Desarrollo Web Front-End con AngularJS

Factory, Provider, Services

- Angular provides us with three ways to create and register our own service.

1) Factory

2) Service

3) Provider

- AngularJS has built-in services. (p.e. Getting External data: $http = built in resource to make rest requests)

Service component

- Angular has many built in Service or Provider components
- $anchorScroll, $http, $window, $timeout

What is a Factory?

- Often referred to as an app service
- Business logic to be used application wide that does not need to be configured
- Also used for shareable data
- Might use for an something like md5
- $resource is a factory that lets you interact with RESTful server side
- Factories are the most popular way to create and configure a service.

More about Factory

When you're using a **Factory** you:

- create an object,
- add properties to it, then return that same object.

When you pass this service into your controller:

- those properties on the object will now be available in that controller through your factory.

Factory

```
app.controller('myFactoryCtrl', function($scope, myFactory){
  $scope.artist = myFactory.getArtist();
});

app.factory('myFactory', function(){
  var _artist = '';
  var service = {};

  service.getArtist = function(){
    return _artist;
  }

  return service;
});
```

What is a Service?

- Produces a service like factory or value, but

When you're using **Service**:

- it's instantiated with the 'new' keyword. Because of that, you'll add properties to 'this' and the service will return 'this'.

When you pass the service into your controller:

- those properties on 'this' will now be available on that controller through your service.
- The biggest thing to know when dealing with creating a Service is that that it's instantiated with the 'new' keyword

Service Example

```
app.controller('myServiceCtrl', function($scope, myService){
    $scope.artist = myService.getArtist();
});

app.service('myService', function(){
    var _artist = 'Nelly';
    this.getArtist = function(){
        return _artist;
    }
});
```

What is a Provider?

- Application wide business logic that does need to be configured
- Twitter API, facebook login etc where values are need to configure the service for it to work (typically APIs)
- **Providers** are the only service you can pass into your .config() function.
- Use a provider when you want to provide module-wide configuration for your service object before making it available.
- Biggest thing to remember about Providers is that they're the only service that you can pass into the app.config portion of your application.

What is a Provider?

```
app.controller('myProviderCtrl', function($scope, myProvider){
  $scope.artist = myProvider.getArtist();
  $scope.data.thingFromConfig = myProvider.thingOnConfig;
});

app.provider('myProvider', function(){
  //Only line 45-46  are available in app.config().
  this._artist = '';
  this.thingFromConfig = '';

  //Only the properties on the object returned from $get are available in the controller.
  this.$get = function(){
    var that = this;
    return {
      getArtist: function(){
        return that._artist;
      },
      thingOnConfig: that.thingFromConfig
    }
  }
});

app.config(function(myProviderProvider){
  myProviderProvider.thingFromConfig = 'This was set in config()';
});
```

What is a Value?

- Application wide settings
- Value objects that other controller or service components might need access to
- Example might be to store currentUser