عنوان: **AngularJS #4**
نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۸/۲۳ ۱۷:۱۵
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

در این قسمت قصد دارم تا یک سیستم ارسال دیدگاه را به کمک Angular پیاده سازی کنم. هدف از این مثال؛ آشنایی با چند Directive توکار Angular و همچنین آموختن چگونگی کار با سرویس $http برای ارتباط با سرور است.
کدهای HTML زیر را در نظر بگیرید:

```
<div ng-app="myApp">
    <div ng-controller="CommentCtrl">

        <div ng-repeat="comment in comments">
            <div style="float:right;cursor:pointer;" ng-click="remove(comment.Id,$index);">X</div>
            <a href="#">
                <img style="width:32px;" ng-src="/Content/user.gif" alt="{{comment.Name}}">
            </a>
            <div>
                <h4>{{comment.Name}}</h4>
                {{comment.CommentBody}}
            </div>
        </div>

        <div>
            <form action="/Comment/Add" method="post">
                <div>
                    <label for="Name">Name</label>
                    <input id="Name" type="text" name="Name" ng-model="comment.Name" placeholder="Your
Name" />
                </div>
                <div>
                    <label for="Email">Email</label>
                    <input id="Email" type="text" name="Email" ng-model="comment.Email"
placeholder="Your Email" />
                </div>
                <div>
                    <label for="CommentBody">Comment</label>
                    <textarea id="CommentBody" name="CommentBody" ng-model="comment.CommentBody"
placeholder="Your Comment"></textarea>
                </div>
                <button type="button" ng-click="addComment()">Send</button>
            </form>
        </div>

    </div>
</div>
```

خب از ابتدا ساختار را مورد بررسی قرار می‌دهم و موارد ناآشنای آن را توضیح می‌دهم:
ng-app : خاصیت ng-app جز خواص پیش فرض HTML نیست و یک خاصیت سفارشی است که توسط Angular به صورت پیش فرض تعریف شده است. این خاصیت به Angular می‌گوید که کدام بخش از DOM باید توسط Angular مدیریت و پردازش شود. در اینجا div ای که با خاصیت ng-app مزین شده است به همراه تمامی عناصر فرزند آن توسط موتور پردازش گر DOM توکار مورد پردازش قرار گرفته و اصطلاحا کامپایل می‌شود. بله! اینجا از لفظ کامپایل شدن برای بیان این فرآیند استفاده کردم. هیچ کدام از این Directiveهای سفارشی به خودی خود برای مرورگر قابل تفسیر نیست و اینجاست که Angular وارد عمل شده و این Directiveها را به کدهای HTML و جاوا اسکریپت که برای مرورگر قابل فهم است تبدیل می‌کند. به همین جهت با ng-app مشخص می‌کنیم که کدام بخش از DOM باید توسط Angular تفسیر و مدیریت شود.شاید این سوال برای شما مطرح شده باشد که در مثال قبلی ng-app مقداری نداشت و برای تگ html تعریف شده بود. پاسخ این است که در مثال قبلی چون برنامه‌ی ما دارای یک ماژول بیشتر نبود می‌توانستیم از مقدار دهی ng-app صرف نظر کنیم؛ اما در این مثال ما قصد داریم کمی هم مفهوم ماژول را در Angular بررسی کنیم. در نتیجه در این مثال برنامه‌ی ما از ماژولی به نام myApp تشکیل شده است. دلیل اینکه در این مثال ng-app بر روی یک div تعریف شده است این است که همین قسمت از DOM توسط Angular تفسیر شود برای ما کفایت می‌کند. هنگامی ng-app را بر روی html تعریف می‌کنیم که قصد داشته باشیم کل صفحه توسط Angular تفسیر شود. **ng-controller** : در Angular کنترلرها تابع سازنده‌ی کلاس‌های ساده‌ی جاوا اسکریپتی هستند که به کمک آن‌ها بخشی از صفحه را مدیریت می‌کنیم. این که کدام بخش از

