# 深入理解ANGULARUI路由\_UI-ROUTER

最近在用 ionic写个webapp 看到几个demo中路由有好几种，搞的有点晕，查下资料研究下，做个笔记，其中大部分为摘抄别人的，做个说明免得被人吐槽。

Angularjs ui-router - 组件：

**$state / $stateProvider**：管理状态定义、当前状态和状态转换。包含触发状态转换的事件和回调函数，异步解决目标状态的任何依赖项，更新$location到当前状态。由于状态包含关联的 url，通过$urlRouterProvider生成一个路由规则来执行转换的状态。

**ui-view**指示器：渲染状态中定义的视图，是状态中定义的视图的一个占位符。

**$urlRouter / $urlRouterProvider**：管理了一套路由规则列表来处理当$location发生变化时如何跳转。最低级的方式是，规则可以是任意函数，来检查$location，并在处理完成时候返回true。支持正则表达式规则和通过$urlMatcherFactory编译的UrlMatcher对象的 url 占位符规则。

**$urlMatcherFactory**：将 url和占位符编译为UrlMatcher对象。除了$routeProvider支持的占位符语法之外，它还支持扩展语法，允许一个正则表达式指定占位符，并且能够提取命名参数和查询url的一部分。

**$templateFactory** - 通过$http / $templateCache来加载模板，供状态配置中使用。

AngularJS不需要任何第三方库，利用自身集成的各个模块便可开发出功能齐全的web应用，不过活跃的AngularJS社区也开发了很多能够最大限度强化web应用的编程库。本文带读者了解专业开发使用的模块AngularUI中的ui-路由(ui-router)。AngularUI库已经被分成了几个模块，用户可以只选择载入自己感兴趣的模块，而不用载入整个库。

## UI-Router

UI-Router被认为是AngularUI为开发者提供的最实用的一个模块，它是一个让开发者能够根据URL状态或者说是'机器状态'来组织和控制界面UI的渲染，而不是仅仅只改变路由（传统AngularJS应用实用的方式）。该模块为开发者提供了很多最视图（view）额外的控制。开发者可以创建嵌套分层的视图、在同一个页面使用多个视图、让多个视图控制某个视图等更多的功能。即使是非常复杂的web应用，UI-Router也可以极佳地驾驭。

### 安装

安装方式可以选择下载发行版本或者使用Bower（前端包管理器）：

$ bower install angular-ui-router --save

同时也需要将源文件包含到页面中：

<script type="text/javascript" src="app/bower\_components/angular-ui-router/release/angular-ui-router.js"></script>

接下来，将UI-Router作为web应用的依赖，注入到主程序：

angular.module('myApp', ['ui.router']);

与集成的ngRoute服务不同的是，UI-Router可以将视图嵌套，因为它基于的是操作状态而仅非URL。与传统做法使用ng-view不同的是，在ngRoute里需要使用ui-view服务。当在ui-router中处理路由和状态时，开发者的重心是当前的状态是什么以及在哪一个页面里。

<div ng-controller="DemoController">

<div ui-view></div>

</div>

和ngRoute一样，为特定状态指定的模板将会放在<div ui-view></div>元素中。在这些模板中也可以包含自己的ui-view,这就是在同一个路由下实现嵌套视图的方法。要定义一个路由，与传统的方法相同：使用.config方式，但使用的不是$routeProvider而是$stateProvider。

.config(function($stateProvider, $urlRouterProvider) {

$stateProvider

.state('start', {

url: '/start',

templateUrl: 'partials/start.html'

})

});

上述代码在设置对象上定义了一个叫start的状态。设置对象stateConfig和路由设置对象的选项是非常相似的。

### 模板，模板路径，模板Provider

开发者可以在每个视图下使用如下方式来设置模板 - template - HTML字符串，或者是返回HTML字符串的函数 - templateUrl - HTML模板的路径，或者是返回HTML模板路径的函数 - templateProvider - 返回HTML字符串的函数 例如：

$stateProvider.state('home', {

template: '<h1>Hello {{ name }}</h1>'

});

### 控制器

和ngRoute相似，开发者可以指定任何已经被注册的控制器，或者在路由里面创建一个作为控制器的函数。但如果没有定义模板，控制器将无效。

### 预载入Resolve

使用预载入功能，开发者可以预先载入一系列依赖或者数据，然后注入到控制器中。在ngRoute中resolve选项可以允许开发者在路由到达前载入数据保证（promises）。在使用这个选项时比使用angular-route有更大的自由度。

预载入选项需要一个对象，这个对象的key即要注入到控制器的依赖，这个对象的value为需要被载入的factory服务。

如果传入的时字符串，angular-route会试图匹配已经注册的服务。如果传入的是函数，该函数将会被注入，并且该函数返回的值便是控制器的依赖之一。如果该函数返回一个数据保证（promise），这个数据保证将在控制器被实例化前被预先载入并且数据会被注入到控制器中。

