**一、什么是ui-router**

地址：https://www.cnblogs.com/VictorYe/p/7099165.html

* ui-router是AngularUI库最有用的组件之一（AngularUI库由AngularJS社区构建）。它是一个第三方路由框架，允许通过状态机制组织接口，而不是简单的URL路由。

**二、作用**

* 和ngRoute功能一样，可以定义在任意状态内的模板都处在<ui-view>中
* 与ngRoute不同的是，每个模板中可以包含自己的<ui-view>中，也就是我们说的嵌套路由

**三、如何使用**

* 下面，将要向大家介绍下使用ui-router建立一个简单的路由，涵盖了ui-router基本的也是常用的方法。
* 首先，需要定义路由，可以使用.config方法，和ngRoute不同的是，路由是要设置在$routeProvider上， 而是将状态设置在$stateProvider上，

             $stateProvider.state(stateName, stateConfig)

             stateName:：字符串

             stateConfig：object对象，可以设置url、template、controller等属性

* 了解完定义状态的用法，我们可以看一下实例代码：

[复制代码](javascript:void(0);)

1 app.config(function($stateProvider) {

2 $stateProvider

3 .state('C1', {

4 url:'/C1',

5 template: '<h1>进入C1状态</h1>'

6

7 })

8 .state('C2', {

9 url: '/C2',

10 templateUrl: 'Htmls/C2.html'

11

12 })

13 .state('C3', {

14 url: '/C3',

15 templateProvider: function() {

16 return '<h1>进入C3状态</h1>';

17 }

18 })

19 });

[复制代码](javascript:void(0);)

* 上面代码中，我们给状态配置对象分配了三个状态，“C1”、“C2”和"C3"。当应用程序状态为以上三个状态时，url会自动切换到定义的地址，而且也会显示相应的html模板。其中，有三种显示html模板的方式

           template： 一个html内容字符串或一个能返回html字符串的函数， 如状态C1处代码；

           templateUrl： 一个html模板的路径字符串或者是一个能返回URL路径字符串的函数，如状态C2处代码；

           templateProvider：一个能返回URL路径字符串的函数，如状态C3处代码；

* 现在，我们来写一下index.html和app.js文件，实现一下ui-router：

[复制代码](javascript:void(0);)

1 <!DOCTYPE html>

2 <html>

3 <head>

4 <meta charset="utf-8" />

5 <title>index</title>

6 <link href="css/index.css" rel="stylesheet" />

7 <script src="js/lib/angular/angular.min.js"></script>

8 <script src="js/lib/angular-ui-router/release/angular-ui-router.min.js"></script>

9 <script src="app.js"></script>

10 </head>

11 <body ng-app="TrialApp" ng-controller="mainController" style="background-color: lightblue;">

12 <h1>index页面</h1>

13 <div style="width: 80%; float: right;background-color: #C0C0C0;">

14 <h1>路由区域</h1>

15 <ui-view> </ui-view>

16 </div>

17 </body>

18 </html>

[复制代码](javascript:void(0);)

* 建立一个定义TrialApp模块文件app.js：

[复制代码](javascript:void(0);)

1 'use strict';

2 // Define `TrialApp` module

3 var app = angular.module('TrialApp', ['ui.router']);

4 // Define routers

5 app.config(function($stateProvider) {

6 $stateProvider

7 .state('C1', {

8 url:'/C1',

9 template: '<h1>进入C1状态</h1>'

10

11 })

12 .state('C2', {

13 url: '/C2',

14 templateUrl: 'Htmls/C2.html'

15

16 })

17 .state('C3', {

18 url: '/C3',

19 templateProvider: function() {

20 return '<h1>进入C3状态</h1>';

21 }

22 })

23 });

[复制代码](javascript:void(0);)

* 这时，并不能一进入页面就能看见路由的效果，需要我们通过写代码来激活状态。激活state有3种方法：

           1. 调用$state.go('stateName');方法；

           2. 在html文档中<ui-view>区域之外的地方，添加<a ui-sref='stateName'>stateName</a>链接，等页面渲染之后可以通过点击该链接进入所选状态所对应的页面；

