AngularJS Custom Services

What's Hot?

Who's excited about what right now?

Review Yesterday

Custom Filters

Today

Services (including Factories, Providers, Constants, Values)

Services

- Services provide a means of **encapsulating functionality that can be reused** by different portions of your app.
- Services are singletons (they are instantiated once per app). They are not destroyed when they are removed from the current view, as is usually the case with directives and controllers. Therefore, they **provide a means of persisting data while the web app is in memory** and a means of sharing data between controllers and components, even as those components are loaded and destroyed.
- Services are lazy (they are instantiated only when a component which depends upon them is loaded).
- Services are provided to components, directives, and other services via **dependency injection**.
- Many built-in services exist ($scope, $http, $location, etc.). All built-in services begin with '$'.

Types of Services

There are five different types of services: Service, Factory, Provider, Const, Value. Some examples below are adapted from ng-book.

Service

Instantiated with 'new' behind the scenes. Therefore, uses a constructor function. So, add properties to 'this' and the service will automatically return 'this'.

```
angular.module('myService', [])
   .service('myService', myService);

myService.$inject = ['$http'];

function myService($http) {
   var ms = this;
   ms.getName = getName;
   function getName() {
      return $http(
         { method: 'GET', url: '/api/user'}
      );
   }
};
```

Factory (Just Use Services!)

"Somewhere deep inside of this Angular world, there's this code that calls Object.create() with the **service** constructor function, when it gets instantiated. However, **a factory function is really just a function that gets called**, which is why we have to return an object explicitly."
http://blog.thoughtram.io/angular/2015/07/07/service-vs-factory-once-and-for-all.html

Create an object, add properties and functions to it, return the object.

```
angular.module('myApp')
      .factory('myService', function() {
            return {
                  username: 'auser',
                   getUserEvents: function(username) {
                        // ...
                  }
            }
      }]);
```

Provider

This is the only type of service you can use with .config(). These are services that you configure before your app finishes bootstrapping. These configurations are available application-wide. Use provider any time you need to make these application-wide configurations prior to the app startup (e.g., setting a locale or base URLs for the APIs you're consuming).

A distinguishing feature of providers is their **$get method**. The $get method is called to create a new instance of the provider.

Example from StackOverflow:

```
angular.module('myApp')
      .provider('helloWorld', function() {
            this.name = 'Default';
            this.$get = function() {
                  var name = this.name;
                  return {
                        sayHello: function() {
                              return "Hello, " + name + "!"
                        }
                  }
            };
            this.setName = function(name) {
                  this.name = name;
            };
      });

// Configuring the provider in the app's .config() method
myApp.config(function(helloWorldProvider){
    helloWorldProvider.setName('World');
});
```

Constant

An injectable constant value.

```
angular.module('myApp')
      .constant('pi', '3.14')
angular.module('myApp')
      .controller('MyController', function($scope, pi) {
            $scope.pi = pi;
      });
```

Value

```
angular.module('myApp')
      .value('pi', '3.14');
```

Note: The value and constant services are less often used. Provider, service, and factory are the primary types.

Constants can be injected anywhere (including module's .config() methods). Values cannot be injected into .config(). Use constants for configuration data that should not change.

Review

Services (including Factories, Providers, Constants, Values)

Project

Work on the ToDo app! Or the Memory Game (here).

Resources

Learn-Angular.org
    http://www.learn-angular.org/#!/lessons/handling-complexity
Angular Docs for Services
    https://docs.angularjs.org/guide/services

Angular Docs for Providers

https://docs.angularjs.org/guide/providers

Comparing Service Types

https://gist.github.com/demisx/9605099

Tomorrow

$http and $q Services, Sharing Data Between Controllers and Services