## Day01

一、常用的设计原则和设计模式

1、常用7大设计原则

①YAGNI

you aren't gonna need it

不要去写不需要的代码

②KISS

keep it simple and stupid

代码越简单越好

③DRY

Don't Repeat Yourself 不要重复自己

善于封装和重用：

提高代码的复用率

降低了测试难度

提高了代码的开发速度

④Least Knowledge Principle

最少知识法则：尽量减少代码（组件、模块、方法。。）之间的关系

⑤SRP

single responsibility principle

单一责任原则：每一个方法、模块只干一件事情

⑥高内聚 低耦合

内聚：一个模块之间内部的相互调用关系

耦合：不同模块之间的关系（降低耦合度--》解耦）

⑦OCP

open closed principle

开闭原则：

开-》扩展开放 方便添加新的功能

闭-》修改关闭 防止代码修改带来的隐藏的bug

（了解、掌握、精通）

2、设计模式

23种设计模式：

①创建型模式 共五种：工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式。

②结构型模式，共七种：适配器模式、装饰器模式、代理模式、外观模式、桥接模式、组合模式、享元模式。

③行为型模式，共十一种：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式。

了解常用的设计模式：

工厂方法模式、单例模式、适配器模式、组合模式、迭代子模式

MVC MVVM MVP MV\*\*

无论是哪一种设计模式，都是为了方便代码的复用，提供代码的质量

什么是MVC？

Model：模型数据

View：视图

Controller：控制器

为啥会出现MVC？

①网页开发功能越来越多，性能要求越来越高，需要编写的代码越来越多，切分职责

②为了代码的复用

③为了后期的迭代和维护

MVC的工作原理：

View中用户进行了操作，比如点了按钮，打了对勾等，在控制器中触发方法，在方法中对数据做一些增删改查等操作，由于数据和视图已经切分开，中间是可以通过数据绑定建立起关系。

https://www.wikipedia.org/

二、AngularJS概述

what？就是一个开源的前端网页框架 用来通过按照MVC的设计模式，来实现SPA(Single Page Application)

Angular（Google） React（Facebook） Vue（尤雨溪）

why? Google

四大特征：

1 采用了MVC的设计模式

2 双向数据绑定

3 依赖注入

4 模块化设计

where/when: SPA

how？引入对应的js文件

三、表达式和指令

1、表达式

AngularJS表达式由变量和运算符构成的，可以用在HTML中执行运算。

语法：<any>{{表达式}}</any>

2、指令

ngApp ：启动AngularJS ;指定ng的作用范围；如果多一个ngApp被调用，第一个生效

了解： ①启动阶段 ②初始化 ③编译连接 ④运行

ngInit：初始化数据 <any ng-init="变量名=值"></any>

初始化多个变量：

<div ng-init="num=2;age=20">

练习：初始化一个对象数组stuList（5个对象）,每一个学生对象都包含：sex、score、name；将对象数组显示在table中

tip：

如果ng的项目在首次加载时 出现了双花括号闪烁的情况，解决方案：ngBind

3、常用指令

① ngRepeat 循环的指令

语法1：

<any ng-repeat="临时变量名 in 集合"></any>

语法2：

<any ng-repeat="(key,value) in 集合"></any>

练习：通过ngRepeat来重构学生数组的练习。

注意事项：如果要遍历的集合中存在重复的数据，会报错的。

解决方案：<li ng-repeat=

"temp in list track by $index">

②ngIf

根据指令后表达式执行的结果的真假，选择是否将元素挂载入到DOM。

练习：学生数组已经显示在了table中，只显示80分以上的学生信息。

③ngShow/ngHide

<any ng-show="表达式"></any>

表达式执行的结果为真，就显示，否则是隐藏。

<any ng-hide="表达式"></any>

表达式执行的结果为真，就隐藏，否则是显示。

④ngSrc

ngSrc是所有的ng指令唯一的比较特殊的一个指令：特殊需要读取变量名必须加上双花括号。

img ng-src="img/{{imgUrl}}"

⑤ngChecked ngDisabled

input type='checkbox' ng-checked='表达式'

根据表达式执行结果的真和假 来决定是否默认被选中

button ng-disabled='表达式'

根据表达式执行结果的真和假 来决定是否禁用

⑥ngClass/ngStyle

ngApp ngInit ngBind ngRepeat ngIf ngShow ngHide ngSrc ngChecked ngDisabled ngClass ngStyle...