           3. 在地址栏中输入state中定义过的url，随后Enter直接访问。

* 现在我们来演示一下上述方法中的前面两种激活状态的方法：

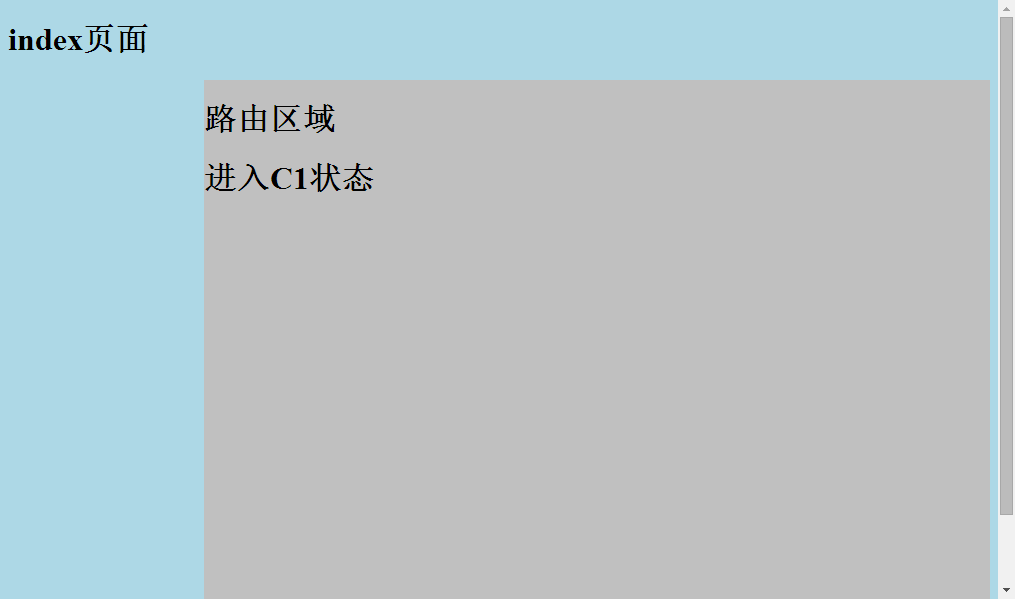
           方法1： 如果需要在页面一加载时就要显示某一状态，需要在app.js后面加一下代码：

1 app.run(function($state) {

2 $state.go('C1');

3 });

上述代码是在加载该模块的时候调用$state.go('C1');，以激活C1状态。运行以后就是以下效果：



       方法2：在html文档中<ui-view>区域之外的地方，添加<a ui-sref='stateName'>stateName</a>链接，代码如下：

1 <a ui-sref='C1'>C1</a><br />

2 <a ui-sref='C2'>C2</a><br />

3 <a ui-sref='C3'>C3</a>

      由于上述效果有点丑，我们调整下html代码：

[复制代码](javascript:void(0);)

1 <!DOCTYPE html>

2 <html>

3 <head>

4 <meta charset="utf-8" />

5 <title>index</title>

6 <link href="css/index.css" rel="stylesheet" />

7 <script src="js/lib/angular/angular.min.js"></script>

8 <script src="js/lib/angular-ui-router/release/angular-ui-router.min.js"></script>

9 <script src="app.js"></script>

10 </head>

11 <body ng-app="TrialApp" ng-controller="mainController" style="background-color: lavender;">

12 <h1>index页面</h1>

13 <div style="width: 20%; float: left;background-color: lightgoldenrodyellow;text-align: center;">

