AngularJs学习笔记--Scope

Lcllao

早日摆脱初级阶段

AngularJs学习笔记--Scope

原版地址：http://code.angularjs.org/1.0.2/docs/guide/scope

一、什么是Scope?

scope（http://code.angularjs.org/1.0.2/docs/api/ng.$rootScope.Scope）是一个指向应用model的object。它也是expression（http://www.cnblogs.com/lcllao/archive/2012/09/16/2687162.html）的执行上下文。scope被放置于一个类似应用的DOM结构的层次结构中。scope可以监测（watch,$watch）expression和传播事件。

二、scope的特性

- scope提供$watch API（http://code.angularjs.org/1.0.2/docs/api/ng.$rootScope.Scope#$watch），用于监测model的变化。
- scope提供$apply API（http://code.angularjs.org/1.0.2/docs/api/ng.$rootScope.Scope#$apply），在“Angular realm”（controller、server、angular event handler）之外，从系统到视图传播任何model的变化。
- scope可以在提供到被共享的model属性的访问的时候，被嵌入到独立的应用组件中。scope通过（原型），从parent scope中继承属性。
- scope在expression求值之时提供上下文环境。例如，{{username}}表达式是无意义的，除非它与一个特定的定义了”username”属性的scope一起进行求值。

三、Scope as Data-Model（scope作为数据模型）

scope是在应用controller与view之间的纽带。在模版linking（http://www.cnblogs.com/lcllao/archive/2012/09/04/2669802.html）的阶段，directive（http://www.cnblogs.com/lcllao/archive/2012/09/09/2677190.html）在scope中设置$watch表达式。$watch让directive能够得知属性的变化，使得directive将更新后的值渲染到DOM中。

controller和directive两者都与scope有引用，但它们两者之间没有（引用）（Both controllers and directives have reference to the scope, but not to each other）。这样的安排，将controller从directive和DOM中隔离开来。这是一个重要的地方，因为它让controller与view是隔离的，极大地提升了应用的可测试性（greatly improves the testing story of the applications）。

```
DOCTYPE HTML<< span>html lang="zh-cn" ng-app<< span>head<< span>meta charset="UTF-8"<< span>title<data-modeltitle<< span
>style type="text/css"<         .ng-cloak {             display: none;}style<head<< span>body class="ng-cloak"<< span>div
ng-controller="MyController"<     你的名字：  < span>input type="text" ng-model="username"/<< span>button ng-click="sayHello
()"<欢迎button<< span>hr/<    {{greeting}}div<< span>script src="../angular-1.0.1.js" type="text/javascript"<script<< spa
n>script type="text/javascript"<function MyController($scope) {         $scope.username ="My Little Dada";         $scope.
sayHello =function() {            $scope.greeting ="Hello~"+ $scope.username +"!";         };     }script<body<html<
```

在上面的例子中我们可以注意到MyController以”My Little Dada”对scope中的username属性进行赋值。然后，scope通知input进行赋值，将username的值预先填入input中。这展示了controller如何做才能够写入数据到scope中。

相似地，controller可以将行为附加在scope中，正如那个当用户点击“欢迎”按钮时触发的sayHello方法一样。sayHello方法可以读取username属性，也可以创建greeting属性。这表明，当它们绑定到HTML input控件时，scope中的属性会自动更新。

逻辑上，显示{{greeting}}涉及以下两点：

- 与定义了{{greeting}}表达式的模版DOM节点一起检索scope。在这个例子中，这个scope与传递到MyController中的scope是相同的。（我们在稍后将会讨论scope的层次结构）
- 通过之前检索的scope，对greeting表达式进行求值，然后将结果作为封闭DOM元素的text的值。

我们可以认为，scope和它自己的属性可以作为数据，用于渲染视图。scope是所有和view相关的东西单一的真相来源（The scope is the single source-of-truth for all things view related）。

