**深入理解 AngularJS 的 Scope**

2013-07-28 18:49

[原文](http://www.lovelucy.info/understanding-scopes-in-angularjs.html?variant=zh) [简体](http://www.lovelucy.info/understanding-scopes-in-angularjs.html?variant=zh-hans) [繁體](http://www.lovelucy.info/understanding-scopes-in-angularjs.html?variant=zh-hant)

21,899 次圍觀

JavaScript 的原型繼承就是奇葩。

之前在 V2EX 上看到討論說，不會 OOP 的 JavaScript 的程序員就是野生程序員。看來我是屬於野生的。

**一、遇到的問題**

問題發生在使用 AngularJS 嵌套 Controller 的時候。因為每個 Controller 都有它對應的 Scope（相當於作用域、控制範圍），所以 Controller 的嵌套，也就意味着 Scope 的嵌套。這個時候如果兩個 Scope 內都有同名的 Model 會發生什麼呢？從子 Scope 怎樣更新父 Scope 里的 Model 呢？

這個問題很典型，比方說當前頁面是一個產品列表，那麼就需要定義一個 ProductListController

|  |
| --- |
| **function** ProductListController($scope, $http) {  $http.**get**('/api/products.json')  .success(**function**(data){  $scope.productList = data;  });  $scope.selectedProduct = {};  } |

你大概看到了在 Scope 里還定義了一個 selectedProduct 的 Model，表示選中了某一個產品。這時會獲取該產品詳情，而頁面通過 AngularJS 中的 $routeProvider 自動更新，拉取新的詳情頁模板，模板中有一個 ProductDetailController

|  |
| --- |
| **function** ProductDetailController($scope, $http, $routeParams) {  $http.**get**('/api/products/'+$routeParams.productId+'.json')  .success(**function**(data){  $scope.selectedProduct = data;  });  } |

有趣的事情發生了，在這裡也有一個 selectedProduct ，它會怎樣影響 ProductListController 中的 selectedProduct 呢？

答案是沒有影響。在 AnuglarJS 里子 Scope 確實會繼承父 Scope 中的對象，但當你試下對基本數據類型（string, number, boolean）的 **雙向數據綁定** 時，就會發現一些奇怪的行為，繼承並不像你想象的那樣工作。子 Scope 的屬性隱藏（覆蓋）了父 Scope 中的同名屬性，對子 Scope 屬性（表單元素）的更改並不更新父 Scope 屬性的值。這個行為實際上不是 AngularJS 特有的，JavaScript 本身的原型鏈就是這樣工作的。開發者通常都沒有意識到 ng-repeat, ng-switch, ng-view 和 ng-include 統統都創建了他們新的子 scopes，所以在用到這些 directive 時也經常出問題。

**二、解決的辦法**

解決的辦法就是不使用基本數據類型，而在 Model 里永遠多加一個點.

|  |
| --- |
| 使用  <input type="text" ng-model="someObj.prop1">  來替代  <input type="text" ng-model="prop1"> |

是不是很坑爹？下面這個例子很明確地表達了我所想表達的奇葩現象

|  |
| --- |
| app.controller('ParentController',function($scope){  $scope.parentPrimitive = "some primitive"  $scope.parentObj = {};  $scope.parentObj.parentProperty = "some value";  });  app.controller('ChildController',function($scope){  $scope.parentPrimitive = "this will NOT modify the parent"  $scope.parentObj.parentProperty = "this WILL modify the parent";  }); |

查看 [在線演示 DEMO](http://jsfiddle.net/xxNxj/5/)

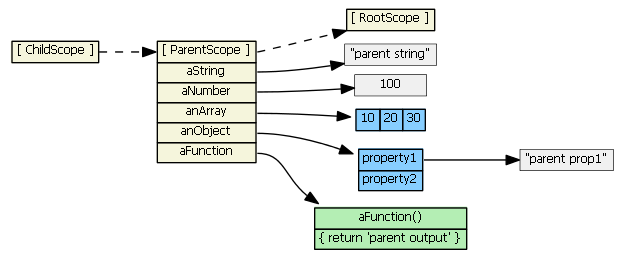
但是我真的確實十分很非常需要使用 string number 等原始數據類型怎麼辦呢？2 個方法——