4、ng采用了MVC的设计模式来写app

①采用了MVC的设计模式

②双向数据绑定

③依赖注入

④模块化设计

通过ng来实现基于MVC的app的基本步骤：

①创建模块、使用模块

var app = angular.module(‘模块名称’,['依赖模块1','依赖模块2']);

ng-app='模块名称'

②创建控制器、调用控制器（Controller）

app.controller(‘控制器名称’,func);

ng-controller='控制器名称'

③在控制器定义数据（Model）

$scope.变量 = 值；

④将数据绑定到视图去显示

通过双花括号等方式，将数据显示在视图中

ngClick 是用来绑定单击事件

ngChange/ngDblClick/...

在js中绑定事件的方式，在ng中都有对应的解决方案

on\*\*\* --> ng\*\*\*

学习ng之前做项目的思路主要是针对DOM操作，

学习ng之后思路要切换到数据。

初始化数据的方式：

①ngInit 全局变量

②通过$scope定义变量：局部变量，作用范围限定在控制器的范围内 （建议）

综合练习：

按照MVC的方式来处理：实现显示一个菜品列表，并且实现一个加载更多，往视图中插入一行数据

①定义数据并显示出来

②实现加载更多的功能

## Day02

一、自定义指令

指令：在一个特定的DOM元素执行的函数

1、指令的自定义和使用

指令的自定义：

app.directive('指令名称',function(){

return {

template:'渲染的视图'

}

})

定义指令时的相关属性：

①template 是一个模板，指定指令被调用时要显示的内容

②restrict 限定指令的使用场景 E(element)A(attribute)C(class)M(comment)

③replace 根据布尔类型的数据决定是否替换调用调用指定时所指定的元素

使用自定义指令：按照烤串式的写法，通过横杆连接起来

testHello --> test-hello

ngApp --> ng-app

ngRepeat --》 循环

ngIf--> 判断

注意事项：

①命名规则

指令在定义是驼峰式，在调用时烤串式

②指令在命名时，一般都是由两部分构成，前缀多数情况都是当前模块或者项目的简写 ，后缀多数都是为了表达它的功能

2、高级功能：给指令传参数

①传递参数

test-name='123'

②接收参数

scope:{

testName:'@'

}

练习：创建一个自定义的指令(tsDirective) 按照元素的方式调用并传递参数(tsNum=100)，将传递来的参数接收到并显示在h1标签中。

面试题：要求通过自定义指令来实现一个多选的指令。

数据 视图

定义一个数组，传递给自定义指令，在指令内部将数组中内容显示。

使用自定义指令传参时，其实有多种方式

@ =

ngApp ngInit ngBind ngRepeat ngIf....

二、双向数据绑定

方向1：将model绑定到view

方式：①双花括号 ②常见的ng指令

方向2：将View中表单组件，用户操作的结果绑定到model

方式：ngModel

监听数据变化的方法：

$scope.$watch('变量名称',

function(newValue,oldValue){

})

如何动态生成select中的option？

<select

ng-model="imgObj"

ng-options="

imgObj.name for imgObj in imgList">

</select>

三、过滤器 常见方法

1、内置过滤器

过滤器：实现对表达式结果的筛选、过滤、格式化

支持多重过滤

语法：{{ 表达式 | 过滤器 | 过滤器}}

currency 货币样式格式化的过滤器

date 日期

uppercase/lowercase 大小写转换

orderBy 排序

number 指定显示小数点后几位

limitTo:3

练习：有一个对象数组 carList,有5条数据，每个car都包含price，brand；

要求显示价格最高的3辆汽车

2、自定义过滤器

过滤器的本质 就是一个方法。

app.directive

app.controller

app.filter('过滤器名字',function(){

return function(myInput,arg1){

return '处理后的数据'