从可测试性来看，controller和view的分离是值得欣喜的，因为它允许我们在没有渲染细节的干扰下（专注于）测试行为。

```
it('should say hello', function() {    var scopeMock = {};    var cntl = new MyController(scopeMock);       // Assert th
at username is pre-filled    expect(scopeMock.username).toEqual('World');       // Assert that we read new username and
```

```
 greet    scopeMock.username = 'angular';    scopeMock.sayHello();    expect(scopeMock.greeting).toEqual('Hello angular!
');});
```

四、Scope Hierarchies（scope层次结构）

每一个angular应用有且只有一个root scope，但可以拥有多个child scope。

应用可以拥有多个child scope，因为一些directive会创建新的child scope（参考directive文档，查看哪些directive可创建新的scope，如ng-repeat）。当新的scope被创建后，他们将作为一个child scope，加入到parent scope中。这样，创建了一个与它们附属的DOM相似的树结构。

当angular对{{username}}求值时，它首先查看与当前元素关联的scope的username属性。如果没有找到对应的属性，它将会一直向上搜索parent scope，直到到达root scope。在javascript中，这个行为被称为“原型继承”，child scope典型地继承自它们的parent。

这个例子说明应用中的scope（是怎样的），属性的原型继承。

```
DOCTYPE HTML<< span>html lang="zh-cn" ng-app<< span>head<< span>meta charset="UTF-8"<< span>title<scope-hierarchiestitle
<< span>style type="text/css"<        .ng-cloak {            display: none;}        .ng-scope {            border: 1px d
ashed red;}style<head<< span>body class="ng-cloak"<< span>div ng-controller="MyController"<    经理：{{employee.name}} [{
{department}}] < span>br/<    报告：    < span>ul<< span>li ng-repeat="employee in employee.reports"<            {{employ
ee.name}} [{{department}}]        li<ul<< span>hr/<    {{greeting}}div<< span>script src="../angular-1.0.1.js" type="tex
t/javascript"<script<< span>script type="text/javascript"<function MyController($scope) {        $scope.department ="某部
";        $scope.employee = {            name:"My Little Dada",            reports: [                {name:"Lcllao"},
           {name:"那个谁^o^"}            ]        };    }script<body<html<
```

注意，angular自动放置ng-scope class到与scope粘附的元素中。定义在上面的例子中，通过红色的虚线，高亮新的scope的范围。因为repeater对{{employee.name}}表达式求值，child scope是必须的，但取决于表达式在哪一个scope进行求值，不同的scope有不同的结果。相似地，{{department}}的值是从root scope中原型继承得来的，只有在那个地方有，才有department属性的定义。

五、Retrieving Scopes from the DOM（从DOM中检索scope）

scope作为$scope数据属性附加到DOM中，可以被用于以调试作为目的的检索。（在应用中通过这个方式检索Scope是不可能的。）附加到的DOM的root scope的位置是通过ng-app directive的位置定义的。通常ng-app是放置在元素中，但它也可以放置在其他元素中，例如，只有一部分视图需要被angular控制。

在debugger中查看scope：

1. 在浏览器中，对着感兴趣的元素点击右键，选择“查看元素”。我们可以看到浏览器debugger高亮了我们选中的元素。

2. debugger允许我们在console中通过$0变量去访问当前选择的元素。

3. 想查看关联的scope，我们可以在console中输入：angular.element($0).scope()

