عنوان: **بررسی سرویس های $on و $emit و $broadcast در AngularJs**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۳/۰۲/۱۴ ۲۳:۴۰
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

در این پست درباره به اشتراک گذاری داده‌ها بین کنترلرهای Angular بحث شد. اما استفاده از Factory و Service فقط زمانی کاربرد دارد که بخواهیم یک منبع داده مشخص را در اختیار مصرف کننده قرار دهیم. اگر قصد داشته باشم بر اساس شرایط خاص، داده یا داده‌های مشخصی در سایر کنترلرها تغییر پیدا کنند چه باید کرد؟ به زبان ساده‌تر برای ایجاد ارتباط بین کنترلرها به طوری که از تغییرات یکدیگر باخبر باشند چه راهکارهایی وجود دارد. $on و $emit و $broadcast برای این منظور تعبیه شده اند.
برای شرح موارد بالا بهترین روش بررسی یک مثال است:

:$emit

دو کنترلر به نام‌های FirstCtrl و SecondCtrl داریم. FirstCtrl به عنوان والد کنترلر Second است(در این مورد در این پست توضیح داده شده است).
پس فایل html نیز به صورت زیر خواهد بود:

```
<body ng-app>
  <div ng-controller="FirstCtrl">
    <p>{{title}}</p>
    <div ng-controller="SecondCtrl">
      <button ng-click="onUpdate()">Update First Ctrl Title</button>
    </div>
  </div>
</body>
```

قصد داریم با کلیک بر روی دکمه Update در کنترلر Second، مقدار خاصیت title در FirstCtrl به روز رسانی شود. اگر دقت کرده باشید حرکت رویداد از پایین به بالاست درنتیجه برای این کار باید از سرویس $emit استفاده کنیم. کافیست در کنترلر دوم (Second) کد زیر را وارد نمایید:

```
function ChildCtrl($scope){
  $scope.onUpdate = function(){
    this.$emit("Update_Title", "Good Bye");
  };
}
```

به وسیله سرویس $emit می‌توان از بروز یک رویداد در کنترلر جاری خبر داد. اگر کنترلر والد یک handler برای این رویداد داشته باشد، می‌تواند مقدار جدید را دریافت نماید. برای تعریف handler باید از سرویس $on استفاده کرد. کدهای کنترلر First را به صورت زیر تغییر دهید.

```
function FirstCtrl($scope){
 $scope.title= "Hello";

 $scope.$on("Update_Title", function(event, message){
   $scope.title= message;
 });
}
```

»سرویس $on برای تعریف handler برای رویداد مورد نظر استفاده می‌شود.
»پارامتر دوم سرویس $on برابر با مقدار جدید ارسال شده توسط سرویس $emit است.
»نام رویدادی که به عنوان پارامتر به $on پاس داده می‌شود باید برابر با نام رویداد پاس داده شده به $emit باشد.
»می‌توان چندین پارامتر را با استفاده از $emit ارسال کرد و در سرویس $on با تعریف متغیر به تعداد پارامترها مقادیر آن‌ها را دریافت نمود.

$broadcast

همان طور که مشاهده کردید SecondCtrl در محدوده FirstCtrl تعریف شده است. در نتیجه به راحتی با استفاده از سرویس $emit توانستیم یک رویداد را منتشر نماییم. اما نکته مهم این است که اگر قصد داشته باشیم یک رویداد را از کنترلر والد (در این جا FirstCtrl است) منتشر نماییم به طوری که در کنترلرهای فرزند قابل دریافت باشد(حرکت رویداد بالا به پایین است)، باید از $broadcast استفاده کنیم.

«$broadcast فقط از نظر کاربرد با $emit متفاوت است و در پیاده سازی کاملا مشابه هستند.

یک مثال:

```
function ParentCtrl($scope){
 $scope.foo = "Hello";

 $scope.$on("UPDATE_PARENT", function(event, message){
   $scope.title= message;

   $scope.$broadcast("DO_BIDDING", {
     buttonTitle : "Taken over",
     onButtonClick : function(){
       $scope.title= "HAHA this button no longer works!";
     }
   });
 });
}

function ChildCtrl($scope){
  $scope.buttonTitle = "Update Parent";
  $scope.onButtonClick = function(){
    this.$emit("UPDATE_PARENT", "Updated");
  };

  $scope.$on("DO_BIDDING", function(event, data){
    for(var i in data){
      $scope[i] = data[i];
    }
  });
}
```

مشاهده خروجی مثال

اگر حالت فرزند و والد بین کنترلرها نباشد چه؟

در این حالت باید $rootScope را به کنترلر مورد نظر تزریق نمایید و سپس با استفاده از سرویس $broadcast یا $emit رویدادتان را منتشر کنید. مثال:

```
'use strict';
angular.module('myAppControllers', [])
  .controller('FirstCtrl', function ($rootScope) {

    $rootScope.$broadcast('UPDATE_ALL');

    Or

    $rootScope.$emit('UPDATE_ALL');
});
```

نکته:

از آن جا که حرکت بالا به پایین event bubbling بسیار هزینه برتر است نسبت به حرکت پایین به بالا در نتیجه سعی کنید تا جای ممکن از $rootScope.$broadcast استفاده **نکنید** . در این جا توضیح کاملی درباره دلایل عدم استفاده از $rootScope.$broadcast داده شده است.

هم چنین می‌توانید یک مثال Live را نیز برای مقایسه بین $emit و $broadcast در این جا مشاهده کنید.

Testing in IE 10.0 32-bit on Windows Server 2008 R2 / 7 64-bit		
Test		**Ops/sec**
$broadcast	`window.$rootScope.$broadcast('fooHappened');`	26,406 ±9.58% 90% slower
$emit	`window.$rootScope.$emit('fooHappened');`	250,522 ±7.21% fastest