**AngularJS中如何对Controller与Service进行分层设计与编码**

**AngularJS中如何对Controller与Service进行分层设计与编码**

**前言**

本文主要针对AngularJS的初学者而编写的一篇关于如何在AngularJS框架下更好的编写代码的一个指南。下文将对AngularJS进行简称，替代为NG。

**初学者的Controller**

在我们当接触NG后，如需要通过访问远程的API获取一系列的数据进行显示，通常我的Controller代码会写成下面的样子：

angular.module('demo')

.controller('myCtrl',['$scope','$http', function($scope,$http){

$http.get("xxx")

.success(function(response)

{

$scope.data = response.data;

});

}]);

这样的在功能上是没有问题的，但是会导致Controller除了负担了与View层的$scope变量的初始化和防范定义还需要额外注入http进行远程的数据调用。  
如当调用API的代码是会大量被引用、或是API变更时候会导致大面的修改Controller代码。

**分离Service**

**Service层和Controller层的分工**

我们将原来全部集中在Controller中代码拆分成两个层面：

* service层：主要负责数据交互和数据处理、处理一些业务领域上的逻辑；
* controller层：主要负责初始化$scope的变量用于传递给view层，并且处理一些页面交互产生的逻辑;

**什么情况下需要编写Service**

当一个功能是设计远程API调用、数据集、业务领悟复杂逻辑、将会大量重复的运算方法时就可以考虑将代码以service形式注入controller层。

**编写Service**

将原先的代码从Controller中抽离处理，代码如下：

angular.module('demo')

.service('myService',['$http',function($http){

return {

getData:function(){

return $http.get("xxx");

}

}

}]);

则Controller的代码将会被注入myService用于获取相关的数据

angular.module('demo')

.controller('myCtrl',['$scope','myService', function($scope,$http,myService){

myService.getData().success(function(response)

{

$scope.data = response.data;

});

}]);

基本上这样的代码看上去很不错了，但是我们依旧在Controller层处理了通讯时的回调函数success,这样controller虽然直接依赖http了，但是还是间接的需要处理http。

**在Service层处理通讯回调，将业务回调传递给Controller层**

这里需要引入deffered将http的通讯级的回调在Service层处理完后，再重新交由controller去处理其他的问题。

angular.module('demo')

.service('myService',['$http','$q',function($http,$q){

return {

getData:function(){

var deferred = $q.defer();

var promise = $http.get("xxx");

promise.then(

// 通讯成功的处理

function(answer){

//在这里可以对返回的数据集做一定的处理,再交由controller进行处理

answer.status = true;

deferred.resolve(answer);

},

// 通讯失败的处理

function(error){

// 可以先对失败的数据集做处理，再交由controller进行处理

error.status = false;

deferred.reject(error);

});

//返回promise对象，交由controller继续处理成功、失败的业务回调

return deferred.promise;

}

}

}]);

相应的在controller中我们也可以进行相关事件的处理，修改代码如下

angular.module('demo')

.controller('myCtrl',['$scope','myService', function($scope,myService){

myService.getData().then(

function(answer){

$scope.data = answer;

},

function(error){

$scope.error = error;

}

);

}]);

这样controller和servic的职业分离，并且controller完全不依赖http而只是依赖service传递的事件和数据。再编写测试代码时，其逻辑也会变得简单。并且多个controller可以调用一个service中相同的方法，而不是通过曾经那种复制的方法来解决。  
分层编写代码的最终目的无非就是

1. 增加代码的复用性;
2. 代码责任简单，不会又做保姆又做司机，可读性强容易理解;
3. 编写测试代码的时候容易编写;
4. 减少对一些框架和环境插件的依赖;
5. 修改逻辑时最小幅度的修改代码
6. 数据层发生变更修改Service,UI层有变化则修改Controller。不用担心改controller把service也一起带到沟里的情况发生。

**最后的最后**

AngularJS在许多框架的设计方面与Java的Spring非常类似，如你有一定的Java基础应该很能理解为赢编写出分层的代码。

[在AngularJS中使用$q同步读取服务器数据](http://blog.csdn.net/lglgsy456/article/details/32335127)

今天在写AngularJS Service，然后希望在Controller中使用Service提供Model。架构上，我希望Controller可以直接获得数据，因此有了“同步读取”数据的想法。

但是，我们都知道，在前端做同步读取显然不是好的实践做法，毕竟JavaScript的很多良好体验都是通过异步请求实现的。而且，同步之后会严重影响前端的体验和性能。所以，这里把标题写成“同步读取服务器数据”，更多是吸引眼球。希望刚刚你不是拿着砖头正准备拍的！

不过，我在网上搜索了一下，还真有人一开始和我想法一样的，见参考资料2。转入正题。

AngularJS提供了一个内置Service $q，它提供了一种承诺/延后（promise/deferred），可以保证我们的调用代码一定能够拿到数据。当然，我们可以猜到，最后去服务器取数据的方式肯定是异步的。只不过这个服务提供了表面上是同步访问的API，当数据获取成功之后，自动将数据提供给调用的代码。

$q – A promise/deferred implementation inspired by [Kris Kowal’s Q](https://github.com/kriskowal/q).

[The CommonJS Promise proposal](http://wiki.commonjs.org/wiki/Promises) describes a promise as an interface for interacting with an object that represents the result of an action that is performed asynchronously, and may or may not be finished at any given point in time.

话不多说，上代码看看：

1. 创建一个Service，去服务器读取数据：

**[javascript]** [view plaincopy](http://blog.csdn.net/myweishanli/article/details/17936147)[在CODE上查看代码片](https://code.csdn.net/snippets/145235)

1. // $q 是内置服务，所以可以直接使用
2. ngApp.factory('UserInfo', ['$http', '$q', **function** ($http, $q) {
3. **return** {
4. query : **function**() {
5. **var** deferred = $q.defer(); // 声明延后执行，表示要去监控后面的执行
6. $http({method: 'GET', url: 'scripts/mine.json'}).
7. success(**function**(data, status, headers, config) {
8. deferred.resolve(data);  // 声明执行成功，即http请求数据成功，可以返回数据了
9. }).
10. error(**function**(data, status, headers, config) {
11. deferred.reject(data);   // 声明执行失败，即服务器返回错误
12. });
13. **return** deferred.promise;   // 返回承诺，这里并不是最终数据，而是访问最终数据的API
14. } // end query
15. };
16. }]);

2. 在Controller上（以同步方式）使用这个Service：

**[javascript]** [view plaincopy](http://blog.csdn.net/myweishanli/article/details/17936147)[在CODE上查看代码片](https://code.csdn.net/snippets/145235)

1. angular.module('ngApp')
2. .controller('MainCtrl', ['$scope', 'UserInfo', **function** ($scope, UserInfo) { // 引用我们定义的UserInfo服务
3. **var** promise = UserInfo.query(); // 同步调用，获得承诺接口
4. promise.then(**function**(data) {  // 调用承诺API获取数据 .resolve
5. $scope.user = data;
6. }, **function**(data) {  // 处理错误 .reject
7. $scope.user = {error: '用户不存在！'};
8. });

# [给你一个承诺 - 玩转 AngularJS 的 Promise](https://segmentfault.com/a/1190000002788733)

<https://segmentfault.com/a/1190000002788733>

[angular通过$http与服务器通信](http://blog.csdn.net/yangnianbing110/article/details/43124679)

<http://blog.csdn.net/yangnianbing110/article/details/43124679>

**AngularJS中$http服务的简单用法**

<http://www.2cto.com/kf/201506/405137.html>