Decorator

Decorator装饰， 在已有的功能上锦上添花

Use case 使用场景：

我有一个module A依赖于另外一个module B。 module B有个service Mail， 这个服务提供 两个方法 setReceiver 和 setBody 分别用来指定邮件的收件人和邮件的内容。 但是在 module A 使用Mail服务的时候， 我希望还可以指定抄送的人。 这个时候我就可以在已有 的service上扩展下（装饰下）加个addCC的方法。

Example

**Module A**

var Mail = function() {

this.receiver = '';

this.body = '';

this.cc = [];

};

Mail.prototype.setReceiver = function(receiver) {

this.receiver = receiver;

};

Mail.prototype.setBody = function(body) {

this.body = body;

};

angular.module('A', []).service('Mail', Mail);

### Module B

angular.module('B', ['A']).config(function($provide) {

$provide.decorator('Mail', function($delegate) {

$delegate.addCC = function(cc) {

this.cc.push(cc);

};

return $delegate;

});

})

.controller('TestCtrl', function($scope, Mail) {

Mail.addCC('jack');

console.log(Mail);

});

### $provide.decorator()

$provide服务提供了在服务实例创建时对其进行拦截的功能，可以对服务进行扩展，或者用另外的内容完全代替它,这就是装饰器。装饰器是非常强大的，它不仅可以应用在我们自己的服务上，也可以对AngularJS的核心服务进行拦截、中断甚至替换功能的操作。事实上AngularJS中很多功能的测试就是借助$provide.decorator()建立的。对服务进行装饰的场景有很多，比如对服务进行扩展，将外部数据缓存进localStorage的功能，或者对服务进行封装以便在开发中进行调试和跟踪等。例如，我们想给之前定义的githubService服务加入日志功能，可以借助decorator()函数方便地实现这个功能，而不需要对原始的服务进行修改。

**$provide.decorator()函数可以接受两个参数:**

1. θ name（字符串）  
   将要拦截的服务名称。
2. θ decoratorFn（函数）  
   在服务实例化时调用该函数，这个函数由injector.invoke调用，可以将服务注入这个函数中。

### $delegate

装饰器依赖于$delegate，$delegate是可以进行装饰的最原始的服务，为了装饰其他服务，需要将其注入进装饰器



var githubDecorator = function($delegate, $log) {

var events = function(path) {

var startedAt = new Date();

var events = $delegate.events(path); // 事件是一个promise events.finally(function() {

$log.info("Fetching events" + " took " + (new Date() - startedAt) + "ms"); }); return events; }; return { events: events }; }; angular.module('myApp') .config(function($provide) { $provide.decorator('githubService', githubDecorator); });