14 <a ui-sref='C1'>C1</a><br />

15 <a ui-sref='C2'>C2</a><br />

16 <a ui-sref='C3'>C3</a>

17 </div>

18 <div style="width: 80%; float: right;background-color: #C0C0C0;">

19 <h1>路由区域</h1>

20 <ui-view> </ui-view>

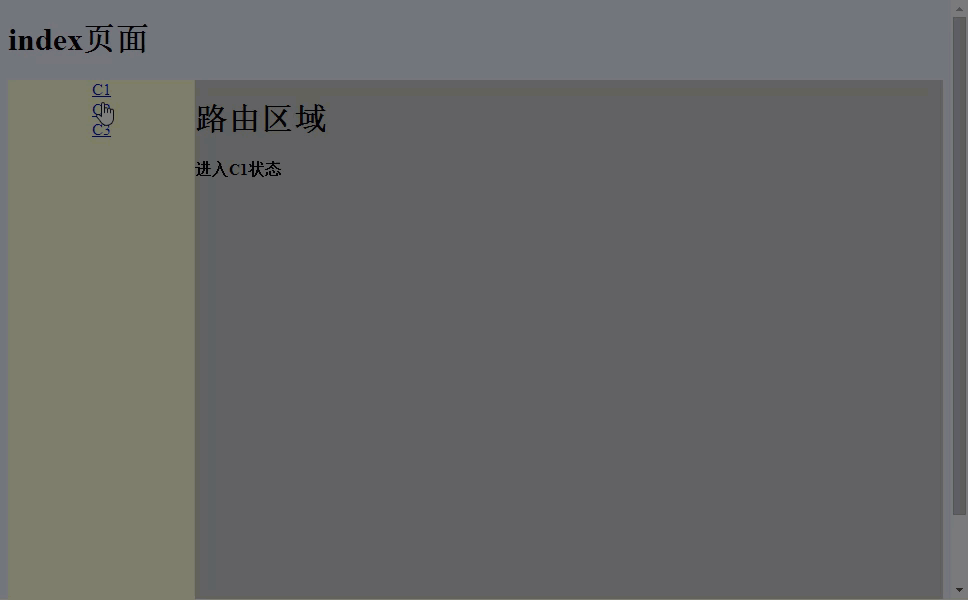
21 </div>

22 </body>

23 </html>

[复制代码](javascript:void(0);)

     以下是改良后的渲染效果，只要点击



* 在本文最后, 演示一下如何在state中加载controller

 controller：

1. 外部文件Controller的名字，注意文件命名时要把controller和前面的单词区分开，比如“MainController”， 最好前面一个单词首字母大写，Controller也是一样，方便解析

2. 直接写一个函数

    现在，我们把app.js的代码再编辑下，如下：

[复制代码](javascript:void(0);)

1 'use strict';

2 // Define `TrialApp` module

3 var app = angular.module('TrialApp', ['ui.router']);

4 // Define routers

5 app.config(function($stateProvider) {

6 $stateProvider

7 .state('C1', {

8 url:'/C1',

9 template: '<h4>进入C1状态</h4>',

10 controller:'C1Controller'

11 })

12 .state('C2', {

13 url: '/C2',

14 templateUrl: 'Htmls/C2.html',

15 controller: function() {

16 this.test = 'world!';

17 },

18 controllerAs: 'C2Ctrl'

19 })

20 .state('C3', {

21 url: '/C3',

22 templateProvider: function() {

23 return '<h4>进入C3状态</h4><br />'

24 + '<p>{{t}}</p>';

25 },

26 controller: function($scope) {

27 $scope.t = 'C3Controller is on!';

28 }

29 })

30 });

31 app.controller('mainController', function() {

32 return alert('hello!');

33 });

34 app.run(function($state) {

35 $state.go('C1');

36 });

[复制代码](javascript:void(0);)

另外创建一个js文件“C1Controller”，代码如下：

1 'use strict';

2 //Define `C1Controller`

3 app.controller('C1Controller', function() {

4 alert('C1Controller is on!');