1. 在子 Scope 中使用 $parent.parentPrimitive。 這將阻止子 Scope 創建它自己的屬性。
2. 在父 Scope 中定義一個函數，讓子 Scope 調用，傳遞原始數據類型的參數給父親，從而更新父 Scope 中的屬性。（並不總是可行）

**三、JavaScript 的原型鏈繼承**

吐槽完畢，我們來深入了解一下 JavaScript 的原型鏈。這很重要，特別是當你從服務器端開發轉到前端，你應該會很熟悉經典的 Class 類繼承，我們來回顧一下。

假設父類 parentScope 有如下成員屬性 aString, aNumber, anArray, anObject, 以及 aFunction。子類 childScope 原型繼承父類 parentScope，於是我們有：

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance1.png)

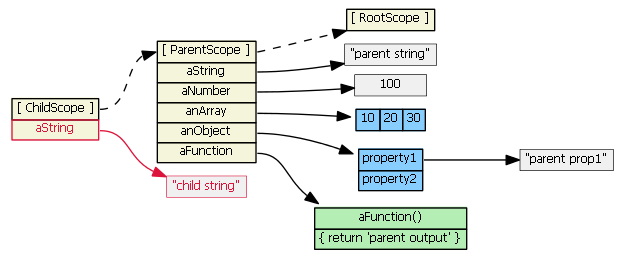
如果子 Scope 嘗試去訪問 parentScope 中定義的屬性，JavaScript 會先在子 Scope 中查找，如果沒有該屬性，則找它繼承的 scope 去獲取屬性，如果繼承的原型對象 parentScope 中都沒有該屬性，那麼繼續在它的原型中尋找，從原型鏈一直往上直到到達 rootScope。所以，下面的表達式結果都是 ture：

|  |
| --- |
| childScope.aString === 'parent string'  childScope.anArray[1] === 20  childScope.anObject.property1 === 'parent prop1'  childScope.aFunction() === 'parent output' |

假設我們執行下面的語句

|  |
| --- |
| childScope.aString = 'child string' |

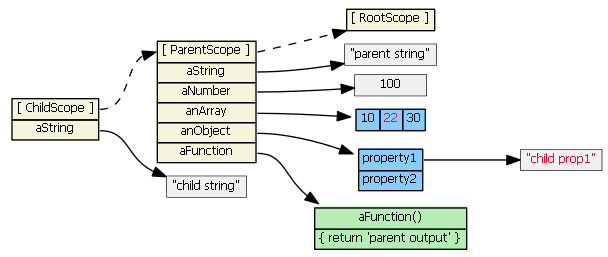
原型鏈並沒有被查詢，反而是在 childScope 中增加了一個新屬性 aString。這個新屬性隱藏（覆蓋）了 parentScope 中的同名屬性。在下面我們討論 ng-repeat 和 ng-include 時這個概念很重要。

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance2.png)

假設我們執行這個操作：

|  |
| --- |
| childScope.anArray[1] = '22'  childScope.anObject.property1 = 'child prop1' |

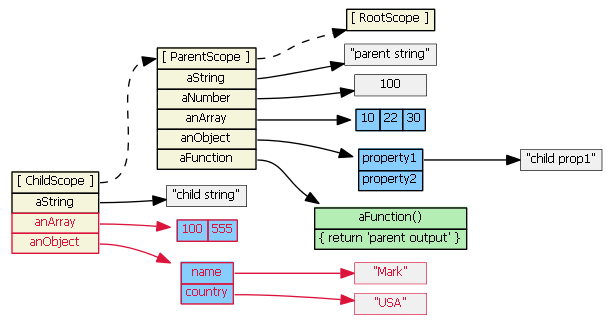
原型鏈被查詢了，因為對象 anArray 和 anObject 在 childScope 中沒有找到。它們在 parentScope 中被找到了，並且值被更新。childScope 中沒有增加新的屬性，也沒有任何新的對象被創建。（註：在 JavaScript 中，array 和 function 都是對象）

