عنوان: **مسیریابی تو در تو در Angularjs با استفاده از UI-Router**
نویسنده: حمید صابری
تاریخ: ۱۳۹۲/۱۲/۲۳ ۱۹:۳۵
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

UI-Router ابزاری برای مسیریابی در AngularJS است که این امکان را برایتان فراهم می‌کند تا بخش‌های برنامه رابط کاربریتان را به شکل یک ماشین حالت ساماندهی کنید. برخلاف سرویس $route که بر اساس مسیریابی URLها ساماندهی شده و کار می‌کند، UI-Router بر اساس حالت‌ها کار می‌کند، که این حالت‌ها می‌توانند در صورت لزوم مسیریابی هم داشته باشند.

UI-Router یکی از افزونه‌های مجموعه Angular-ui ، و پاراگراف بالا معرفی آن در صفحه خانگیش است (تقریبا!). این افزونه جزئیات مفصلی دارد و در این مطلب تنها به معرفی آن خواهم پرداخت (بر اساس مطالب صفحه خانگیش). پیش از ادامه پیشنهاد می‌کنم اگر مطالب زیر را نخوانده‌اید ابتدا آن‌ها را مرور کنید:
مسیریابی در AngularJs #بخش اول
مسیریابی در AngularJs #بخش دوم
مسیریابی در AngularJs #بخش سوم
برای استفاده از UI-Router باید:

فایل جاوا اسکریپت آن را دانلود کنید (released یا minified).
در صفحه اصلی برنامه‌تان پس از include کردن فایل اصلی AngularJS فایل angular-ui-router.js (یا angular-ui-router.min.js) را include کنید.
'ui.router' را به لیست وابستگی‌های ماژول اصلی اضافه کنید.
نتیجه چیزی شبیه این خواهد بود:

```
<!doctype html>
<html ng-app="myApp">
<head>
    <script src="//ajax.googleapis.com/ajax/libs/angularjs/1.1.5/angular.min.js"></script>
    <script src="js/angular-ui-router.min.js"></script>
    <script>
        var myApp = angular.module('myApp', ['ui.router']);
        // For Component users, it should look like this:
        // var myApp = angular.module('myApp', [require('angular-ui-router')]);
    </script>
    ...
</head>
<body>
    ...
</body>
</html>
```

حالت‌ها و viewهای تو در تو قابلیت اصلی UI-Router امکان تعریف حالت‌ها و viewهای تو در تو است. در مطلب مسیریابی در AngularJs #بخش اول دایرکتیو ng-view معرفی شده است. هنگام استفاده از سرویس $route با این دایرکتیو می‌توان محل مورد نظر برای بارگذاری محتویات مربوط به مسیرها را مشخص کرد. دایرکتیو ui-view در UI-Router همین نقش را دارد. فرض کنید این کد فایل index.html باشد:

```
<!-- index.html -->
<body>
    <div ui-view></div>
    <!-- We'll also add some navigation: -->
    <a ui-sref="state1">State 1</a>
    <a ui-sref="state2">State 2</a>
</body>
```

همانطور که ملاحظه می‌کنید در تگ‌های a از دایرکتیو ui-sref استفاده شده است. این دایرکتیو علاوه بر مدیریت تغییر حالت،

خصوصیت href تگ a را در صورتی که حالت مشخص شده URL داشته باشد تولید می‌کند. البته برای استفاده از UI-Router ملزم به استفاده از دایرکتیو ui-sref نیستید و می‌توانید href را مشخص کنید. ولی با استفاده از ui-sref لازم نیست مسیر یک حالت را به یاد داشته باشید، و یا در صورت تغییر آن، همه hrefها را به روز کنید.
در ادامه برای هر کدام از حالت‌ها یک template اضافه می‌کنیم:
فایل state1.html:

```
<!-- partials/state1.html -->
<h1>State 1</h1>
<hr/>
<a ui-sref="state1.list">Show List</a>
<div ui-view></div>
```