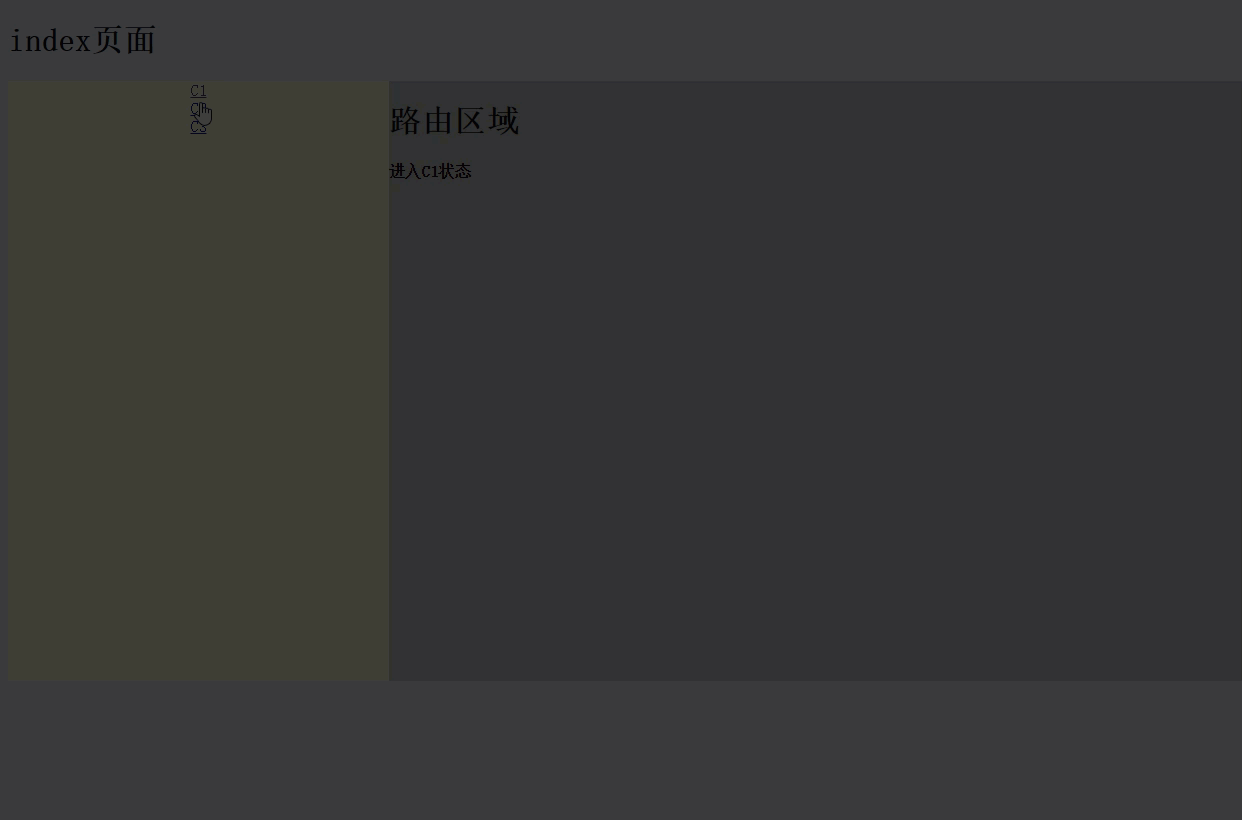
5 });

C2.html代码修改如下：

1 <h4>进入状态C2</h4><br />

2 <p>Hello, {{C2Ctrl.test}}</p>

 下面是整个应用的效果：



* 上面的代码使用了上述两种方法加载了controller。对于第一种方法，记得创建完相应的js外部文件之后，**在index.html加上该文件的路径**，不然，编译器在解析代码的时候找不到名字所对应的文件会报错Error: [$controller:ctrlreg]，

          这是都因为controller文件没有被注册成功，所以在编写完之后要注意检查这一点。

* 还有一点值得注意的是，在状态C2和C3中，分别都用了内置controller的方法，在方法中创建变量，并在对应页面中显示其值。

          在C2中，用了this.test（this指当前对象）声明并初始化变量，为了能在C2对应页面中显示变量值，需要将controller设置别别名，controllerAs: ‘C2Ctrl’，

