عنوان: **Lazy Loading در AngularJS**
نویسنده: حمید صابری
تاریخ: ۱۳۹۳/۰۱/۰۸ ۱۶:۵۰
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

وقتی پروژه انگیولاری‌تان کمی گسترش پیدا کند، تعداد زیادی فایل شامل کنترلرها، سرویس‌ها، دایرکتیوها و ... خواهید داشت. واضح است که همه این اجزا همراه با هم مورد نیاز نیستند و برای افزایش سرعت بارگذاری سایت و صرفه جویی در مصرف پهنای باند بهتر است هرکدام از آن‌ها را در هنگام نیاز بارگذاری کنیم. این یعنی همان lazy loading خودمان!
در AngularJS امکانی برای lazy loading فایل‌ها پیشبینی نشده است، پس باید از ابزارهای دیگری که این امکان را فراهم می‌کنند استفاده کرد. من در ادامه از Script.js برای این کار استفاده خواهم کرد، ولی شما می‌توانید از هر کتابخانه دیگری استفاده کنید.
اما مسئله دیگری که پیش از lazy loading فایل‌ها باید تکلیفش را معلوم کنیم، این است که چطور می‌توانیم اجزایی را به ماژولی که قبلا راه‌اندازی (bootstrap) شده اضافه کنیم. اگر بخواهیم برای مثال کنترلری را در یک فایل مجزا تعریف کنیم، باید آن را به شکلی در ماژول برنامه‌مان ثبت کنیم. فرض کنید این کار را به این ترتیب انجام دهیم:

```
angular.module('app').controller('SomeLazyController', function($scope)
{
    $scope.key = '...';
});
```

در این صورت اگر این کنترلر را در قسمتی از برنامه به صورت 'ng-controller='SomeLazyController استفاده کنیم با این خطا مواجه خواهیم شد:

```
Error: Argument ‘SomeLazyController’ is not a function, got undefined
```

برای این کار (افزودن اجزایی به ماژولی که قبلا راه اندازی شده) می‌توانیم بجای استفاده از APIهای ماژول، از provider های AngularJS استفاده کنیم. به این ترتیب برای ثبت یک کنترلر باید از $controllerProvider ، برای ثبت یک directive از $compileProvider ، برای ثبت فیلترها از $filterProvider و برای ثبت سایر اجزا در ماژول از $provide استفاده کنیم:

```
// Registering a controller after app bootstrap
$controllerProvider.register('SomeLazyController', function($scope)
{
   $scope.key = '...';
});

// Registering a directive after app bootstrap
$compileProvider.directive('SomeLazyDirective', function()
{
    return {
        restrict: 'A',
        templateUrl: 'templates/some-lazy-directive.html'
    }
})

// etc
```

اما نکته‌ای که درباره providerها وجود دارد این است که آن‌ها تنها در روال config یک ماژول در دسترس هستند. بنا بر این برای دسترسی به آن‌ها پس از اجرای این روال، ارجاعی به آنها را باید نگهداری کنیم:

```
(function () {
    app = angular.module("app", []);

    app.config([
        '$controllerProvider',
        '$compileProvider',
        '$filterProvider',
        '$provide',
```

```
    function ($controllerProvider, $compileProvider, $filterProvider, $provide) {
        //برای رجیستر کردن غیر همروند اجزای انگیولاری در آینده
        app.lazy =
        {
            controller: $controllerProvider.register,
            directive: $compileProvider.directive,
            filter: $filterProvider.register,
            factory: $provide.factory,
            service: $provide.service
        };
    }]);
})();
```

اکنون SomeLazyController را به این ترتیب می‌توانیم ثبت کنیم:

```
angular.module('app').lazy.controller('SomeLazyController', function($scope)
{
    $scope.key = '...';
});
```