六、Scope Events Propagation（Scope事件传播）

scope可以以类似于DOM事件的方式进行事件传播。事件可以被broadcast（http://code.angularjs.org/1.0.2/docs/api/ng.$rootScope.Scope#$broadcast）到child scope或者emit（http://code.angularjs.org/1.0.2/docs/api/ng.$rootScope.Scope#$emit）到parent scope中。（当前scope如果有监听，也会执行）

```
DOCTYPE HTML<< span>html lang="zh-cn" ng-app<< span>head<< span>meta charset="UTF-8"<< span>title<scope-event-propagatio
ntitle<< span>style type="text/css"<        .ng-cloak {            display: none;}style<head<< span>body class="ng-cloak
"<< span>div ng-controller="MyController"<    root scope count:{{count}}    < span>ul<< span>li ng-repeat="i in [1]" ng-
controller="MyController"<< span>button ng-click="$emit('MyEvent')"<$emit("MyEvent")button<< span>button ng-click="$broa
dcast('MyEvent')"<$broadcast("MyEvent")button<< span>br/<            middle scope count:{{count}}            < span>ul<<
 span>li ng-repeat="item in [1,2]" ng-controller="MyController"<                    Leaf scope count:{{count}}
      li<ul<li<ul<div<< span>script src="../angular-1.0.1.js" type="text/javascript"<script<< span>script type="text/jav
ascript"<function MyController($scope) {        $scope.count =0;        $scope.$on("MyEvent", function() {            $s
cope.count++;        });    }script<body<html<
```

七、Scope Life Cycle（scope生命周期）

浏览器正常的事件流中，当浏览器接收到事件后，它会执行一个相应的javascript回调。一旦回调函数执行完毕后，浏览器将会重绘DOM，并返回到继续等待事件的状态。

当浏览器在angular执行环境外调用javascript代码时，这意味着angular是不知道model的改变的。要正确处理model的修改，这个命令必须通过使$apply方法进入angular执行环境。只有在$apply方法中的model变更，才会正确地被angular统计。例如，一个directive监听了DOM事件，例如ng-click，它必须在$apply方法中对表达式进行求值。

在对表达式求值之后，$apply方法执行一个$digest。在$digest阶段里，scope检查所有$watch监听的表达式，将现在的值与旧的值作比较。脏检查（dirty checking）是异步的。这意味着赋值语句（例如$scope.username=”angular”）将不会马上导致一个$watch被通知，反而，$watch的通知将会延迟到$digest阶段。这个延迟是必须的，因为它把多个model更新联合到一个$watch通知中，这保证了在$watch通知的过程中，没有其他$watch在执行。如果一个$watch改变了model的值，那么它将会强制增加一个$digest周期。

1) Creation（创建scope）

root scope是在应用启动的过程中，被$injector（http://code.angularjs.org/1.0.2/docs/api/AUTO.$injector）创建的。在模版linking的过程中，一些directive会创建新的child scope。

2) Watcher registration（注册watcher）

在模版linking过程中，directive在scope中注册$watch。这些watch将会被用作向DOM传播model的值。

3) Model mutation（Model变化）

为了让变化被正确地检测，我们需要将他们包裹在scope.$apply中。（angular API 已经隐式地做了这部操作，所以，当在controller中做同步的工作或者与$http或者$timeout一起做异步工作的时候，不需要额外的$apply调用）。

4) Mutation observation（变化监测）

在$apply的结尾，angular会在root scope执行一个$digest周期，这将会传播到所有child scope中。在$digest周期中，所有注册了$watch的表达式或者function都会被检查，判断model是否发生了改变，如果改变发生了，那么对应的$watch监听器将会被调用。

5) Scope destruction（scope销毁）

当child scope不再是必须的时候，child scope的产生者有责任通过scope.$destroy() API销毁它们（child scope）。这将会停止$digest的调用传播传播到child scope中，让被child scope model使用的内存可以被gc（garbage collector）回收。

1. Scopes and Directives

在编译阶段中，compiler依靠DOM模版匹配directive。directive通常可以分为两大类：

- 观察型directive（Observing directives），例如dobule-curly表达式{{expression}}，使用$watch方法注册监听器。无论什么时候，表达式（的值）发生改变，这类directive必须被通知，从而更新view。
- 监听型directive（Listener directive），例如ng-click，注册一个监听器到DOM中。当DOM的监听器触发时，directive会执行相关的表达式，并通过使用$apply方法更新视图。

当一个外部的事件（例如用户动作、timer或者XHR）被监听到，相关的expression必须通过$apply方法应用到scope中，让所有监听器能够正确地更新。