          然后在对应html模板中用{{C2Ctrl.test}}

          而在C3中，在controller中用$scope创建变量，在对应html模板中用{{t}}，这里$scope对应的就是该controller的作用域，所以在与其关联的html文本中，直接写出该变量即可。

* 那这里，有人也许会问，我在这里也写一个contorllerAs，给controller取个别名，然后再{{controller.变量}}行不行？

         答案是不行（亲测有效）。记得AngularJS官网的phonecatApp示例中，曾提到过一句话，如果应用规模大，controller数量多而杂的情况下，在设置controller过程中，尽量避免$scope。

         个人觉得是因为，$scope代表一个controller的作用域，如果有多个controller，那就会有多个$scope，那如果大家都有$scope，那放在一起，谁会清楚这是那个作用域的属性？所以个人

         觉得controller给controller起个别名是一个比较好的方法，这样一眼就是识别这是哪个controller，方便代码后期维护，也方便别人（你的同事们）解读。

* 好了，今天主要是对ui-router入门的讲解。希望能通过这一简单的示例，让大家能对ui-router的一个基本原理有所了解。下一次将会进阶地讲ui-router的东西，比如多个视图、路由嵌套及与ngRoute的对比。如果本文有哪些讲解得不够细致或有错误的地方，欢迎读者指正。

**四、多个视图**

地址：https://blog.csdn.net/victoryzn/article/details/74315843

在此之前，上一篇文章讲述了如何使用ui-router的一些基本功能，也通过简单的示例。众所周知，ui-router除了拥有ng-route所没有的嵌套视图功能外，

还有一个非常强大的特性就是在一个状态内设置多个命名视图。我们先来做个示例来体验一下这一特性，小试牛刀：

首先设置以下路由：

在ui-router中设置了一个state ‘B’, 定义了url（必须定义不然无法显示）以及views参数。

注意：这时，如果views参数外面再定义template、templateUrl、templateProvider均不会返回页面。只有views参数面设置才有效。