نکته دیگر این است که کجا باید lazy loadign را انجام دهیم. به نظر می‌رسد مناسب‌ترین محل برای انجام این کار خصوصیت resolve مسیریابی است. در این مطلب و این مطلب resolve در $route و UI-Router معرفی شده است:

```
$stateProvider
            .state('state1', {
                url: '/state1',
                template: '<div>{{st1Ctrl.msg}}</div>',
                controller: 'state1Controller as st1Ctrl',
                resolve: {
                    fileDeps: ['$q', '$rootScope', function ($q, $rootScope) {
                        var deferred = $q.defer();
                        var deps = [
                            'app/messageService.js',
                            'app/state1Controller.js'];
                        $script(deps, function () {
                            $rootScope.$apply(function () {
                                deferred.resolve();
                            });
                        });
                        return deferred.promise;
                    }]
                }
            })
            .state('state2', {
                url: '/state2',
                template: '<div>{{st2Ctrl.msg}}</div>',
                controller: 'state2Controller as st2Ctrl',
                resolve: {
                    fileDeps: ['$q', '$rootScope', function ($q, $rootScope) {
                        var deferred = $q.defer();
                        var deps = [
                            'app/messageService.js',
                            'app/state2Controller.js'];
                        $script(deps, function () {
                            $rootScope.$apply(function () {
                                deferred.resolve();
                            });
                        });
                        return deferred.promise;
                    }]
                }
            });
    }]);
```

کنترلر state1Controller که در فایلی با همین نام پیاده‌سازی شده است تنها در مسیر state1/ مورد نیاز است، و state2Controller تنها در مسیر state2/ لازم است بارگذاری شود. هردوی این کنترلرها به messageService وابستگی دارند که در messageService.js پیاده سازی شده است (همانطور که در این مطلب اشاره شده می‌توانیم یک حالت انتزاعی به عنوان پدر دو حالت موجود تعریف کرده و وابستگی مشترک را به آن منتقل کنیم).

برای بارگذاری فایل‌های مورد نیاز در ابتدای کار و راه اندازی اولیه برنامه هم می‌توان به این ترتیب عمل کرد:

```
<script type="text/javascript">
        // ----Script.js----
        !function (a, b, c) { function t(a, c) { var e = b.createElement("script"), f = j; e.onload =
e.onerror = e[o] = function () { e[m] && !/^c|loade/.test(e[m]) || f || (e.onload = e[o] = null, f = 1,
c()) }, e.async = 1, e.src = a, d.insertBefore(e, d.firstChild) } function q(a, b) { p(a, function (a)
{ return !b(a) }) } var d = b.getElementsByTagName("head")[0], e = {}, f = {}, g = {}, h = {}, i =
"string", j = !1, k = "push", l = "DOMContentLoaded", m = "readyState", n = "addEventListener", o =
"onreadystatechange", p = function (a, b) { for (var c = 0, d = a.length; c < d; ++c) if (!b(a[c]))
return j; return 1 }; !b[m] && b[n] && (b[n](l, function r() { b.removeEventListener(l, r, j), b[m] =
"complete" }, j), b[m] = "loading"); var s = function (a, b, d) { function o() { if (!--m) { e[l] = 1,
j && j(); for (var a in g) p(a.split("|"), n) && !q(g[a], n) && (g[a] = []) } } function n(a) { return
a.call ? a() : e[a] } a = a[k] ? a : [a]; var i = b && b.call, j = i ? b : d, l = i ? a.join("") : b, m
= a.length; c(function () { q(a, function (a) { h[a] ? (l && (f[l] = 1), o()) : (h[a] = 1, l && (f[l] =
1), t(s.path ? s.path + a + ".js" : a, o)) }) }, 0); return s }; s.get = t, s.ready = function (a, b,
c) { a = a[k] ? a : [a]; var d = []; !q(a, function (a) { e[a] || d[k](a) }) && p(a, function (a) {
return e[a] }) ? b() : !function (a) { g[a] = g[a] || [], g[a][k](b), c && c(d) }(a.join("|")); return
s }; var u = a.$script; s.noConflict = function () { a.$script = u; return this }, typeof module !=
"undefined" && module.exports ? module.exports = s : a.$script = s }(this, document, setTimeout)

        $script('Scripts/angular.js', function () {
            $script('Scripts/angular-ui-router.js', function () {
                $script('app/app.js', function () {
                    angular.bootstrap(document, ['app']);
                });
            });
        });
    </script>
```

توجه داشته باشید که لازم نیست بارگذاری فایل‌ها حتما یکی پس از دیگری باشد. ترتیب بارگذاری فایل‌ها تنها در آن‌هایی که وابستگی به هم دارند باید رعایت شود. همچنین، می‌توانید همه فایل‌های مورد نیاز در این مرحله را Bundle کنید. از اینجا می‌توانید پروژه بسیار ساده‌ای که در آن lazy loading پیاده شده است را دانلود کرده و مطالب توضیح داده شده را مشاهده کنید.

