عنوان: **کنترلرهای تو در تو در angularJs و نحوه ارث بری متدها و property ها در آن - بخش دوم**
نویسنده: مهرداد کاهه
تاریخ: ۱۳۹۴/۰۵/۲۵ ۱۷:۳۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, AngularJS

در بخش قبلی به چگونگی ساخت کنترلرهای تو در تو در AngularJs پرداختیم. همچنین بررسی نمودیم که propertyهای تعریف شده در کنترلر ما چگونه قابل استفاده توسط کنترلر فرزند میباشند.
حال روشی دیگر را برای ارث بری تابع‌ها و propertyها، در کنترلرهای تو در تو معرفی می‌نماییم. لازم به ذکر است که سورس پروژه را می‌توانید از لینک زیر دریافت نمایید: AngularJsNestedController.zip

کد جاوااسکریپت زیر سه کنترلر تو در تو را پیاده سازی می‌کند:

```
var firstControllerObj = function ($scope) {
    // Initialize the model object
    $scope.firstModelObj = {
        firstName: "John"
    };
};

var secondControllerObj = function ($scope) {
    // Initialize the model object
    $scope.secondModelObj = {
        lastName: "Doe"
    };

    // Define utility functions
    $scope.getFullName = function () {
        return $scope.firstModelObj.firstName + " " +
          $scope.secondModelObj.lastName;
    };
};

var thirdControllerObj = function ($scope) {
    // Initialize the model object
    $scope.thirdModelObj = {
        middleName: "Al",
        lastName: "Smith"
    };

    // Define utility functions
    $scope.getFullName = function () {
        return $scope.firstModelObj.firstName + " " +
          $scope.thirdModelObj.middleName + " " +
          $scope.thirdModelObj.lastName;
    };
};
```

کد html زیر هم از کنترلرهای تعریف شده‌ی در بالا، به صورت زیر استفاده کرده است. نگاهی کوتاه و مختصر به این فایل و سپس به شرح فرآیند ارث بری propertyها می‌پردازیم:

```
<div ng-controller="firstControllerObj">
        <h3>First controller</h3>
        <strong>First name:</strong> {{firstModelObj.firstName}}<br />
        <br />
        <label>Set the first name: <input type="text" ng-model="firstModelObj.firstName" /></label><br
/>
        <br />


        <div ng-controller="secondControllerObj">
            <h3>Second controller (inside First)</h3>
            <strong>First name (from First):</strong> {{firstModelObj.firstName}}<br />
            <strong>Last name (from Second):</strong> {{secondModelObj.lastName}}<br />
            <strong>Full name:</strong> {{getFullName()}}<br />
            <br />
            <label>Set the first name: <input type="text" ng-model="firstModelObj.firstName"
/></label><br />
            <label>Set the last name: <input type="text" ng-model="secondModelObj.lastName"
/></label><br />
            <br />
```

```
            <div ng-controller="thirdControllerObj">
                <h3>Third controller (inside Second and First)</h3>
                <strong>First name (from First):</strong> {{firstModelObj.firstName}}<br />
                <strong>Middle name (from Third):</strong> {{thirdModelObj.middleName}}<br />
                <strong>Last name (from Second):</strong> {{secondModelObj.lastName}}<br />
                <strong>Last name (from Third):</strong> {{thirdModelObj.lastName}}<br />
                <strong>Full name (redefined in Third):</strong> {{getFullName()}}<br />
                <br />
                <label>Set the first name: <input type="text" ng-model="firstModelObj.firstName"
/></label><br />
                <label>Set the middle name: <input type="text" ng-model="thirdModelObj.middleName"
/></label><br />
                <label>Set the last name: <input type="text" ng-model="thirdModelObj.lastName"
/></label>
            </div>
        </div>
    </div>
```

اگر نگاهی به کنترلرهای فوق بیاندازیم می‌بینیم که در firstControllerObj تنها یک شیء درون scope قرار داده شده که firstModelObj نامیده شده است و این شی تنها دارای یک property با نام firstName است. در کنترلر دوم نیز یک شیء جدید با نام secondModelObj ساخته شده است؛ با این تفاوت که property آن lastName نام دارد. حال به بدنه‌ی فایل html دو کنترلر توجه کنیم. می‌بینیم که در کنترلر دوم مستقیما پراپرتی firstName از شیء درون کلاس اول فراخوانی شده است. با اجرای پروژه‌ی نمونه خواهیم دید که با تغییر input شامل firstName در کنترلر دوم، این مدل در کنترلر اول نیز تغییر خواهد کرد.
بنابراین ما می‌توانیم در کنترلر فرزند به objectهای درون کنترلر والد دسترسی داشته باشیم و مقادیر آنرا تغییر دهیم. باید توجه داشت که این نوع تعریف مدل با آنچه که در بخش اول گفته شد، تفاوتی اساسی دارد. در بخش قبل با تغییر مدل توسط کنترلر فرزند، آن مدل در کنترلر والد تغییری نمی‌کرد؛ اما در روشی که در این مقاله شرح داده شد مشاهده نمودیم که با تغییر مدل در کنترلر فرزند مقدار مدل در کنترلر والد نیز تغییر می‌کند.