angular是用来构建单页面的应用。

使用的MVC模式，

M mode 模型 管理数据（处理json格式数据）

V view 视图 DOM

C Controller控制器 用来处理M和V之间的关系

通过以上实现了数据与界面的管理。

表达式支持：1，数据，2，运算3，数组，4，对象属性。

防压缩：压缩的时候会把函数的传入参数精简化，但是有些是不能精简的，为了防止压缩出错需要将函数以数组形式写入，并将形参用数组形式添加到前面。

例：[“$scope”,”$http”,function($scope,$http){}]

注意：

使用bootstarp的轮播时若在非主页添加的轮播需要更改左右按钮a链接的href属性的锚点为class，使用id会造成与路由的冲突。

## 首先

在底部导入框架，与控制文件js，底部导入有利于页面加载。

1，页面从ng-app开始，管理区域范围。

2，控制器Controller

3，Controller中的$scope充当MVC中的Data-Model角色，

$scope是表达式的执行环境。

$scope是一个树形结构，与DOM标签平行。

子$scope对象会继承父$scope上的属性。

每一个Angular应用只有一个根$scope对象，（一般位于ng-app上）。

$scope可以传播事件，类似DOM事件，可以向上也可以向下。

$scope的生命周期：创建，

4，依赖注入

除了自动注入$scope之外，还有$location,$window等等都可以注入。

5，模块化Module，作为启动点的模块。

1. 一切都是从angular。Module开始的。Controller(),directive(),filter()等API，都定义在module实例上。
2. 定义和特性
   1. 模块是Service，Directive,Filter配置信息的集合。
   2. 使用angular.module(..)创建或者获取模块
   3. 模块之间可以相互依赖
   4. 模块可以动态加载
   5. 切分成小模块便于进行单元测试和集成测试

|  |  |  |
| --- | --- | --- |
| $index  $index是从  0开始计：算的。 | [**数**](https://docs.angularjs.org/) | 重复元素的迭代器偏移量（0..length-1）**直接获取父级下标，只有两个ng-repate写在一起的时候，使用$ parent。$ index即可获取，若没多一级ng-if之类的判断条件，则接着在往上一级寻找即可。** |
| $first | [**布尔**](https://docs.angularjs.org/) | 如果重复的元素在迭代器中是第一个，则为true。 |
| $middle | [**布尔**](https://docs.angularjs.org/) | 如果重复元素在迭代器中的第 一个和最后一个之间，则为true。 |
| $last | [**布尔**](https://docs.angularjs.org/) | 如果重复的元素在迭代器中是最后的，则为true。 |
| $even | [**布尔**](https://docs.angularjs.org/) | 如果迭代器位置$index为偶数（否则为false），则为true。 |
| $odd | [**布尔**](https://docs.angularjs.org/) | 如果迭代器位置$index为奇数（否则为false），则返回true。 |

## Angular指令

angular是扩展的HTML属性，带有前缀ng-

1. ng-app指令初始化一个Angular应用程序。例：ng-app=”myApp”
2. ng-init指令初始化应用程序数据。例：ng-init="myMoney='1000'"
3. ng-model指令把元素值（比如输入域的值）绑定到应用程序。双向同步。

{{firstName}}是通过ng-model=”firstName”进行同步。双括号内是变量，后面是数据。

1. ng-cloak可以降低初始打开网页的乱码效果。
2. ng-bind可以将数据接收后在进行显示在网页。
3. ng-repeat指令会重复输出元素，并复制标签。ng-repeat="x in records"
4. ng-click指令可以设置点击事件。
5. ng-value绑定输入框
6. ng-class根据是否满足来添加class，例如：ng-class="active:$index=num.cid"
7. ng-style根据数据更改样式变化
8. ng-repeat对后台数据进行遍历，同时赋值所在区域。
9. ng-options可以直接应用于下拉列表，不用循环输出即可事件输出下拉列表

<select name="" ng-model="sh条件条件eng" ng-options="pro.name for pro in addre">前面的为想要输出的内容，for后面为循环得到的数组中的元素，后面就是个遍历输出。  
 <option value="">请选择省份</option>  
</select>

1. ng-model="radi" value="1"，单选中，使用不同的value，相同的绑定对象来输出不一样的值。
2. 多选按钮，游泳<input type="checkbox" ng-model="chebox.swim" ng-true-value="1" ng-false-value="0">，绑定的为不同的对象属性，因为多选按钮默认的是true和false所以需要更改为1和0，便于后台识别。

