Share for AngularJS

From introduction to use

WangShiwei - May 12, 2016

Introduction

AngularJS 诞生于2009年，由Misko Hevery 等人创建，后为Google所收购。是一款优秀的前端JS框架，已经被用于Google的多款产品当中。AngularJS有着诸多特性，最为核心的是：**MVVM、模块化、自动化双向数据绑定、语义化标签、依赖注入**等等。

一、MVC

MVC适用于大型可扩展的Web应用的开发,它强制性地将应用程序的输入、处理和输出分开,将其划分为**模型（M）、视图（V）和控制器（C）**3个核心部分,使它们各司其职,各自完成不同的任务,其中任何一部分的修改都不会影响其他两部分。

模型 – model

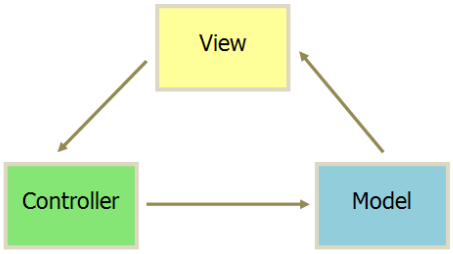
模型封装了应用问题的核心数据、逻辑关系和业务规则,提供了业务逻辑的处理过程。模型一方面被控制器调用,完成问题处理的操作过程,另一方面为视图获取显示数据提供了访问数据的操作。

视图 – view

视图是MVC模式下用户看到的并与之交互的界面。视图从模型处获得数据,其更新由控制器控制。视图不包含任何业务逻辑的处理,它只是作为一种输出数据的方式。

控制器 – controller

响应于用户输入并执行交互数据模型对象，在MVC模式中,控制器主要起导航的作用,它根据用户的输入调用相应的模型和视图去完成用户的请求。



前端开发的MVC架构核心是事件流；后端开发的核心是数据流。

由于前端越来越多地承担了应用逻辑的功能（毕竟用户输入都是绑定在 DOM 上的），所以承担“用户和系统之间连接”的更多是 View。这时 View 其实是“控件”而不是“视图”了。

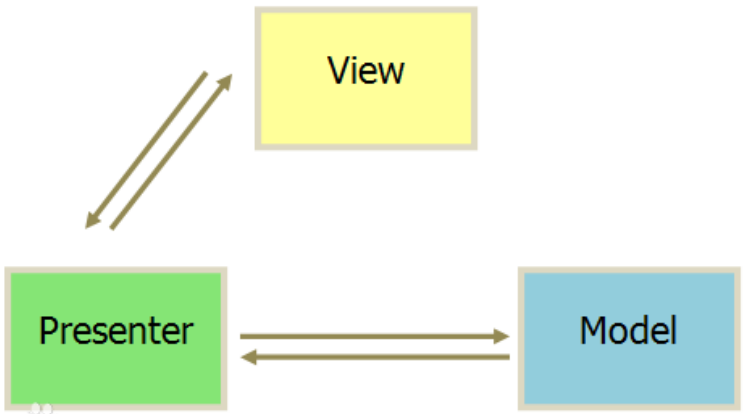
前端既有URL (hashtag /push state) 也有DOM event 都会对View造成影响 所以前端经常称自己为MVVM。

MVVM是Model-View-ViewModel的简写。MVVM（Model–View–ViewModel）框架的由来是MVP（Model–View–Presenter）模式与WPF结合的应用方式时发展演变过来的一种新型架构框架。它是原有MVP框架并且把WPF的新特性糅合进去，以应对客户日益复杂的需求变化。

那么问题来了，MVP是什么。

MVP 是从经典的模式MVC演变而来，它们的基本思想有相通的地方：Controller/Presenter负责逻辑的处理，Model提供数据，View负责显示。

MVP与MVC有着一个重大的区别：在MVP中View并不直接使用Model，它们之间的通信是通过Presenter (类似MVC中的Controller)来进行的，所有的交互都发生在Presenter内部，而在MVC中View会从直接Model中读取数据而不是通过 Controller。



ps：所有mvx的设计模式都是通过x实现model和view的解耦。例如：MVC、MVP、MVVM。

下面是干货！！！！

AngularJs 官方API　https://docs.angularjs.org/api

二、AngularJS的名词（术语）

1. 表达式，可以理解为插值，双大括号内：{{ expression }}， 将数据绑定到html模板中，与ng-bind指令异曲同工；

2. 指令，通过新的属性来扩展html，添加新功能，可以自定义扩展指令。带有前缀ng-。ng-app 指令初始化一个 AngularJS 应用程序。ng-init 指令初始化应用程序数据。ng-model 指令把元素值（比如输入域的值）绑定到应用程序。

内置基础属性指令：ng-href, ng-src, ng-disabled, ng-checked, ng-readonly, ng-selected, ng-class, ng-style, ng-include, ng-switch, ng-repeat, ng-view, ng-controller, ng-if, ng-show, ng-hide, ng-model, ng-form, ng-change, ng-click, ng-select, ng-option, ng-submit, ng-attr-(suffix)

3. 控制器，ng-controller 指令定义了应用程序控制器。控制器是 JavaScript 对象，由标准的 JavaScript 对象的构造函数 创建。

4. 过滤器，可以使用一个管道字符（|）添加到表达式和指令中。自带的有：currency、filter、lowecase、orderBy、uppercase、date、limitTo、json、number等。