以下是HTML代码：

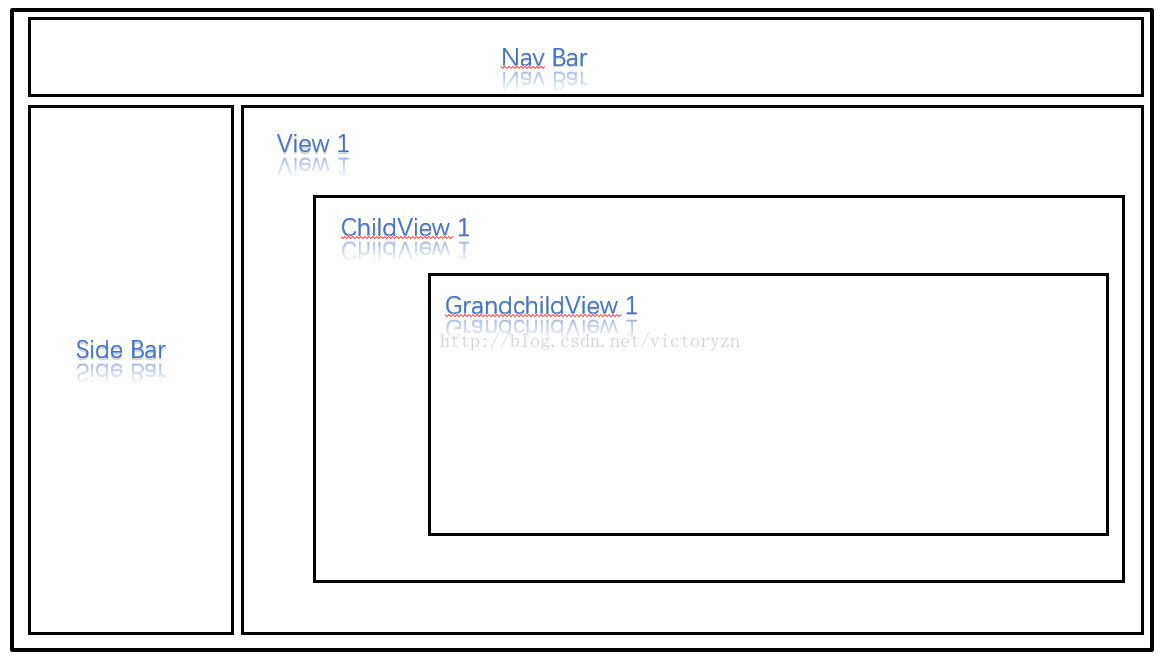
**五、嵌套视图**

地址：https://blog.csdn.net/victoryzn/article/details/74454481

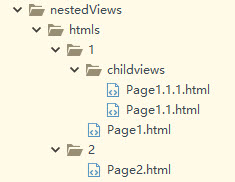
嵌套视图是ui-router不同于ng-route的最大区别之一，也是ui-router受到大众青睐的主要原因。接下来跟小编直接上手做一下简单的嵌套视图（Nested Views）。