$stateProvider.state('home', {

resolve: {

//这个函数的值会被直接返回，因为它不是数据保证

person: function() {

return {

name: "Ari",

email: "ari@fullstack.io"

}

},

//这个函数为数据保证, 因此它将在控制器被实例化之前载入。

currentDetails: function($http) {

return $http({

method: 'JSONP',

url: '/current\_details'

});

},

//前一个数据保证也可作为依赖注入到其他数据保证中！（这个非常实用）

facebookId: function($http, currentDetails) {

$http({

method: 'GET',

url: 'http://facebook.com/api/current\_user',

params: {

email: currentDetails.data.emails[0]

}

})

}

},

//定义控制器

controller: function($scope, person,

currentDetails, facebookId) {

$scope.person = person;

}

})

### URL

url选项将会为该应用的状态指定一个URL基于用户浏览该应用所在的状态。这样当在浏览该应用的时候便能实现深度链接的效果。 该选项与ngRoute的URL相似，但可以被视为对ngRoute主要的升级，在接下来的文章里你便会认可这一点。开发者可以这样指定一个基本的路由。

$stateProvider

.state('inbox', {

url: '/inbox',

template: '<h1>Welcome to your inbox</h1>'

});

当用户浏览到/inbox时，该应用将状态改为inbox同时向主ui-view元素中插入模板中的内容('Welcome to your inbox')。URL参数有多个选项，因此它非常强大。开发者可以像设置ngRoute一样设置最基本的参数：

$stateProvider

.state('inbox', {

url: '/inbox/:inboxId',

template: '<h1>Welcome to your inbox</h1>',

controller: function($scope, $stateParams) {

$scope.inboxId = $stateParams.inboxId;

}

});

现在将:inboxId最为URL的第二个部分，例如：访问/inbox/1,那么$stateParams.inboxId就为1（$stateParams为{inboxId:1}）。同时也可使用不同的语法：

url: '/inbox/{inboxId}'

路径必须匹配URL，与ngRoute不同的是，当用户访问到/inbox/时，上面的的路径会被激活，然而当访问到/inbox时不会被激活。路径同时也使开发者可以使用正则表达式来匹配，例如：

// 限定id为6位16进制数字

url: '/inbox/{inboxId:[0-9a-fA-F]{6}}',

// 或者

// 匹配任何在 `/inbox`后面的url（慎用）并匹配值到indexId

url: '/inbox/{inboxId:.\*}'

注意，在路由中目前还无法使用路由组，路由数据预载入器无法预载入。

在路径里也可以指定查询参数：

// /inbox?sort=ascending 将会被匹配

url: '/inbox?sort'

### 嵌套路由

使用url参数可以实现嵌套的路由，有了嵌套路由便可在同一个模板同一个路由实现多层次的ui-view，例如在/inbox中嵌入更多路由：

$stateProvider

.state('inbox', {

url: '/inbox/:inboxId',

template: '<div><h1>Welcome to your inbox</h1>\

<a ui-sref="inbox.priority">Show priority</a>\

<div ui-view></div>\

</div>',

controller: function($scope, $stateParams) {

$scope.inboxId = $stateParams.inboxId;

}

})

.state('inbox.priority', {

url: '/priority',

template: '<h2>Your priority inbox</h2>'

});

第一个路由是传统的，注意第二个，它是/inbox下的一个子路由：state( . )语法指定了它使子路由。/inbox/1将匹配第一个路由，而/index/1/priority会匹配第二个路由。使用这种语法，在父视图中的ui-view元素将会由第二个路由控制。

### Params 路由参数

params选项是一个包含路径中的参数和正则表达式匹配结果的数组。该选项不能和url选项混用！当某状态被激活时，应用将这个数组赋值给$stateParams服务。

### Views 视图

开发者可以在一个状态中设置多个有名称的视图。该功能在ui-router中很强大，开发者可以在同一个模板中改变和切换不同的视图。

<如果设置了视图选项，则该状态的‘template’，‘templateUrl’及‘templateProvider’将被忽略。如果想在路由里包含父级模板，就需要创建一个包含模板的抽象模板。

例如有这样的视图：

<div>

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

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

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

接下来就可以创建将被分别被插入到上述ui-view的有命名的视图了，每个子视图可以包含自己的模板、控制器和预载入数据。

$stateProvider

.state('inbox', {

views: {

'filters': {

template: '<h4>Filter inbox</h4>',

controller: function($scope) {}

},

'mailbox': {

templateUrl: 'partials/mailbox.html'

},

'priority': {

template: '<h4>Priority inbox</h4>',

resolve: {

facebook: function() {

return FB.messages();

}

}

}

}

});