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance3.png)

假設我們執行這個操作：

|  |
| --- |
| childScope.anArray = [100, 555]  childScope.anObject = { name: 'Mark', country: 'USA' } |

原型鏈沒有被查詢，並且子 Scope 新加入了兩個新的對象屬性，它們隱藏（覆蓋）了 parentScope 中的同名對象屬性。

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance4.png)

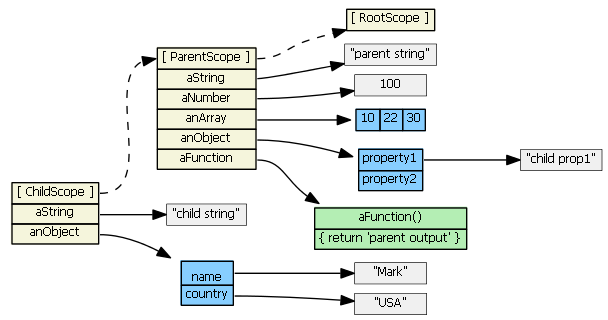
應該可以總結

* 如果讀取 childScope.propertyX，並且 childScope 有屬性 propertyX，那麼原型鏈沒有被查詢。
* 如果設置 childScope.propertyX，原型鏈不會被查詢。

最後一種情況，

|  |
| --- |
| **delete** childScope.anArray  childScope.anArray[1] === 22 *// true* |

我們從 childScope 刪除了屬性，則當我們再次訪問該屬性時，原型鏈會被查詢。刪除對象的屬性會讓來自原型鏈中的屬性浮現出來。

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance5.png)

**四、AngularJS 的 Scope 繼承**

* 創建新的 Scope，並且原型繼承：ng-repeat, ng-include, ng-switch, ng-view, ng-controller, directive with scope: true, directive with transclude: true
* 創建新的 Scope，但不繼承：directive with scope: { ... }。它會創建一個獨立 Scope。

註：默認情況下 directive 不創建新 Scope，即默認參數是 scope: false。

**ng-include**

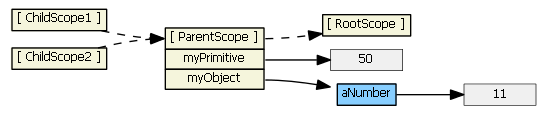
假設在我們的 controller 中，

|  |
| --- |
| $scope.myPrimitive = 50;  $scope.myObject = {aNumber: 11}; |

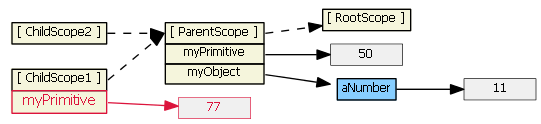
HTML 為：

|  |
| --- |
| <**script** type="text/ng-template" id="/tpl1.html">  <**input** ng-model="myPrimitive">  </**script**>  <**div** ng-include src="'/tpl1.html'"></**div**>    <**script** type="text/ng-template" id="/tpl2.html">  <**input** ng-model="myObject.aNumber">  </**script**>  <**div** ng-include src="'/tpl2.html'"></**div**> |

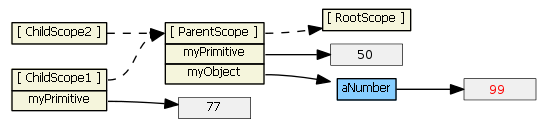
每一個 ng-include 會生成一個子 Scope，每個子 Scope 都繼承父 Scope。

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance6.png)

輸入（比如”77″）到第一個 input 文本框，則子 Scope 將獲得一個新的 myPrimitive 屬性，覆蓋掉父 Scope 的同名屬性。這可能和你預想的不一樣。

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance7.png)