فایل state2.html:

```
<!-- partials/state2.html -->
<h1>State 2</h1>
<hr />
<a ui-sref="state2.list">Show List</a>
<div ui-view></div>
```

دو نکته قابل توجه در این templateها وجود دارد. اول اینکه همانطور که می‌بینید templateها خود شامل تگی با دایرکتیو ui-view هستند. و دوم مقدار دایرکتیو ui-sref است که به صورت state1.list و state2.list آمده است. این جدا سازی با نقطه نشان دهنده سلسله مراتب حالت‌هاست. یعنی حالت‌های state1 و state2 هرکدام حالت فرزندی به نام list دارند. در ادامه وقتی حالت‌ها و مسیریابی را در app.config() تعریف کنیم این مسائل از هاله‌ای از ابهام که در آن هستند خارج می‌شوند! فعلا بیاید با راهنمای UI-Router پیش برویم و فایل‌های template حالت‌های فرزند را تعریف کنیم. templateهایی که قرار است در ui-view پدرانشان بارگذاری شوند:

```
<!-- partials/state1.list.html -->
<h3>List of State 1 Items</h3>
<ul>
  <li ng-repeat="item in items">{{ item }}</li>
</ul>
```

```
<!-- partials/state2.list.html -->
<h3>List of State 2 Things</h3>
<ul>
  <li ng-repeat="thing in things">{{ thing }}</li>
</ul>
```

خوب! حالا برویم سراغ شعبده بازی! برای اینکه از UI-Router استفاده کنید لازم است $stateProvider و $urlRouterProvider را به عنوان وابستگی به app.config() تزریق کنید:

```
myApp.config(['$stateProvider', '$urlRouterProvider',
function($stateProvider, $urlRouterProvider) {
  //
  // For any unmatched url, redirect to /state1
  $urlRouterProvider.otherwise("/state1");
  //
  // Now set up the states
  $stateProvider
    .state('state1', {
      url: "/state1",
      templateUrl: "partials/state1.html"
    })
    .state('state1.list', {
      url: "/list",
      templateUrl: "partials/state1.list.html",
      controller: function($scope) {
        $scope.items = ["A", "List", "Of", "Items"];
      }
    })
```

```
    .state('state2', {
      url: "/state2",
      templateUrl: "partials/state2.html"
    })
    .state('state2.list', {
      url: "/list",
        templateUrl: "partials/state2.list.html",
        controller: function($scope) {
          $scope.things = ["A", "Set", "Of", "Things"];
        }
      })
    }]);
```

در ابتدا با متد ()$urlRouterProvider.otherwise مسیر پیشفرض مشخص شده است. متد otherwise را باید از مقالات مسیریابی در AngularJS به یاد داشته باشید. سپس حالت‌های برنامه با استفاده از متد state تعریف شده است. این متد دو پارامتر ورودی دارد؛ اولی نام حالت و دومی یک شی شامل خصوصیات حالت. همانطور که می‌بینید این شی خصوصیاتی شبیه به همان‌ها که در متد ()$routeProvider.when وجود داشت دارد. می‌شود گفت این خصوصیات همان‌ها هستند و همان عملکرد را دارند.

خصوصیت url مشخص کننده مسیر حالت است. این خصوصیت همان مقداریست که به عنوان پارامتر اول به ()$routeProvider.when پاس می‌شد. در این پارامتر می‌شود متغیرهای url را هم به همان ترتیب تعریف کرد. مثلا اگر حالت state1 در آدرسش یک پارامتر id داشته باشد می‌شود آن را به این ترتیب تعریف کرد:

```
.state('state1', {
      url: "/state1/:id",
      templateUrl: "partials/state1.html"
    })
```

برای خواندن مقدار این متغیر باید از $stateParams استفاده کرد:

```
$stateParams.id
```

