# 深入理解ANGULAR中的$APPLY()以及$DIGEST()

$apply()和$digest()在AngularJS中是两个核心概念，但是有时候它们又让人困惑。而为了了解AngularJS的工作方式，首先需要了解$apply()和$digest()是如何工作的。这篇文章旨在解释$apply()和$digest()是什么，以及在日常的编码中如何应用它们。

探索$apply()和$digest()

AngularJS提供了一个非常酷的特性叫做双向数据绑定(Two-way Data Binding)，这个特性大大简化了我们的代码编写方式。数据绑定意味着当View中有任何数据发生了变化，那么这个变化也会自动地反馈到scope的数据上，也即意味着scope模型会自动地更新。类似地，当scope模型发生变化时，view中的数据也会更新到最新的值。那么AngularJS是如何做到这一点的呢？当你写下表达式如{{ aModel }}时，AngularJS在幕后会为你在scope模型上设置一个watcher，它用来在数据发生变化的时候更新view。这里的watcher和你会在AngularJS中设置的watcher是一样的：

****[javascript]**** [view plain](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "view plain)[copy](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "copy)[print](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "print)[?](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "?)

1. $scope.$watch(‘aModel’, function(newValue, oldValue) {
2. //update the DOM with newValue
3. });

传入到$watch()中的第二个参数是一个回调函数，该函数在aModel的值发生变化的时候会被调用。当aModel发生变化的时候，这个回调函数会被调用来更新view这一点不难理解，但是，还存在一个很重要的问题！AngularJS是如何知道什么时候要调用这个回调函数呢？换句话说，AngularJS是如何知晓aModel发生了变化，才调用了对应的回调函数呢？它会周期性的运行一个函数来检查scope模型中的数据是否发生了变化吗？好吧，这就是$digest循环的用武之地了。

在$digest循环中，watchers会被触发。当一个watcher被触发时，AngularJS会检测scope模型，如果它发生了变化，那么关联到该watcher的回调函数就会被调用。那么，下一个问题就是$digest循环是在什么时候以各种方式开始的？

在调用了$scope.$digest()后，$digest循环就开始了。假设你在一个ng-click指令对应的handler函数中更改了scope中的一条数据，此时AngularJS会自动地通过调用$digest()来触发一轮$digest循环。当$digest循环开始后，它会触发每个watcher。这些watchers会检查scope中的当前model值是否和上一次计算得到的model值不同。如果不同，那么对应的回调函数会被执行。调用该函数的结果，就是view中的表达式内容(译注：诸如{{ aModel }})会被更新。除了ng-click指令，还有一些其它的built-in指令以及服务来让你更改models(比如ng-model，$timeout等)和自动触发一次$digest循环。

目前为止还不错！但是，有一个小问题。在上面的例子中，AngularJS并不直接调用$digest()，而是调用$scope.$apply()，后者会调用$rootScope.$digest()。因此，一轮$digest循环在$rootScope开始，随后会访问到所有的children scope中的watchers。

现在，假设你将ng-click指令关联到了一个button上，并传入了一个function名到ng-click上。当该button被点击时，AngularJS会将此function包装到一个wrapping function中，然后传入到$scope.$apply()。因此，你的function会正常被执行，修改models(如果需要的话)，此时一轮$digest循环也会被触发，用来确保view也会被更新。

Note: $scope.$apply()会自动地调用$rootScope.$digest()。**$apply()方法有两种形式。第一种会接受一个function作为参数，执行该function并且触发一轮$digest循环。第二种会不接受任何参数，只是触发一轮$digest循环。**我们马上会看到为什么第一种形式更好。

什么时候手动调用$apply()方法？

如果AngularJS总是将我们的代码wrap到一个function中并传入$apply()，以此来开始一轮$digest循环，那么什么时候才需要我们手动地调用$apply()方法呢？实际上，AngularJS对此有着非常明确的要求，就是它只负责对发生于AngularJS上下文环境中的变更会做出自动地响应(即，在$apply()方法中发生的对于models的更改)。AngularJS的built-in指令就是这样做的，所以任何的model变更都会被反映到view中。但是，**如果你在AngularJS上下文之外的任何地方修改了model，那么你就需要通过手动调用$apply()来通知AngularJS。这就像告诉AngularJS，你修改了一些models，希望AngularJS帮你触发watchers来做出正确的响应。**

**比如，如果你使用了JavaScript中的setTimeout()来更新一个scope model，那么AngularJS就没有办法知道你更改了什么。这种情况下，调用$apply()就是你的责任了，通过调用它来触发一轮$digest循环。**类似地，如果你有一个指令用来设置一个DOM事件listener并且在该listener中修改了一些models，那么你也需要通过手动调用$apply()来确保变更会被正确的反映到view中。

让我们来看一个例子。假如你有一个页面，一旦该页面加载完毕了，你希望在两秒钟之后显示一条信息。你的实现可能是下面这个样子的：

HTML:

****[html]**** [view plain](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "view plain)[copy](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "copy)[print](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "print)[?](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "?)