2. Directives that Create Scopes

在大多数的情况中，directive和scope是相互影响的，但不会创建新的scope实例。然而，一些directive（例如ng-controller和ng-repeat）会创建新scope，附加child scope到对应的DOM元素中。我们通过使用angular.element(aDomElement).scope()查看任意DOM元素的scope。

3. Controllers and Scopes

在以下的情况中，scope与controller是相互影响的：

- controller使用scope暴露controller方法到模版中（查看ng-controller（http://code.angularjs.org/1.0.2/docs/api/ng.directive:ngController））。
- controller定义方法（行为），可以改变model（scope上的属性）。
- controller可能在model中注册watch。这些watch会在controller行为执行之后马上执行。

4. Scope $watch Performance Considerations（Scope $watch的性能考虑）

在angular中，为了检测属性的变化而对scope进行脏检测（Dirty checking），是一个普遍的操作。为此，这要求dirty checking函数必须是高效的。应小心dirty checking函数不要做任何DOM访问操作，因为DOM访问的速度比访问javascript对象属性的速度要慢好几个数量级。

By Lcllao.

14211590.22197
分类: AngularJS
标签: AngularJs, 笔记, 教程

· 马化腾兴趣创业启示录
· 硅谷寻求将人类寿命延长到120岁以上
» 更多新闻...
最新知识库文章
· 小团队的技术管理
· 高效编程之欲擒故纵
· 互联网组织的未来：剖析GitHub员工的任性之源
· 内存数据库中的索引技术
· 如何实现一个malloc
» 更多知识库文章...

公告

昵称：Lcllao
园龄：2年9个月
粉丝：89
关注：8
+加关注

< a>	2012年9月	<

日	一	二	三	四	五	六
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

AngularJs(21)
教程(21)
笔记(21)
javascript(2)
闭包(1)
理解(1)
浏览器(1)
疑问(1)
canvas(1)
foreach(1)
更多

随笔分类

AngularJS(21)
坑(3)

随笔档案

2013年2月 (1)
2012年10月 (5)
2012年9月 (16)
2012年4月 (3)

文章分类

原创(1)

最新评论

1. Re:AngularJs学习笔记--directive
@Echofool
你是用的是回调获取数据吗？
--limerick
2. Re:AngularJs学习笔记--directive
我使用局部scope，使用=绑定，但是我发现那个是单向的，不是双向的绑定，貌似它和function传参是一个机制，如果你传的是一个引用对象，就可以都改变，但是普通对象是不会改变controller里的$scope的指定对象。
--limerick
3. Re:AngularJs学习笔记--directive
楼主，我用$http获取数据，angularjs 刷新加载了部分ui之后，我想在重新加载的某些元素上应用样式，我照着你写的，写了一个link function的 postLink方法，但是获取不到子元素啊。这是咋回事，求指导。
--Echofool
4. Re:AngularJs学习笔记--directive
楼主辛苦了！
--最爱海贼王
5. Re:AngularJs学习笔记--concepts(概念)
楼主辛苦了！
--最爱海贼王

阅读排行榜

1. AngularJs学习笔记--directive(33931)
2. AngularJs学习笔记--bootstrap(32250)
3. AngularJs学习笔记--Guide教程系列文章索引(17619)
4. AngularJs学习笔记--Modules(14848)
5. AngularJs学习笔记--Scope(12909)

评论排行榜

1. AngularJs学习笔记--directive(14)
2. AngularJs学习笔记--Guide教程系列文章索引(9)
3. AngularJs学习笔记--html compiler(8)
4. AngularJs学习笔记--concepts(概念)(6)
5. AngularJs学习笔记--expression(3)

推荐排行榜

1. AngularJs学习笔记--Guide教程系列文章索引(10)
2. AngularJs学习笔记--bootstrap(7)
3. AngularJs学习笔记--directive(6)
4. AngularJs学习笔记--html compiler(5)
5. AngularJs学习笔记--concepts(概念)(4)