2']);
```

- In above declaration ‘myApp1’ module has dependency of ‘myApp2’

- ‘myApp1’ can access data and methods of ‘myApp2’. This is called inter module data access.
- Demo on intermodule code access:
  http://ctgit/silviap/angularjsdemosandlabs/blob/master/Day4/day4_Demos/1_Inter_module_codeaccess.html

CitiusTech

Inter-Module Code Access (2/2)

```
/*  Example depicting  'app2'  module using
'MathService' of app1   */
var app = angular.module('app1', []);
 app.service('MathService', function() {
  this.add = function(a, b) { return a + b };
  this.multiply = function(a, b) { return a * b};
   });

//'app1' module injected as dependency
var a =angular.module('app2',['app1']);
a.controller('CalculatorController',
function($scope, MathService) {
 $scope.doSquare = function() {
$scope.answer =
MathService.multiply($scope.number1,$scope
.number1);
   }
 });
```

```
<div ng-app="app2">
<div ng-controller="CalculatorController">
 Enter a number:
<input type="number" ng-model="number1" />
 <button ng-click="doSquare()">X<sup>2</sup>
      </button>
       <div>Answer: {{answer}}</div>
    </div>
</div>
```

Agenda

- Modules Introduction
  - What is a Module?
  - Why Modules?
- Creating Modules
- Creating Component in a Module
- Inter-Module Code Access
- **Module Lifecycle**
- Module Loading & Dependencies

Module Lifecycle

- Modules have two distinct phases in their lifecycle:
  - The configuration phase - Constants are set and module **components are registered and configured**. Components can be registered in any order.
  - The run phase - Module **components are instantiated** and injected with dependencies as needed.

- Modules have nothing to do with loading of physical scripts into a VM.

- As modules do nothing at load time they can be loaded into the VM in any order, script loaders can take advantage of this property and parallelize the loading process.

Agenda

- Modules Introduction
  - What is a Module?
  - Why Modules?
- Creating Modules
- Creating Component in a Module
- Inter-Module Code Access
- Module Lifecycle
- **Module Loading & Dependencies**

Module Loading & Dependencies (1/3)

Configuration blocks

- Angular executes blocks of configuration during the service provider registration and configuration phases in the bootstrapping of the module
- Config block ensures that service gets registered and configured before it gets instantiated
- We can *only inject and configure* providers and constants into config block.
- Service registration/configuration should complete before its instantiation so service can't be injected into the config block
- We can define multiple config blocks for a module. Angular runs them in the order in which they are written

Config block:

```
angular.module('myModule', []).
config(function (injectables) { // provider-injector
   // This is an example of config block.
   // You can have as many of these as you want.
   // You can only inject Providers (not instances)
   // into config blocks.
})
```

Example:

```
angular.module('myApp', [])
.config(function ($provide, $compileProvider) {

  $provide.factory('myFactory', function () {
      var service = {};
      return service;
   });
   $compileProvider.directive('myDirective',
   function () {
      return { template: '<button>Click me</button>'
      }
   })  });
```

*Note: refer service ppt for info on service, provider

Module Loading & Dependencies (2/3)

Run blocks

- Contains code to be run after the configuration phase but before the module at large kicks into gear
- Executed after all of the services have been configured and the injector has been created (to inject the dependant objects)
- Its similar to the main method
- Contains code that is typically hard to unit test and is related to the general app
- Are places where we'll set up event listeners that should happen at the global scale of the app

Run block:

```
angular.module('myModule', [])
.run(function (injectables) { // instance-injector
   // This is an example of a run block.
   // You can have as many of these as you want.
   // You can only inject instances (not Providers)
   // into run blocks
});
```

```
/* Example: run block to set up listeners for routing events
or unauthenticated  requests  */
angular.module('myApp', ['ngRoute'])
.run(function ($rootScope, AuthService) {
   $rootScope.$on('$routeChangeStart',
  function (evt, next, current) {
     // If the user is NOT logged in
     if (!AuthService.userLoggedIn()) {
        if (next.templateUrl === "login.html") {
                // Already heading to the login route so
                                  no need to redirect
                } else {
           $location.path('/login');
        }
     }  }); });
```

CitiusTech

Module Loading & Dependencies (3/3)

- Module.factory() method that we commonly use for registration of service is same as $provide.factory() used in module config block. (as shown below)
- Both methods are same and do the same task of registering a service in config block , at module load time.

```
//convenience methods (easy style provided by angular)
            angular.module('myModule', []). value('a', 123)
             .factory('a', function() { return 123; })
            . directive('directiveName', ...)
            . filter('filterName', ...);

           // above 'easy-to-use' methods are same as below methods…
           angular.module('myModule', [])
           . config(function($provide, $compileProvider, $filterProvider)
            {
                $provide.value('a', 123);
                $provide.factory('a', function() { return 123; });
                                      $compileProvider.directive('directiveName', ...);
                $filterProvider.register('filterName', ...);
             });
```

CitiusTech

Modules - Finding your way (1/2)

Technical Questions:

1. How to show data from two controllers in two different modules in View?

2. If two modules have service component with same name, then how name collision is resolved during dependancy injection?

As you start to work with AngularJS Modules, you will frequently encounter technical issues which are not covered by this training.

How will you resolve these technical issues on AngularJS Modules?

Modules - Finding your way (2/2)

Resources	Remarks
https://docs.angularjs.org/guide/module	AngularJS official doc site
https://github.com/jedrichards/angularjs-handbook#modules	Good handbook on AngularJS that includes modules
CurioCT - https://interct/SitePages/CurioCT.aspx	
CTCourses – Modules	In addition to updated course material, CTCourse contains reference sites (Library) and list of project teams with expertise on Modules

CitiusTech

CurioCT - CitiusTech's Technology Q & A Forum

- In case of any questions please log on to https://interct/SitePages/CurioCT.aspx

Thank You