

assum

关于angularjs

- AngularJS主要考虑的是构建CRUD应用。
- 构建一个CRUD应用可能用到的全部内容包括：数据绑定、基本模板标识符、表单验证、路由、深度链接、组件重用、依赖注入。
- 如游戏，图形界面编辑器，这种DOM操作很频繁也很复杂的应用，和CRUD应用就有很大的不同，它们不适合用AngularJS来构建。像这种情况用一些更轻量、简单的技术如jQuery可能会更好。

angularjs——MVC模式

- angular中的 MVC模式 将web应用分解成各自独立的三层（视图、逻辑、数据）组件，使应用更加结构化，各层间耦合性非常低。
- 提升服务器性能：服务端不用提供jsp/php页面响应，仅提供API数据支持，服务器的负载大大降低，性能得到提升。
- 前后端分离：前端无需关注服务端实现过程，只需知道实现REST的API与专注于页面互动与用户体验的实现；后端也无需关注前端页面效果，只需专注API接口开发与性能提升。

angularjs——依赖注入（Dependency Injection）

- 概念：依赖注入DI，当一个对象在建立时，需要依赖于另一个对象，这是代码层中的依赖关系，当在代码中声明了依赖关系后，ng通过注入器将依赖的对象进行“注入”操作。
- 优势：通过ng中特有的依赖注入方式，将依赖的对象轻松注入任意需要的地方，而且不必关注被注入对象本身的逻辑，这种方式减轻了代码开发量，并且提高了工作效率。
- DI（依赖注入）是一种软件设计模式，主要为了解决组件获取它的依赖组件的问题。AngularJS使用一个专门的子系统（\$injector）进行DI管理，这个过程包括了创建组件、解析、获取它的依赖组件，并将这些依赖组件回传给请求组件。
- AngularJS 提供很好的依赖注入机制。以下5个核心组件用来作为依赖注入：
 - value
 - factory
 - service
 - provider
 - constant

//Value 是一个简单的 javascript 对象，用于向控制器传递值（配置阶段）：

// 定义一个模块

```
var mainApp = angular.module("mainApp", []);
```

// 创建 value 对象 "defaultInput" 并传递数据

```
mainApp.value("defaultInput", 5);
```

...

// 将 "defaultInput" 注入到控制器

```
mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {  
    $scope.number = defaultInput;
```

```

$scope.result = CalcService.square($scope.number);

$scope.square = function() {
    $scope.result = CalcService.square($scope.number);
}
});

```

```

//factory 是一个函数用于返回值。
//在 service 和 controller 需要时创建。通常我们使用 factory 函数来计算或返回值。
// 定义一个模块
var mainApp = angular.module("mainApp", []);
// 创建 factory "MathService" 用于两数的乘积 provides a method multiply to return multiplication
of two numbers
mainApp.factory('MathService', function() {
    var factory = {};

    factory.multiply = function(a, b) {
        return a * b
    }
    return factory;
});
// 在 service 中注入 factory "MathService"
mainApp.service('CalcService', function(MathService){
    this.square = function(a) {
        return MathService.multiply(a,a);
    }
});
...

```

```

//AngularJS 中通过 provider 创建一个 service、factory等(配置阶段)。
//Provider 中提供了一个 factory 方法 get(), 它用于返回 value/service/factory。
// 定义一个模块
var mainApp = angular.module("mainApp", []);
...
// 使用 provider 创建 service 定义一个方法用于计算两数乘积
mainApp.config(function($provide) {
    $provide.provider('MathService', function() {
        this.$get = function() {
            var factory = {};

            factory.multiply = function(a, b) {
                return a * b;
            }
            return factory;
        };
    });
});

```

```
//constant(常量)用来在配置阶段传递数值，注意这个常量在配置阶段是不可用的。  
mainApp.constant("configParam", "constant value");
```

angularJs 常用控件

- `angular-ui-router` : ui-router 支持嵌套视图多视图，用于替代ng内置的路由ngRoute服务以弥补其不足
- `angular-file-upload` : 基于angular开发的文件上传插件。
- `angular-ui-bootstrap` : 基于ng和bootstrap开发的UI组件指令，包含分页、风琴、按钮组件、日期控件、模态框等等。
- `angular-ui-tree` : 基于ng开发的树插件，支持双向数据绑定，以及树的增删查改。
- `angular-animate` : 基于ng的动画插件。
- `restangular` : 基于ng开发的用于简化ng请求服务的插件。
- `angular-cookies` : angular的cookie插件。

