## AngularJS 的 Scope 继承

* 创建新的 Scope，并且原型继承：ng-repeat, ng-include, ng-switch, ng-view, ng-controller, directive with scope: true, directive with transclude: true
* 创建新的 Scope，但不继承：directive with scope: { ... }。它会创建一个独立 Scope。

注：默认情况下 directive 不创建新 Scope，即默认参数是 scope: false。

**ng-include**

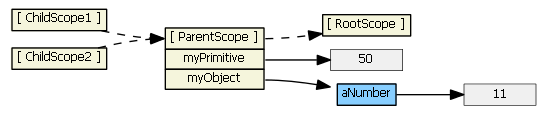
假设在我们的 controller 中，

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

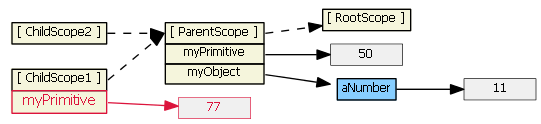
HTML 为：

|  |
| --- |
| <**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**> |

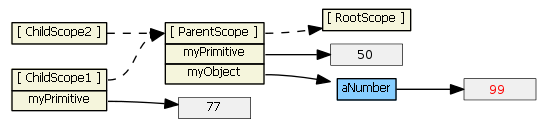
每一个 ng-include 会生成一个子 Scope，每个子 Scope 都继承父 Scope。

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

输入（比如”77″）到第一个 input 文本框，则子 Scope 将获得一个新的 myPrimitive 属性，覆盖掉父 Scope 的同名属性。这可能和你预想的不一样。

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

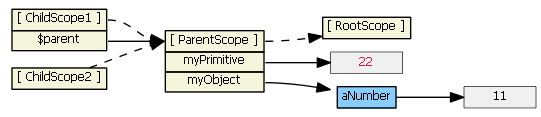
输入（比如”99″）到第二个 input 文本框，并不会在子 Scope 创建新的属性，因为 tpl2.html 将 model 绑定到了一个对象属性（an object property），原型继承在这时发挥了作用，ngModel 寻找对象 myObject 并且在它的父 Scope 中找到了。

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

如果我们不想把 model 从 number 基础类型改为对象，我们可以用 $parent 改写第一个模板：

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

输入（比如”22″）到这个文本框也不会创建新属性了。model 被绑定到了父 scope 的属性上（因为 $parent 是子 Scope 指向它的父 Scope 的一个属性）。

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

对于所有的 scope （原型继承的或者非继承的），Angular 总是会通过 Scope 的 $parent, $$childHead 和 $$childTail 属性记录父-子关系（也就是继承关系），图中为简化而未画出这些属性。

在没有表单元素的情况下，另一种方法是在父 Scope 中定义一个函数来修改基本数据类型。因为有原型继承，子 Scope 确保能够调用这个函数。例如，

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

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

**ng-switch**

ng-switch 的原型继承和 ng-include 一样。所以如果你需要对基本类型数据进行双向绑定，使用 $parent，或者将其改为 object 对象并绑定到对象的属性，防止子 Scope 覆盖父 Scope 的属性。

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

**ng-repeat**

ng-repeat 有一点不一样。假设在我们的 controller 里：

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

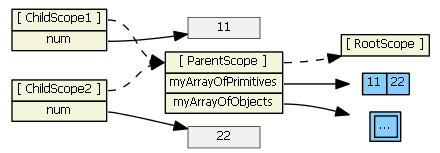
还有 HTML：

|  |
| --- |
| <**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**> |

对于每一个 Item，ng-repeat 创建新的 Scope，每一个 Scope 都继承父 Scope，但同时 item 的值也被赋给了新 Scope 的新属性（新属性的名字为循环的变量名）。Angular ng-repeat 的源码实际上是这样的：

