

LEARN. DO. EARN

ACADGILD



FRONT END DEVELOPMENT (WITH ANGULARJS)



Website : <http://www.acadgild.com>
LinkedIn : <https://www.linkedin.com/company/acadgild>
Facebook : <https://www.facebook.com/acadgild>

© copyright ACADGILD



Session 16 – AngularJS Service





Agenda – AngularJS Service

1. **Introduction to AngularJS Service**
2. **AngularJS Service Types**
3. **Object - value**
4. **Function - factory**
5. **Object - service**
6. **provider**
7. **constant**





Introduction to AngularJS Service

- In Angular a service is used to organize and share our code across application.
- It is the best place to put all our business logic.
- Service can be injected easily in another services or controller.





AngularJS Service Types

There are 5 different ways to create services in Angular:

1. Value
2. Factory
3. Service
4. Provider
5. Constant





Object - value

- **value** is simple JavaScript object.
- It is used to pass values to controller during config phase.

Example:

```
//define a module
var mainApp = angular.module("mainApp", []);
//create a value object as "defaultInput" and pass it a data.
mainApp.value("defaultInput", 5);
...
//inject the value in the controller using its name "defaultInput"
mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {
    $scope.number = defaultInput;
    $scope.result = CalcService.square($scope.number);

    $scope.square = function() {
        $scope.result = CalcService.square($scope.number);
    }
});
```





Function - factory

- **factory** is a function which is used to return value.
- It creates value on demand whenever a service or controller requires.
- It normally uses a factory function to calculate and return the value

Example:

//define a module

```
var mainApp = angular.module("mainApp", []);
```

**//create a factory "MathService" which provides a method multiply to return
//multiplication of two numbers**

```
mainApp.factory('MathService', function() {  
  var factory = {};  
  factory.multiply = function(a, b) { return a * b }  
  return factory; });
```

**//inject the factory "MathService" in a service to utilize the multiply method of
factory.**

```
mainApp.service('CalcService', function(MathService){  
  this.square = function(a) {  
    return MathService.multiply(a,a); } });
```





Object - service

service is a singleton JavaScript object containing a set of functions to perform certain tasks.

Services are defined using **service()** functions and then injected into controllers.

//define a module

```
var mainApp = angular.module("mainApp", []);...
```

//create a service which defines a method square to return square of a number.

```
mainApp.service('CalcService', function(MathService){  
  this.square = function(a) {  
    return MathService.multiply(a,a);  }  
});
```

//inject the service "CalcService" into the controller

```
mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {  
  $scope.number = defaultInput;  
  $scope.result = CalcService.square($scope.number);  
  $scope.square = function() {  
    $scope.result = CalcService.square($scope.number);  }  
});
```





provider

- **provider** is used by AngularJS internally to create services, factory etc. during config phase(phase during which AngularJS bootstraps itself).
- Below mention script can be used to create MathService that we've created earlier. Provider is a special factory method with a method **get()** which is used to return the value/service/factory.

Example:

//define a module

```
var mainApp = angular.module("mainApp", []); ...
```

//create a service using provider which defines a method square to return square

//of a number.

```
mainApp.config(function($provide) {  
  $provide.provider('MathService', function() {  
    this.$get = function() {  
      var factory = {}; factory.multiply = function(a, b) {  
        return a * b; }  
      return factory; }; }); });
```





constant

constants are used to pass values at config phase considering the fact that value can not be used to be passed during config phase.

Syntax:

```
mainApp.constant("configParam", "constant value");
```





Lets Discuss Assignments