例：<!--ng-app定义区间-->  
<div ng-app="myMoney" ng-controller="MyControler as mc">

<!--as 意为转为 mc意为实例 -->  
输入工资：<input type="text" ng-model="myMoney" ng-init="myMoney='1000'">  
<p ng-model></p>

<!--双向绑定-->  
<p>{{myMoney}}</p>  
 <!--直接应用-->  
 <p no-cloak>{{myMoney}}</p>  
 <p ng-bind="myMoney +'是成绩'"></p>  
 <!--//单向绑定-->  
</div>

1. 自定义指令：一个单词无所谓，两个以上单词使用驼峰法来命名一个指令， runoobDirective, 但在使用它时需要以 - 分割,

<runoob-directive></runoob-directive>

应用方法有：1，标签，2，属性，3，注释，4，类名， 5标签前面可以加标识符

**return** {  
// template:"<h1>自定义标签</h1>"  
 templateUrl:"work.html" ,/\*可以传入标签也可以是页面\*/  
 restrict:"EA", /\*限制使用自定义标签的方式\*/

E:元素，A：属性，C：类名，M：注释，使用多中方法则输入多个，一个也可

replace：决定是否替代原数据  
 controller:**function** ($scope) {  
/\*将控制器写在自定义标签内，则控制范围即为该标签。控制器也可以写外边，但是写在外边是对\*/  
 $scope.name="豆豆" ;/\*控制器内可以访问数据\*/  
 }  
 }

.directive("studentList",**function** () {  
 **return**{  
 templateUrl:"work1.html", /\*可以继续后续往下写\*/  
 /\*controller:function ($scope,$attrs,$element) {  
/!\*控制器作为指令中的时有$attrs属性：得到自定义标签的所有属性；$element得到自定义标签的对象\*!/  
 $scope.name="宝宝";  
 }\*/

Link控制器  
// link方法的作用和controller几乎一样，通常两个写一个就行，依赖注入的参数不同。  
 link:**function** (scope,attrs,element) {/\*后面两个参数的名字可以自定义\*/  
 scope.info="哈哈";  
 }  
 }  
 })

作用与controller差不多，但写法略不同，用于自定义命令内部。

例： /\*controller:function ($scope,$attrs,$element) {  
/!\*控制器作为指令中的时有$attrs属性：得到自定义标签的所有属性；$element得到自定义标签的对象\*!/  
 $scope.name="宝宝"  
 }\*/  
// link方法的作用和controller几乎一样，通常两个写一个就行，依赖注入的参数不同。  
 link:**function** (scope,attrs,element) {/\*后面两个参数的名字可以自定义\*/  
 scope.info="哈哈";  
 }

### Ng-repeat过滤器

过滤器应用在ng-repeat中，对遍历出来的数据进行过滤后输出。使用一个管道字符（|）添加到表达式和指令中，AngularJS 过滤器可用于转换数据。

|  |  |
| --- | --- |
| **过滤器** | **描述** |
| currency | 格式化数字为货币格式。后接冒号加字符可以更改符号。 |
| filter | 从数组项中选择一个子集。在区域的所有内容中查询符合条件的，无法选择多个，或内区间。  过滤出相交内容filter:{num:0,price:2}；或：filter:num:price  过滤出或的内容：filter:serchs filter是过滤掉判断结果为false的内容，而保留结果为ture的；  $scope.seachs=function (it) {  return it.name.indexOf($scope.iport)!=-1||it.Ename.indexOf($scope.iport)!=-1  } |
| lowercase | 格式化字符串为小写。 |
| orderBy | 根据某个表达式排列数组。若想从大到小排列则后接true；后接属性，可以根据属性来进行排列；若用下拉框的方式来排列则将对象属性名写到下拉框的vlue中。 |
| uppercase | 格式化字符串为大写。 |