上面是本次示例的布局，有导航栏、侧边栏、视图1及其子孙视图。

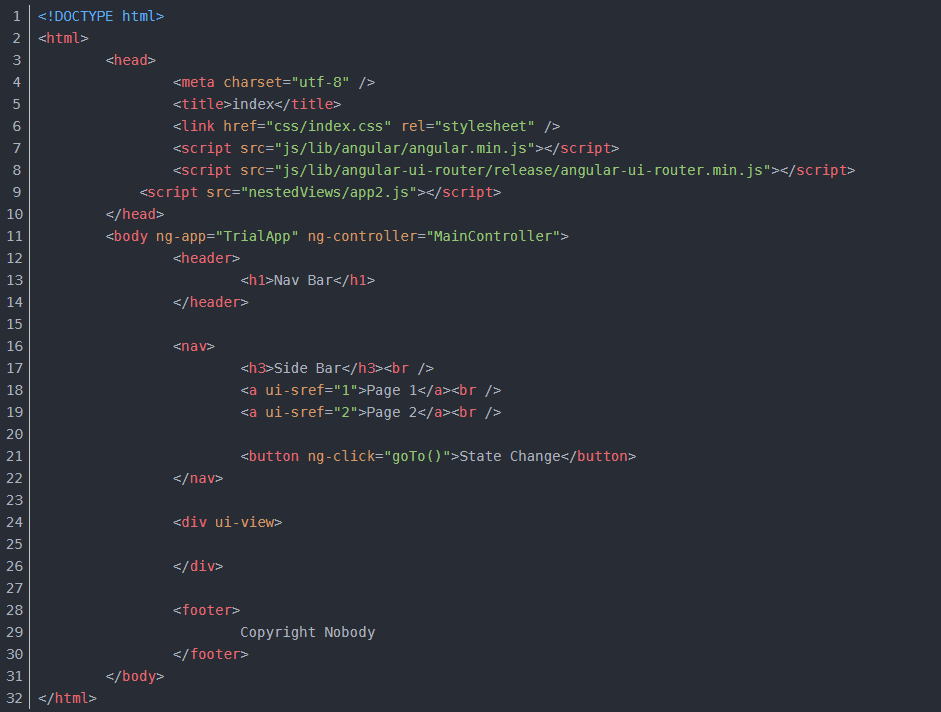
这个是整个示例的布局：



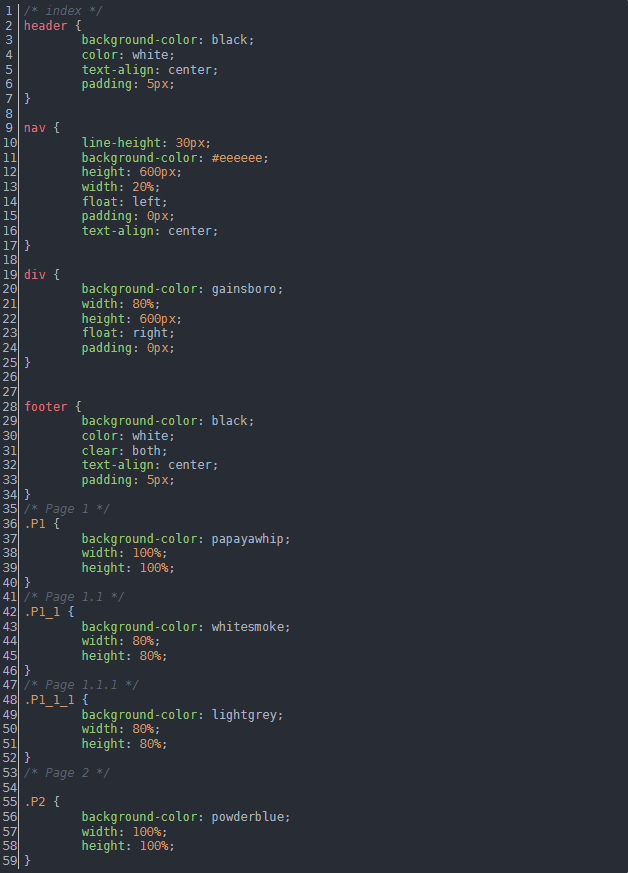
文件结构：



首先是首页demo2.html代码:



以及css文件index.css:



app2.js代码：



上述代码中，设置了嵌套视图，状态1-->状态1.1-->状态1.1.1，子视图状态名前必须加上其上一层视图的状态名加‘.’，例如father.child。多级嵌套也是如此，方便辨认。

Page1.html代码：

<div class="P1">

<h3>View 1</h3>

<button ng-click="change()">ChildViews</button>

<ui-view id="view1"></ui-view>

</div>

在Page1页面中，需要设置ui-view，这是为了给其子视图显示的地方。

Page1.1.html代码：

<div class="P1\_1">

<h3>ChildView 1.1</h3>

<button ng-click="change()">ChildViews</button>

<ui-view id="view1.1"></ui-view>

</div>

在Page1.1页面中，也需要同Page1一样设置ui-view，给其子视图显示。

Page1.1.1.html代码：

<div class="P1\_1\_1">

<h3>GrandchildView 1.1.1</h3>

</div>

Page2.html代码：

<div class="P2">

<h3>View 2</h3>

<button>ChildViews</button>

<ui-view id="view2"></ui-view>

</div>

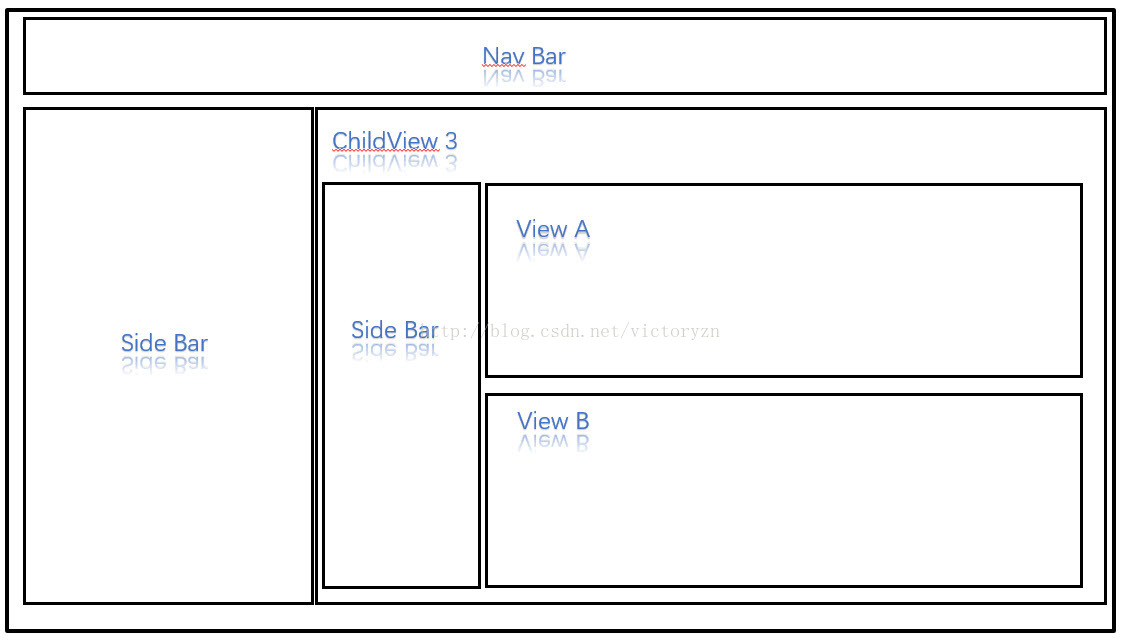
Page2 也是一样为了给之后的子视图显示，设置了ui-view。好了下面显示以下效果：