صفحه توسط کدام کلاس کنترل و مدیریت شود، توسط ng-controller مشخص می‌شود. در اینجا هم عنصری که با ng-controller مشخص شده به همراه تمامی فرزندانش، توسط کلاس جاوا اسکریپتی به نام CommentCtrl مدیریت می‌شود. در حقیقت ما به کمک ng-controller مشخص می‌کنیم که کدام قسمت از View توسط کدام Controller مدیریت می‌شود. مرسوم است که در Angular نام کنترلرها با Ctrl خاتمه یابد. **ng-repeat** : همه‌ی نظرات دارای یک قالب html یکسان هستند که به ازای داده‌های متفاوت تکرار شده اند. اگر می‌خواستیم نظرات را استفاده از موتور نمایشی Razor نشان دهیم از یک حلقه‌ی foreach استفاده می‌کردیم. خبر خوب این است که ng-repeat هم دقیقا به مانند حلقه‌ی foreach عمل می‌کند.در اینجا عبارت comment in comments دقیقا برابر با آن چیزی است که در یک حلقه‌ی foreach می‌نوشتیم. Comments در اینجا یک لیست به مانند آرایه ای از comment هست که در کنترلر مقدار دهی شده است. پس اگر با حلقه‌ی foreach مشکلی نداشته باشید با مفهوم ng-repeat هم مشکلی نخواهید داشت و دقیقا به همان شکل عمل می‌نماید. **ng-click** : همان طور که گفتیم Directiveهای تعریف شده می‌توانند یک event سفارشی نیز باشند. ng-click هم یک Directive تو کار است که توسط Angular به صورت پیش فرض تعریف شده است. کاملا مشخص است که یک تابع به نام remove تعریف شده است که به هنگام کلیک شدن، فراخوانی می‌شود. دو پارامتر هم به آن ارسال شده است. اولین پارامتر Id دیدگاه مورد نظر است تا به سرور ارسال شود و از پایگاه داده حذف شود. دومین پارامتر $index است که یک متغیر ویژه است که توسط Angular در هر بار اجرای حلقه‌ی ng-repeat مقدارش یک واحد افزایش می‌یابد. $index هم به تابع remove ارسال می‌شود تا بتوان فهمید در سمت کلاینت کدام نظر باید حذف شود. **ng-src** : از این Directive برای مشخص کردن src عکس‌ها استفاده می‌شود. البته در این مثال چندان تفاوتی بین ng-src و src معمولی وجود ندارد. ولی اگر آدرس عکس به صورت Content/{{comment.Name}}.gif می‌بود دیگر وضع فرق می‌کرد. چرا که مرورگر با دیدن آدرس در src سعی به لود کردن آن عکس می‌کند و در این حالت در لود کردن آن عکس با شکست روبرو می‌شود. ng-src سبب می‌شود تا در ابتدا آدرس عکس توسط Angular تفسیر شود و سپس آن عکس توسط مرورگر لود شود. **{{comment.Name}}** : آکلودهای دوتایی برای انقیاد داده (Data Binding) با view-model استفاده می‌شود. این نوع اقیاد داده در مثال‌های قبلی مورد بررسی قرار گرفته است و نکته‌ی بیشتری در اینجا مطرح نیست. **ng-model** : به کمک ng-model می‌توان بین متن داخل textbox و خاصیت شی مورد نظر انقیاد داده بر قرار کرد و هر دو طرف از تغییرات یکدیگر آگاه شوند. به این عمل انقیاد داده دوطرفه (Two-Way Data-Binding) می‌گویند.برای مثال textbox مربوط به نام را به comment.Name و textbox مربوط به email را به comment.Email مقید(bind) شده است. هر تغییری که در محتوای هر کدام از طرفین صورت گیرد دیگری نیز از آن تغییر با خبر شده و آن را نمایش می‌دهد.

