## AngularJS 作用域Scope的继承

\*\*\*\*\*\***会创建子作用域的几个指令：**

**ng-include**

**ng-swtich**

**ng-view**

**ng-repeat**

**ng-controller**

**用scope:true和transclude：true创建directive**

### 提示:

* **以下方式会创建新的子作用域，并且进行原型继承： ng-repeat、ng-include、ng-switch、ng-view、ng-controller, 用scope: true和transclude: true创建directive。**
* 以下方式会创建新的独立作用域，不会进行原型继承：用scope: { ... }创建directive。这样创建的作用域被称为"Isolate"作用域。

**注意：**默认情况下创建directive使用了scope: false，不会创建子作用域。

进行原型继承即意味着父作用域在子作用域的原型链上，这是JavaScript的特性。AngularJS的作用域还存在如下内部定义的关系：

* **scope.$parent指向scope的父作用域；**
* **scope.$$childHead指向scope的第一个子作用域；**
* **scope.$$childTail指向scope的最后一个子作用域；**
* **scope.$$nextSibling指向scope的下一个相邻作用域；**
* **scope.$$prevSibling指向scope的上一个相邻作用域；**

这些关系用于AngularJS内部历遍，如$broadcast和$emit事件广播，$digest处理等。

### ng-include

In controller:

$scope.myPrimitive = 50;

$scope.myObject = {aNumber: 11};

In HTML:

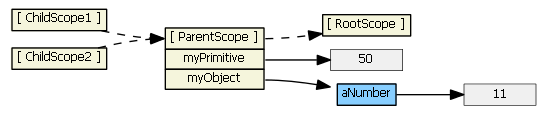
<script type="text/ng-template" id="/tpl1.html">

<input ng-model="myPrimitive"></script><div ng-include src="'/tpl1.html'"></div>

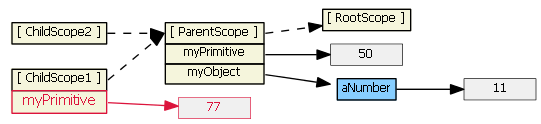
<script type="text/ng-template" id="/tpl2.html">

<input ng-model="myObject.aNumber"></script><div ng-include src="'/tpl2.html'"></div>

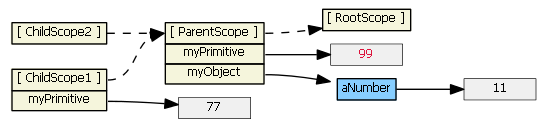
每一个ng-include指令都创建一个子作用域, 并且会从父作用域进行原型继承。



在第一个input框输入"77"将会导致子作用域中新建一个同名属性，其值为77，这不是你想要的结果。



在第二个input框输入"99"会直接修改父作用域的myObject对象，这就是JavaScript原型继承机制的作用。

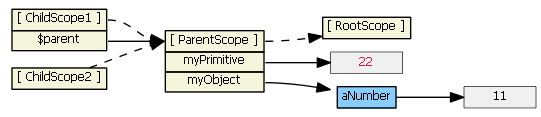


(**注：**上图存在错误，红色99因为是50，11应该是99)

如果我们不想把model由原始类型改成引用类型——对象，我们也可以使用$parent直接操作父作用域：

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

输入"22"我们得到了想要的结果。



另一种方法就是使用函数，在父作用域定义函数，子作用域通过原型继承可运行该函数：

*// in the parent scope*

$scope.setMyPrimitive = function(value) {

$scope.myPrimitive = value;}

请参考：

[sample fiddle](http://jsfiddle.net/mrajcok/jNxyE/" \t "http://blog.csdn.net/whisper_a/article/details/_blank) that uses this "parent function" approach. (This was part of a[Stack Overflow post](http://stackoverflow.com/a/14104318/215945" \t "http://blog.csdn.net/whisper_a/article/details/_blank).)

[http://stackoverflow.com/a/13782671/215945](http://stackoverflow.com/a/13782671/215945" \t "http://blog.csdn.net/whisper_a/article/details/_blank)

[https://github.com/angular/angular.js/issues/1267](https://github.com/angular/angular.js/issues/1267" \t "http://blog.csdn.net/whisper_a/article/details/_blank).