}

})

3、方法

记住：功能、输入的参数、返回值

angular.toJson 将对象或者其他的集合 转换为json格式的字符串，方便我们进行发送给服务器或者本地的存储等。。。

angular.fromJson 将json格式的字符串 转换为 方便操作的对象或者其他的集合。

。。

四、服务的基础知识

服务的本质就是一个单例对象（符合单个实例模式的对象：在应用程序只存在一个实例，如果尝试着创建同一类型的该实例，内部会做处理，保证整个app只有一个实例 ），在对象指定数据和方法。

服务存在的意义就是把需要反复用到的数据或者方法给封装在服务中，在需要使用时直接注入到对象去使用，而无需关心服务的内部实现。

常用的服务：

$scope/$location/$window/$interval/$timeout/$http...

服务的使用方式：

①在创建的ng对象中，注入进来 function($scope,$location)

②调用服务

$location.host()

$location.port()

。。。。

面试题：

$scope 和 $rootScope关系

①控制器之间的$scope是相互隔离的，没有关系的

②$scope就是$rootScope的子作用域对象

③$rootScope的id是1，第一个被加载的控制器的$scope的id就是2，其实控制器的$scope的id是按照控制器的加载顺序递增的

④可以将要共享的数据 放在$rootScope中，这样一来，所有的自作用对象，都可以读取该数据

面试题：不同的控制器之间有哪些通信方式

①借助于$rootScope

②借助于控制器之间的嵌套

③借助于事件

ngModel就是一个双向数据绑定的指令，既可以将数据绑定到视图，又可以将视图中用户操作的结果绑定到数据。

综合练习：实现全选

0、准备工作（模块、控制器）

1、初始化数据

是一个对象数组：对象包括名字、生日、是否选中

2、显示在table

3、根据选中结果做一个逻辑运算，根据结果判断是否要全部选中

4、将全部选中的结果 赋值给每一个部分选中的数据

## Day03

一、借助于事件完成不同控制器之间的通信

方法：

//完成事件的绑定

$scope.$on('eventName',func)

//完成事件的触发并传递参数(父控制器向子控制器传递传值)

$scope.$broadcast('eventName',data)

//完成事件的触发并传递参数(子控制器向父控制器传递传值)

$scope.$emit('eventName',data)

二、ng内置的常用服务

1、双向数据绑定的工作原理：

每一次去完成绑定的时候，在ng框架背后都会自动添加一个监听watcher,每一个监听都对应着有一个处理函数将自动的更新的DOM。

在ng中有一个循环，这个循环会不断的判断数据是否发生了变化，如果发生了变化，就去执行DOM操作的函数，将这个循环称之为$digest

当执行$scope.$digest()，手工的去触发当前作用域对象的循环.

$scope.$apply 触发的是$rootScope以及所有的子作用域对象

2、定时器相关的服务

$interval $timeout

promise = $interval(func,500) $interval.cancel(promise)

promise = $timeout(func,500) $timeout.cancel(promise)

3、$http

在与后台进行交互，有哪些注意事项？？

①请求的地址和端口号、参数

②出错了，调试错误

2.1 查看控制台的报错信息

2.2 查看network:

参数、地址、端口、header设置是否正确,

返回的数据是否和预期的一致。

$http

.get('url')

.success(function(data){})

.error(function(){//异常处理})

$http({method:'get',url:'data/test.json'})

.success().error()

实现一个功能：

点击视图中的按钮时，发送请求给服务器，同时传递一个参数(name=zhangsan)；服务器端接收到数据之后，返回一个json对象 {result:'hello zhangsan'}

