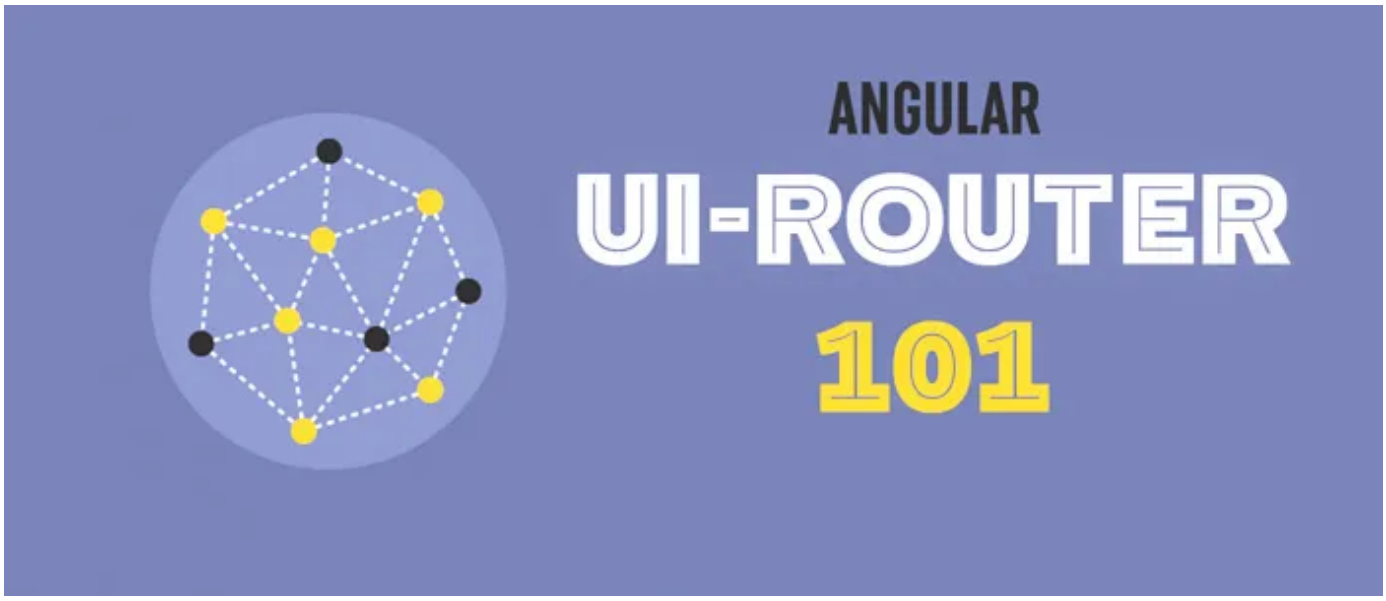
A Developer Diary

{about:"code learn and share"}

October 16, 2015 By Abhisek Jana — 1 Comment (Edit)

How use ui-router with Angular JS



I have already posted an article on ui-router which explains the basics of Angular routing and ui-router. This is an extension of that article where we will learn almost everything on how use ui-router with Angular JS.

You can find the previous article in the Part 5 of **The Best Way to Master Angular JS** series.

Nested State & View:

One of the main reason we use ui-router instead of ngRoute is the Nested State & View. We have a very interesting example here. We will build a Setup Wizard using nested Views & States.

We will have two parent view, `Setup` and `About`. On click of Setup we will display the Setup Wizard, `Step 1` and `Step 2` then we will finish the setup and display a confirmation message.

Here is demo of the application. Click on Setup and the Wizard will be displayed.

The index.html is very simple and has only two `ui-sref` and one `ui-view` to display the content. The first `ui-sref` is new, we will discuss about that in a min.

- Setup
- About

Below is the `config()` function of the `app.js`. At first we are defaulting to `/home` in order to display the `home.template.html`. So when the application is loaded the contents of the `home.template.html` (**Welcome !**) will be displayed.

