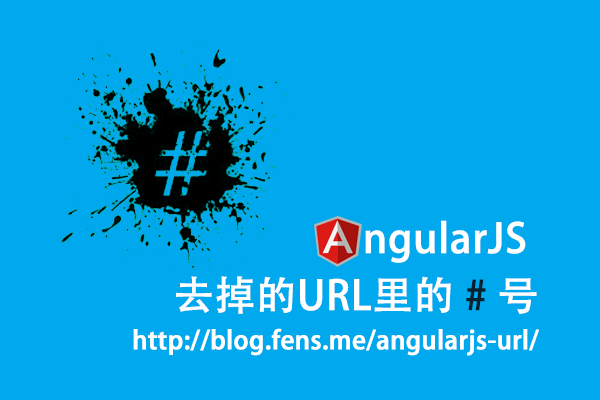
**AngularJS去掉的URL里的#号**

[AngularJS体验式编程系列文章](http://blog.fens.me/series-angular/)，将介绍如何用angularjs构建一个强大的web前端系统。angularjs是由Google团队开发的一款非常优秀web前端框架。在当前如此多的web框架下，angularjs能脱颖而出，从架构设计上就高人一等，双向数据绑定，依赖注入，指令，MVC，模板。Angular.js创新地把后台技术融入前端开发，扫去jQuery一度的光芒。用angularjs就像写后台代码，更规范，更结构化，更可控。

**转载出处：**  
<http://blog.fens.me/angularjs-url/>

[](http://blog.fens.me/wp-content/uploads/2014/08/angularjs-url.png)

**前言**

天天都在用AngularJS，各类文档也都看过好几遍，但总是些编程上的事找不到优雅的解决办法。今天终于把AngularJS的项目访问路径URL里的#号去掉了，这个问题不见得有多难，关键是花多长时间去理解AngularJS框架本身。

**目录**

1. URL的#号问题
2. 找到错误原因
3. 静态网站的解决方案
4. 动态网站的解决方案

**1. URL的#号问题**

使用AngularJS的朋友都应该了解，AngularJS框架定义了自己的前端路由控制器，通过不同URL实现单面(ng-app)对视图(ng-view)的部署刷新，并支持HTML5的历史记录功能，详细介绍可以参考文章：[AngularJS路由和模板](http://blog.fens.me/angularjs-route-template/)。

对于默认的情况，是不启动HTML5模式的，URL中会包括一个#号，用来区别是AngularJS管理的路径还是WebServer管理的路径。

比如：下面的带#号的URL，是AngularJS管理的路径。

* http://onbook.me/
* http://onbook.me/#/
* http://onbook.me/#/book
* http://onbook.me/#/about

这种体验其实是不太友好的，特别是像我这种喜欢简洁设计的人，#号的出现非我自愿的，怎么看怎么难受。AngularJS框架提供了一种HTML5模式的路由，可以直接去掉#号。

通过设置$locationProvider.html5Mode(true)就行了。

book.config(['$routeProvider', '$locationProvider', function ($routeProvider, $locationProvider) {

//..省略代码

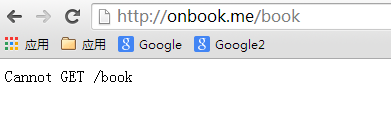
$locationProvider.html5Mode(true);

}]);

支持HTML5的路由URL。

* http://onbook.me/
* http://onbook.me/book
* http://onbook.me/about

接下来的问题就来了，当用这种方式设置了路径以后。如果用户从首页(http://onbook.me/)开始访问，然后跳转到 图书页(http://onbook.me/book)一切正常。但如果用户直接打开 图书页(http://onbook.me/book) ，就会出现404错误。

[](http://blog.fens.me/wp-content/uploads/2014/08/url1.png)

