ngRoute 实用文章：

<http://www.tuicool.com/articles/jqMveaB>

在使用 ngRoute 之前， 必须要加载 angular-route.js 或者压缩版 angular-route.min.js

<script src="https://code.angularjs.org/1.6.0/angular.js"></script>

<script src="https://code.angularjs.org/1.6.0/angular-route.js"></script>

定义模块时，必须要注入 ngRoute 服务

var app = angular.module("myApp", ["ngRoute"]);

**使用说明**

ngRoute 模块中包含以下内容，

| **名称** | **所属** | **作用** |
| --- | --- | --- |
| ngView | DIRECTIVE | 提供不同路由模板插入的视图层 |
| $routeProvider | PROVIDER | 提供路由配置 |
| $route | SERVICE | 用于构建各个路由的url、view、controller这三者的关系 |
| $routeParams | SERVICE | 解析返回路由中带有的参数 |
|  |  |  |

1. 注册配置路由： config( function($routeProvider) { } );
2. $locationProvider.hashPrefix('');

## $routeProvider - Methods

* **when(path, route);**

**route = {**

**controller: string | function,**

**controllerAs: string,**

**template: html string | function – 函数返回值应该是html string,**

**templateUrl: html file | function – 函数返回值应该是html file,**

**resolve: {Object.<string, function>=}**

**resolveAs: string,**

**redirectTo: string | function,**

**resolveRedirectTo: function,**

**reloadOnSearch: boolean – default true,**

**caseInsensitiveMatch: Boolean – default false**

**}**

* **otherwise(params);**

**params: string|object string-path object–{redirectTo:string|function}**

**path:** 我们也可以用\*进行模糊匹配，如：/show\*/:name将匹配/showInfo/tom。

$location service – 此服务对于路由很重要，所以先介绍此对象

### $location服务的配置

要配置$location服务，检索$locationProvider并把参数设置成以下这样：

* html5Mode(模式): {boolean}  
  true - 参阅HTML5模式  
  false - 参阅Hashbang模式  
  default: false
* hashPrefix(前缀): {string}  
  Hashbang URLs的前缀 (在Hashbang模式中或者低级浏览器中使用)  
  default: '!'

var app = angular.module("myApp", ["ngRoute", "MYRoute"]);

app.config( function($provide,$routeProvider, $locationProvider){

$locationProvider.hashPrefix('');

}

只读属性：

$location服务为URL只读部分(absUrl, protocol, host, port)提供只读方法，为可读写部分（url, path, search, hash）提供读写方法：

$scope.myss = $sce.trustAsHtml(

'<br>ReadOnly: <br>'

+ "absUrl: " + $location.absUrl() + '<br>'

+ 'host: ' + $location.host("www.sohu.com") + '<br>'

+ 'protocol: ' + $location.protocol("https") + '<br>'

+ 'port: ' + $location.port(443) + '<br>'

+ '<br>Read and Write<br>'

+ 'url: ' + $location.url() + '<br>'

+ 'path: ' + $location.path() + '<br>'