تا به اینجای کار قالب مربوط به HTML را بررسی کردیم. حال به سراغ کدهای جاوا اسکریپت می‌رویم:

```
var app = angular.module('myApp', []);

app.controller('CommentCtrl', function ($scope, $http) {
    $scope.comment = {};
    $http.get('/Comment/GetAll').success(function (data) {
        $scope.comments = data;
    })
    $scope.addComment = function () {
        $http.post("/Comment/Add", $scope.comment).success(function () {
            $scope.comments.push({ Name: $scope.comment.Name, CommentBody: $scope.comment.CommentBody
});
            $scope.comment = {};
        });
    };
    $scope.remove = function (id, index) {
        $http.post("/Comment/Remove", { id: id }).success(function () {
            $scope.comments.splice(index, 1);
        });
    };
});
```

در تعریف ng-app اگر به یاد داشته باشید برای آن مقدار myApp در نظر گرفته شده بود. در اینجا هم ما به کمک متغیر سراسری angular که توسط خود کتابخانه تعریف شده است، ماژولی به نام myApp را تعریف کرده ایم. پارامتر دوم را فعلا توضیح نمی‌دهم، ولی در این حد بدانید که برای تعریف وابستگی‌های این ماژول استفاده می‌شود که من آن را برابر یک آرایه خالی قرار داده ام.
در سطر بعد برای ماژول تعریف شده یک controller تعریف کرده ام. شاید دفعه‌ی اول است که تعریف کنترلر به این شکل را مشاهده می‌کنید. اما چرا به این شکل کنترلر تعریف شده و به مانند قبل به شکل تابع سازنده‌ی کلاس تعریف نشده است؟
پاسخ این است که اکثر برنامه نویسان از جمله خودم دل خوشی از متغیر سراسری ندارند. در شکل قبلی تعریف کنترلر، کنترلر به شکل یک متغیر سراسری تعریف می‌شد. اما استفاده از ماژول برای تعریف کنترلر سبب می‌شود تا کنترلرهای ما روی هوا تعریف نشده باشند و هر یک در جای مناسب خود باشند. به این شکل مدیریت کدهای برنامه نیز ساده‌تر بود. مثلا اگر کسی از شما بپرسد که فلان کنترلر کجا تعریف شده است؛ به راحتی می‌گویید که در فلان ماژول برنامه تعریف و مدیریت شده است.
در تابعی که به عنوان کنترلر تعریف شده است، دو پارامتر به عنوان وابستگی درخواست شده است. scope $ که برای ارتباط با view-model و انقیاد داده به کار می‌رود و http $ که برای ارتباط با سرور به کار می‌رود. نمونه‌ی مناسب هر دوی این پارامترها توسط سیستم تزریق وابستگی تو کار angular در اختیار کنترلر قرار می‌گیرد.
قبلا چگونگی استفاده از scope $ برای اعمال انقیاد داده توضیح داده شده است. نکته‌ی جدیدی که مطرح است چگونگی استفاده از سرویس $http برای ارتباط با سرور است. سرویس http $ دارای 4 متد get ، post ، put و delete است.
واقعا استفاده از این سرویس کاملا واضح و روشن است. در متد addComment وقتی که دیدگاه مورد نظر اضافه شد، به آرایه‌ی کامنت‌ها یک کامنت جدید می‌افزاییم و چون انقیاد داده دو طرفه است، بالافاصله دیدگاه جدید نیز در view به نمایش در می‌آید.کار تابع remove هم بسیار ساده است. با استفاده از index ارسالی، دیدگاه مورد نظر را از آرایه‌ی کامنت‌ها حذف می‌کنیم و ادامه‌ی کار توسط انقیاد داده دو طرفه انجام می‌شود. همان طور که مشاهده می‌شود مفاهیم انقیاد داده دو طرفه و تزریق وابستگی خودکار سرویس‌های مورد نیاز، کار با angularjs را بسیار ساده و راحت کرده است. اصولا در بسیاری از موارد احتیاجی به باز اختراع چرخ نیست و کتابخانه‌ی angular آن را برای ما از قبل تدارک دیده است.
کدهای این مثال ضمیمه شده است. این کدها در Visual Studio 2013 و به کمک ASP.NET MVC 5 و Entity Framework 6 نوشته شده است. سعی شده تا مثال نوشته شده به واقعیت نزدیک باشد. اگر دقت کنید مدل کامنت در مثالی که نوشتم به گونه ای است که دیدگاه‌های چند سطحی به همراه پاسخ هایش مد نظر بوده است. به عنوان تمرین نمایش درختی این گونه دیدگاه‌ها را به کمک Angular انجام دهید. کافیست Treeview in Angular را جست و جو کنید؛ مطمئنا به نتایج زیادی می‌رسید. گرچه در مثال ضمیمه شده اگر جست و جو کنید من پیاده سازیش را انجام دادم. هدف از جست و جو در اینترنت مشاهده این است که بیشتر مسائل در Angular از پیش توسط دیگران حل شده است و احتیاجی نیست که شما با چالش‌های جدیدی دست و پنجه نرم کنید.
پس به عنوان تمرین، دیدگاه‌های چند سطحی به همراه پاسخ که نمونه اش را در همین سایتی که درحال مشاهده آن هستید می‌بینید را به کمک AngularJS پیاده سازی کنید.
در مقاله‌ی بعدی چگونگی انتقال منطق تجاری برنامه از کنترلر به لایه سرویس و چگونگی تعریف سرویس جدید را مورد بررسی قرار می‌دهم. AngularSample1.rar