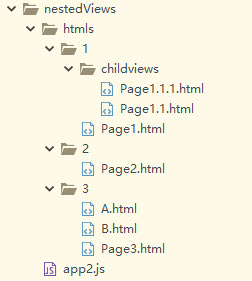
**六、ui-router嵌套视图与多个视图组合使用**

地址：地址：https://blog.csdn.net/victoryzn/article/details/74454481

这是组合使用的布局：



文件结构：



这是更新的demo2.html代码：



在index.css文件中加上以下代码：



app2.js代码:



Page3.html代码：

<div class="P3">

<h3>View 3</h3>

<section class"side">

<a ui-sref="3.child">Multiple Views</a>

</section>

<div ui-view="A" id="A"></div>

<div ui-view="B" id="B"></div>

</div>

A.html代码：

<div class="A">

<h3>Page A</h3>

</div>

B.html代码：

<div class="B">

<h3>Page B</h3>

</div>

演示效果：



视图命名（相对命名与绝对命名）

上述示例中，我们演示了views属性中，视图的相对命名。那什么是绝对命名呢？

一般，只要视图名只要包含”@“符号的就属于绝对命名。

相对命名：相对于父视图

绝对命名：指定相对于哪个视图

从原理来看，views属性中的每个视图都以”视图名@状态名“ 命名。我们动手来理解一下视图命名的奥妙。

我们修改下app2.js的代码：

.state('3.child', {

url:'/child',

views: {

'A@3' : {

templateUrl:'nestedViews/htmls/3/A.html'

},

'B@3' : {

templateUrl:'nestedViews/htmls/3/B.html'

}

}

})

//在相同的A和B视图中，切换html模板

.state('3.child1', {

url: '/child1',

views: {

'A@3': {

template: '<h3><strong>A:</strong>View has been changed!</h3>'

},

'B@3': {

template: '<h3><strong>B:</strong>View has been changed!</h3>'

}

}

})

Page3.html代码：（增加了一个状态”3.child1“，为了在多个同级视图情况下，切换html模板）

<div class="P3">

<h3>View 3</h3>

<section class"side">

<a ui-sref="3.child">Multiple Views</a><br />

<a ui-sref="3.child1">Change Views</a>

</section>

<div ui-view="A" id="A"></div>

<div ui-view="B" id="B"></div>

</div>

上述app2.js代码中：

A@3代表的是，在状态“3”所对应的html模板（Page3.html）中<div ui-view='A'>的地方；B@3代表的是，在状态“3”所对应的html模板（Page3.html）中<div ui-view='B'>的地方。

其实就是指明一个地点，就像上海市黄浦区，@前面市指定视图名，也是就是ui-view的名字，@后面则指定状态名所对应的html模板。后者相当于一个大范围，比如说城市，后者则是这个大范围里面某一个精确的地方，城市当中的某一个区。

那如果@前面没有东西，那就对应的是<div ui-view></div>，而倘若@后面没有东西，那就是默认根html模板，如果根模板中没有A、B视图，就不会显示任何效果。下面是效果展示：