به خصوصیت url دو حالت state1.list و state2.list دقت کنید. هردو برابر '/list' است. یعنی هردو یک مسیر دارند؟ نه! بلکه مسیر state1.list برابر 'state1/list/' و مسیر state2.list برابر 'state2/list/' است. در واقع حالت state1.list یعنی list فرزند state1 و به همین ترتیب state2.list یعنی list فرزند state2. و می‌توان گفت UI-Router آدرس url حالت فرزند را، آدرسی نسبی، نسبت به url حالت پدر می‌داند. این رابطه سلسله مراتبی و پدر و فرزندی را می‌توان با استفاده از خصوصیت parent به صورت صریح‌تری مشخص کرد:

```
.state('list', {
    parent: "state1",
    url: "/list",
    templateUrl: "partials/state1.list.html",
    controller: function($scope) {
        $scope.items = ["A", "List", "Of", "Items"];
    }
})
.state('list', {
    parent: "state2",
    url: "/list",
    templateUrl: "partials/state2.list.html",
    controller: function($scope) {
        $scope.items = ["A", "List", "Of", "Items"];
    }
})
```

تا اینجای کار، اگر آدرس "/state1" وارد شود، فایل "partials/state1.html" در "ui-view" فایل "index.html" بارگذاری خواهد شد. اگر آدرس "/state1/list" وارد شود، ابتدا فایل "partials/state1.html" در "ui-view" فایل "index.html" بارگذاری شده، سپس فایل "partials/state1.list.html" در "ui-view" آمده در فایل فایل "partials/state1.html" بارگذاری می‌شود. این همان امکان حالت‌ها و viewهای تو در تو است که UI-Router فراهم می‌کند. اینجا می‌توانید خروجی کدهای بالا را مشاهده کنید.

اگر مستقیما url یک حالت فرزند وارد شود، یا به عبارت دیگر، اگر بخواهیم مستقیما برنامه به حالتی که فرزند حالت دیگر است برود، UI-Router برنامه را ابتدا به حالت پدر، و پس از آن به حالت فرزند خواهد برد. حالت فرزند دو چیز را از حالت پدر به ارث

می‌برد:
وابستگی‌های فراهم شده در حالت پدر به وسیله " resolve "

داده‌های سفارشی مشخص شده در خصوصیت data حالت پدر

استفاده از resolve در UI-Router مشابه استفاده از آن در $route است. ولی افزودن داده‌های سفارشی کمی متفاوت است.
برای افزودن داده‌های سفارشی باید از خصوصیت data یک حالت استفاده کرد:

```
.state('state1', {
    url: "/state1",
    templateUrl: "partials/state1.html",
    data:{
        foodata: 'addorder'
    }
})
```

برای دسترسی به این داده‌ها هم می‌توان از $state.current.data استفاده کرد:

```
$state.current.data.foodata
```

Viewهای نامگذاری شده و چندگانه

یکی دیگر از قابلیت‌های کاربردی UI-Router امکان داشتن چند ui-view در هر template است (استفاده همزمان از این قابلیت و حالت‌های تو در تو، امکان مدیریت واسط کاربری را به خوبی فراهم می‌کند). برای توضیح این قابلیت، با راهنمای UI-Router همراه شویم:

1. دستورالعمل برپایی UI-Router که در بالا آمده را اجرا کنید.
2. یک یا چند ui-view به برنامه‌تان اضافه کنید و آن‌ها را نامگذاری کنید:

```
<!-- index.html -->
<body>
    <div ui-view="viewA"></div>
    <div ui-view="viewB"></div>
    <!-- Also a way to navigate -->
    <a ui-sref="route1">Route 1</a>
    <a ui-sref="route2">Route 2</a>
</body>
```

3. حالت‌های برنامه‌تان را در روال config ماژول تعریف کنید:

```
myApp.config(function ($stateProvider) {
    $stateProvider
      .state('index', {
          url: "",
          views: {
              "viewA": { template: "index.viewA" },
              "viewB": { template: "index.viewB" }
          }
      })
      .state('route1', {
          url: "/route1",
          views: {
              "viewA": { template: "route1.viewA" },
              "viewB": { template: "route1.viewB" }
          }
      })
      .state('route2', {
          url: "/route2",
          views: {
              "viewA": { template: "route2.viewA" },
              "viewB": { template: "route2.viewB" }
          }
```

```
    })
});
```

4. خروجی کدهای بالا را اینجا مشاهده کنید.