نظرات خوانندگان

نویسنده: علی فخرائی
تاریخ: ۱۳۹۳/۰۱/۱۰ ۱۵:۵۲

با تشکر.
اگر ممکن است یک نمونه از تعریف دایرکتیو توسط lazy را هم بنویسید.

نویسنده: حمید صابری
تاریخ: ۱۳۹۳/۰۱/۱۰ ۱۸:۲۲

سلام.
یک لینک به index.html اضافه کنید:

```
<div style="direction: rtl">
        <a href="#/state1">1 حالت</a> |
        <a href="#/state2">2 حالت</a> |
        <a href="#/state3">3 حالت</a>
        <div ui-view style="font-weight:bold; text-align:center;"></div>
    </div>
```

فرض کنید محتویات مورد نظر برای این حالت که در فایل app/state3.html قرار دارد، شامل یک دایرکتیو است: state3.html:

```
تگ زیر یک دایرکتیو دارد:
<br/>
<div ng-hello-directive></div>
```

ng-hello-directive در فایل app/helloDirective.js به این صورت تعریف شده است:

```
angular.module('app').lazy.directive('ngHelloDirective', function () {
    return function (scope, elem, attr) {
        elem.html('سلام دایرکتیو تنبل!');
    };
});
```

و در نهایت حالت sate3 را با آدرس state3/ در app.js تعریف کنید:

```
.state('state3', {
            url: '/state3',
            templateUrl: 'app/state3.html',
            resolve: {
                fileDeps: ['$q', '$rootScope', function ($q, $rootScope) {
                    var deferred = $q.defer();
                    var deps = ['app/helloDirective.js'];
                    $script(deps, function () {
                        $rootScope.$apply(function () {
                            deferred.resolve();
                        });
                    });
                    return deferred.promise;
                }]
            }
        });
```

از اینجا می‌توانید پروژه مثال را که این دایرکتیو به آن افزوده شده دانلود کنید.
دقت کنید که در این حالت، این دایرکتیو تنها در ماژولی با نام app که خصوصیتی به نام lazy به صورت توضیح داده شده دارد ثبت می‌شود. اگر تابحال دایرکتیو آماده‌ای را دریافت کرده باشید، دیده‌اید که این دایرکتیوها به این صورت تعریف می‌شوند:

```
angular.module('moduleOfDirective', []).directive('ngDirectiveName', ...
```

همانطور که می‌بینید یک ماژول جدید تعریف شده و دایرکتیو در آن ثبت شده است. برای استفاده از چنین دایرکتیوی باید ماژولِ

دایرکتیو را به وابستگی‌های ماژول خودتان اضافه کنید:

```
app = angular.module("app", ['ui.router', 'moduleOfDirective']);
```

در این حالت حتما باید فایل دایرکتیو را پیش از فایل app خود بارگذاری کرده باشید. یا اینکه تعریف دایرکتیو را تغییر دهید و بجای تعریف ماژول جدید، آن را به همان ماژول خودتان اضافه کنید. یعنی تعریف دایرکتیو را به این شکل تغییر دهید:

```
angular.module('app', []).lazy.directive('ngDirectiveName', ...
```

حالا این دایرکتیو را هم می‌توانید *تنبلانه!* بارگذاری کنید.

نویسنده: علی فخرائی
تاریخ: ۱۳۹۳/۰۱/۱۴ ۱۲:۴۸

بسیار ممنون جناب صابری.
شما در قسمت مسیریابی هم نام کنترلر را وارد کرده اید:

```
.state('state2', {
                url: '/state2',
                template: '<div>{{st2Ctrl.msg}}</div>',
                controller: 'state2Controller as st2Ctrl',
```

و هم فایل مرتبط را به عنوان وابستگی تعریف کرده اید:

```
var deps = ['app/messageService.js',
                'app/state2Controller.js'];
```

اما چرا محتوای کنترلر state2Controller.js دو بار اجرا میشود؟یعنی با هر بار تغییر مسیر، 2 بار کل محتوای کنترلر اجرا میشود. مثلا اگر 1 تابع را در کنترلر صدا زده باشیم، این تابع 2 بار اجرا میشود.

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۳/۰۱/۱۴ ۱۳:۳۹

چک کنید ببینید در قسمت کدهای HTML ، ویژگی ای به نام ng-controller که به کنترلر شما اشاره کند وجود نداشته باشد