You should already be familiar with the generic `home` and `about` states from my previous article. We can create a nested state by a `.` (dot) separator. In this instance we have created two nested state of the `home` state, `home.step1` and `home.step2`. Each nested state has a template (or templateUrl), url and controller attribute (same as any state). We can access the Nested State by the name, which is `home.step1` or `home.step2` in this case. Whenever we route to a nested state it will by default also display its parent state. So you want to have all common information in the parent state. In our example here, we are displaying a message in the parent (`home`) state.

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

        /* defaulting to /home to display
home.template.html */
    $urlRouterProvider.otherwise('/home');
    $stateProvider
        .state('home', {
            url: "/home",
            templateUrl:'app/partials/home.template.html',
            controller:'homeController'
        })
                        .state('home.step1', {
                                url: "/step1",

templateUrl:'app/partials/step1.template.html'
```

```
                        })
                        .state('home.step2', {
                                url: "/step2",

templateUrl:'app/partials/step2.template.html'
                        })
        .state('about', {
            url: "/about",
            template:"
```

This is a Setup Wizard !

```
",
        });
});
```

The `home.template.html` has another `ui-view` element to display the nested state/view. So we know where exactly to place the nested views.

{{msg}}

The `step1.template.html` and `step2.template.html` are displaying the wizard. We are changing the state to `home.step2` on click on the **Next** button in `step1.template.html`. The **Previous** button in `step2.template.html` will change the state again to `home.step1`. We are using the `ng-click()` to set the `$parent.msg` and then `ui-sref` to change to state to home.

Note : A nested state can access the parent controller's `$scope` using `$parent`.

Step 1

Previous

Next

Step 2

Previous

Done

Here is the `homeController` code.

```
app.controller('homeController',
    function($scope){
        $scope.msg="Welcome !"
    });
```

You can find the above code in github. Click here. Also let me know if you have any additional questions.

Abstract States :

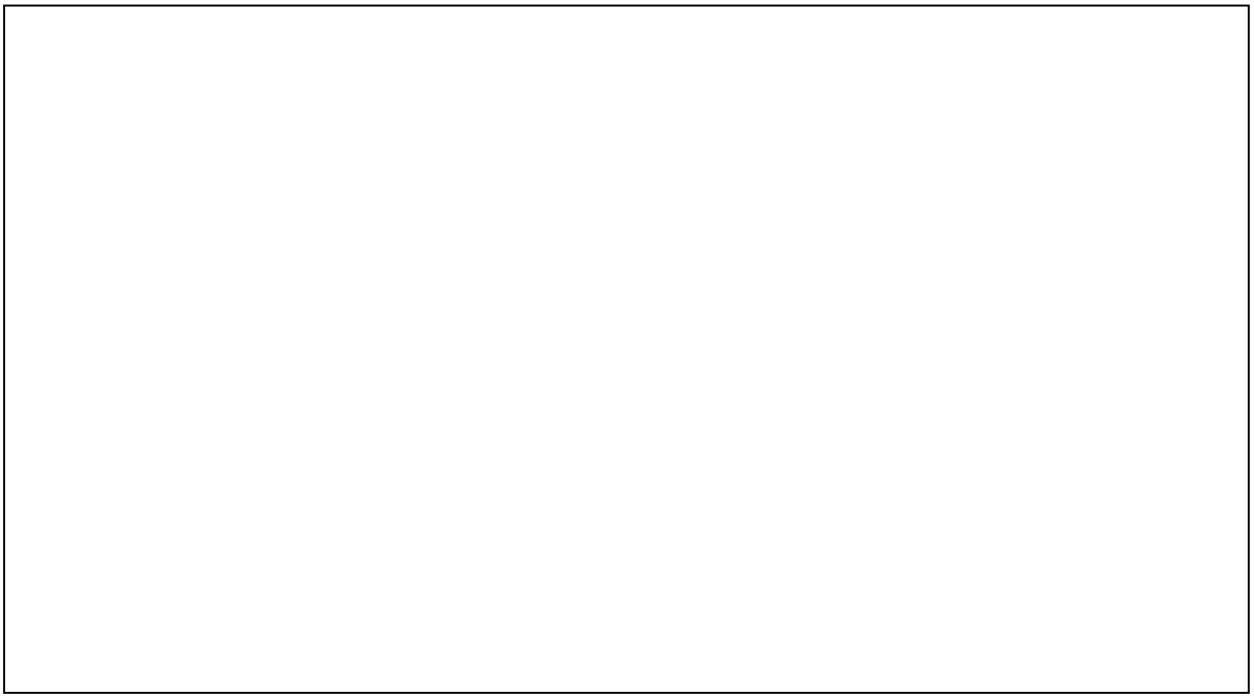
If you define a state as abstract then you can't activate the state directly, however you can activate it from the child state. If you have a header shared across multiple child state then you should define the parent state abstract so that it can't be activated directly.

In order to make a state abstract, just set the value of abstract to true. [`abstract=true`]

