# 介绍

AngularJS是一款由Google公司开发维护的前端MVC框架，其克服了[HTML](http://baike.baidu.com/view/692.htm" \t "_blank)在构建应用上的诸多不足，从而降低了开发成本提升了开发效率。

## 特点

AngularJS与我们之前学习的jQuery是有一定的区别的，jQuery更准确来说只一个类库（类库指的是一系列函数的集合）以DOM做为驱动（核心），而AngularJS则一个框架（诸多类库的集合）以数据和逻辑做为驱动（核心）。

框架对开发的流程和模式做了约束，开发者遵照约束进行开发，更注重的实际的业务逻辑。

AngularJS有着诸多特性，最为核心的是：模块化、双向数据绑定、语义化标签、依赖注入等。

与之类似的框架还有BackBone、KnockoutJS、Vue、React等。

## 下载

1、通过[AngularJS](https://angularjs.org/)官网下载，不过由于国内特殊的国情，需要翻墙才能访问。

2、通过npm下载，npm install angular

3、通过bower下载，bower install angular

## 体验AngularJS

见代码示例1-01.html

## MVC

MVC是一种开发模式，由模型（Model）、视图（View）、控制器（Controller）3部分构成，采用这种开发模式为合理组织代码提供了方便、降低了代码间的耦合度、功能结构清晰可见。

模型（Model）一般用来处理数据（读取/设置），一般指操作数据库。

视图（View）一般用来展示数据，比如通过HTML展示。

控制器（Controller）一般用做连接模型和视图的桥梁。



通过ThinkPHP来演示后端MVC的执行流程，其重点在于理解。

MVC更多应用在后端开发程序里，后被引入到前端开发中，由于受到前端技术的限制便有了一些细节的调整，进而出现了很多MVC的衍生版（子集）如MVVM、MVW、MVP、MV\*等。

注：做为初学可以不必过于在意这些概念。

# 模块化

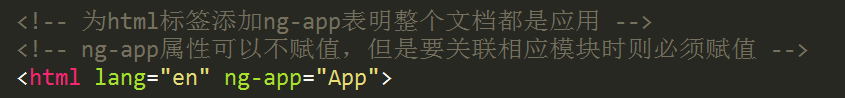
使用AngularJS构建应用（App）时是以模块化（Module）的方式组织的，即将整个应用划分成若干模块，每个模块都有各自的职责，最终组合成一个整体。