نظرات خوانندگان

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۸/۲۳ ۱۸:۴۹

ممنون از مطلبتون. یک سوال کوچک :
شما با استفاده از کتابخانه Newtonsoft.Json لیست خودتون رو سریالایز کردید و بعد بازگشت دادید :

```
var comments = _db.Comments.Include(x => x.Children).ToList().Where(x => x.Parent == null).ToList();

var result = JsonConvert.SerializeObject(comments, Formatting.Indented,
        new JsonSerializerSettings
        {
              ReferenceLoopHandling = ReferenceLoopHandling.Ignore,
        });

return Content(result);
```

در حالی که با این روش هم میشه پاسخ داد هر چند در شی‌های تو در تو ابتدا باید فیلدها رو مشخص کنیم :

```
var comments = _db.Comments.Include(x => x.Children).ToList().Where(x => x.Parent == null).ToList();
return Json(comments , JsonRequestBehavior.AllowGet);
```

تفاوت این دو آیا در حجم مقدار دیتای ارسالی تفاوتی ندارند؟
طول محتوا با روش شما برای من 1278 و زمان 584ms و در روش دوم طول محتوا به 760 و زمان 135ms کاهش پیدا کرد. در مثالی دیگر منتها با همین دو روش بالا. البته اینها ربطی به مطلب شما نداشت.فقط میخواستم بدونم تفاوت این دو غیر از طول محتوا و زمان ،با همدیگه چیه؟
بیصبرانه منتظر مطلب بعدی شما هستم.

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۸/۲۳ ۱۹:۲۷

علت استفاده من از Newtonsoft.Json توانایی سریالایز کردن اشیا تودرتو است. فکر کنم با کمک یک خاصیت به نام IgnoreJsonConvert بتوانیم بگیم که چه خواصی را سریالایز نکند. همچنین این کتابخانه خیلی توانایی دیگر هم دارد. یک نمونش که می‌تونه مفید باشه اینه که نام خواص اشیا را به صورت استاندارد camelCase سریالایز کنه. همچنین از نظر سرعت هم نسبت به نمونه‌ی توکار برتری قابل توجهی داره.