```
var app = angular.module('MyApp', ['ui.router']);
app.config(function($stateProvider, $urlRouterProvider){

  //You can active a child state by
/(parent_state)/(child_state)
  $urlRouterProvider.otherwise('/home/step1');

  $stateProvider
    .state('home', {
        url: "/home",
        abstract: true,
        templateUrl:'home.template.html'
    })
        .state('home.step1', {
```

```
            url: "/step1",
            templateUrl:'step1.template.html'
        })
        .state('home.step2', {
            url: "/step2",
            templateUrl:'step2.template.html'
        });
});
```

Check this demo below, each menu activates a state, however you can't activate the `home` state.



Activate Child State :

In order to activate a child state using the url you need to mention the parent state fist and them provide the child state name. Here is an example.

```
//You can active a child state by
/(parent_state)/(child_state)
```

```
$urlRouterProvider.otherwise('/home/step1');
```

Incase you want to activate them directly.You need to add the ^ in front of the url in the state. Refer the example below.

```
$urlRouterProvider.otherwise('/step1');

.state('home.step1', {
        url: "^/step1",
    templateUrl:'step1.template.html'
})
```

Resolve :

You can use resolve to resolve any dependency before initiating the controller for any view. You can define more than one object inside the resolve.Until the promise(s) are resolved the controller will not be initiated. The promise object can be injected in the controller directly.

Please refer the code below. The `promiseObject` object will be resolved first then only the `homeController` will be initiated.

```
var app = angular.module('MyApp', ['ui.router']);
app.config(function($stateProvider, $urlRouterProvider){

  $urlRouterProvider.otherwise('/home');

  $stateProvider
    .state('home', {
        resolve:{
          promiseObject:function($http){
```

```
            return
$http.get('https://api.github.com/users/adeveloperdiary/repos');
          }
        },
        url: "/home",
        templateUrl:'home.template.html',
        controller:'homeController'
    });
});