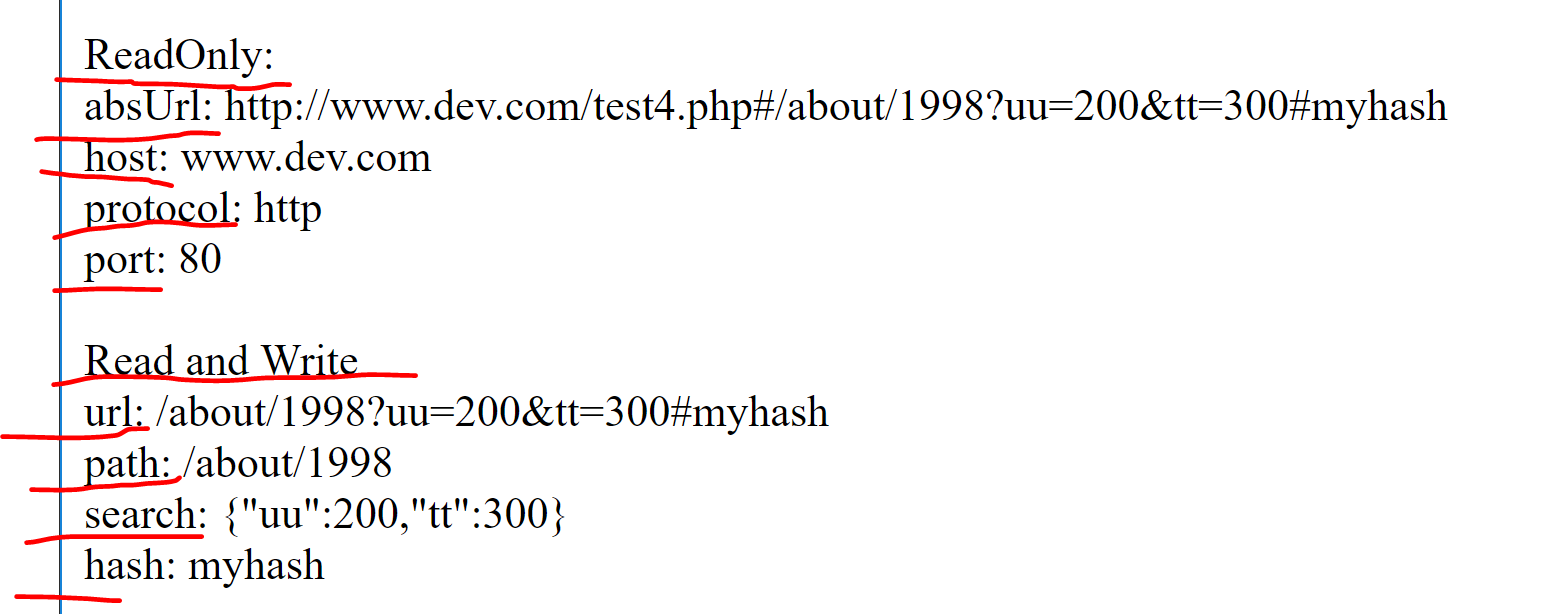
+ 'search: ' + angular.**toJson**($location.search()) + '<br>'

+ 'hash: ' + $location.hash() + '<br>'

);

<span ng-bind-html='myss'></span>

如果浏览器的URL : http://www.dev.com/test4.php#/about/20008?uu=200&tt=300#myhash



看看 $location.url()