采用模块化的组织方式，可以最大程度的实现代码的复用，可以像搭积木一样进行开发。



## 定义应用

通过为任一HTML标签添加ng-app属性，可以指定一个应用，表示此标签所包裹的内容都属于应用（App）的一部分。



## 定义模块

AngularJS提供了一个全局对象angular，在此全局对象下存在若干的方法，其中angular.module()方法用来定义一个模块。



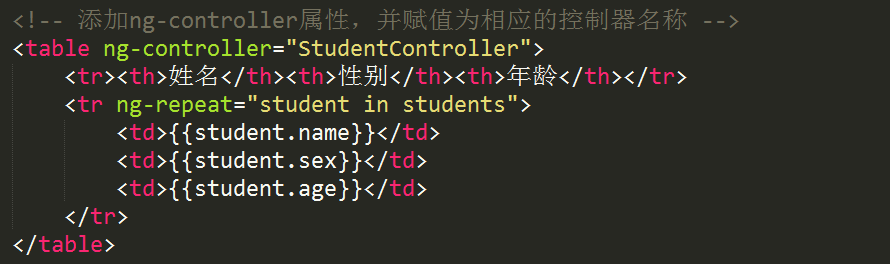
注：应用（App）其本质也是一个模块（一个比较大的模块）。

## 定义控制器

控制器（Controller）作为连接模型（Model）和视图（View）的桥梁存在，所以当我们定义好了控制器以后也就定义好了模型和视图。



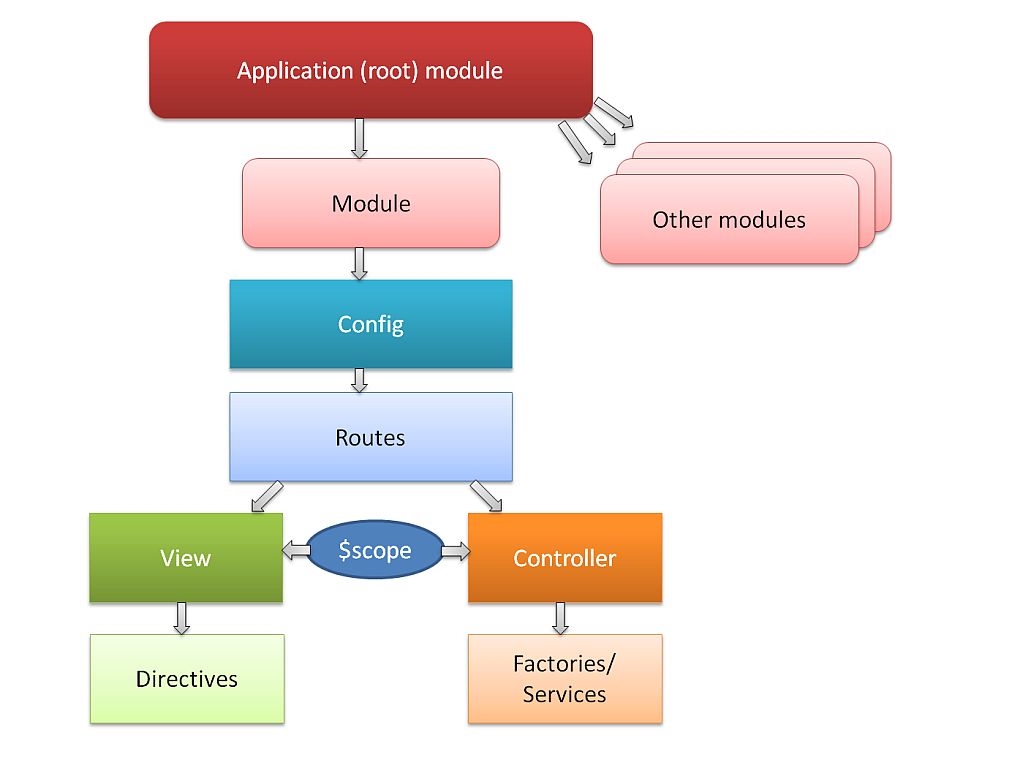
模型（Model）数据是要展示到视图（View）上的，所以需要将控制器（Controller）关联到视图（View）上，通过为HTML标签添加ng-controller属性并赋值相应的控制器（Controller）的名称，就确立了关联关系。



见代码示例2-01.html

以上步骤就是AngularJS最基本的MVC工作模式。

下图是AngularJS的结构，学习AngularJS会围绕下图的结构展开。



# 指令

HTML在构建应用（App）时存在诸多不足之处，AngularJS通过扩展一系列的HTML属性或标签来弥补这些缺陷，所谓指令就是AngularJS自定义的HTML属性或标签，这些指令都是以ng-做为前缀的，例如ng-app、ng-controller、ng-repeat等。

## 内置指令

ng-app 指定应用根元素，至少有一个元素指定了此属性。

ng-controller 指定控制器

ng-show控制元素是否显示，true显示、false不显示

ng-hide控制元素是否隐藏，true隐藏、false不隐藏

ng-if控制元素是否“存在”，true存在、false不存在

ng-src增强图片路径

ng-href增强地址

ng-class控制类名

ng-include引入模板

ng-disabled表单禁用

ng-readonly表单只读

ng-checked单/复选框表单选中

ng-selected下拉框表单选中

见代码示例3-01.html

注：后续学习过程中还会介绍其它指令。

## 自定义指令

AngularJS允许根据实际业务需要自定义指令，通过angular全局对象下的directive方法实现。



见代码示例3-02.html

# 数据绑定

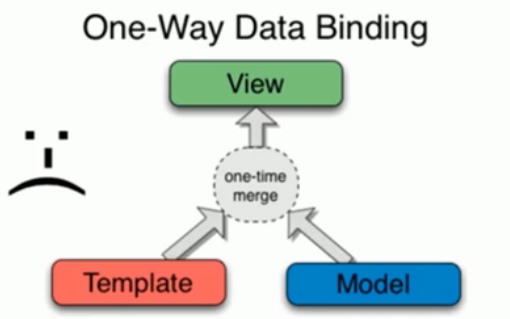
AngularJS是以数据做为驱动的MVC框架，所有模型（Model）里的数据经由控制器（Controller）展示到视图（View）中。