项目用得比较多的 `angular-ui-router` , `angular-ui-bootstrap` , `restangular`

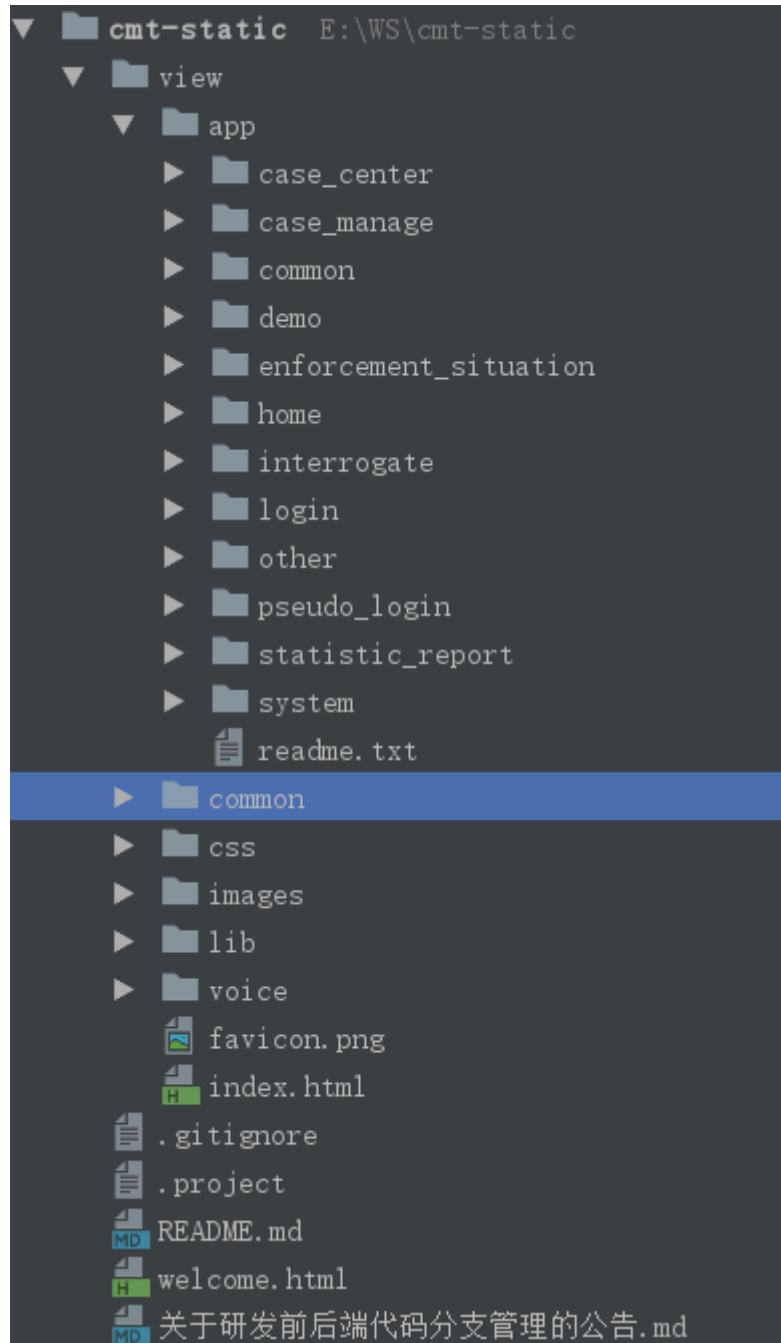
ng-route不能够嵌套深层路由，而ui-router可以,ui路由，ui-router和官方的ng-route相比较，它的处理方式更加简洁和易用，尤其是涉及到项目中大量路由嵌套时，使用ui路由能更加快捷方便的完成项目中路由的跳转处理。

项目中使用 `angular-ui-bootstrap` 中的模态框比较多，提供了一个关闭弹窗的弹窗回调会比较方便。

```
define(['restangular'], function () {  
    var module = angular.module('models', ['restangular', 'constants']);  
    module.factory('Models', function(Restangular, constant_models_url) {  
        var rest = Restangular.withConfig(function(RestangularConfigurer) {  
            RestangularConfigurer.setBaseUrl(constant_models_url);  
            RestangularConfigurer.setDefaultHeaders(  
                {  
                    'Access-Control-Allow-origin' : '*',  
                    'Access-Control-Allow-Headers' : 'X-Requested-With',  
                    'Access-Control-Allow-Methods' : 'GET',  
                    'X-Requested-With' : 'XMLHttpRequest',  
                    'If-Modified-Since': '0',  
                    'Cache-Control': 'no-cache'  
                }  
            );  
        });  
        /**  
         * 公共模块  
         */  
        rest.File = rest.all('common/file');//指纹图片上传  
        rest.ImgUploader = rest.all('image/upload');//图片上传  
        rest.ImgSaveSize = rest.all('image/savesize');//保存图片像素大小  
        rest.ImgSizeSave = rest.all('size_images');//保存图片像素大小  
        rest.ImgDelfiles = rest.all('image/deletefiles');//删除图片  
        rest.Attachments = rest.all('common/attachments');//系统附件  
        rest.OpeUploader = rest.all('operation/upload');//运维管理系统文件上传  
        return rest;  
    });  
});
```

```
Models.Devops.all('service/getCheckList')
  .post({magicid:'f498cf74b47fab88bd72b7d7fe18454b'}).then(function(res){
    if(res.state.code === 200){
      console.log(res);
    }
  });
```

