Agenda: AngularJS Services

- Understanding Services
- Developing Creating Services
- Using a Service
- Injecting Dependencies in a Service
- Complete Example

Understanding Services

- Services are reusable and stateless JavaScript objects with functions used to carry out a specific set of tasks.
- AngularJS supports the concept of ***Separation of Concerns*** using services architecture.
- While controllers are used to bind data model to views using @scope, service can be used to fetch the data, organize and share code across your app. Eg: The business logic or logic to call HTTP url to fetch data from server can be put within a service object.
- They are responsible to perform only specific and independent tasks. This makes them individual entities which are maintainable and testable.
- The controllers and filters can call them on requirement basis.

Angular services are:

- **Lazily instantiated** – Angular only instantiates a service when an application component depends on it.
- **Singletons** – Each component dependent on a service gets a reference to the single instance generated by the service factory.

Angular offers several useful services (like **$http**), but for most applications you'll also want to create your own. Application developers are free to define their own services by registering the service's name and **service factory function**, with an Angular module.

Developing Custom Service

There are two ways to create a service:

- Service method
- Factory method

Using Service Method

In this method, we define a service and then assign method to it.

```
mainApp.service('MathService', function(){
   this.multiply = function(a,b) {
                        return a * b;
            }
   this.divide = function(a,b) {
                        return a / b;
            }
});
```

When declaring serviceName (MathService) as an injectable argument you will be provided with the **instance of a function** passed to module.service.

Using Factory Method

In this method, we first define a factory and then assign method to it.

```
    var mainApp = angular.module("mainApp", []);
    mainApp.factory('MathService', function() {
      var factory = {};
      factory.multiply = function(a,b) {
        return a * b
      }
      factory.divide = function(a,b) {
        return a / b
      }
      return factory;
    });
```

When declaring factoryName as an injectable argument you will be provided the **value that is returned by invoking the function reference** passed to module.factory.

Injecting dependencies in services

Angularjs provides out of the box support for dependency management.

In general the wikipedia definition of dependency injection is:

Dependency injection is a software design pattern that allows the removal of hard-coded dependencies and makes it possible to change them, whether at run-time or compile-time.

If CalculatorService is dependent on MathService:

```
myApp.service('CalculatorService', function (MathService) {
   this.doSquare = function (a) { return MathService.multiply(a, a); };
   this.doCube = function (a) { return MathService.multiply(this.doSquare(a), a); };
});
```

Complete Example:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title></title>
   <script src="angular.js"></script>
</head>
<body>
   <div ng-app="myApp" ng-controller="myController">
      <form name="myForm" novalidate>
         Name:
         <input type="text" name="num" ng-model="Number" integer /><br />
         <input type="button" name="btnDouble" value="Square" ng-click="Square()" />
      </form>
   </div>
   <script>
      var myApp = angular.module("myApp", [])
      myApp.service('MathService', function () {
         this.multiply = function (a, b) {
            return a * b;
         }
         this.divide = function (a, b) {
            return a / b;
         }
      });
      myApp.service('CalculatorService', function (MathService) {
         this.doSquare = function (a) { return MathService.multiply(a, a); };
         this.doCube = function (a) { return MathService.multiply(this.doSquare(a), a); };
      });
      myApp.controller('myController', ['$scope', 'CalculatorService', function ($scope, CalculatorService) {
         $scope.Number = 10;
```

```
        $scope.Square = function()
        {
            $scope.Number = CalculatorService.doSquare($scope.Number);
        }
    }]);
  </script>
</body>
</html>
```