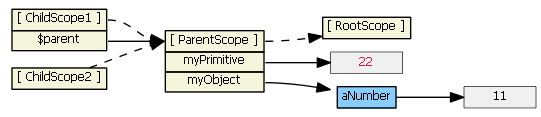
輸入（比如”99″）到第二個 input 文本框，並不會在子 Scope 創建新的屬性，因為 tpl2.html 將 model 綁定到了一個對象屬性（an object property），原型繼承在這時發揮了作用，ngModel 尋找對象 myObject 並且在它的父 Scope 中找到了。

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance8.png)

如果我們不想把 model 從 number 基礎類型改為對象，我們可以用 $parent 改寫第一個模板：

|  |
| --- |
| <**input** ng-model="$parent.myPrimitive"> |

輸入（比如”22″）到這個文本框也不會創建新屬性了。model 被綁定到了父 scope 的屬性上（因為 $parent 是子 Scope 指向它的父 Scope 的一個屬性）。

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance9.png)

對於所有的 scope （原型繼承的或者非繼承的），Angular 總是會通過 Scope 的 $parent, $$childHead 和 $$childTail 屬性記錄父-子關係（也就是繼承關係），圖中為簡化而未畫出這些屬性。

在沒有表單元素的情況下，另一種方法是在父 Scope 中定義一個函數來修改基本數據類型。因為有原型繼承，子 Scope 確保能夠調用這個函數。例如，

|  |
| --- |
| *// 父 Scope 中*  $scope.setMyPrimitive = **function**(value) {  $scope.myPrimitive = value;  } |

[查看 DEMO](http://jsfiddle.net/mrajcok/jNxyE/)。參考 [StackOverflow](http://stackoverflow.com/questions/12977894/what-is-the-angularjs-way-to-databind-many-inputs/13782671#13782671)。

**ng-switch**

ng-switch 的原型繼承和 ng-include 一樣。所以如果你需要對基本類型數據進行雙向綁定，使用 $parent，或者將其改為 object 對象並綁定到對象的屬性，防止子 Scope 覆蓋父 Scope 的屬性。

參考 [AngularJS, bind scope of a switch-case?](http://stackoverflow.com/questions/12405005/angularjs-bind-scope-of-a-switch-case/12414410)

**ng-repeat**

ng-repeat 有一點不一樣。假設在我們的 controller 里：

|  |
| --- |
| $scope.myArrayOfPrimitives = [ 11, 22 ];  $scope.myArrayOfObjects = [{num: 101}, {num: 202}] |

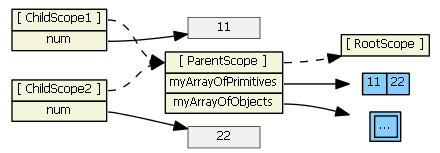
還有 HTML：

|  |
| --- |
| <**ul**>  <**li** ng-repeat="num in myArrayOfPrimitives">  <**input** ng-model="num">  </**li**>  <**ul**>  <**ul**>  <**li** ng-repeat="obj in myArrayOfObjects">  <**input** ng-model="obj.num">  </**li**>  <**ul**> |

對於每一個 Item，ng-repeat 創建新的 Scope，每一個 Scope 都繼承父 Scope，但同時 item 的值也被賦給了新 Scope 的新屬性（新屬性的名字為循環的變量名）。Angular ng-repeat 的源碼實際上是這樣的：

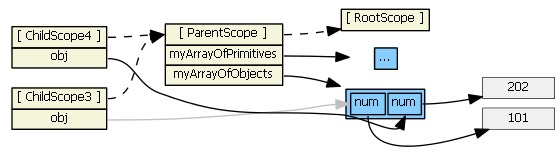
|  |
| --- |
| childScope = scope.$new(); *// 子 scope 原型繼承父 scope ...*  childScope[valueIdent] = value; *// 創建新的 childScope 屬性* |