步骤：

①编写php文件

启动xampp/apache

按照要求 编写php文件

测试

②在前端中指定按钮的点击处理函数

发起请求

发起post请求,传递参数，请注意：

//设置请求头消息

app.run(function($http){

$http.defaults.headers.post ={

'Content-Type':

'application/x-www-form-urlencoded'

}

})

$http.post('url',data).success().error();

三、自定义服务

1、service

app.service('服务名称',function(){

this.变量 = 值；

this.方法名字 = function(){

}

})

练习：通过service创建一个$show的服务，在服务中

实现一个showAlert的方法，在方法中通过alert,显示一个msg。

在视图中，放置一个按钮，点击按钮时，调用自定义服务中的showAlert方法，显示'Hello customService'

2、借助于factory

app.factory('服务名称',function(){

return {

key1:value1,

key2:func

}

}

)

无论时哪一种创建的服务的方式，用起来时和内置的服务用法是一致的。

练习：

心跳连接的服务 $heartBeat，通过factory去创建服务，并封装两个方法，一个开始心跳startBeat(启动周期性定时器，每隔0.5s，在控制台打印一句话),一个停止心跳stopBeat(结束掉周期性定时器)。

在视图中有两个按钮，点击其中一个开始心跳，另外一个结束心跳。

3、借助于constant/value去创建一个常量服务或者一个变量服务

创建一个常量服务，可以把应用当中一些不会变化的数据存储在常量服务中

app.constant('$baseConfig',{

baseUrl:'10.20.30.40'

})

创建一个变量服务，把应用中经常变化的数据存储在变量服务中

app.value('$test',{

token:''

})

四、依赖注入

依赖注入3种方式：

推断式（猜测式）

标记式

行内式

代码在开发环境下编写完毕之后，真正的发布到线上都需要做哪些工作？

①修改连接服务器的地址（开发服务器-》生产服务器）

②压缩

uglify yui-compressor ...

③发布到线上

压缩所进行的操作：

①删除了注释

②没有语义的空白字符去掉了

③方法定义的形参被简化，尽可能的简化变量的名称、函数名、形参名称（混淆），但是有些关键的数据（数组、集合、字符串..）不会给处理的.

1、推断式（不建议）

app.controller('myCtrl',function ($scope,$show) {

$scope.handleClick = function () {

$show.showAlert();

}

})

2、标记式

var ctrlFunc = function ($scope,$show) {

$scope.handleClick = function () {

$show.showAlert();

}

}

//做标记

ctrlFunc.$inject = ['$scope','$show'];

app.controller('myCtrl',ctrlFunc)

注意事项：在标记的数组中的服务顺序 要和在设置ng对象时的回调函数注入的顺序是要保持一致。

3、行内式依赖注入（推荐）

要求在创建ng对象时，允许将个数组作为参数，数组有要求，数组中依赖的服务名称,数组的最后一个元素要求必须是一个方法。

app.controller(

'myCtrl',

['$scope','$location',

function ($scope,$location) {

$scope.count = 10;

}]

);