نویسنده: زمان
تاریخ: ۱۳۹۲/۰۸/۲۳ ۱۹:۳۶

با سلام. تشکر بابت مقاله و یک سوال.
شما عملیات CRUD رو در کلاس کنترلر تعریف کردید. در برخی مقالات دیدم که اینها توسط کنترلرهای WebApi مدیریت میشوند. آیا تفاوتی از لحاظ کارایی بین این دو روش وجود دارد یا شما ترجیحاً از روش اول استفاده کردید؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۸/۲۳ ۲۰:۱

There is more than one way to skin a cat

On The Coexistence of ASP.NET MVC and WebAPI

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۸/۲۳ ۲۰:۷

شخصا علاقه ای به استفاده از webapi برای دریافت اطلاعات به فرمت json ندارم. webapi محدودیت‌ها و مزیت‌های خاص خودش را دارد که در اینجا کنترلرهای معمولی بدون محدودیتی کار مورد نیاز من را انجام می‌دهند.

نویسنده: آریو
تاریخ: ۱۳۹۲/۱۱/۱۵ ۰:۱۷

با سلام. من طبق آموزش‌های شما که واقعا عالیه پیش رفتم و توی یه کار عملی به یه مشکلی خوردم.

فایل اسکریپت من اینه :

```
var saman = angular.module('SamanApplication', []);

saman.service('loginService', ['$http', function (http) {
    var loginData = [];
    this.login = function () {

        http.get('Saman/LogOn/IsLogedIn').success(function (data, status, headers, config) {
            loginData = data;
        });
        return loginData;
    };
}]);

        saman.controller('loginController', function ($scope, loginService) {
            $scope.response = [];
            $scope.click = function () { $scope.response = loginService.login() };
        });
```

فایل html هم :

```
<body ng-app="SamanApplication">
    <div ng-controller="loginController">
        <button ng-click="click()">Test</button>
        {{data}}
    </div>
</body>
```

اما مشکل اینجاست که بار اول که کلیک میکنم اطلاعات از سرور میاد ولی در {{data}} نمایش داده نمیشه. اما بار دوم که کلیک میکنم نمایش داده میشه !
امکانش هست راهنمایی کنید ؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۱۵ ۱:۱۲

http.get به صورت async اجرا میشه و نه synchronous. یعنی زمانیکه فراخوانی شد، سطر return بعدی اجرا میشه و صبر نمی‌کنه تا به اینجا برسه. بهتره از promises استفاده کنید (راه حل استانداردش): ^ و ^

نویسنده: محمد
تاریخ: ۱۳۹۲/۱۱/۲۳ ۱۱:۶

سلام؛ من از ngResource دارم برای گرفتن اطلاعات از سرور استفاده میکنم. آیا استفاده از این ماژول به جای $http مشکلی به وجود نمیاره ؟ کلا میشه در مورد ngResource بیشتر توضیح بدین ؟

نویسنده: سعید رضایی
تاریخ: ۱۳۹۲/۱۲/۲۱ ۱۲:۱۰

با عرض سلام من از آموزشتون استفاده کردم وقتی دیتا رو ذخیره می‌کنم مقادیر به صورت null ذخیره میشه تو دیتابیس اینم کدم:

html

```
<div ng-app="myApp" id="ng-app">



<div ng-controller="MenuCtrl" style="width:300px">

  <div style="height:200px;overflow:auto;">
  <div ng-repeat="menu in menu" >
<div style="float:right;cursor:pointer;" ng-click="remove(menu.ID,$index);">X</div>
<a href="#">
<img style="width:32px;" ng-src="/Content/user.gif" alt="{{menu.Title}}">
</a>
<div>
<h4>{{menu.Title}}</h4>
{{menu.Url}}
</div>
  </div>
</div>

  <form action="/Menu/Add" method="post">
<div>
<label for="Title">عنوان</label>
<input id="Title" type="text" name="Title" ng-model="menu.Title" placeholder="عنوان" />
</div>
<div>
<label for="Url">آدرس</label>
<input id="Url" type="text" name="Url" ng-model="menu.Url" placeholder="آدرس" />
</div>
<div>
<label for="ParentID">والد</label>
<input id="ParentID" type="text" name="ParentID" ng-model="menu.ParentID" placeholder="والد" />
</div>


<button type="button"  ng-click="addmenu()">ذخیره</button>
  </form>
</div>
</div>
```

