

angularJS讲解与回顾

华苏科技

课程说明

- 对实战负责，不对说法负责，谨请异步[angular官网](#)
- angularJS –v 1.6.4
- 配合angular-ui-router –v 1.0.3做路由

angularJS的简单介绍

- 用来开发SPA的
- 吸收了MVC思想的JS框架
- 扩展了HTML
- 提供了一个非常快速的前端开发解决方案

启动应用

- 引入angularJS（下简NG）和路由JS文件
- 指明NG应用的根元素 `<html lang="zh-CN" ng-app="app">`
- 配置NG的接口，根模块 `var app = angular.module('app', ['ui.router'])`
- 手动启动NG应用 `angular.bootstrap(element, [modules], [config]);`

路由配置

```
.config(function ($urlRouterProvider, $stateProvider, $sceDelegateProvider) {  
    /**  
     * when:      对于给定的规则进行路由重定向    可以是正则表达式  
     * otherwise: 对未匹配到的进行路由重定向  
     */  
    $urlRouterProvider.when('', '/home').otherwise('home');  
    /**  
     * state 状态配置  
     */  
    $stateProvider.state('home', {  
        url: '/home',  
        template: 'hello',  
        controller: function ($scope, $rootScope, resolveFactory) {...},  
        /**  
         * 预加载  
         * resolve 属性里的值会在路由成功前被预先设定好，然后注入到控制器中  
         */  
        resolve: {  
            resolveFactory: function () {...}  
        }  
    }).state('jquery', {  
        /**  
         * 参数占位符  
         */  
        url: '/jquery?param1',  
        templateUrl: 'jquery.html'  
    });  
});
```

Models和controller

- 新建模块 `var app = angular.module('app', ['ui.router'])`
- 没有主要模块，而是声明性地指定应该如何引导
- 新建controller `app.controller('model', ['$scope', '$timeout', function (sp, timeout) {...}])`
- 通过ng-controller 指令指明要控制的DOM

作用域

- 创建新作用域并以原型继承：ng-repeat , ng-include , ng-switch , ng-view , ng-controller , ng-if , scope: true或transclude: true的指令
- 创建新作用域但不会原型继承： scope : {}的指令
- 作用域の説明、 AngularJS Inspector

NG表达式

- 类似JavaScript的代码片段
- 以undefined或null代替JS的ReferenceError 或TypeError
- 不能使用函数声明、条件、循环、正则、new、位运算
- 通常以简单计算、三元运算符、插值绑定的形式出现

双向数据绑定

- 数据层和视图层同步更新
- 一次性绑定 `<h1>{{::list[0].name}}</h1>`
- 只有发生在NG上下文环境中的变更才会做出自动地响应
- 延伸了JS事件——\$digest循环（包括\$watch队列）——\$apply
- input , select , textarea上的ng-model
- 配置model : ng-model-options
- 相关 : \$watch、 ng-change
- 最佳实践 : 采用对象形式绑定数据

DOM操作

- 只在指令里操作DOM
- 数据驱动、单元测试、丰富和可扩展的指令
- angular.element：将原始的DOM包装成jQuery元素
- jqLite：简化的jQuery，支持的[方法](#)
- 如果引入jQuery必须在angular之前引入

指令

- 通过DOM元素上的标记指示NG的HTML编译器 (\$compile) 将指定的行为附加到该DOM元素
- 指令的命名采用驼峰命名法，并且在调用时必须转化为-命名，即headerNav 转 header-nav
- ng-repeat
- ng-if、 ng-show、 ng-hide、 ng-switch
- ng-click、 ng-blur、 ng-copy、 ng-keydown、 ng-mouseover
- ng-class、 ng-style
- ng-options、 ng-selected

自定义指令

```
app.directive('headerNav', function () {
  return {
    /** 可选字符串参数，用以设置这个指令在DOM中以何种形式被声明 ...*/
    restrict: 'E',
    /** (布尔型) 默认为false(模板内容会加载到标签内部), true(模板内容会替换当前标签) ...*/
    replace: true,
    /** 当你希望创建一个可以包含任意内容的指令时 ...*/
    transclude: true,
    /** 当你需要其他指令的控制器支持时 ...*/
    require: '^?ngModule',
    /** 控制作用域 ...*/
    scope: {"config": '='...},
    /** HTML模板路径或直接通过template指定一段HTML片段 ...*/
    templateUrl: 'header.html',
    /** 创建内联控制器 ...*/
    controller: function ($scope,$http) {...},
    /** 通常在这里做DOM操作 ...*/
    link: function (sp, ele, att) {...},
    /** 如果我们希望在angular进行编译之前进行dom操作, 使用compile ...*/
    compile: function (ele, att) {...}
  };
});
```