所谓数据绑定指的就是将模型（Model）中的数据与相应的视图（View）进行关联，分为单向绑定和双向绑定两种方式。

## 单向绑定

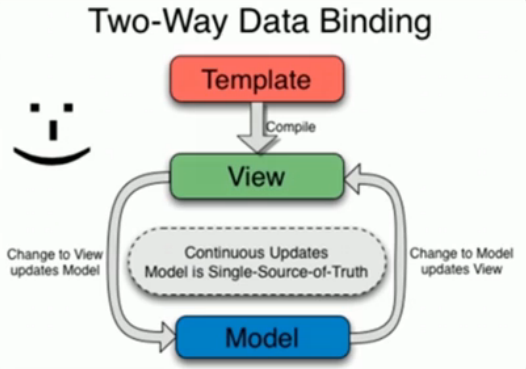
单向数据绑定是指将模型（Model）数据，按着写好的视图（View）模板生成HTML标签，然后追加到DOM中显示，如之前所学的artTemplate 模板引擎的工作方式。

如下图所示，只能模型（Model）数据向视图（View）传递。



## 双向绑定

双向绑定则可以实现模型（Model）数据和视图（View）模板的双向传递，如下图所示。



## 相关指令

在AngularJS中通过“{{}}”和ng-bind指令来实现模型（Model）数据向视图模板（View）的绑定，模型数据通过一个内置服务$scope来提供，这个$scope是一个空对象，通过为这个对象添加属性或者方法便可以在相应的视图（View）模板里被访问。

注：“{{}}”是ng-bind的简写形式，其区别在于通过“{{}}”绑定数据时会有“闪烁”现象，添加ng-cloak也可以解决“闪烁”现象，通过ng-bind-template可以绑定多个数据。

解决原理：加载网页时候，Angular会生成一段css代码，ng-cloak是个属性选择器，选中的标签 Display:none，这时候这个标签不会显示，等到Angular加载完毕后，才会使起显示。

见代码示例4-01.html

通过为表单元素添加ng-model指令实现视图（View）模板向模型（Model）数据的绑定。

见代码示例4-02.html

通过ng-init可以初始化模型（Model）也就是$scope。

见代码示例4-03.html

AngularJS对事件也进行了扩展，无需显式的获取DOM元素便可以添加事件，易用性变的更强。通过在原有事件名称基础上添加ng-做为前缀，然后以属性的形式添加到相应的HTML标签上即可。如ng-click、ng-dblclick、ng-blur等。

见代码示例4-04.html

通过ng-repeat可以将数组或对象数据迭代到视图模板中，ng-switch、on、ng-switch-when可以对数据进行筛选。

见代码示例4-05.html

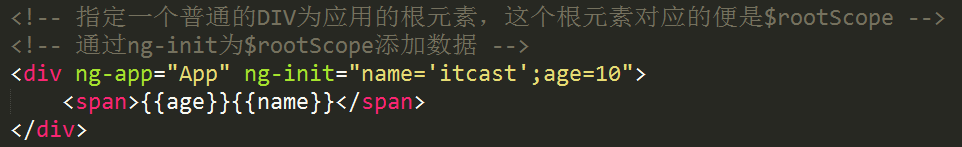
# 作用域

通常AngularJS中应用（App）是由若干个视图（View）组合成而成的，而视图（View）又都是HTML元素，并且HTML元素是可以互相嵌套的，另一方面视图都隶属于某个控制器（Controller），进而控制器之间也必然会产生嵌套关系。

每个控制器（Controller）又都对应一个模型（Model）也就是$scope对象，不同层级控制器（Controller）下的$scope便产生了作用域。

## 根作用域

一个AngularJS的应用（App）在启动时会自动创建一个根作用域$rootScope，这个根作用域在整个应用范围（ng-app所在标签以内）都是可以被访问到的。