注意事项：在标记的数组中的服务顺序 要和在设置ng对象时的回调函数注入的顺序是要保持一致。

4、完成依赖注入 $injector

$injector.has(‘$location’) //判断一个服务是否存在

$injector.get(‘$location’) //得到一个服务的实例

每日一练：

采用service创建一个自定义服务$cart,在服务中封装2个方法，一个向购物车添加数据，一个从购物车中删除数据。

在注入$cart服务时，要求是采用行内式依赖注入。

# Day04

一、知识点回顾以及练习讲解

二、模块化设计

ng的四大特征：

①采用了MVC的设计模式

②双向数据绑定 ngModel

③依赖注入

④模块化设计

1、模块之间如何相互调用？？

在创建自定义模块时，在依赖模块列表中，直接指定所需要依赖的模块名称，在自定义模块注入所需要用到的服务等去使用就可以

2、一个模块可以包含哪些？

控制器

服务

过滤器

指令

三、路由模块 ngRoute

angular就是一个基于MVC的开源的实现SPA的js框架

ngRoute实现SPA（single page application）,整个应用程序中只有一个完整的页面，其余的都是一些代码片段。

SPA的工作原理：

①当地址栏加载 http://127.0.0.1/index.html#/start

②首先请求完整的页面 index.html

③解析url中的路由地址 start

④根据路由词典，找到路由地址所对应的配置信息有一个url：真实的页面地址 \*\*\*\*/start.html

⑤请求真实的页面ajax

⑥将页面显示在视图中

1、具体实现SPA步骤：

①引入angular.js angular-route.js

script src=''

②创建自定义模块，指定依赖于ng和ngRoute

angular.module('myApp',['ng','ngRoute'])

③指定一个容器 盛放代码片段

div ng-view

④创建应用程序中需要用到的代码片段

⑤配置路由词典

app.config(function($routeProvider){

$routeProvider

.when('/login',

{templateUrl:''})

.otherwise({redirectTo:'/login'})

})

⑥测试

在地址栏去修改#后边的路由地址，能够加载对应的代码片段并显示

2、代码片段之间跳转的方式

①直接修改地址栏中的路由地址（开发人员自己测试）

②通过a标签

<a href='路由地址'></a>

// <a href='/testStart'></a>

③通过js的方式

$location.path('路由地址');

// $location.path('/tsLogin');

练习：通过ngRoute来实现一个SPA，实现3个代码片段： login register myPerson

要求：

login 有两个按钮：登录(跳转到myPerson)、注册(跳转到register)

register 有一个按钮：注册完成(跳转到login)

myPerson： 有一个超链接:退出登录 (跳转login)

3、页面传参

①明确发送、接收

②配置接收方的路由

when('/myLogin',{})

==>

when('/myLogin/:num',{})

③接收发送方所传递过来的数据

$routeParams.num

④发送数据

案例：

创建一个SPA，有2个代码片段,list detail

要求：点击list中列表项，能够将数组的下标发送给detail，detail接收到数据显示在视图中。

步骤：

①创建一个SPA

②基本的跳转

③传参

3.1 list --> detail（接收）

3.2 配置接收方的路由

3.3 接收参数

3.4 发送参数

（4:05 - 4:25）

综合练习：

创建一个SPA，有3个代码片段，分别是：check.html pay.html send.html

check:商品的查看页面

按钮（去支付） -->跳转到pay，将商品的价格10传递给pay

pay: 商品的支付页面

接收到发来的价格并显示在视图中

button(支付完成) --》点击跳转send

send:商品的发货页面

a 查看详情 --》 跳转到check

传参：

①发送check --> 接收 pay

②pay 配置路由地址

/myPay --> /myPay/:price

③接收参数

$routeParams.price

④发送参数

$location.path('/myPay/10');

<a href="#/myPay/10"></a>

①通过创建SPA的基本步骤

②跳转3种方式

③传参的基本步骤

四、动画模块

如何使用动画模块？

①引入对应的angular.js angular-animate.js

②自定义模块时指定依赖列表: ng ngAnimate

③如何使用

ng-leave

ng-leave-active

ng-enter

ng-enter-activ

五、ng处理表单验证

$dirty 处理过的表单元素

$pristine 未处理过的表单元素

$valid 有效

$invalid 无效

$error 对象

基本步骤：

①给表单以及输入组件 指定name属性

②给验证的表单组件 指定上使用ngModel指令

③使用属性

myForm.u\_name.$valid/$dirty/$error.email...

练习：

创建一个表单，在表单中要求输入用户名和密码，要求用户名和密码都是必须输入的，表单中有一个提交的按钮，是否能够点击要和当前表单是否通过验证绑定在一起。可以点击，点击在控制台打印当前的用户名和密码