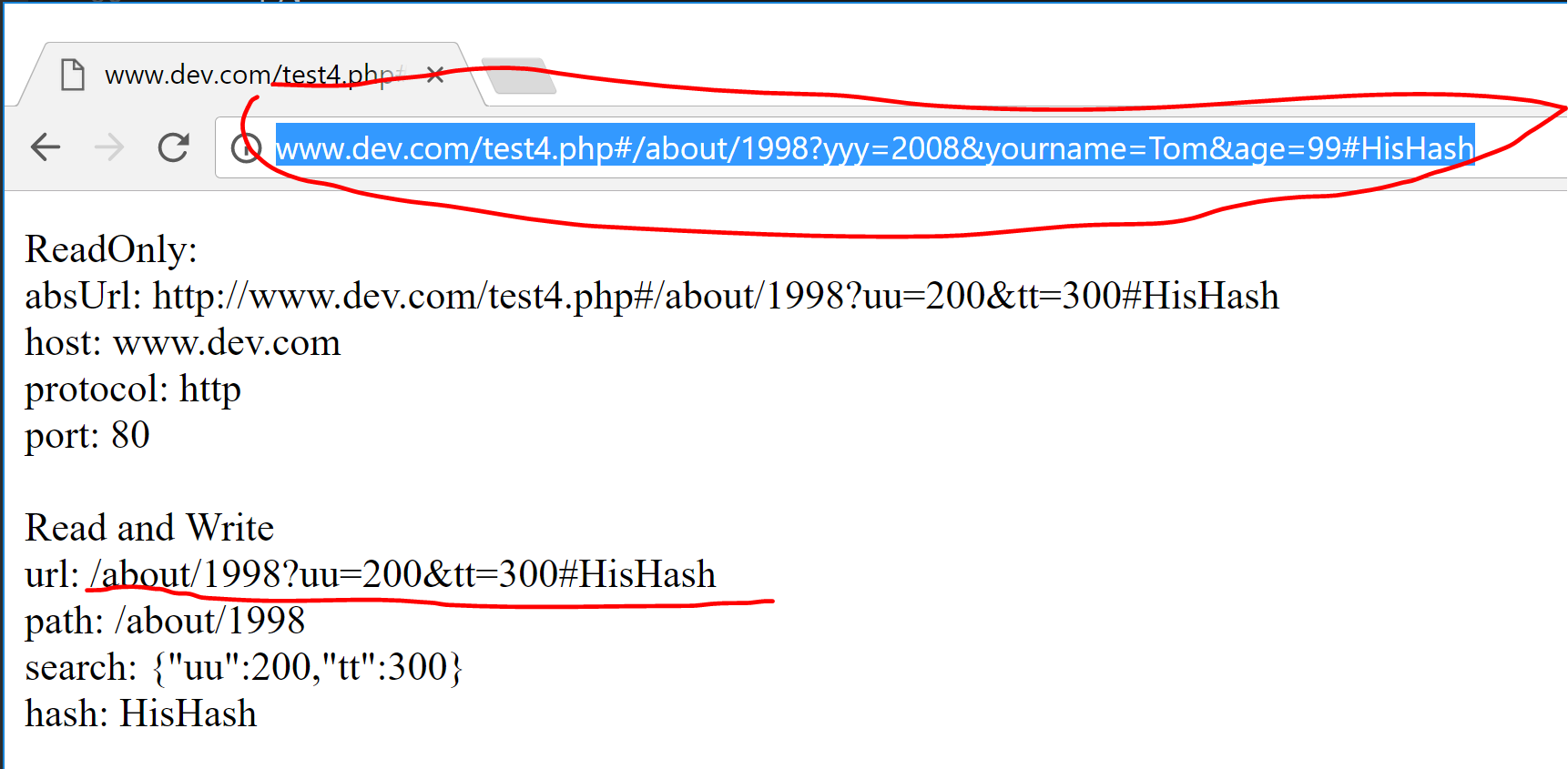
虽然我们在浏览器地址栏里输入：http://www.dev.com/test4.php#/about/20008?uu=200&tt=300#myhash

最后会跳转到 url 指定的地址：

app.controller("ct22", function($scope, $location, $sce, $routeParams) {

$location.url("/about/1998?yyy=2008&yourname=Tom&age=99#HisHash")

}



$location 读写方法对重定向的影响：

$routeParams.uu = 'newChart';

$location.search("zzz", 201800);

$location.url("/about/1998?yyy=2008&yourname=Tom&age=99#HisHash")

* **生成的 search( {yyy: , yourname: , age: } ) 会保留；**

$location.hash("OKKK");

$location.path("/about/" + ($routeParams.uu + 100000) );

http://www.dev.com/test4.php#/about/newChart100000?yyy=2008&yourname=Tom&age=99#OKKK

$location.search(“zzz” , 201800);

会被 $location.url( ?yyy=2008&yourname=Tom&age=99 重新定义给覆盖掉

$routeParams.uu = 'newChart';

$location.search("zzz", 201800);

$location.hash("OKKK");

$location.url("/about/1998");

$location.path("/about/" + ($routeParams.uu + 100000) );