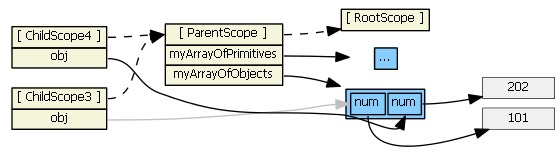
|  |
| --- |
| childScope = scope.$new(); *// 子 scope 原型继承父 scope ...*  childScope[valueIdent] = value; *// 创建新的 childScope 属性* |

如果 item 是一个基础数据类型（就像 myArrayOfPrimitives），本质上它的值被复制了一份赋给了新的子 scope 属性。改变这个子 scope 属性值（比如用 ng-model，即 num）不会改变父 scope 引用的 array。所以上面第一个 ng-repeat 里每一个子 scope 获得的 num 属性独立于 myArrayOfPrimitives 数组：

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

这样的 ng-repeat 和你预想中的不一样。在 Angular 1.0.2 及更早的版本，向文本框中输入会改变灰色格子的值，它们只在子 Scope 中可见。Angular 1.0.3+ 以后，输入文本不会再有任何作用了。（参考 [StackOverflow 上的解释](http://stackoverflow.com/a/13723990/215945)）我们希望的是输入能改变 myArrayOfPrimitives 数组，而不是子 Scope 里的属性。为此我们必须将 model 改为一个关于对象的数组（array of objects）。

所以如果 item 是一个对象，则对于原对象的一个引用（而非拷贝）被赋给了新的子 Scope 属性。改变子 Scope 属性的值（使用 ng-model，即 obj.num）也就改变了父 Scope 所引用的对象。所以上面第二个 ng-repeat 可表示为：

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

这才是我们想要的。输入到文本框即会改变灰色格子的值，该值在父 Scope 和子 Scope 均可见。

参考 [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)。

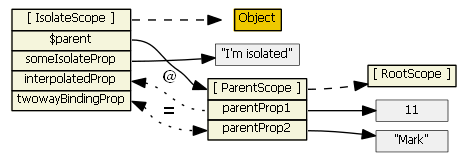
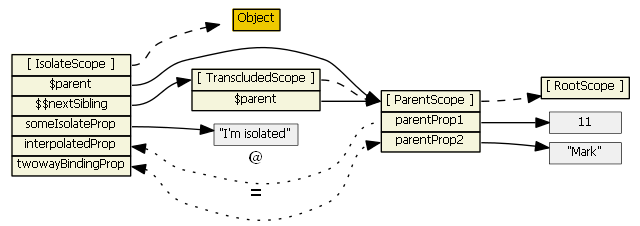
**ng-controller**

使用 ng-controller 进行嵌套，结果和 ng-include 和 ng-switch 一样是正常的原型继承。所以做法也一样不再赘述。然而“两个 controller 使用 $scope 继承来共享信息被认为是不好的做法”（来自 [这里](http://onehungrymind.com/angularjs-sticky-notes-pt-1-architecture/)），应该使用 service 在 controller 间共享数据。

如果你确实要通过继承来共享数据，那么也没什么特殊要做的，子 Scope 可以直接访问所有父 Scope 的属性。参考 [Controller load order differs when loading or navigating](http://stackoverflow.com/questions/13825419/controller-load-order-differs-when-loading-or-navigating/13843771#13843771)。