例：

<tr ng-repeat="g in goods | orderBy:'price' :true |filter:'双汇'">  
 <td>{{g.name}}</td>  
 <td>{{g.num}}</td>  
 <td>{{g.price | currency : "￥"}}</td>  
 <td>{{g.No | uppercase}}</td>  
</tr>

date 格式化

{{1490161945000 | date:"yyyy-MM-dd HH:mm:ss"}} // 2017-03-22 13:52:25前面的编号为：

number 格式化（保留小数）

{{149016.1945000 | number:2}}

{ 250 | currency:"RMB ￥ " }} // 结果：RMB ￥ 250.00

limitTo 截取

{{"1234567890" | limitTo :6}} // 从前面开始截取6位

{{"1234567890" | limitTo:-4}} // 从后面开始截取4位

创建自定义过滤器

app.filter('reverse', function() { //可以注入依赖

return function(text) { 传入的第一形参为左面的第一个，第二个或者之后的都是传入的，用冒号隔开即可。

return text1.concat("元") 返回一个自定义的筛选的方法。

}

});

## 创建模块对象

例：**var** app=angular.module("myApp",[]);//模块对象

### 创建一个控制器，名称为：MyController

**写法1：**

app.controller("MyController",**function** () {  
 //实现控制器创建controller属性，将名字添加在形参内，需要用大写；

后面是对名称的实现函数，写的是匿名函数。  
 //初始化数据  
**this**.price=5.9;  
**this**.num=20;  
//定义事件函数  
**this**.add=**function** () {  
 console.log("run");  
 **this**.num++;  
}  
});

在显示网页应转换为实例，ng-controller="MyController as mc" mc为实例名。

显示页调用为：mc.salary；

**写法2：**

//$scope作用域  
app.controller("MyController",**function** ($scope) {  
 //数据初始化  
 //price num add()  
 $scope.price=5.9;  
 $scope.num=20;  
 $scope.total=30;//总量

页面展示直接写属性即可，无需加 对象.

两种写法都可以，没有优劣性。

## 事件

angular中的事件只是更改了前缀，其他没变，前面的on不要了。例：onclick=ng-click;

## $scope服务

$scope是一个容器 ,保存在容器中的内容，可以被控制器所访问到。作用于，归属于控制器，又属于模块。

例：// $scope.name="Tom";  
// $scope.age=30;  
// $scope.address="瑞达路";  
$scope.student={  
 name:"Merry",  
 age:90,  
 address:"瑞达路"  
};

将数据保存在scope里面，例如：$scope.prise=6;

$scope是一个容器，放在容器内的内容可以被控制器所访问到。

## 路由

使用angular-ui-router.js文件。angular不刷新页面且更新内容，与ajax的不同是：可以后退，有历史可寻，且路由应该在配置器之前配置。

### 路由指令

ui-view路由的自定义标签是路由的视口标识。是系统自定义的路由，没有写名称不代表不能写名称，路由可以写多个。

### 定义区域后需要建立依赖，

例：var app=angular.module("myApp",["ui.router"]);

### 使用config方法配置路由，并在方法中配置接受数据，并设置参数

例：*app*.config(function (*$stateProvider*) { config方法获取返回的数据  
 *$stateProvider*.state("about",{ 方法  
 url:"/about", 路由名锚点，用来在网页中唯一标识页面，可随意起名。  
 templateUrl:"about.html" 设置显示的目标

template:”<h1>文字标签<h1>” 也可添加标签，方法相同。  
 }).state("more",{ 可以重复引用  
 url:"/more",  
 templateUrl:"more.html"  
 })  
});

### 页面添加

例：<**body** ng-app=**"myApp"**>  
<**div** ui-view></**div**> 内容更新区域。  
</**body**>