如果 item 是一個基礎數據類型（就像 myArrayOfPrimitives），本質上它的值被複制了一份賦給了新的子 scope 屬性。改變這個子 scope 屬性值（比如用 ng-model，即 num）不會改變父 scope 引用的 array。所以上面第一個 ng-repeat 里每一個子 scope 獲得的 num 屬性獨立於 myArrayOfPrimitives 數組：

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance10.png)

這樣的 ng-repeat 和你預想中的不一樣。在 Angular 1.0.2 及更早的版本，向文本框中輸入會改變灰色格子的值，它們只在子 Scope 中可見。Angular 1.0.3+ 以後，輸入文本不會再有任何作用了。（參考 [StackOverflow 上的解釋](http://stackoverflow.com/a/13723990/215945)）我們希望的是輸入能改變 myArrayOfPrimitives 數組，而不是子 Scope 里的屬性。為此我們必須將 model 改為一個關於對象的數組（array of objects）。

所以如果 item 是一個對象，則對於原對象的一個引用（而非拷貝）被賦給了新的子 Scope 屬性。改變子 Scope 屬性的值（使用 ng-model，即 obj.num）也就改變了父 Scope 所引用的對象。所以上面第二個 ng-repeat 可表示為：

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance11.png)

這才是我們想要的。輸入到文本框即會改變灰色格子的值，該值在父 Scope 和子 Scope 均可見。

參考 [Difficulty with ng-model, ng-repeat, and inputs](http://stackoverflow.com/questions/13714884/difficulty-with-ng-model-ng-repeat-and-inputs) 以及 [ng-repeat and databinding](http://stackoverflow.com/a/13782671/215945)。

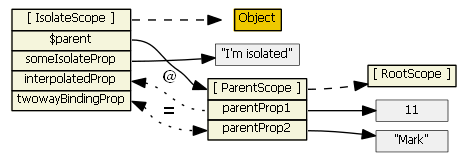
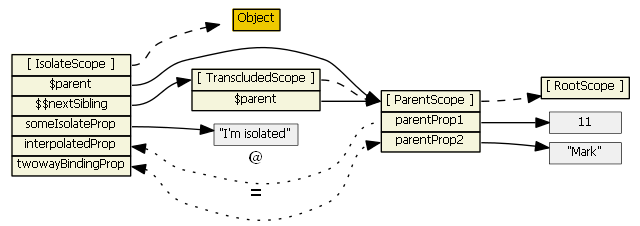
**ng-controller**

使用 ng-controller 進行嵌套，結果和 ng-include 和 ng-switch 一樣是正常的原型繼承。所以做法也一樣不再贅述。然而“兩個 controller 使用 $scope 繼承來共享信息被認為是不好的做法”（來自 [這裡](http://onehungrymind.com/angularjs-sticky-notes-pt-1-architecture/)），應該使用 service 在 controller 間共享數據。

如果你確實要通過繼承來共享數據，那麼也沒什麼特殊要做的，子 Scope 可以直接訪問所有父 Scope 的屬性。參考 [Controller load order differs when loading or navigating](http://stackoverflow.com/questions/13825419/controller-load-order-differs-when-loading-or-navigating/13843771#13843771)。