چند نکته

UI-Router جزئیات فراوانی دارد و آنچه آمد تنها پرده برداری از آن بود. دلم می‌خواست می‌توانستم بیش از این آن را معرفی کنم، اما متاسفانه این روزها وقت آزاد کافی ندارم. در انتها می‌خواهم به چند نکته اشاره کنم: **روش controller as**

برای استفاده از روش controller as در UI-Router باید به این ترتیب عمل کنید:

```
.state('list', {
    parent: "state1",
    url: "/list",
    templateUrl: "partials/state1.list.html",
    controller: "state1ListController as listCtrl1"
    }
})
.state('list', {
    parent: "state2",
    url: "/list",
    templateUrl: "partials/state2.list.html",
    controller: "state2ListController as listCtrl2"
    }
})
```

حالت‌های انتزاعی

حالت انتزاعی حالتی است که url ندارد و در نتیجه برنامه نمی‌تواند در آن حالت قرار گیرد. حالت‌های انتزاعی بسیار به درد خور هستند! مثلا فرض کنید چند حالت دارید که اشتراکاتی با هم دارند (همه باید در template مشابهی بارگذاری شود، یا وابستگی‌های یکسانی دارند، یا حتی سطح دسترسی یکسان). با تعریف یک حالت انتزاعی و جمع کردن همه وابستگی‌ها در آن، و تعریف حالت‌های مورد نظرتان به عنوان فرزندان حالت انتزاعی، می‌توانید اشتراکات حالت‌های برنامه را ساده‌تر مدیریت کنید.

حساسیت به حروف بزرگ و کوچک

در سرویس $route با مقداردهی خصوصیت caseInsensitiveMatch می‌توانستیم مشخص کنیم که بزرگ و کوچک بودن حروف در تطبیق آدرس صفحه با پارامتر route در نظر گرفته بشود یا نه . خودمانیش اینکه url به حروف بزرگ و کوچک حساس باشد یا نه. متاسفانه در UI-Router از این امکان خبری نیست (البته فعلا) و آدرس‌های تعریف شده به حروف بزگ و کوچک حساس هستند. اینجا روشی برای حل این مشکل پیشنهاد شده، به این ترتیب که همه urlهای وارد شده به حروف کوچک تبدیل شود (راستش من این راه حل را نمی‌پسندم!). چند روز قبل هم تغییراتی در کد UI-Router داده شده که امکان حساس نبودن به حروف کوچک و بزرگ فراهم شود . این تغییر هنوز در نسخه نهایی فایل UI-Router نیامده است. هرچند اگر بیاید هم آنچه تا امروز (23 اسفند 92) انجام شده مشکل را حل نمی‌کند.

اگر شما هم مثل من می‌خواهید کلا آدرس‌ها به حروف بزرگ و کوچک حساس نباشند، و فرصت حل کردن اساسی مشکل را هم ندارید به این ترتیب عمل کنید:

در فایل "angular-ui-router.js" عبارت "(new RegExp(compiled" را پیدا کرده و آن را به "('RegExp(compiled, 'i" تبدیل کنید.

و یا در "angular-ui-router.min.js" (هرکدام از فایل‌ها که استفاده می‌کنید) عبارت new RegExp(o) را پیدا کرده و آن را به new RegExp(o, "i") تبدیل کنید. همین؛ صدایش را هم در نیاورید!