见代码示例5-01.html

## 子作用域

通过ng-controller指令可以创建一个子作用域，新建的作用域可以访问其父作用域的数据。

见代码示例5-02.html

# 过滤器

在AngularJS中使用过滤器格式化展示数据，在“{{}}”中使用“|”来调用过滤器，使用“:”传递参数。

## 内置过滤器

1、currency将数值格式化为货币格式

2、date日期格式化，年（y）、月（M）、日（d）、星期（EEEE/EEE）、时（H/h）、分（m）、秒（s）、毫秒（.sss），也可以组合到一起使用。

3、filter在给定数组中选择满足条件的一个子集，并返回一个新数组，其条件可以是一个字符串、对象、函数

4、json将Javascrip对象转成JSON字符串。

5、limitTo取出字符串或数组的前（正数）几位或后（负数）几位

6、lowercase将文本转换成小写格式

7、uppercase将文本转换成大写格式

8、number数字格式化，可控制小位位数

9、orderBy对数组进行排序，第2个参数可控制方向

见代码示例6-01.html

## 自定义过滤器

除了使用AngularJS内建过滤器外，还可以根据业务需要自定义过滤器，通过模块对象实例提供的filter方法自定义过滤器。



见代码示例6-02.html

# 依赖注入

AngularJS采用模块化的方式组织代码，将一些通用逻辑封装成一个对象或函数，实现最大程度的复用，这导致了使用者和被使用者之间存在依赖关系。

所谓依赖注入是指在运行时自动查找依赖关系，然后将查找到依赖传递给使用者的一种机制。