### ng-switch

ng-switch与ng-include一样。

参考： [AngularJS, bind scope of a switch-case?](http://stackoverflow.com/questions/12405005/angularjs-bind-scope-of-a-switch-case/12414410" \t "http://blog.csdn.net/whisper_a/article/details/_blank)

### ng-view

ng-view与ng-include一样。

### ng-repeat

Ng-repeat也创建子作用域，但有些不同。

In controller:

$scope.myArrayOfPrimitives = [ 11, 22 ];

$scope.myArrayOfObjects = [{num: 101}, {num: 202}]

In 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">

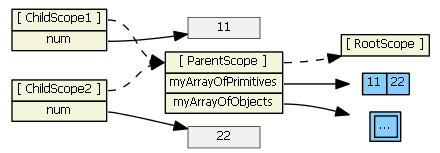
</li></ul>

ng-repeat对每一个迭代项Item都会创建子作用域, 子作用域也从父作用域进行原型继承。 **但它还是会在子作用域中新建同名属性，把Item赋值给对应的子作用域的同名属性**。 下面是AngularJS中ng-repeat的部分源代码：

childScope = scope.$new(); *// child scope prototypically inherits from parent scope ...*

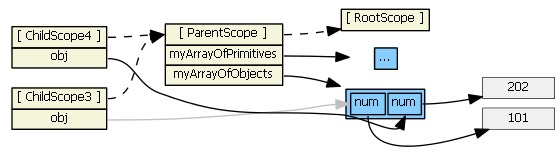
childScope[valueIdent] = value; *// creates a new childScope property*

如果Item是原始类型(如myArrayOfPrimitives的11、22), 那么子作用域中有一个新属性（如num），它是Item的副本（11、22）. 修改子作用域num的值将**不会**改变父作用域myArrayOfPrimitives，所以在上一个ng-repeat，每一个子作用域都有一个num 属性，该属性与myArrayOfPrimitives无关联：



显然这不会是你想要的结果。我们需要的是在子作用域中修改了值后反映到myArrayOfPrimitives数组。我们需要使用引用类型的Item，如上面第二个ng-repeat所示。

myArrayOfObjects的每一项Item都是一个对象——引用类型，ng-repeat对每一个Item创建子作用域，并在子作用域新建obj属性，obj属性就是该Item的一个引用，而不是副本。



我们修改子作用域的obj.num就是修改了myArrayOfObjects。这才是我们想要的结果。

参考：

[Difficulty with ng-model, ng-repeat, and inputs](http://stackoverflow.com/questions/13714884/difficulty-with-ng-model-ng-repeat-and-inputs" \t "http://blog.csdn.net/whisper_a/article/details/_blank)

[ng-repeat and databinding](http://stackoverflow.com/a/13782671/215945" \t "http://blog.csdn.net/whisper_a/article/details/_blank)

### ng-controller

使用ng-controller与ng-include一样也是创建子作用域，会从父级controller创建的作用域进行原型继承。但是，利用原型继承来使父子controller共享数据是一个糟糕的办法。 ["it is considered bad form for two controllers to share information via $scope inheritance"](http://onehungrymind.com/angularjs-sticky-notes-pt-1-architecture/" \t "http://blog.csdn.net/whisper_a/article/details/_blank)，controllers之间应该使用 service进行数据共享。

(如果一定要利用原型继承来进行父子controllers之间数据共享，也可以直接使用。 请参考： [Controller load order differs when loading or navigating](http://stackoverflow.com/questions/13825419/controller-load-order-differs-when-loading-or-navigating/13843771" \l "13843771" \t "http://blog.csdn.net/whisper_a/article/details/_blank))

### directives

1. 默认 (scope: false) - directive使用原有作用域，所以也不存在原型继承，这种方式很简单，但也很容易出问题——除非该directive与html不存在数据绑定，否则一般情况建议使用第2条方式。
2. scope: true - directive创建一个子作用域, 并且会从父作用域进行原型继承。 如果同一个DOM element存在多个directives要求创建子作用域，那么只有一个子作用域被创建，directives共用该子作用域。

scope: { ... } - directive创建一个独立的“Isolate”作用域，没有原型继承。这是创建可复用directive组件的最佳选择。因为它不会直接访问/修改父作用域的属性，不会产生意外的副作用。这种directive与父作用域进行数据通信有如下四种方式（更详细的内容请参考[Developer Guide](http://docs.angularjs.org/guide/directive" \t "http://blog.csdn.net/whisper_a/article/details/_blank)）：