5. 服务，服务是js的一个函数或对象。常用的$location、$http、$interval等。可以通过工厂方法自定义。过滤器其实也算是一种服务。通常服务负责一些逻辑运算复杂的业务逻辑。

三、双向数据绑定

顾名思义，双向数据绑定就是在model（伪）和view之间进行的数据绑定，当model数据改变时，自动更新到view，如果通过ng-model绑定到某个input里面，改变输入时候，页面也会更新。可以理解为模型状态的映射。

这时候，视图可以理解成为实时模板，不会将数据和模板合并之后再更新DOM

demo

双向数据绑定的原理：$digest循环和脏值检查

1.当绑定数据之后，$scope.name,就会将name属性添加到watch列表，并注册一个监视函数。

2.AngularJS会记录数据模型包含的数据在特定时间点的值，而不是原始值。

3.当用户输入时，触发AngularJS遍历watch列表，判断是否是脏值，如果是true，启用新值，并重新遍历。（如果遍历超过10次就会抛出异常）。如果为false，重新绘制DOM。

model提供了$watch等方法，可以手动监听和自定义回调函数。

重大意义在于，不用对DOM进行手动的修改或者手动监控变化，专注于业务逻辑。

四、angular 的表单

在angular中为了让form表单更富有样式更智能，原生添加了

1. ng-valid (验证成功)
2. ng-invalid （验证失败）
3. ng-pristine (从未输入过)
4. ng-dirty (输入过)

dome

建议使用扩展指令ng-message

五、angularJS中的服务定义语法糖

Angular services are singletons objects or functions that carry out specific tasks common to web apps.

factory

使用一个对象工厂函数定义服务，调用该工厂函数将返回服务实例。

service

使用一个类构造函数定义服务，通过new操作符将创建服务实例。

value

使用一个值定义服务，这个值就是服务实例。

constant
使用一个常量定义服务，这个常量就是服务实例。

1、factory()

Angular里面创建service最简单的方式是使用factory()方法。

factory()让我们通过返回一个包含service方法和数据的对象来定义一个service。在service方法里面我们可以注入services，比如 $http 和 $q等。

```
angular.module('myApp.services')
.factory('User', function($http) { // injectables go here
  var backendUrl = "http://localhost:3000";
  var service = {    // our factory definition
    user: {},
    setName: function(newName) {
      service.user['name'] = newName;
    },
    setEmail: function(newEmail) {
      service.user['email'] = newEmail;
    },
    save: function() {
      return $http.post(backendUrl + '/users', {
        user: service.user
      });
    }
  }; return service;
});
```

在应用里面使用factory()方法

在应用里面可以很容易地使用factory ， 需要到的时候简单地注入就可以了

```
angular.module('myApp')
.controller('MainCtrl', function($scope, User) {
  $scope.saveUser = User.save;
});
```

什么时候使用factory()方法

在service里面当我们仅仅需要的是一个方法和数据的集合且不需要处理复杂的逻辑的时候，factory()是一个非常不错的选择。

注意：需要使用.config()来配置service的时候不能使用factory()方法

2、service()

service()通过构造函数的方式让我们创建service，我们可以使用原型模式替代javaScript原始的对象来定义service。

和factory()方法一样我们也可以在函数的定义里面看到服务的注入

```
angular.module('myApp.services')
.service('User', function($http) { // injectables go here
  var self = this; // Save reference
  this.user = {};
  this.backendUrl = "http://localhost:3000";
  this.setName = function(newName) {
    self.user['name'] = newName;
  }
  this.setEmail = function(newEmail) {
    self.user['email'] = newEmail;
  }
  this.save = function() {
    return $http.post(self.backendUrl + '/users', {
      user: self.user
```

```
    })
  }
});
```

这里的功能和使用factory()方法的方式一样，service()方法会持有构造函数创建的对象。

在应用里面使用service()方法

```
angular.module('myApp')
.controller('MainCtrl', function($scope, User) {
  $scope.saveUser = User.save;
});
```

什么时候适合使用service()方法

service()方法很适合使用在功能控制比较多的service里面

注意：需要使用.config()来配置service的时候不能使用service()方法

3、provider()

provider()是创建service最底层的方式，这也是唯一一个可以使用.config()方法配置创建service的方法

不像上面提到的方法那样，我们在定义的this.$get()方法里面进行依赖注入

```
angular.module('myApp.services')
.provider('User', function() {
  this.backendUrl = "http://localhost:3000";
```

```
  this.setBackendUrl = function(newUrl) {
    if (url) this.backendUrl = newUrl;
  }
  this.$get = function($http) { // injectables go here
    var self = this;
    var service = {
      user: {},
      setName: function(newName) {
        service.user['name'] = newName;
      },
      setEmail: function(newEmail) {
        service.user['email'] = newEmail;
      },
      save: function() {
        return $http.post(self.backendUrl + '/users', {
          user: service.user
        })
      }
    };
    return service;
  }
});
```

在应用里面使用provider()方法

为了给service进行配置，我们可以将provider注入到.config()方法里面

```
angular.module('myApp')
.config(function(UserProvider) {
  UserProvider.setBackendUrl("http://myApiBackend.com/api");
})
```

这样我们就可以和其他方式一样在应用里面使用这个service了

```
angular.module('myApp')
.controller('MainCtrl', function($scope, User) {
  $scope.saveUser = User.save;
});
```

什么时候使用provider()方法

当对service进行配置的时候就需要使用到provider()。比如，我们需要配置services在不同的部署环境里面（开发，演示，生产）使用不同的后端处理的时候就可以使用到了。