常见的AngularJS内置服务有$http、$location、$timeout、$rootScope等

## 推断式注入

没有明确声明依赖，AngularJS会将函数参数名称当成是依赖的名称。



这种方式会带来一个问题，当代码经过压缩后函数的参数被压缩，这样便会造成依赖无法找到。

见代码示例7-01.html

## 行内注入

以数组形式明确声明依赖，数组元素都是包含依赖名称的字符串，数组最后一个元素是依赖注入的目标函数。



推荐使用这种方式声明依赖

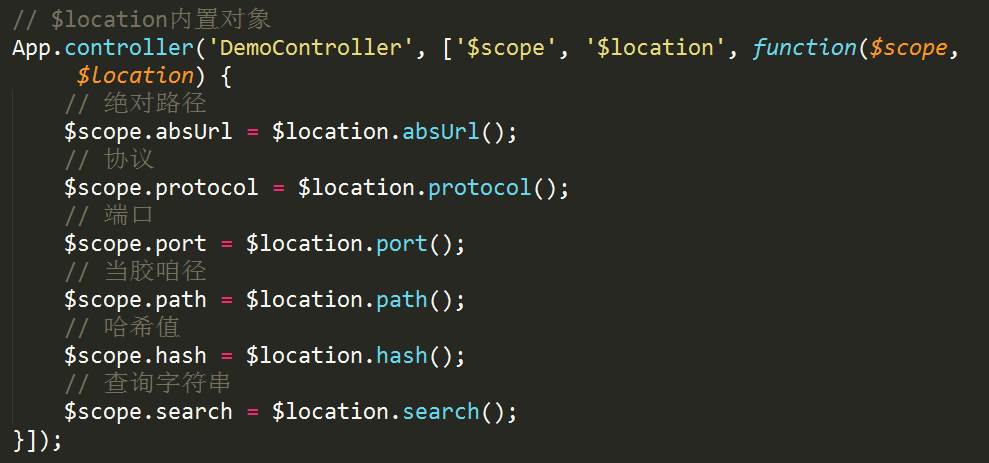
见代码示例7-02.html

# 服务

服务是一个对象或函数，对外提供特定的功能。

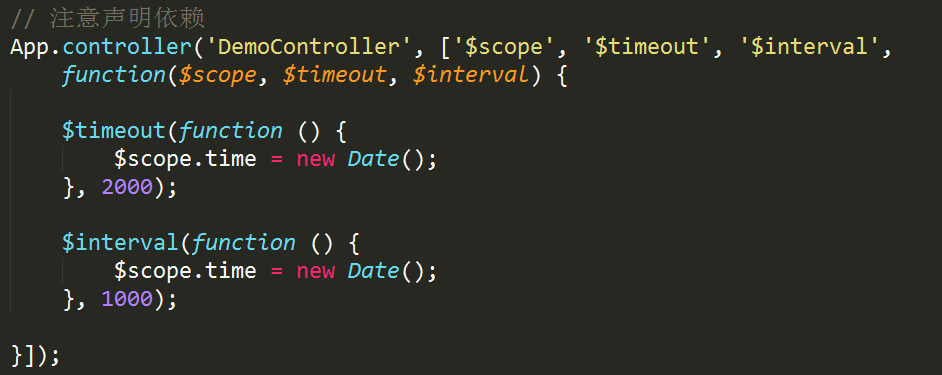
## 内置服务

1、$location是对原生Javascript中location对象属性和方法的封装。



见代码示例8-01.html

1. $timeout&$interval对原生Javascript中的setTimeout和setInterval进行了封装.

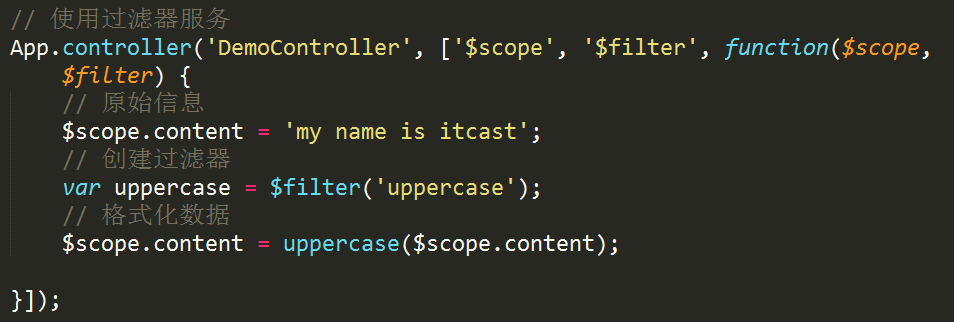


退出循环：

**var** timer = $interval(**function** () {$scope.**now** = **new** Date;  
 }, 1000);

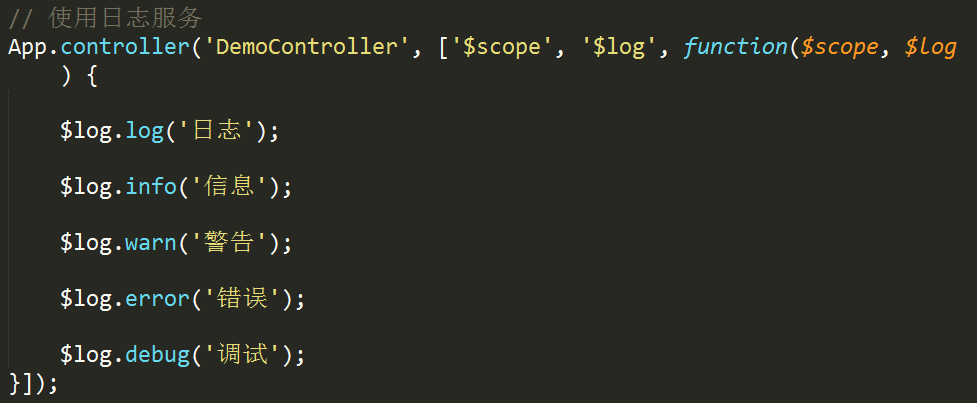
$interval.cancel(timer);