项目架构



模块化是用AMD模块加载方式的requirejs进行。

```

var app = angular.module('app', [
    'home', //路由模块
    'login', //登录模块
    'other', //其他模块
    "services", //服务放在一个模块
    "filters", //过滤器放在一个模块
    "ui.bootstrap", //UI用了第三次插件，也是一个模块
    "directives" //指令放在一个模块
]);

```

angular.module('myApp', [])

后面的数组包含了一个字符串变量组成的列表，每个元素都是一个模块名称，本模块依赖于这些模块，依赖需要在在本模块加载之前由注入器进行预加载。

```

//过滤器
var filters = angular.module('filters', []);
filters.filter('trusthtml', function($sce) { // html加入可信任非转义字符
    return function(text, defaultNull) {
        var showText = defaultNull || '';
        if(typeof(text) == 'undefined'){
            return $sce.trustAsHtml(showText);
        }else{
            if(text && isNaN(text)){
                return $sce.trustAsHtml(text);
            }else{
                return $sce.trustAsHtml('<span>'+text+'</span>');
            }
        }
    }
});
/**
 * 小数转换为百分比
 */
filters.filter('percentage', function() {
    return function(point) {
        return Number(point*100).toFixed(1)+'%';
    };
});
//格式化文字，字典翻译

```

```

//service
var services = angular.module('services', []);
services.service('dateUtil', function() {
    /**
     * 返回距 1970 年 1 月 1 日之间的毫秒数(可用于比较时间先后)
     * @param {} Date 格式为：yyyy-mm-dd
     */
    this.formatTimesFromDate = function(Date){
        var arr = Date.split("-");
        var newDate = new Date(arr[0],arr[1],arr[2]);
    };
});

```

```

        var resultDate = newDate.getTime();
        return resultDate;
    }
});
//调用 注入dateUtil
var date = dateUtil.formatTimesFromDate ( new Date() )
//服务的话有日期服务 外设服务 功能性的服务（数组去重，判断是否为函数，移除对象的空属性，判断是否为IE等个功能）

```

```

//directives
var directives = angular.module('directives', ['services']);
//switch开关
directives.directive("commonToggle", function ($compile, normalUtil, $timeout, modalExt) {
    return {
        restrict: "E",
        replace: true,
        scope: {
            ngDisabled: '=',
            toggleConfig: '='
        },
        template: "<div class='common-toggle-container' ng-class='{true:'active'} [toggleConfig.disabled]'\><div ng-click='switchToggle()'\><div class='toggle-bar'\></div><div class='toggle-button'\></div></div>",
        link: function (scope, ele, attrs) {
            var toggleConfig = {
                disabled: false,
                onSelect: function () {}
            };
            scope.toggleConfig = angular.extend(toggleConfig, scope.toggleConfig);
            if (scope.ngDisabled) {
                scope.toggleConfig.disabled = scope.ngDisabled;
            }
            scope.switchToggle = function () {
                /* scope.toggleConfig.disabled = !scope.toggleConfig.disabled;
                scope.ngDisabled = scope.toggleConfig.disabled; */
                scope.toggleConfig.onSelect(scope.toggleConfig.disabled);
            }
            scope.$watch("ngDisabled", function (newVal, oldVal) {
                scope.toggleConfig.disabled = newVal;
            });
        }
    }
});

```

还有一些内置的服务器比如 `$controller` 和 `$filter` , `$http`

描述项目

遇到的问题

- (1) 没有状态管理，基于广播的通讯方式不太方便
- (2) 没有生命周期函数（钩子函数），不确定在什么时候调用方法，不太容易把控组件的进度
- (3) 业务复杂，不沟通好的话可能会推倒重做
- (4) 文档比较少，比较旧，查询不方便，没有建立起属于自身项目的一套前端文档说明
- (5) IE8测试不方便，仅有一台win7系统的IE8，调试过程没有谷歌的控制台那么强大

可以值得提升的点

- (1) 使用代码压缩混淆工具，或者使用webpack