myapp

```
var app = angular.module('myApp', ['ngAnimate']);

app.controller('MenuCtrl', function ($scope, $http) {

    $scope.menu = {};

    $http.get('/Menu/GetAll').success(function (data) {

        $scope.menu = data;

    })
    $scope.addmenu= function () {

        $http.post("/Menu/Add", $scope.menu).success(function () {

            $scope.menus.push({ Title: $scope.menu.Title, Url: $scope.menu.Url, ParentID:
$scope.menu.ParentID });

            $scope.menu = {};

        });
    };

    $scope.remove = function (ID, index) {

        $http.post("/Menu/Remove", { ID: ID }).success(function () {

            $scope.menu.splice(index, 1);

        });
    };

});
```

controller

```
public class MenuController : Controller
    {
        //
        // GET: /Menu/

        MyContext _db = new MyContext();

        public ActionResult GetAll()
        {
            var menu = _db.Menus.ToList();

            var result = JsonConvert.SerializeObject(menu, Formatting.Indented,
                            new JsonSerializerSettings
                            {
                                ReferenceLoopHandling = ReferenceLoopHandling.Ignore,
                            });

            return Content(result);
        }

        public ActionResult Add(Menu menu)
        {
            _db.Menus.Add(menu);

            _db.SaveChanges();

            return Json("1");
        }

        public ActionResult Remove(int id)
        {
            var selectedMenu = new Menu { ID = id };

            _db.Menus.Attach(selectedMenu);

            _db.Menus.Remove(selectedMenu);

            _db.SaveChanges();

            return Json("1");
        }

        public ActionResult Index()
        {
            return View();
        }

    }
```

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۲/۲۱ ۱۳:۹

دیباگ کنید چه اطلاعات JSON ایی به سرور ارسال میشه؟

نویسنده: سعید رضایی
تاریخ: ۱۳۹۲/۱۲/۲۱ ۱۳:۱۳

null ارسال می‌شه یعنی هیچکدوم از فیلدها مقدار ندارن

نویسنده: علی رضایی
تاریخ: ۱۳۹۳/۰۱/۰۳ ۱۲:۴۳

سلام. با تشکر فراوان برای این آموزش.
در زمان اجرای این برنامه اگر پس از ورود اطلاعات جدید بخواهیم رکوردی را حذف کنیم (قبل از ریفرش)، حذف انجام نمیشود؛ دلیل آن عدم وجود Id نظری است که جدیدا ثبت شده است؛ راه حل آن به شرح ذیل است:

ابتدا در سمت سرور اکشن Add باید به شکل ذیل تغییر یابد:

```
public ActionResult Add(Comment comment)
        {
            _db.Comments.Add(comment);

            _db.SaveChanges();

            return Json(comment.Id);
        }
```

و سپس در سمت کلاینت متد AddComent به شکل ذیل تغییر یابد:

```
$scope.addComment = function () {

        $http.post("/Comment/Add", $scope.comment).success(function (id) {

            $scope.comments.push({Id:id ,Name: $scope.comment.Name, CommentBody:
$scope.comment.CommentBody });

            $scope.comment = {};

        });
    };
```

نویسنده: مهدی
تاریخ: ۱۳۹۳/۰۳/۰۲ ۱۱:۳۷

آموزشهای خیلی عالی هستن. با اینکه من ASP کار نمیکنم ولی این مقوله برای همه زبانهای تحت وب هست. دستتون درد نکنه.
ممنون

نویسنده: علی اسدی
تاریخ: ۱۳۹۳/۰۳/۱۴ ۱۹:۲۸

```
//define
app.service('objUser', function ($http) {

    this.user = [{
        id: null,
        firstName: null,
        lastName: null,
        email: null
    }];

  this.userList = function () {
        var promise = $http.get('api/user')
            .success(function (res) {
                return res;
            });
        return promise;
    };
});

//call
app.controller('UserListCtrl', function ($scope, objUser) {
    $scope.user = objUser.user;
    objUser.userList().then(function (promise) {
        $scope.user = promise.data;
    });
});
```