AngularJS : Service vs factory vs provider

## Introduction

AngularJS Service, Factory or Provider all are used for the same purpose of creating utility function that can be used throughout the page with inject-able object. However, the way it is created and the way it is used are different. Here we shall try to understand them clearly.

##### **AngularJS Service**

Angular Service is used to for sharing utility functions with the service reference in the controller. [Service](http://techfunda.com/Howto/angularjs/460/services) is singleton in nature so for once service only one instance is created in the browser and the same reference is used throughout the page.  
In the service, we create function names as property with this object.

##### **AngularJS Factory**

The purpose of Factory is also same as Service however in this case we create a new object and add functions as properties of this object and at the end we return this object.

##### **AngularJS Provider**

The purpose of this is again same however Provider gives the output of it's $get function

## http://www.dotnetfunda.com/UserFiles/ArticlesFiles/Sheonarayan_9721_Capture.JPG

Now let's try to understand Service, Factory and Provider by creating and using them.

#### Creating Service, Factory and Provider

In below code snippet, we shall see how to create a Service, Factory and a Provider. First, we have created a module.

Next is the Service creation where we have created a service using .service method. Notice that in the service both functions "Hello" and "Sum" have been created on "this" object.

Then we have created a factory using .factory method. Here, we have created both functions "Hello" and "Sum" on the new object "factoryObject" created and then we are returning that object at the end of the factory method.

In the last, we have created a provider using .provider method in which we are returning an object having two functions "Hello" and "Sum" to the $get function.

var app = angular.module("app", []);  
  
    // create utility function with service  
    app.service("myService", function () {  
        // here we expose the function on this object  
        this.Hello = function () {  
            return "Hello";  
        };  
        this.Sum = function (a, b) {  
            return a + b;  
        };  
    });  
  
    // create utility function with factory  
    app.factory("myFactory", function () {  
        // here we return the object  
        var factoryObject = {};  
        factoryObject.Hello = function () {  
            return "Hello";  
        }  
        factoryObject.Sum = function (a, b) {  
            return a + b;  
        }  
        return factoryObject;  
    });  
  
    // create utlity function with provider  
    app.provider("myProvider", function () {  
        this.$get = function () {  
            return {  
                Hello: function () {  
                    return "Hello";  
                },  
                Sum: function (a, b) {  
                    return a + b;  
                }  
            };  
        };  
    });

Notice that despite that all three have the same functions "Hello" and "Sum" having same functionality but the way of declaration is different. This is the major difference between Service, Factory and Provider.

#### Using Service, Factory and Provider

To use them simply inject those into the controller definition and start using those reference to call functions "Hello" and "Sum" defined in them.

Below code snippet is pretty simple. We are simply calling "Hello" and "Sum" functions defined on respective Service, Factory and Provider reference objects.

app.controller("myController", function ($scope, myService, myFactory, myProvider) {  
        // service function call  
        $scope.ServiceOutput = "Look for service output here.";  
        $scope.HelloService = function () {  
            $scope.ServiceOutput = myService.Hello();  
        };  
        $scope.SumService = function () {  
            $scope.ServiceOutput = "The sum is : " + myService.Sum(2, 5);  
        };  
  
        // factory function call  
        $scope.FactoryOutput = "Look for factory output here.";  
        $scope.HelloFactory = function () {  
            $scope.FactoryOutput = myFactory.Hello();  
        };  
        $scope.SumFactory = function () {  
            $scope.FactoryOutput = "The sum is : " + myFactory.Sum(22, 5);  
        };  
  
        // provider function call  
        $scope.ProviderOutput = "Look for factory output here.";  
        $scope.HelloProvider = function () {  
            $scope.ProviderOutput = myProvider.Hello();  
        };  
        $scope.SumProvider = function () {  
            $scope.ProviderOutput = "The sum is : " + myProvider.Sum(22, 52);  
        };  
    });

#### Calling functions defined in Service, Factory and Providers

The HTML code looks like below. Where we have three separate sections each for Service, Factory and Provider. On click respective buttons, functions defined in the controller $scope is being called and the output appears on the page.

<div ng-app="app" ng-controller="myController">  
    <h3>Using Service</h3>  
    <button ng-click="HelloService()">Hello Service</button>  
    <button ng-click="SumService()">Sum Service</button>  
    <div ng-bind="ServiceOutput"></div>  
  
  
    <h3>Using Factory</h3>  
    <button ng-click="HelloFactory()">Hello Factory</button>  
    <button ng-click="SumFactory()">Sum Factory</button>  
    <div ng-bind="FactoryOutput"></div>  
  
    <h3>Using Provider</h3>  
    <button ng-click="HelloProvider()">Hello Service</button>  
    <button ng-click="SumProvider()">Sum Service</button>  
    <div ng-bind="ProviderOutput"></div>  
</div>

To see [live working code, click here](http://techfunda.com/Code/Demo?demoid=457).

## Conclusion

Broadly there is no difference in terms of output/functionality between Service, Factory and Provider however the difference comes when we try to create them. All three have different way of creation and function implementations.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>Demo</title>  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>  </head>  <body>  <h2>Service vs. Factory vs. Provider</h2>  <hr />  <script>  var app = angular.module("app", []);  // create utility function with service  app.service("myService", function () {  // here we expose the function on this object  this.Hello = function () {  return "Hello";  };  this.Sum = function (a, b) {  return a + b;  };  });  // create utility function with factory  app.factory("myFactory", function () {  // here we return the object  var factoryObject = {};  factoryObject.Hello = function () {  return "Hello";  }  factoryObject.Sum = function (a, b) {  return a + b;  }  return factoryObject;  });  // create utlity function with provider  app.provider("myProvider", function () {  this.$get = function () {  return {  Hello: function () {  return "Hello";  },  Sum: function (a, b) {  return a + b;  }  };  };  });  // , myFactory, myProvider  app.controller("myController", function ($scope, myService, myFactory, myProvider) {  // service function call  $scope.ServiceOutput = "Look for service output here.";  $scope.HelloService = function () {  $scope.ServiceOutput = myService.Hello();  };  $scope.SumService = function () {  $scope.ServiceOutput = "The sum is : " + myService.Sum(2, 5);  };  // factory function call  $scope.FactoryOutput = "Look for factory output here.";  $scope.HelloFactory = function () {  $scope.FactoryOutput = myFactory.Hello();  };  $scope.SumFactory = function () {  $scope.FactoryOutput = "The sum is : " + myFactory.Sum(22, 5);  };  // provider function call  $scope.ProviderOutput = "Look for factory output here.";  $scope.HelloProvider = function () {  $scope.ProviderOutput = myProvider.Hello();  };  $scope.SumProvider = function () {  $scope.ProviderOutput = "The sum is : " + myProvider.Sum(22, 52);  };  });  </script>  <div ng-app="app" ng-controller="myController">  <h3>Using Service</h3>  <button ng-click="HelloService()">Hello Service</button>  <button ng-click="SumService()">Sum Service</button>  <div ng-bind="ServiceOutput"></div>  <h3>Using Factory</h3>  <button ng-click="HelloFactory()">Hello Factory</button>  <button ng-click="SumFactory()">Sum Factory</button>  <div ng-bind="FactoryOutput"></div>  <h3>Using Provider</h3>  <button ng-click="HelloProvider()">Hello Service</button>  <button ng-click="SumProvider()">Sum Service</button>  <div ng-bind="ProviderOutput"></div>  </div>  </body>  </html> |