* 1. **= or =attr** “Isolate”作用域的属性与父作用域的属性进行双向绑定，任何一方的修改均影响到对方，这是最常用的方式；
  2. **@ or @attr** “Isolate”作用域的属性与父作用域的属性进行单向绑定，即“Isolate”作用域只能读取父作用域的值，并且**该值永远的String类型**；
  3. **& or &attr** “Isolate”作用域把父作用域的属性包装成一个函数，从而以函数的方式读写父作用域的属性，包装方法是$parse，详情请见[API-$parse](http://docs.angularjs.org/api/ng.$parse" \t "http://blog.csdn.net/whisper_a/article/details/_blank)；

“Isolate”作用域的\_\_proto\_\_是一个标准[Scope](http://docs.angularjs.org/api/ng.$rootScope.Scope" \t "http://blog.csdn.net/whisper_a/article/details/_blank) object (the picture below needs to be updated to show an orange 'Scope' object instead of an 'Object'). “Isolate”作用域的$parent同样指向父作用域。它虽然没有原型继承，但它仍然是一个子作用域。

如下directive：

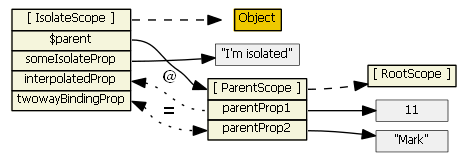
<my-directive interpolated="{{parentProp1}}" twowayBinding="parentProp2">

scope：

scope: { interpolatedProp: '@interpolated', twowayBindingProp: '=twowayBinding' }

link函数中：

scope.someIsolateProp = "I'm isolated"

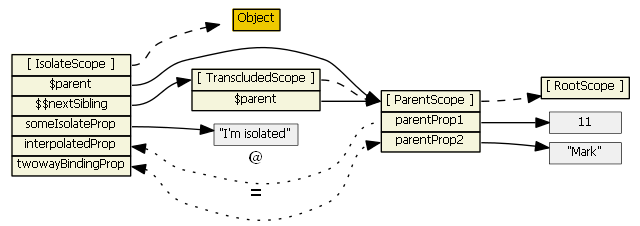


请注意，我们在link函数中使用attrs.$observe('interpolated', function(value) { ... }来监测@属性的变化。

更多请参考： [http://onehungrymind.com/angularjs-sticky-notes-pt-2-isolated-scope/](http://onehungrymind.com/angularjs-sticky-notes-pt-2-isolated-scope/" \t "http://blog.csdn.net/whisper_a/article/details/_blank)

transclude: true - directive新建一个“transcluded”子作用域，并且会从父作用域进行原型继承。需要注意的是，“transcluded”作用域与“Isolate”作用域是相邻的关系（如果“Isolate”作用域存在的话） -- 他们的$parent属性指向同一个父作用域。“Isolate”作用域的$$nextSibling指向“transcluded”作用域。

更多请参考： [AngularJS two way binding not working in directive with transcluded scope](http://stackoverflow.com/a/14484903/215945" \t "http://blog.csdn.net/whisper_a/article/details/_blank)



demo: [fiddle](http://jsfiddle.net/mrajcok/7g3QM/" \t "http://blog.csdn.net/whisper_a/article/details/_blank)

## 总结

AngularJS存在四种作用域:

1. 普通的带原型继承的作用域 -- ng-include, ng-switch, ng-controller, directive with scope: true；
2. 普通的带原型继承的，并且有赋值行为的作用域 -- ng-repeat，**ng-repeat**为每一个迭代项创建一个普通的有原型继承的子作用域，但同时在子作用域中创建新属性存储迭代项；
3. “Isolate”作用域 -- directive with scope: {...}， 该作用域没有原型继承，但可以通过'=', '@', 和 '&'与父作用域通信。
4. “transcluded”作用域 -- directive with transclude: true，它也是普通的带原型继承的作用域，但它与“Isolate”作用域是相邻的好基友。