### abstract 抽象模板

抽象模板不能被激活，但是它的子模板可以被激活。抽象模板可以提供一个包括了多个有名的视图的模板，或者它可以传递作用域变量$scope给子模板。使用它可以在同一个url下传递自定义数据或者预载入的依赖。除了需要添加abstract属性外，其他设置和设定一个常规状态是相同的：

$stateProvider

.state('admin', {

abstract: true,

url: '/admin',

template: '<div ui-view></div>'

})

.state('admin.index', {

url: '/index',

template: '<h3>Admin index</h3>'

})

.state('admin.users', {

url: '/users',

template: '<ul>...</ul>'

});

### onEnter,onExit 回调函数

当应用进入或者离开当前状态的视图时会调用这两个函数。这两个函数可以访问预载入的数据。这两个回调函数使开发者可以根据状态改变来采取某些动作，例如在用户要离开时可以弹出对话框‘你确定吗？’以及防止意外操作等。

### Data 数据

自定义数据也可以被附加到状态控制对象state configObject.该数据和预载入数据resolve属性相似，但是该数据不会被注入到控制器中，promise也不会被预载入，它的用途是从父状态传递数据到子状态。

# 事件

和ngRoute相同的是，angular-route服务会在不同的状态生命周期lifecycle里启动某些事件events。监听$scope对象便可以捕获这些事件然后采取不同的响应或者操作。如下的事件将会在$rootScope上触发，因此在任何$scope对象上都可以监听到这些事件。

## 状态改变事件

$scope.$on('$stateChangeStart', function(evt, toState, toParams, fromState, fromParams), {

// 如果需要阻止事件的完成

evt.preventDefault();});

可以触发的事件包括：

### stateChangeStart

当状态改变开始的时候被触发

### $stateChangeSuccess

当状态改变成功后被触发

### $stateChangeError

当状态改变遇到错误时被触发，错误通常是目标无法载入，需要预载入的数据无法被载入等。

## 视图载入事件

视图载入阶段ui-router也提供了一些事件

### $viewContentLoading

当视图正在被载入且在DOM被渲染之前触发。

$scope.$on('$viewContentLoading', function(event, viewConfig){

// 获取任何视图设置的参数，以及一个特殊的属性：viewConfig.targetView});

### $viewContentLoaded

当视图被载入且DOM已经渲染完成后被触发。

## $stateParams 状态参数

在上面提及使用$stateparams来提取在url中的不同参数。该服务的作用是处理url的不同部分。例如，当上述的inbox状态是这样时：

url: '/inbox/:inboxId/messages/{sorted}?from&to'

//当用户访问者链接时：'/inbox/123/messages/ascending?from=10&to=20'

$stateParams对象的值为：

{inboxId: '123', sorted: 'ascending', from: 10, to: 20}

### $urlRouterProvider

和ngRoute一样，开发者可以在该对象上设定特定的URL被激活时做什么的规则。由于设定好的状态在特定的url被访问是会自动激活，所以$urlRouterProvider没有必要用来管理激活和载入状态。但当需要管理哪些被发生在当前状态之外的作用域scope时它会非常有用，例如在重定向或者安全验证的时候。在模块的设置函数里便可使用$urlRouterProvider。

#### when()

该函数需要两个参数：1.当前的路径，2.需要重定向到的路径（或者是需要在路径被访问是运行的函数）。设置重定向前需要为$urlRouterProvider设置when函数来接受一个字符串。例如，当希望重定向一个空的路由到/inbox：

.config(function($urlRouterProvider) {

$urlRouterProvider.when('', '/inbox');});

如果传递的是函数，在路径被匹配时该函数会被执行，处理器返回如下3个值中的一个： - falsy，该回应告诉$urlRouter没有匹配到当前url规则，应该尝试匹配新的路径，这样能保证用户访问了正常的路径。 - 字符串，$urlRouter将该字符串当做重定向的路径。 - TRUE 或者 undefined，该回应告诉$urlRouter，url已被处理

#### otherwise()

和ngRoute的otherwise()函数相似，在用户提交的路径没有被定义的时候它将重定向到指定的页面。这是个创建’默认‘路径的好方法。 otherwise()只接受一个参数，要么函数要么字符串，字符串必须为合法的url路由地址，函数则会在没有任何路径被匹配的时候被运行。

.config(function($urlRouterProvider) {

$urlRouterProvider.otherwise('/');

// or

$urlRouterProvider.otherwise(

function($injector, $location) {

$location.path('/');

});});

#### rule()

如果想越过任何URL的匹配或者在其他路由前做路由修改，则可以使用rule()函数。在使用它的时候必须返回一个合法的代表路径的字符串。

app.config(function($urlRouterProvider){

$urlRouterProvider.rule(

function($injector, $location) {

return '/index';

});})

# 总结

本文涵盖了ui-router深度及几乎全部的功能。希望你也发现这个库的强大和实用，并在下一个项目中实用这些强大的功能。

--- 转自 http://www.cnblogs.com/hughtxp/p/3965916.html ---