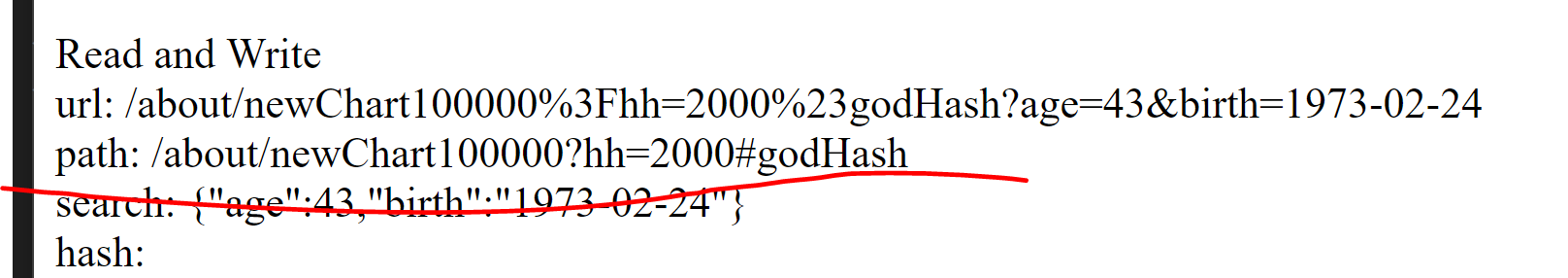
返回： http://www.dev.com/test4.php#/about/newChart100000

$location.url(“/about/1998”) 具有绝对的覆盖权，即没有search, hash

$location.path("/about/" + ($routeParams.uu + 100000) + "?hh=2000#godHash" );

Path : 只能替换路由路径， 虽然有 ?hh=2000#godHash， 但是会urlencode 当成字符处理, 所以不要在 path 里带上search 和 hash 参数。

http://www.dev.com/test4.php#/about/newChart100000%3Fhh=2000%23godHash



容易混乱的影子数据：

app.controller("ct22", function($scope, $location, $route, $sce, $routeParams) {

$routeParams.uu = 9999;

$location.search("age", 43);

$location.search("birth", "1973-02-24");

$location.hash("OKKK");

$scope.myss = $sce.trustAsHtml(

'<br>ReadOnly: <br>'

+ "absUrl: " + $location.absUrl() + '<br>'

+ 'host: ' + $location.host("www.sohu.com") + '<br>'

+ 'protocol: ' + $location.protocol("https") + '<br>'

+ 'port: ' + $location.port(443) + '<br>'

+ '<br>Read and Write<br>'

+ 'routeParams.uu: ' + $routeParams.uu + '<br>'

+ 'url: ' + $location.url() + '<br>'

+ 'path: ' + $location.path() + '<br>'

+ 'search: ' + angular.toJson($location.search()) + '<br>'

+ 'hash: ' + $location.hash() + '<br>'

);

$scope.getSS = function($sce) {

return $sce.trustAsHtml($scope.myss);

}

$routeParams.uu = 'newChart';

$location.search("zzz", 201800);

$location.hash("OKKK");

$location.url("/about/1998");

明明 url (“/about/1998”) 没有search 参数， 为何又有呢

原因是， 当 url 重定向回来同一个路由路径时， controller 一开始就开始改写这些值：

$routeParams.uu = 9999;

$location.search("age", 43);

$location.search("birth", "1973-02-24");

$location.hash("OKKK");

所以造成了 url , path 与实际的不相符， 理解这点很重要， 否则容易代码逻辑错误。

但是为何 hash 没有得到修改？？ 这个目前不清楚？

$location.search() 的使用 – 返回的是JSON 键值对

如何访问 search 的参数：

$location.search().xxxx 或者 $location.search()[“xxxx”]

$location.search("age", 22);

$location.search("zzz", 201800);

$location.search("age", 99);

console.log( $location.search().age + " : " + $location.search()["age"]);

如果是同名则修改，如果没有则新增：

$location.search("age", 22);

$location.search("zzz", 201800);

$location.search("age", 99); **- 修改**

$location.search("age", 22);

$location.search("zzz", 201800);

$location.search("age", 99);

$location.search({ hhh: 11, JJ: 22 }); - **JSON 则是删除以后再添加， 也就是之前的全删除了**

[http://www.dev.com/test4.php#/about/1998?hhh=11&JJ=22#OKKK](http://www.dev.com/test4.php#/about/1998?hhh=11&JJ=22)

删除某个参数：

$location.search("age", undefined); ok

$location.search("age", null); ok

$location.search("age", false); - **不是删除** 是 age=false&zzz=201800

$location.search("age", “”); - **不是删除** 是 age=&zzz=201800

删除所有：

$location.search({});