过滤器 (filter)

- 用于格式化表达式的值
- 在视图里和JS里都可以使用
- date `{{1288323623006 | date:'yyyy-MM-dd HH:mm:ss Z'}}`
- limitTo `$filter('limitTo')(input, limit, begin);`
- lowercase、 uppercase、 orderBy、
- number : 返回的是字符串且带千位分隔符
- filter
- 可以连续过滤

自定义filter

```
/**
 * 第一个参数是名字, 第二个参数是一个返回一个用于过滤的函数的函数
 * 在视图上这样调用: {{128 | percent:0:'9.8'}}
 */
app.filter('percent', function () {
  /**
   * data: 作为第一个参数为遍历数组传进来的值
   * num 作为后来的参数作为参数使用, 可以增加多个
   */
  return function (data, num, assist) {
    if (data === 0 || data) {
      var v = parseFloat(data);
      /**
       * 返回一个值作为过滤的结果
       */
      return Number(Math.round(v * 10000) / 100).toFixed(num) + "%" + assist;
    }
  };
});
```

表单验证

- 每一个form指令都会创建一个FormController，用于跟踪和状态检测
- 必须添加name属性，以指明记录状态的变量
- 状态：\$dirty、\$invalid、\$error、\$submitted
- class：ng-valid、ng-submitted
- 方法：\$setPristine、\$setDirty、\$setUntouched
- ng-pattern：如果不是RegExp对象，则new RegExp('^str\$')，并且不使用g修饰符

自定义表单验证

```
app.directive('equals', function () {
  return {
    require: 'ngModel',
    link: function (scope, elm, attrs, ngModelCtrl) {
      function validateEqual(myValue) {
        /**
         * $eval: 执行angular表达式并返回结果
         */
        var valid = (myValue === scope.$eval(attrs.equals));
        /**
         * 更改有效性状态, 并通知表单
         */
        ngModelCtrl.$setValidity('equals', valid);
        /**
         * 返回期望的有效值
         */
        return valid ? myValue : undefined;
      }
      /**
       * 当通过NG方式更改时作为等待执行的函数数组
       */
      ngModelCtrl.$parsers.push(validateEqual);
      /**
       * 当通过NG方式更改时作为等待执行的函数数组
       * 最后一个返回值用作实际的DOM值
       */
      ngModelCtrl.$formatters.push(validateEqual);
    }
  };
});
```

[链接](#)

服务

- \$http、\$jsonpCallbacks
- \$compile、\$templateCache、\$timeout
- \$q
- \$sce

自定义服务

```
app.factory('Pool', ['$q', '$http', function (q, http) {
    var get = function (url, data) {
        var defer = q.defer(),
            promise = defer.promise,
            id = url + JSON.stringify(data); // 作为session的KEY

        if (sessionStorage.getItem(id)) {...} else {
            http.post('/deeplan.son.mro' + url, data && $.param(data)).then(function (res) {
                if (res.data.resultCode === 1) {


                    window.onbeforeunload = function () {...}; // 如果刷新页面则清空session, 做退步机制

                    if (['/netOverlayCommon/getVendors'].indexOf(url) > -1) {...} // 只有指定的请求才做缓存

                    defer.resolve(res.data);
                } else {
                    defer.reject(res.data);
                }
            }, function (res) {
                layer.msg('请求厂商数据接口出错 , ' + res.statusText);
            });
        }
        return promise;
    };
    return {
        vendorList: function () {
            return get('/netOverlayCommon/getVendors');
        }
    }
}]);
```

常用函数

- `angular.copy(source, [destination]);`
- `angular.equals(o1, o2);`
- `angular.forEach(obj, iterator, [context]);`
- `angular.isObject(value);`
- `angular.extend(dst, src);`



感谢