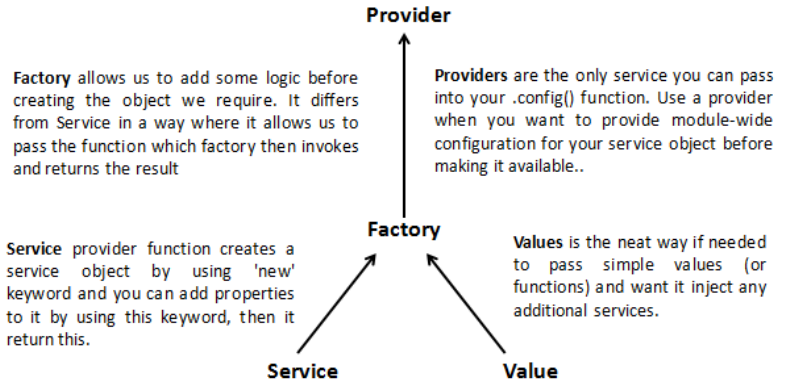

Features / Recipe type	Factory	Service	Value	Constant	Provider
can have dependencies	yes	yes	no	no	yes
uses type friendly injection	no	yes	yes*	yes*	no
object available in config phase	no	no	no	yes	yes**
can create functions	yes	yes	yes	yes	yes
can create primitives	yes	no	yes	yes	yes

The difference between factory and service is just like the difference between a function and an object

Factory Provider

- Gives us the function's return value ie. You just 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. (Hypothetical Scenario)
- Singleton and will only be created once
- Reusable components
- Factory are a great way for communicating between controllers like sharing data.
- Can use other dependencies
- Usually used when the service instance requires complex creation logic
- Cannot be injected in `.config()` function.
- Used for non configurable services
- If you're using an object, you could use the factory provider.
- Syntax: `module.factory('factoryName', function);`

Service Provider

- Gives us the instance of a function (object)- You just instantiated with the ‘new’ keyword and 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. (Hypothetical Scenario)
- Singleton and will only be created once
- Reusable components
- Services are used for communication between controllers to share data
- You can add properties and functions to a service object by using the `this` keyword
- Dependencies are injected as constructor arguments
- Used for simple creation logic
- Cannot be injected in `.config()` function.
- If you're using a class you could use the service provider
- Syntax: `module.service(‘serviceName’, function);`

Sample Demo

In below example I have define `MyService` and `MyFactory`. Note how in `.service` I have created the service methods using `this.methodname`. In `.factory` I have created a factory object and assigned the methods to it.

AngularJS .service

```
module.service('MyService', function() {

    this.method1 = function() {
            //..method1 logic
        }
```

```
    this.method2 = function() {
            //..method2 logic
        }
});
```

AngularJS .factory

```
module.factory('MyFactory', function() {

    var factory = {};

    factory.method1 = function() {
            //..method1 logic
        }

    factory.method2 = function() {
            //..method2 logic
        }

    return factory;
});
```

Also Take a look at this beautiful stuffs

Confused about service vs factory

AngularJS Factory, Service and Provider

Angular.js: service vs provider vs factory?

edited May 23 at 10:31
Community ♦
1 1

answered Apr 15 '14 at 6:24
Nidhish Krishnan
15k 6 40 62

2 This makes sense to me now! So I could conceivably make a service behave like a factory, but that kind of goes against what services are designed to be used for. Further, if I don't return anything when I use module.factory, things won't work properly, correct? – Cameron Ball Apr 15 '14 at 6:46

Is it just me or does it seem like a service is kinda pointless? – Oliver Dixon Aug 11 '15 at 17:16

The last link is not valid anymore. – Swanidhi Sep 26 '15 at 18:08

I dont like the factory example, as its identical to service (altough valid). Factories can return functions, but the conceptual difference between the two should be about variables - services are like class (New = empty object) and factories like object (= static data/values). – Z. Khullah Nov 7 '16 at 12:38

`Factory` and `Service` is a just wrapper of a `provider`.

Factory

`Factory` can return anything which can be a `class(constructor function)`, `instance of class`, `string`, `number` or `boolean`. If you return a `constructor` function, you can instantiate in your controller.

```
 myApp.factory('myFactory', function () {

  // any logic here..

  // Return any thing. Here it is object
  return {
    name: 'Joe'
  }
}
```

Service

Service does not need to return anything. But you have to assign everything in `this` variable. Because service will create instance by default and use that as a base object.

```
myApp.service('myService', function () {

  // any logic here..

  this.name = 'Joe';
}
```

Actual angularjs code behind the service

```
function service(name, constructor) {
    return factory(name, ['$injector', function($injector) {
        return $injector.instantiate(constructor);
    }]);
}
```

It just a wrapper around the `factory`. If you return something from `service`, then it will behave like `Factory`.

`IMPORTANT`: The return result from Factory and Service will be cache and same will be returned for all controllers.

When should i use them?

`Factory` is mostly preferable in all cases. It can be used when you have `constructor` function which needs to be instantiated in different controllers.

`Service` is a kind of `Singleton` Object. The Object return from Service will be same for all controller. It can be used when you want to have single object for entire application. Eg: Authenticated user details.

For further understanding, read

http://iffycan.blogspot.in/2013/05/angular-service-or-factory.html

http://viralpatel.net/blogs/angularjs-service-factory-tutorial/

edited Apr 16 '14 at 4:08

answered Apr 15 '14 at 4:56

Fizer Khan
24.4k 12 98 111

7 Why would you choose one over the other, then? And can you give me a solid example of something factory can do that service cannot? I still don't really understand the difference. – Cameron Ball Apr 15 '14 at 4:58

1 Updated the answer for when you need to use them. – Fizer Khan Apr 15 '14 at 5:03

1 Since a Factory lets you return the object, you have control over what is exposed, meaning you can kind of have private methods in factories. With services, the entire object is exposed. – aet Apr 15 '14 at 5:26

1 @eet, In service also we can have private method. Whatever you assign to `this` object will be exposed. Other functions are private. Service just reduce to create object yourself. – Fizer Khan Apr 15 '14 at 5:29

2 Service and factory both are singleton. – shruti Oct 19 '16 at 14:08

- If you use a service you will get the instance of a function ("this" keyword).
- If you use a factory you will get the value that is returned by invoking the function reference (the return statement in factory)

Factory and Service are the most commonly used recipes. The only difference between them is that Service recipe works better for objects of custom type, while Factory can produce JavaScript primitives and functions.

Reference

answered Apr 15 '14 at 4:53

Jayram Singh
8,788 5 32 54

3 I don't really understand what that means. Can you give me an example that shows something you can do with a factory but not with a service, or vica versa? – Cameron Ball Apr 15 '14 at 4:54

$provide service

They are technically the same thing, it's actually a different notation of using the `provider` function of the `$provide` service.

- If you're using a class: you *could* use the *service* notation.
- If you're using an object: you *could* use the *factory* notation.

The *only* difference between the `service` and the `factory` *notation* is that the service is *new-ed* and the factory is not. But for everything else they both *look*, *smell* and *behave* the same. Again, it's just a shorthand for the *$provide.provider* function.

```
// Factory

angular.module('myApp').factory('myFactory', function() {

  var _myPrivateValue = 123;

  return {
    privateValue: function() { return _myPrivateValue; }
  };

});

// Service

function MyService() {
  this._myPrivateValue = 123;
}
```

```
MyService.prototype.privateValue = function() {
  return this._myPrivateValue;
};

angular.module('myApp').service('MyService', MyService);
```

edited Apr 15 '14 at 5:41

answered Apr 15 '14 at 5:31

null
6,547 2 26 34