$location.search("")

添加search 参数，使用链条方式

$location.search("age", 22).search("zzz", 201800).search("age", 99)

.search({"XXX":"World"}).search("PPP", "Peter"); - **从{“xxx”:”World”}处开始覆盖并重定义新的**

http://www.dev.com/test4.php#/about/1998?XXX=World&PPP=Peter#OKKK

$route.updateParams(); - 修改路由参数会直接触发新的 url 加载，从而 controller也被重新初始化。

$scope.changeURL = function(){

$route.updateParams({uu: 1668});

}

其他的设置，会无效， 如 $location.search() 并不起作用。

$location.search({}).search("HHHH", 8999);

$route.updateParams({uu: 1668});

$routeParams - 路由参数： $routeParams 包含了 $location.search() 的内容

.when( "/about/:uu",

{

template: "<span> About Us {{ myid }} <br>Search: <br> <span ng-bind-html='myss'></span></span>",

controller: "ct22"

}

)

.when( "/about/:uu/:tt?",

{

template: "<span> Cayon <br>Search: <br> <span ng-bind-html='myss'></span></span>",

controller: "ct22"

}

)

.when( "/about/:uu/:tt?/:ss?",

{

template: "<span> Canon <br>Search: <br> <span ng-bind-html='myss'></span></span>",

controller: "ct22"

}

)

路由路径: 不算 $location.search() 部分 ?key=value&key1=value1

可以选参数： /:p1/:p2?/:p3? p1 是必须的， p2 , p3 是可选参数

<li><a href="#/about/20008?id=200&name=Willtom#EBUY88">About</a></li>

Here is View:

Cayon   
Search:   
  
ReadOnly:   
absUrl: http://www.dev.com/test4.php#/about/2008?id=200&name=Willtom#EBUY88  
host: www.dev.com  
protocol: http  
port: 80  
  
Read and Write  
routeParams: {"id":"200","name":"Willtom","uu":"2008"}  
url: /about/2008?id=200&name=Willtom#EBUY88  
path: /about/2008  
search: {"id":"200","name":"Willtom"}  
hash: EBUY88

$scope.changeURL = function(){

$location.search({}).search("HHHH", 8999);

$route.updateParams({uu: 1668});

}

$scope.refreshURL = function() {

$route.reload();

}

<div ng-controller="ct22">

<button ng-click="changeURL()">SQL Call</button>

<button ng-click="refreshURL()">Refresh Call</button>

</div>

resolve – 同步返回数据： 可以定义多个 resolve 提供商 respData; respData1 ….

.when( "/about/:uu",

{

template: "<span> Resole Return: {{ rrr }} </span>",

controller: "ct22",

resolve: {

respData: function($q, $timeout, $http) {

return { mon: 4000, tue: 5000 };

},

respData1: function($route) {

return $route.current.params;

}

}

}

)

app.controller("ct22", function($scope, respData, respData1) {

$scope.rrr = angular.toJson(respData) + " : " + angular.toJson(respData1);

}

http://www.dev.com/test4.php#/about/2008?id=200&name=Willtom#EBUY88

Here is View:

Resole Return: {"mon":4000,"tue":5000} : {"id":"200","name":"Willtom","uu":"2008"}

在 resolve 里面访问 $route.current.params 等同于 $routeParams, 同样也允许访问 $location

respData1: function($route, $routeParams, $location) {

return $routeParams;

}

resolve – 异步访问， 则必须返回 promise , 但是必须要触发成功事件才能成功返回。 否则路由失败, 页面不会跳转。

.when( "/about/:uu",

{

template: "<span> Resole Return: {{ rrr }} </span>",

controller: "ct22",

resolve: {

respData: function($q, $timeout, $route, $http) {

var ddd = $q.defer();

$timeout( function(){

console.log("ddd callback");

ddd.resolve( {good: $route.current.params.uu, message:"Homer"});

}, 1000 );

return ddd.promise;

},

respData1: function($route, $q, $timeout) {

var dd1 = $q.defer();