3、$filter格式化数据。



见代码示例8-03.html

4、$log打印调试信息



见代码示例8-04.html

1. $http用于向服务端发起异步请求。

$http其实是对JS原生的XMLHttpRequest的再包装。



同时还支持多种快捷方式如$http.get()、$http.post()、$http.jsonp。

注：各参数含义见代码注释。

见代码示例8-05.html

## 自定义服务

通过上面例子得知，所谓服务是将一些通用性的功能逻辑进行封装方便使用，AngularJS允许将自定义服务。

1、factory方法



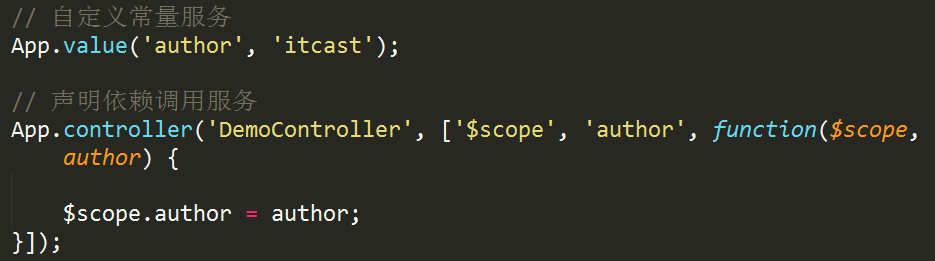
见代码示例8-06.html

2、service方法



见代码示例8-07.html

3、value方法定义常量

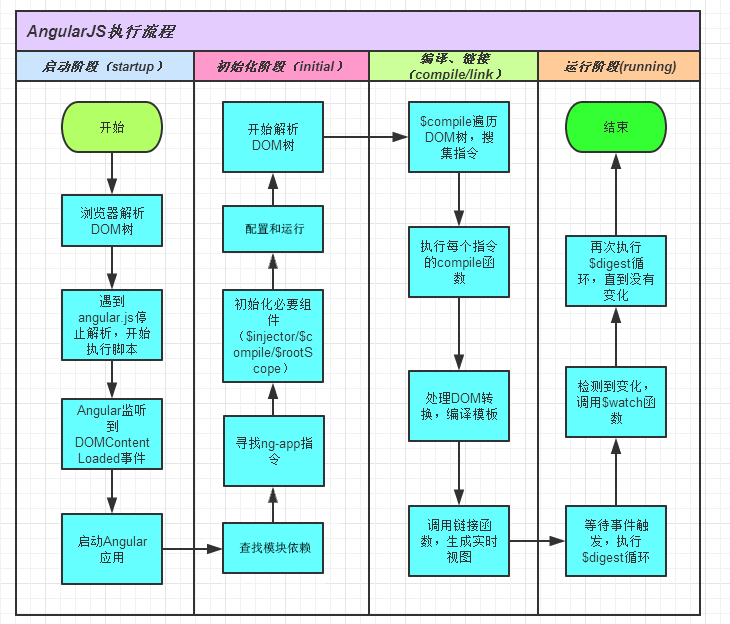


见代码示例8-08.html

在介绍服务时曾提到服务本质就是一个对象或函数，所以自定义服务就是要返回一个对象或函数以供使用。

# 模块加载

AngularJS模块可以在被加载和执行之前对其自身进行配置。我们可以在应用的加载阶段配置不同的逻辑。



性能优化：雅虎十三条

## 配置块

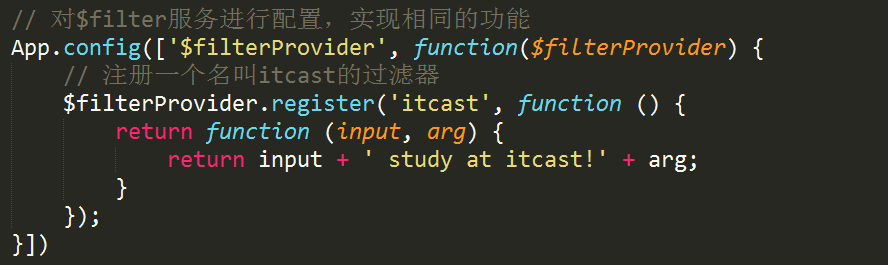
1、通过config方法实现对模块的配置，AngularJS中的服务大部分都对应一个“provider”，用来执行与对应服务相同的功能或对其进行配置。

比如$log、$http、$location都是内置服务，相对应的“provider”分别是$logProvider、$httpProvider、$locationPorvider。

下图以$log为例进行演示，修改了配置



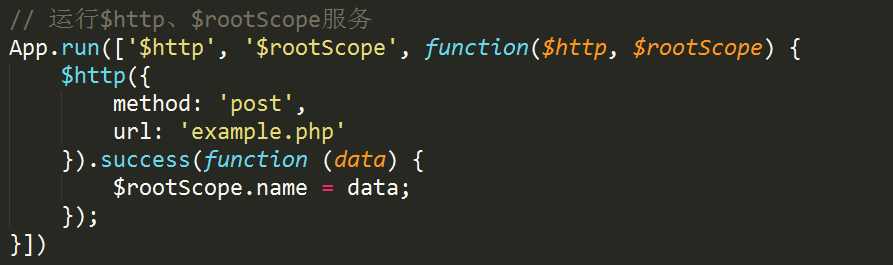
下图以$filter为例进行演示，实现相同功能



见代码示例9-01.html

## 运行块

服务也是模块形式存在的对且对外提供特定功能，前面学习中都是将服务做为依赖注入进去的，然后再进行调用，除了这种方式外我们也可以直接运行相应的服务模块，AngularJS提供了run方法来实现。