1. <body ng-app=“myApp”>
2. <div ng-controller=“MessageController”>
3. Delayed Message: {{message}}
4. </div>
5. </body>

JavaScript:

****[javascript]**** [view plain](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "view plain)[copy](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "copy)[print](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "print)[?](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "?)

1. /\* What happens without an $apply() \*/
2. angular.module(‘myApp’,[]).controller(‘MessageController’, function($scope) {
3. $scope.getMessage = function() {
4. setTimeout(function() {
5. $scope.message = ‘Fetched after 3 seconds';
6. console.log(‘message:’+$scope.message);
7. }, 2000);
8. }
9. $scope.getMessage();
10. });

通过运行这个例子，你会看到过了两秒钟之后，控制台确实会显示出已经更新的model，然而，view并没有更新。原因也许你已经知道了，就是我们忘了调用$apply()方法。因此，我们需要修改getMessage()，如下所示：

****[javascript]**** [view plain](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "view plain)[copy](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "copy)[print](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "print)[?](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "?)

1. /\* What happens with $apply \*/
2. angular.module(‘myApp’,[]).controller(‘MessageController’, function($scope) {
4. $scope.getMessage = function() {
5. setTimeout(function() {
6. $scope.$apply(function() {
7. //wrapped this within $apply
8. $scope.message = ‘Fetched after 3 seconds';
9. console.log(‘message:’ + $scope.message);
10. });
11. }, 2000);
12. }
14. $scope.getMessage();
16. });

如果你运行了上面的例子，你会看到view在两秒钟之后也会更新。唯一的变化是我们的代码现在被wrapped到了$scope.$apply()中，它会自动触发$rootScope.$digest()，从而让watchers被触发用以更新view。

Note:顺便提一下，你应该使用$timeout service来代替setTimeout()，因为前者会帮你调用$apply()，让你不需要手动地调用它。

而且，注意在以上的代码中你也可以在修改了model之后手动调用没有参数的$apply()，就像下面这样：

****[javascript]**** [view plain](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "view plain)[copy](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "copy)[print](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "print)[?](http://blog.csdn.net/dm_vincent/article/details/38705099" \o "?)

1. $scope.getMessage = function() {
2. setTimeout(function() {
3. $scope.message = ‘Fetched after two seconds';
4. console.log(‘message:’ + $scope.message);
5. $scope.$apply(); //this triggers a $digest
6. }, 2000);
7. };

以上的代码使用了$apply()的第二种形式，也就是没有参数的形式。需要记住的是你总是应该使用接受一个function作为参数的$apply()方法。这是因为当你传入一个function到$apply()中的时候，这个function会被包装到一个try…catch块中，所以一旦有异常发生，该异常会被$exceptionHandler service处理。

$digest循环会运行多少次？

当一个$digest循环运行时，watchers会被执行来检查scope中的models是否发生了变化。如果发生了变化，那么相应的listener函数就会被执行。这涉及到一个重要的问题。如果listener函数本身会修改一个scope model呢？AngularJS会怎么处理这种情况？

答案是$digest循环不会只运行一次。在当前的一次循环结束后，它会再执行一次循环用来检查是否有models发生了变化。这就是脏检查(Dirty Checking)，它用来处理在listener函数被执行时可能引起的model变化。因此，$digest循环会持续运行直到model不再发生变化，或者$digest循环的次数达到了10次。因此，尽可能地不要在listener函数中修改model。

Note: $digest循环最少也会运行两次，即使在listener函数中并没有改变任何model。正如上面讨论的那样，它会多运行一次来确保models没有变化。

结语

我希望这篇文章解释清楚了$apply和$digest。需要记住的最重要的是AngularJS是否能检测到你对于model的修改。如果它不能检测到，那么你就需要手动地调用$apply()。