$timeout( function(){

console.log("dd1 callback");

-- **dd1.reject() 是呼叫失败的回调函数**

dd1.reject({bad: $route.current.name}); }, 1500);

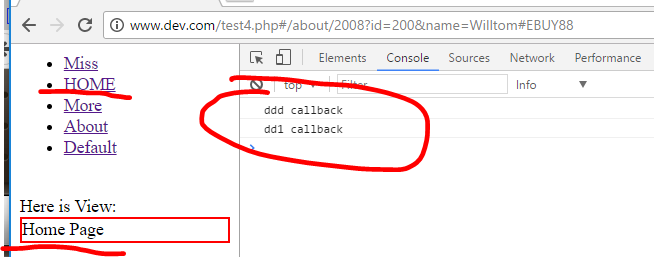
return dd1.promise;

}

}

}

)



所以仍然停留在 Home Page, 并没有跳转， 但是上面的url 已经改变

resolve - with $http, $route

.when( "/about/:uu",

{

template: "<span> Resole Return: {{ rrr }} </span>",

controller: "ct22",

resolve: {

respData: function($q, $route, $http) {

return $http({

method: "GET",

url: "test4\_data.php"

})

.then(

function(rdata){

console.log("http then");

return rdata.data;

},

function(rdata){}

);

},

respData1: function($route, $q, $timeout) {

var dd1 = $q.defer();

$timeout( function(){

console.log("dd1 callback");

dd1.resolve({well: $route.current.params.name}); }, 1500);

return dd1.promise;

}

}

}

)

return $http() - 返回的是 promise;

而resolve: respData 相当于是promise.then( successFunc( httpResp.data), errorFunc() );

$http 回调函数里的 参数是 httpResponseData, 而真正的数据是 httpReponseData.data.

.when( "/about/:uu",

{

template: "<span> Resole Return: {{ rrr }} </span>",

controller: "ct22",

resolve: {

respData: function($q, $route, $http) {

return $http({

method: "GET",

url: "test4\_data.php"

})

.then(

function(rdata){

console.log("http then");

console.log(rdata);

return rdata.data;

},

function(rdata){}

)

.then(

function(rdata) {

console.log("http then11");

rdata.data.data += 7500;

rdata.data.date += " 21:00pm";

return rdata;

},

function(rdata) { }

);

},

respData1: function($route, $q, $timeout) {

var dd1 = $q.defer();

$timeout( function(){ console.log("dd1 callback"); dd1.resolve({well: $route.current.params.name}); }, 1500);

return dd1.promise;

}

}

}

)

<?php

$response = array();

$response["data"]["data"] = 1000;

$response["data"]["date"] = "2017-10-15";

  echo json\_encode($response);

?>

Here is View:

Resole Return: {"data":{"data":8500,"date":"2017-10-15 21:00pm"}} : {"well":"Willtom"}

也可以在最后阶段抛出错误而终止路由页面跳转，也就是说，可以添加更多的业务逻辑

respData: function($q, $route, $http) {

return $http({

method: "GET",

url: "test4\_data.php"

})

.then(

function(rdata){

console.log("http then");

console.log(rdata);

return rdata.data;

},

function(rdata){}

)

.then(

function(rdata) {

console.log("http then11");

rdata.data.data += 7500;

rdata.data.date += " 21:00pm";

throw rdata;

},

function(rdata) { }

);

},

可以使用任何 ajax 技术： 如 $.ajax JQuery Ajax

关键是要使用 $q $q.defer() 和 resolve, reject, notify ,

返回 $q.defer().promise

respData: function($q, $route, $http) {

var ddd = $q.defer();

$.ajax({

data: {},

dataType: "json",

error: function(xhr, tStatus, errorTh ) {

ddd.resolve({error:1});

},

success: function(req, tStatus) {

console.log("Ajax Done");

console.log(req);

ddd.resolve(req);

},

type: "GET",

url: "test4\_data.php"

});

return ddd.promise;

Here is View:

Resole Return: {"data":{"data":1000,"date":"2017-10-15"}} : {"well":"Willtom"}

resolve - with $http, $route

var app = angular.module('CommunityInfo', ['ngRoute', 'chart.js'])