### 多层路由， $stateParams服务，根据路由中的目标继续路由

页面在点击跳转的a链接可进行回传参数，回传需要跳转的页面的标识。

例：ui-sref="page({id:a.id})"，将遍历出的a中的ID属性赋值给id传给后台，并同时进行跳转到page路由。

回传的信息在后台的路由信息中可以进行字符串拼接的形式指向不同的路径。例：url: "/page/:id",

若想在控制器中调取则必须使用$stateParams服务来调取前台数据，例如：$stateParams.id

js区域

angular.*module*("works", ["ui.router"])  
 .config(**function** ($stateProvider) {//$stateProvider 依赖注入  
 $stateProvider.state("workR", {  
 /\*路由名称\*/  
 url: "", /\*锚点信息\*//\*若为空则页面加载好之后直接执行\*/ templateUrl: "more.html", /\*要显示的内容\*/  
 controller: "phoneController"/\*通过控制器来获取数据并填充路由的界面\*/  
 }).state("page", { /\*对于不同的路由最好设置不同的控制器\*/  
 url: "/page/:id",/\*拼接锚点，后接前台获取的id属性\*/  
 templateUrl: "page.html",  
 controller: "phoneDetailsController",  
 });  
 // 获取手机信息的控制器  
 }).controller("phoneController", **function** ($http, $scope) {  
 $scope.seach = "";  
 $scope.sort = "age";  
 $http.get("phones/json/phones.json").success(**function** (data) {  
 $scope.phones = data;  
 });  
 // 获取每个手机的具体信息，$stateParams可以获取到前台返回的数据，例：$stateParams.id  
}).controller("phoneDetailsController", **function** ($http, $scope, $stateParams) {  
 // console.log($stateParams);//获取每一行的内容  
 **var** fileName = "phones/json/" + $stateParams.id + ".json";/\*拼接手机所对应的文件json\*/  
 $http.get(fileName).success(**function** (data) {子控制器获取get数据，  
 $scope.phone = data;  
 $scope.bImg = $scope.phone.images[0];/\*大图的src属性为获取到的phone的imges路径的第一个\*/  
 });  
 $scope.bigImg = **function** (dImg) {/\*鼠标进入事件，传入函数的为图片路径  
 大图的src路径为当前图片的路径\*/  
 $scope.bImg = dImg;  
 }  
});

HTML前端

**第一个子页面**

<ul>  
 <div class="seach">seach <input type="text" ng-model="seach">  
 排序方式 <select ng-model="sort">  
 <option value="age">名称</option>  
 <option value="name">发布日期</option>  
 </select></div>  
  
 <li ng-repeat="a in phones | filter : seach:a.name |orderBy : sort">  
 <img ng-src="phones/{{a.imageUrl}}" style="width: 200px;height: auto;">  
 <div class="right">  
 <!--angular自定义的src属性，值为跳转的需要使用路由名称，从路由配置进行转发-->  
 <!--把路由名称看做一个函数，内容为键值对函数，获得的内容为遍历内容的属性，赋值给id传给后台-->  
 <a ui-sref="page({id:a.id})"><h1>{{a.name}}</h1></a>  
 <p>{{a.snippet}}</p>  
 </div>  
 </li>  
</ul>

**第二个子页面**

<div>  
 <h3>{{phone.name}}</h3>  
 <img ng-src="phones/{{bImg}}">  
 <div>  
 <img style="width: 200px;height: auto;" ng-src="phones/{{img}}"  
 ng-repeat=" img in phone.images" ng-mouseenter="bigImg(img)">  
 </div> <!--将当前对象属性传回函数bigImg-->  
</div>

### **嵌套路由**

路由嵌套常用与路由页面中的再次路由，

.state("userinfo.details", { 意为该路由在userinfo中的嵌套路由  
 url: "/details",  
 views: { 定义路由  
 "userView": { 路由名称，直接使用的ui-view是angular自定义的

此处为自己定义的不同路由名称  
 templateUrl: "views/details.html",跳转路径  
 controller: "detailsPageController"嵌套路由中的控制器需要写在定义的路由中  
 }  
 }  
  
}

应用：将ui-view必须用属性方式显示，在后面输入路由的名称来唯一标识路由。

<div ui-view="userView"></div>

## 监听器

监听自定义标签的内容改变，用controller和link都可以

例：**return** {  
 template:"<input type='text' ng-model='name'>",  
 controller:**function** ($scope) {  
 $scope.name="豆豆";  
 $scope.$watch(**function** () {/\*当监听目标改变时会两次\*/  
 console.log("改变了")  
 });  
 $scope.name="dou";  
 },  
 link:**function** (scope,ele,attr) {

//ele是Dom对象,attr是jquery对象  
 scope.name="宝宝",  
 scope.$watch(**function** () {/\*当监听目标改变时会改变一次\*/  
 console.log("改变了")  
 })  
 }  
}

## 表单验证

必须绑定数据，form必须设置name

用户名：<input type="text" name="userName" required ng-model="user">  
<!--name为myForm中的userName的错误情况，若输入框为空则显示。ng-show的值为flase则不显示。-->  
<span style="color: red" ng-show="myForm.userName.$error.required">用户名不可为空</span><br>  
密码：<input type="password" ng-model="pwd" name="pwd" required ng-minlength="6" ng-maxlength="12">  
<span ng-show="myForm.pwd.$error.required">密码不可为空</span>  
<span style="color: red" ng-show="myForm.pwd.$error.minlength">密码长度不能低于6位</span>  
<span style="color: red" ng-show="myForm.pwd.$error.maxlength">密码长度不能高于12位</span><br>