**directives**

这个要分情况来讨论。

1. 默认 scope: false – directive 不会创建新的 Scope，所以没有原型继承。这看上去很简单，但也很危险，因为你会以为 directive 在 Scope 中创建了一个新的属性，而实际上它只是用到了一个已存在的属性。这对编写可复用的模块和组件来说并不好。
2. scope: true – 这时 directive 会创建一个新的子 scope 并继承父 scope。如果在同一个 DOM 节点上有多个 directive 都要创建新 scope，则只有一个新 Scope 会创建。因为有正常的原型继承，所以和 ng-include， ng-switch 一样要注意基础类型数据的双向绑定，子 Scope 属性会覆盖父 Scope 同名属性。
3. scope: { ... } – 这时 directive 创建一个独立的 scope，没有原型继承。这在编写可复用的模块和组件时是比较好的选择，因为 directive 不会不小心读写父 scope。然而，有时候这类 directives 又经常需要访问父 scope 的属性。对象散列（object hash）被用来建立这个独立 Scope 与父 Scope 间的双向绑定（使用 ‘=’）或单向绑定（使用 ‘@’）。还有一个 ‘&’ 用来绑定父 Scope 的表达式。这些统统从父 Scope 派生创建出本地的 Scope 属性。注意，HTML 属性被用来建立绑定，你无法在对象散列中引用父 Scope 的属性名，你必须使用一个 HTML 属性。例如，<div my-directive> 和 scope: { localProp: '@parentProp' } 是无法绑定父属性 parentProp 到独立 scope的，你必须这样指定： <div my-directive the-Parent-Prop=parentProp> 以及 scope: { localProp: '@theParentProp' }。独立的 scope 中 \_\_proto\_\_ 引用了一个 Scope 对象（下图中的桔黄色 Object），独立 scope 的 $parent 指向父 scope，所以尽管它是独立的而且没有从父 Scope 原型继承，它仍然是一个子 scope。  
     
   下面的图中，我们有 <my-directive interpolated="{{parentProp1}}" twowayBinding="parentProp2"> 和 scope: { interpolatedProp: '@interpolated', twowayBindingProp: '=twowayBinding' }。  
   同时，假设 directive 在它的 link 函数里做了 scope.someIsolateProp = "I'm isolated"  
     
   [](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance12.png)  
     
   注意：在 link 函数中使用 attrs.$observe('attr\_name', function(value) { ... } 来获取独立 Scope 用 ‘@’ 符号替换的属性值。例如，在 link 函数中有 attrs.$observe('interpolated', function(value) { ... } 值将被设为 11. （scope.interpolatedProp 在 link 函数中是 undefined，相反scope.twowayBindingProp 在 link 函数中定义了，因为用了 ‘=’ 符号）  
   更多参考 <http://onehungrymind.com/angularjs-sticky-notes-pt-2-isolated-scope/>
4. transclude: true – 这时 directive 创建了一个新的 “transcluded” 子 scope，同时继承父 scope。所以如果模板片段中的内容（例如那些将要替代 ng-transclude 的内容）要求对父 Scope 的基本类型数据进行双向绑定，使用 $parent，或者将 model 一个对象的属性，防止子 Scope 属性覆盖父 Scope 属性。  
     
   transcluded 和独立 scope （如果有）是兄弟关系，每个 Scope 的 $parent 指向同一个父 Scope。当模板中的 scope 和独立 Scope 同时存在，独立 Scope 属性 $$nextSibling 将会指向模板中的 Scope。  
   更多关于 transcluded scope 的信息，参考 [AngularJS two way binding not working in directive with transcluded scope](http://stackoverflow.com/a/14484903/215945)。  
     
     
   在下图中，假设 directive 和上个图一样，只是多了 transclude: true  
     
   [](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance13.png)  
   查看 [在线 DEMO](http://jsfiddle.net/mrajcok/7g3QM/)，例子里有一个 showScope() 函数可以用来检查独立 Scope 和它关联的 transcluded scope。

**总结**

一共有四种 Scope：

1. 普通进行原型继承的 Scope —— ng-include, ng-switch, ng-controller, directive with scope: true
2. 普通原型继承的 Scope 但拷贝赋值 —— ng-repeat。 每个 ng-repeat 的循环都创建新的子 Scope，并且子 Scope 总是获得新的属性。
3. 独立的 isolate scope —— directive with scope: {...}。它不是原型继承，但 ‘=’, ‘@’ 和 ‘&’ 提供了访问父 Scope 属性的机制。
4. transcluded scope —— directive with transclude: true。它也遵循原型继承，但它同时是任何 isolate scope 的兄弟。

对于所有的 Scope，Angular 总是会通过 Scope 的 $parent, $$childHead 和 $$childTail 属性记录父-子关系。