就是这个问题纠结了我好长时间，让我不得不用带#号的URL。

**2. 找到错误原因**

那么，这个问题的原因出在哪里了呢？ 在路径解析上出错了。

让我从头说起，AngularJS是单页应用，一个ng-app对应一个页面，一个URL。AngularJS实现了自己的前端路由，让一个ng-app可以管理多个URL，再对应到多个ng-vew上面。当我们去访问URL(http://onbook.me/book) 的时候，怎么确定这个路径是 WebServer 后台管理的URL还是AngularJS前台管理的URL呢？

分2种情况看：

* 1. 用户如果是先访问 首页(http://onbook.me)，然后再跳转到 页面(http://onbook.me/book)，则这个跳转是由AngularJS前台管理的URL，访问是正常的。
* 2. 用户直接访问 页面(http://onbook.me/book)时，请求是先被提交到了WebServer后台，后台路由没有对应页面(http://onbook.me/book)的路由管理，就会出现404的错误。

如果能把这层想明白，技术上就非常容易解决了。我们让WebServer把属于AngularJS管理的路由URL，都发转到ng-app就可以解决404的问题了，同时，没有#号，还支持HTML5的历史记录查询！！

实现起来分为2种解决方案：

* 1. 静态网站：纯前台网站(JS+HTML+CSS)，通过Nginx提供Web服务。
* 2. 动态网站：前台(JS + HTML + CSS) + 后台Node.js提供Web服务。

**3. 静态网站的解决方案**

静态网站，我们需要修改的地方包括3个文件

* index.html : ng-app的定义文件
* app.js : 对应ng-app的控制文件
* nginx.conf : nginx的网站配置文件

编辑 index.html，增加base标签。

<html lang="zh-CN" ng-app="book">

<head>

<base href="/">

// 省略代码

</head>

编辑app.js，增加 $locationProvider.html5Mode(true);

book.config(['$routeProvider', '$locationProvider', '$sceProvider', 'tplProvider', function ($routeProvider, $locationProvider, $sceProvider, tplProvider) {

$routeProvider

.when('/', {templateUrl: tplProvider.html('welcome'), controller: 'WelcomeCtrl'})

.when('/book', {templateUrl: tplProvider.html('book'), controller: 'BookCtrl'}) //图书

.when('/book-r1', {templateUrl: tplProvider.html('book-r1'), controller: 'BookR1Ctrl'}) //R的极客理想

.when('/video', {templateUrl: tplProvider.html('video'), controller: 'VideoCtrl'}) //视频

.when('/about', {templateUrl: tplProvider.html('about'), controller: 'AboutCtrl'}) //关于作者

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

$locationProvider.html5Mode(true);

}]);

编辑nginx的配置文件，增加try\_files配置。

server {

set $htdocs /www/deploy/mysite/onbook;

listen 80;

server\_name onbook.me;

location / {

root $htdocs;

try\_files $uri $uri/ /index.html =404;

}

}

这样，静态网站就搞定了，没有麻烦的#号了，可以直接访问和任意页面的刷新。

**4. 动态网站的解决方案**

动态网站，我们同样需要修改的地方包括3个文件。

* index.html : ng-app的定义文件
* app.js : 对应ng-app的控制文件
* server.js : Express框架的路由访问控制文件

index.html 和 app.js两个文件修改，同静态网站的解决方案。动态网站，一般不是通过Nginx直接路由，而是通过Web服务器管理路由。假设我们使用的是Node.js的Express的Web框架。

打开Express框架的路由访问控制文件server.js，增加路由配置。

app.use(function (req, res) {

console.log(req.path);

if(req.path.indexOf('/api')>=0){

res.send("server text");

}else{ //angular启动页

res.sendfile('app/index.html');

}

});

设置当 站内路径(req.path) 不包括 /api 时，都转发到 AngularJS的ng-app(index.html)。所以，我们再直接访问地址 (http://onbook.me/book)时，/book 不包括 /api，就会被直接转发到AngularJS进行路由管理。我们就实现了路由的优化！

我已用了AngularJS有8-9个月了，主要功能都用到了，但还不见其全貌。经常会遇到各种问题，不过比起jQuery的各种无解的问题，还是值得的。优秀的框架值得我们的研究和使用，在摸索中前进！