不但如此，run方法还是最先执行的，利用这个特点我们可以将一些需要优先执行的功能通过run方法来运行，比如验证用户是否登录，未登录则不允许进行任何其它操作。

见代码示例9-02.html

注：此知识点意在了解AngularJS的加载机制。

# 路由

一个应用是由若个视图组合而成的，根据不同的业务逻辑展示给用户不同的视图，路由则是实现这一功能的关键。

## SPA

SPA（Single Page Application）指的是通单一页面展示所有功能，通过Ajax动态获取数据然后进行实时渲染，结合CSS3动画模仿原生App交互，然后再进行打包（使用工具把Web应用包一个壳，这个壳本质上是浏览器）变成一个“原生”应用。

在PC端也有广泛的应用，通常情况下使用Ajax异步请求数据，然后实现内容局部刷新，局部刷新的本质是动态生成DOM，新生成的DOM元素并没有真实存在于文档中，所以当再次刷新页面时新添加的DOM元素会“丢失”，通过单页面应可以很好的解决这个问题。

## 路由

在后端开发中通过URL地址可以实现页面（视图）的切换，但是AngularJS是一个纯前端MVC框架，在开发单页面应用时，所有功能都在同一页面完成，所以无需切换URL地址（即不允许产生跳转），但Web应用中又经常通过链接（a标签）来更新页面（视图），当点击链接时还要阻止其向服务器发起请求，通过锚点（页内跳转）可以实现这一点。

实现单页面应用需要具备：

a、只有一页面

b、链接使用锚点

见代码实例10-01.html

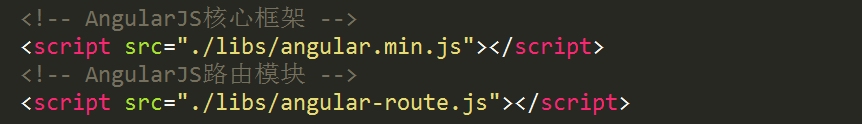
通过上面的例子发现在单一页面中可以能过hashchange事件监听到锚点的变化，进而可以实现为不同的锚点准不同的视图，单页面应用就是基于这一原理实现的。不能使用onclick事件去做的原因是：重复点击会重复发送请求。

AngularJS对这一实现原理进行了封装，将锚点的变化封装成路由（Route）,这是与后端路由的根本区别。

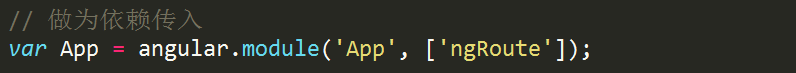
在1.2版前路由功能是包含在AngularJS核心代码当中，之后的版本将路由功能独立成一个模块，[下载angular-route.js](https://code.angularjs.org/) <https://code.angularjs.org/>

### 使用

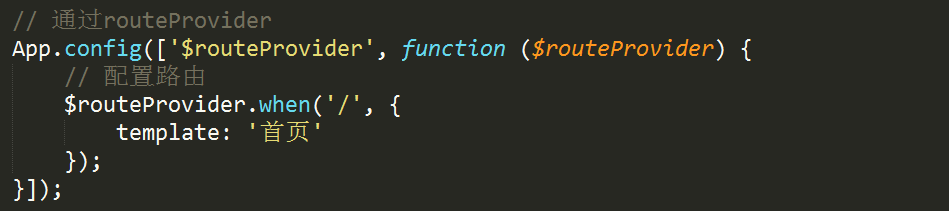
1、引入angular-route.js



2、实例化模块（App）时，当成依赖传进去（模块名称叫ngRoute）。

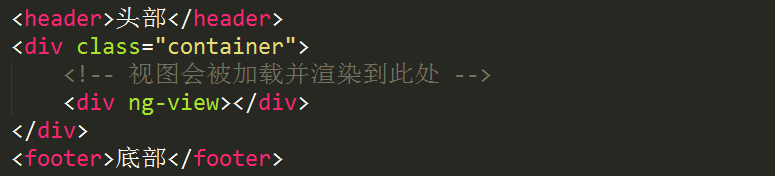


3、配置路由模块



4、布局模板

通过ng-view指令布局模板，路由匹配的视图会被加载渲染到些区域。



### 路由参数

1、提供两个方法匹配路由，分别是when和otherwise，when方法需要两个参数，otherwise方法做为when方法的补充只需要一个参数，其中when方法可以被多次调用。