app.controller('homeController',function($scope,promiseObject)
{
  $scope.header=promiseObject.data;
});
```

In the demo below, the $http.get() will be resolved first, then the `promiseObject` will be injected in the `homeController`. We are displaying the name of the repositories under adeveloperdiary in github.

Named Views :

The named views are more complex ones and you may need them only for specific scenarios. You also probably want to avoid using these. Here is a simple example of named views.

The named views can be defined with `views` attribute. The empty `''` will refer to the parent template (`home.template.html` in this case). When you define the `ui-view` you can assign a specific name, then you can refer that from the `views` section. Here `step1@home` and `step2@home` will refer to the parent view's `ui-view="step1"` and `ui-view="step2"` respectively.

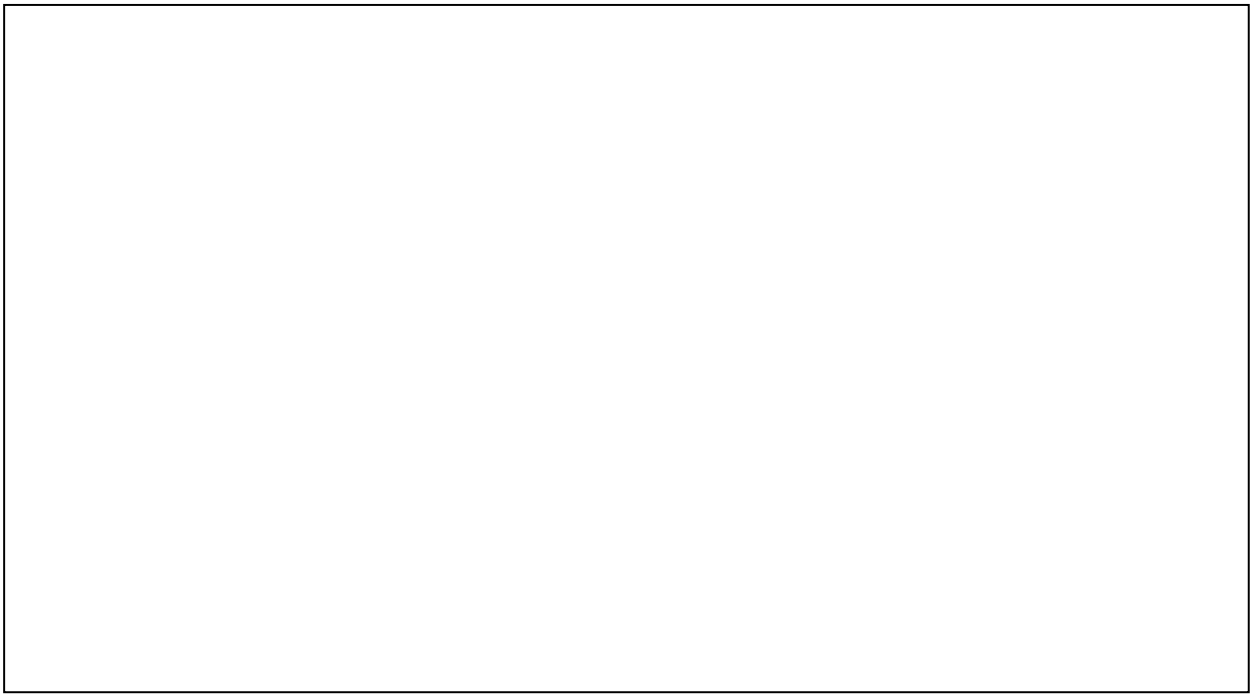
```
$stateProvider
        .state('home', {
            url: "/home",
            views:{
                                    // Parent Template
                '':{

templateUrl:'app/partials/home.template.html',
                },
                                    //Named Views
                'step1@home':{

templateUrl:"app/partials/step1.template.html"

                },
                                    //Named Views
                'step2@home':{

templateUrl:'app/partials/step2.template.html'
                }
            }
        })
```

Here is the home.template.html. The named views could be much complex than what I have explained here. In typical scenarios you will not use them and it's always a good idea to avoid so that the code is more maintainable.

Welcome !

Save

Here is the demo.



You can find the full source code in github.

If you need the more complex implementations of name view, check out this code.

In Angular JS 1.5 we will have native support for Nested View. Stay tuned to get more details on that.

Related

How to Learn and Master Angular easily – Part5
In "Angular 1.x"

How to Learn and Master Angular easily – Part3
In "Angular 1.x"

How to Learn and Master Angular easily – Part6
In "Angular 1.x"

Filed Under: Angular 1.x, JavaScript | Tagged With: Angular JS, Java Script, Nested State, Programming, State, step by step, tutorial, ui-router, WebDevelopment

Comments

Andre says

October 14, 2016 at 9:50 am

(Edit)

Thanks for the info you share. How about an forms implementation where a step has sub-steps.. Instead of Home Step1 Step2.. Its PersonalInfo->Step1&Step2. Interests->Misc1&Misc2

Reply

Leave a Reply

Logged in as Abhisek Jana. Edit your profile. Log out? Required fields are marked *

Comment *

Post Comment

This site uses Akismet to reduce spam. Learn how your comment data is processed.