.config(function ($routeProvider, $httpProvider){

$routeProvider

// main page (listing)

.when('/members', {

controller: 'MembersController',

templateUrl: self.location.pathname + '/views/members-list.html'

})

// member of the list (load additional data from request.php)

.when('/member/:memberID', {

controller: 'MembersController',

// ----- new code (start) -----

resolve: {

MemberDataDB: function($http, $route){

return $http.

get(

self.location.origin +

self.location.pathname +

'request.php?rop=info&battleTag=' +

$route.current.params.memberID

)

.then(

function(response){

return response.data;

}

)

}

},

// ----- new code (end) -----

templateUrl: self.location.pathname + '/views/member-data.html'

})

})

.factory('MembersFactory', function($http, MembersListData, ChartsData){

// factory functions & initialization

})

.controller('MembersController',

function($scope, $routeParams, $location, MembersFactory, MemberDataDB){

function init(){

// other initialization

if (memberID){

// copy part of the current data

$scope.memberData = $scope.membersList[memberID];

if (!$scope.memberData){

// member not found - redirect to list

$location.path('/members');

}

// add the new data

angular.merge($scope.memberData, MemberDataDB); // new code

}

}

init();

});

疑难问题：

1. 路由路径 #/a 变成 #!/#%2Fa ， 也就是 / 变成了 !/#%2F

解决方案：$locationProvider 运行此方法 $locationProvider.hashPrefix('');

app.config(function($routeProvider, $locationProvider) {

$locationProvider.hashPrefix('');

$routeProvider.when("/a", {

templateUrl: 'a.html',

controller: 'aController'

})

.when("/b", {

template: '<span>BBB Path: {{bval}}</span>',

controller: 'bController'

})

.otherwise('/');

});

1. Ajax 跨域访问 ASP.NET Web API

解决方法：

1. 安装 Nuget – Microsoft Web Api Cors 程序包
2. 然后在路由注册配置文件添加以下两行代码即可：

var cors = new System.Web.Http.Cors.EnableCorsAttribute("www.my.com", "\*", "\*");

config.EnableCors(cors);

如果不想指定来自特定的网站， 可以用 \* 替代 www.my.com

原文：

<http://stackoverflow.com/questions/11786562/angularjs-asp-net-web-api-cross-domain-issue>

### $location的双向绑定

AngularJS的编译器目前不支持对$location对象的双向绑定（参看问题列表）。如果你需要对$location对象（在input元素上使用ngModel指令）进行双向绑定，你需要指定一个带有两个监听者的额外的模型属性（比如locationPath），这两个监听者各负责一个方向。

<!-- html -->

<input type="text" ng-model="locationPath" />

// js - controller

$scope.$watch('locationPath', function(path) {

$location.path(path);

});

$scope.$watch('$location.path()', function(path) {

scope.locationPath = path;

});

1. $location服务有一个特殊的replace方法可以用来告诉 $lacation服务下一次自动和浏览器同步，上一条浏览记录应该被替换而不是创建一个新的。这在重定向的时候很好用。不这样的话容易使后退按钮失效（点后退时会又触发重定向）。要改变URL而不添加新的历史记录，你可以这样做：

$location.path('/someNewPath');

$location.replace();

// or you can chain these as: $location.path('/someNewPath').replace();

相关文章：

<http://www.tuicool.com/articles/jqMveaB>

<https://www.undefinednull.com/2014/02/17/resolve-in-angularjs-routes-explained-as-story/>

<http://blog.csdn.net/qq_20282263/article/details/55001188>

<http://www.2cto.com/kf/201412/363151.html>

<http://stackoverflow.com/questions/23602181/angularjs-ngareq-argument-fn-is-not-a-function-got-string>

<http://stackoverflow.com/questions/43439265/angularjs-ngroute-does-not-work-for-me>

<https://docs.angularjs.org/api/ngRoute/provider/$routeProvider>

<http://stackoverflow.com/questions/38913371/angularjs-route-provider-changes-into-2f>