2、第1个参数是一个字符串，代表当前URL中的hash值。

3、第2个参数是一个对象，配置当前路由的参数，如视图、控制器等。

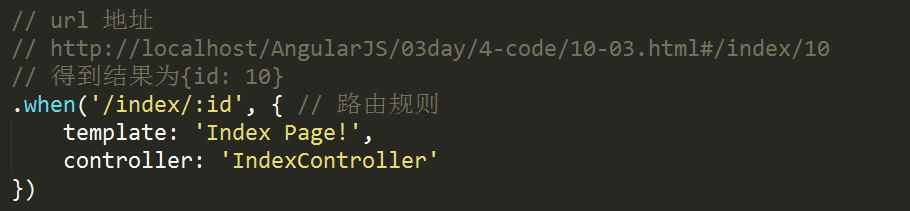
a、template 字符串形式的视图模板

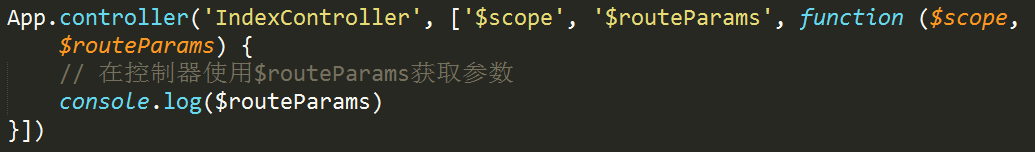
b、templateUrl 引入外部视图模板

c、controller 视图模板所属的控制器

d、redirectTo跳转到其它路由

4、获取参数，在控制中注入$routeParams可以获取传递的参数





# 其它

## jQuery

AngularJS内建了一个简版的jQuery Lite，通过angular.element不能选择元素，但可以将一个DOM元素转成jQuery对象，该对象具备jQuery对象的部分功能:https://docs.angularjs.org/api/ng/function/angular.element

|  |
| --- |
| **var *box*** = **document**.querySelector(**"box"**); *//转成jQuery对象* ***box*** = **angular**.**element**(***box***); ***box***.css(**'color'**,**'red'**); |

如果引提前引入了jQuery则angular.element则完全等于jQuery。

## bower

基于NodeJS的一个静态资源管理工具，由twitter公司开发维护，解决大型网站中静态资源的依赖问题。(就是用来下载插件的！！)

1、依赖NodeJS环境和git工具。

2、npm install -g bower --registry=https://registry.npm.taobao.org安装bower

3、bower search 查找资源信息

4、bower install 安装（下载）资源，通过#号可以指定版本号

5、bower info 查看资源信息

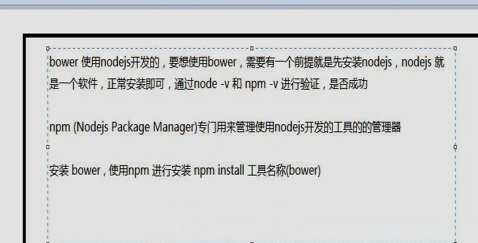
6、bower uninstall 卸载（删除）资源

7、bower init初始化，用来记录资源信息及依赖。

（bower init在windows下只能使用cmd来操作，且必须使用下列命令：

bower init --config.interactive

）



npm install -g bower --registry=https://registry.npm.taobao.org

-g表示 global 全局安装

bower可以指定所下载插件的版本：

bower info 查看资源信息（包括版本）

bower install jquery#1.7.2

bower可以自动帮我们解决依赖，比如我们下载一个插件A，但是这个插件会依赖于另外一款插件B，bower会将这2个插件全部下载。这时候我们就无法卸载这个插件B，

我们要卸载B，必须卸载A，然后bower会自动将依赖插件B删除。