**directives**

這個要分情況來討論。

1. 默認 scope: false – directive 不會創建新的 Scope，所以沒有原型繼承。這看上去很簡單，但也很危險，因為你會以為 directive 在 Scope 中創建了一個新的屬性，而實際上它只是用到了一個已存在的屬性。這對編寫可復用的模塊和組件來說並不好。
2. scope: true – 這時 directive 會創建一個新的子 scope 並繼承父 scope。如果在同一個 DOM 節點上有多個 directive 都要創建新 scope，則只有一個新 Scope 會創建。因為有正常的原型繼承，所以和 ng-include， ng-switch 一樣要注意基礎類型數據的雙向綁定，子 Scope 屬性會覆蓋父 Scope 同名屬性。
3. scope: { ... } – 這時 directive 創建一個獨立的 scope，沒有原型繼承。這在編寫可復用的模塊和組件時是比較好的選擇，因為 directive 不會不小心讀寫父 scope。然而，有時候這類 directives 又經常需要訪問父 scope 的屬性。對象散列（object hash）被用來建立這個獨立 Scope 與父 Scope 間的雙向綁定（使用 ‘=’）或單向綁定（使用 ‘@’）。還有一個 ‘&’ 用來綁定父 Scope 的表達式。這些統統從父 Scope 派生創建出本地的 Scope 屬性。注意，HTML 屬性被用來建立綁定，你無法在對象散列中引用父 Scope 的屬性名，你必須使用一個 HTML 屬性。例如，<div my-directive> 和 scope: { localProp: '@parentProp' } 是無法綁定父屬性 parentProp 到獨立 scope的，你必須這樣指定： <div my-directive the-Parent-Prop=parentProp> 以及 scope: { localProp: '@theParentProp' }。獨立的 scope 中 \_\_proto\_\_ 引用了一個 Scope 對象（下圖中的桔黃色 Object），獨立 scope 的 $parent 指向父 scope，所以儘管它是獨立的而且沒有從父 Scope 原型繼承，它仍然是一個子 scope。  
     
   下面的圖中，我們有 <my-directive interpolated="{{parentProp1}}" twowayBinding="parentProp2"> 和 scope: { interpolatedProp: '@interpolated', twowayBindingProp: '=twowayBinding' }。  
   同時，假設 directive 在它的 link 函數里做了 scope.someIsolateProp = "I'm isolated"  
     
   [](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance12.png)  
     
   注意：在 link 函數中使用 attrs.$observe('attr\_name', function(value) { ... } 來獲取獨立 Scope 用 ‘@’ 符號替換的屬性值。例如，在 link 函數中有 attrs.$observe('interpolated', function(value) { ... } 值將被設為 11. （scope.interpolatedProp 在 link 函數中是 undefined，相反scope.twowayBindingProp 在 link 函數中定義了，因為用了 ‘=’ 符號）  
   更多參考 <http://onehungrymind.com/angularjs-sticky-notes-pt-2-isolated-scope/>
4. transclude: true – 這時 directive 創建了一個新的 “transcluded” 子 scope，同時繼承父 scope。所以如果模板片段中的內容（例如那些將要替代 ng-transclude 的內容）要求對父 Scope 的基本類型數據進行雙向綁定，使用 $parent，或者將 model 一個對象的屬性，防止子 Scope 屬性覆蓋父 Scope 屬性。  
     
   transcluded 和獨立 scope （如果有）是兄弟關係，每個 Scope 的 $parent 指向同一個父 Scope。當模板中的 scope 和獨立 Scope 同時存在，獨立 Scope 屬性 $$nextSibling 將會指向模板中的 Scope。  
   更多關於 transcluded scope 的信息，參考 [AngularJS two way binding not working in directive with transcluded scope](http://stackoverflow.com/a/14484903/215945)。  
     
     
   在下圖中，假設 directive 和上個圖一樣，只是多了 transclude: true  
     
   [](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance13.png)  
   查看 [在線 DEMO](http://jsfiddle.net/mrajcok/7g3QM/)，例子里有一個 showScope() 函數可以用來檢查獨立 Scope 和它關聯的 transcluded scope。

**總結**

一共有四種 Scope：

1. 普通進行原型繼承的 Scope —— ng-include, ng-switch, ng-controller, directive with scope: true
2. 普通原型繼承的 Scope 但拷貝賦值 —— ng-repeat。 每個 ng-repeat 的循環都創建新的子 Scope，並且子 Scope 總是獲得新的屬性。
3. 獨立的 isolate scope —— directive with scope: {...}。它不是原型繼承，但 ‘=’, ‘@’ 和 ‘&’ 提供了訪問父 Scope 屬性的機制。
4. transcluded scope —— directive with transclude: true。它也遵循原型繼承，但它同時是任何 isolate scope 的兄弟。

對於所有的 Scope，Angular 總是會通過 Scope 的 $parent, $$childHead 和 $$childTail 屬性記錄父-子關係。