其他的验证机制

|  |  |
| --- | --- |
| $dirty | 表单有填写记录 |
| $valid | 字段内容合法的 |
| $invalid | 字段内容是非法的 |
| $pristine | 表单没有填写记录 |

## 动画

ngAnimate 模型并不能使 HTML 元素产生动画，但是 ngAnimate 会监测事件，类似隐藏显示 HTML 元素 ，如果事件发生 ngAnimate 就会使用预定义的 class 来设置 HTML 元素的动画。

AngularJS 添加/移除 class 的指令:

* ng-show
* ng-hide
* ng-class
* ng-view
* ng-include
* ng-repeat
* ng-if
* ng-switch

ng-show 和 ng-hide 指令用于添加或移除 ng-hide class 的值。

其他指令会在进入 DOM 会添加 ng-enter 类，移除 DOM 会添加 ng-leave 属性。

当 HTML 元素位置改变时，ng-repeat 指令同样可以添加 ng-move 类 。

此外， 在动画完成后，HTML 元素的类集合将被移除。例如： ng-hide 指令会添加一下类：

* ng-animate
* ng-hide-animate
* ng-hide-add (如果元素将被隐藏)
* ng-hide-remove (如果元素将显示)
* ng-hide-add-active (如果元素将隐藏)
* ng-hide-remove-active (如果元素将显示)

例：类名为命令的样式

.ng-hide {  
    height: 0;  
}

## 表单校验

对input加上name属性，

对form加 novalidata 关闭默认校验

required不能为空

ng-shwo=”regForm.username.$dirty&&regForm.username.$invalid”

$dirty以为输入后不能为空，就是不能输入后又删除完。

ng-pattern=”正则函数”,不满足该函数则为true。

ng-show=”regForm.phone.$error.pattern”

## $http服务

可以在控制器中注入$http服务，$state服务， $state服务可实现页面的跳转。

### Post方式获取传输数据

.controller("regController", ["$http", "$scope","$state","$rootScope",**function** ($http, $scope,$state,$rootScope) {  
 $http({  
 url: "http://localhost:3000/user/reg", 请求路径  
 method: "post", 获取数据服务方式  
 data: { post上传参数方式为data  
 name: $scope.user  
 }  
 }).then(**function** success(data) { 回调中成功与失败返回函数，可获取返回内容  
 console.log(data);  
 **if** (data.data.resultCode == "0000") { 约定的内容为成功返回，当满足则执行  
 $rootScope.loginUser=data.config.data.name;   
 $state.go("home"); 成功后直接跳转，值为路由名称  
 }  
 }, **function** error(err) { 失败返回函数  
 console.log(err)  
 })  
}]);

### Get方式获取数据

$scope.changeObj = **function** () {  
 $http({  
 url: "http://localhost:3000/prolist",  
 method: "get",  
 params: {  
 cid: $scope.cids,  
 status: $scope.status   
 }  
 }).then(**function** success(data) {  
 console.log(data.data.result);  
 **var** num = Math.ceil(data.data.result.countAll / $scope.counts);  
 **for** (**var** i = 0; i < num; i++) {  
 $scope.numArr.push(i + 1)  
 }  
 $scope.ress = data.data.result.list;  
 })  
};

## 自定义服务

Angular自带有很多的服务，使用自定义服务可以减少重复服务，封装一个服务可以重复使用。在进行相同的angular操作时可以使用自己的服务。用传入参数的形式传入数据。但是在接受时存在异步的问题，

自定义服务方法：

1，用回调函数解决。

在服务中调用函数并传入参数为值，在控制层定义，服务层调用。

2，$q内置服务拿到promise就是承诺。

Var deferd=$q.dfder()

成功时：deferd.resolve(data)

失败时：deferd.reject(ere)

失败或者成功都会返回。

## Value服务

使用value服务可以对url或其他的字符内容进行统一的管理，使用该服务有利于共同管理可变的字符内容。使整个界面显得更加清晰更加整洁。

value(“服务名”，{

“服务中的操作”：<http://localhost:3000/use/IndexInfo>,

})