理解AngularJS指令 -- ng-view

[张小俊128](http://www.html-js.com/user/526) 发布在[用Angular开发web应用](http://www.html-js.com/article/column/50)2014年3月9日view：27743

**理解AngularJS指令 – ng-view**

在本文中我们将探索ng-view指令内部的实现方式，并且创建一个“ngMultiView”指令，

从AngularJS 1.2开始，ngView指令以及$route service 都被移动到了一个单独的ngRoute模块中。于是，如果你需要使用ngView和route的话，必须显式的声明这个模块作为依赖。另 外，otherwise语法也和之前的AngularJS版本有所不同。下面就是一个route的例子，它创建了两个路由和一个默认选项：

var app = angular.module('ngViewExampleApp', ['ngRoute']);

app.controller('RootCtrl', ['$scope', function($scope){

$scope.title = "Home Page";

}]);

app.controller('CatsCtrl', ['$scope', function($scope){

$scope.title = "Cats Page";

}]);

app.config(['$routeProvider', function($routeProvider){

$routeProvider

.when('/', {

controller : 'RootCtrl',

template : '<h1>{{title}}</h1>'

})

.when('/cats', {

controller : 'CatsCtrl',

template : '<h1>{{title}}</h1>'

})

.otherwise({

redirectTo : '/'

});

}]);

**理解ngView的特性**

在我们深入探索ngView背后的代码去之前，我们先要来说说两个文档中不存在的属性：‘onload’和’autoscroll‘。onload 属性将会接收任何AngularJS表达式并在视图发生变化时执行。AutoScroll使用$autoScroll service并且基于$location.hash()的当前值滚动到一个特定的元素。最后，在指令的最后，link(currentScope)之 后’$viewContentLoaded’时间将会在当前的作用域内被发射 – 你可以在你的控制器中使用这个时间。下面的代码是上面的例子的一个修改版本，其中包括一个onload属性。

var app = angular.module('ngViewExampleApp', ['ngRoute']);

app.controller('RootCtrl', ['$scope', function($scope){

$scope.title = "Home Page";

}]);

app.controller('CatsCtrl', ['$scope', function($scope){

$scope.title = "Cats Page";

}]);

app.controller('AppCtrl', ['$scope', function($scope){

$scope.onViewLoad = function(){

console.log('view changed');

};

}]);

app.config(['$routeProvider', function($routeProvider){

$routeProvider

.when('/', {

controller : 'RootCtrl',

template : '<h1>{{title}}</h1>'

})

.when('/cats', {

controller : 'CatsCtrl',

template : '<h1>{{title}}</h1>'

})

.otherwise({

redirectTo : '/'

});

}]);

**ngView是怎样运行的？**

为了理解ngView，我们现在来创建一个ngView的简化版本。下面是ngView的简化版本ngViewLite，它并补办扩作用域清除或者动画，除此之外和ngView基本上没有什么区别。

var app = angular.module("app", ['ngRoute']);

app.directive("ngViewLite", ['$route', '$compile', '$controller', function($route, $compile, $controller){

return {

terminal: true,

priority: 400,

transclude: 'element',

compile : function(element, attr, linker){

return function(scope, $element, attr) {

var currentElement;

scope.$on('$routeChangeSuccess', update);

update();

// update view

function update(){

var locals = $route.current && $route.current.locals,

template = locals && locals.$template;

if(template){

var newScope = scope.$new();

linker(newScope, function(clone){

clone.html(template);

$element.parent().append(clone);

if(currentElement){

currentElement.remove();

}

var link = $compile(clone.contents()),

current = $route.current;

currentElement = clone;

current.scope = newScope;

if (current.controller) {

locals.$scope = newScope;

var controller = $controller(current.controller, locals);

clone.data('$ngControllerController', controller);

clone.children().data('$ngControllerController', controller);

}

link(newScope);

newScope.$emit('$viewContentLoaded');

});

}else{

//清除上一次的视图

}

}

}

}

}

}]);

app.controller('RootCtrl', ['$scope', function($scope){

$scope.title = "Home Page";

}]);

app.controller('CatsCtrl', ['$scope', function($scope){

$scope.title = "Cats Page";

}]);

app.config(['$routeProvider', function($routeProvider){

$routeProvider

.when('/', {

controller : 'RootCtrl',

template : '<h1>{{title}}</h1>'

})

.when('/cats', {

controller : 'CatsCtrl',

template : '<h1>{{title}}</h1>'

})

.otherwise({

redirectTo : '/'

});

}]);

首先，绑定了一个函数update到事件$routeChangeSuccess；当路由发生变化时，update函数将会被调用。在将函数绑定到事件上之后，我们马上调用update()将初始化内容载入页面中。

update函数会检查对于当前路由是否有定义好的模板，如果有它将会调用linker函数，为它传递一个新的作用域，以及一个毁掉函数。这个回调 函数中唯一的参数句式克隆的元素，它的html会被当前路由的模板所替代。克隆的元素接着被追加到具有ng-view-lite属性的div中。在此之后 我们将前面的内容从视图中移除。

最后，模板必须被编译($compile(clone.contents()))，同时一个新的作用域会被注入到其中(link(newScope))。在这两个步骤之间我们会检查路由是否具有一个相关联的控制器，如果有我们就用这个newScope以及当前路由的本地变量初始化控制器。

**编写一个ngMultiView**

ngView运行的很好，但是如果你想要根据url来改变多个视图怎么办。根据文档我们知道在一个应用中ngView只能被使用一次。为了完成我们 的ngMultiView，我们需要稍稍修改ngView，并创建一个AngularJS值（MultiViewPaths）来保存 urls，views，controllers以及templates之间的映射。

在ngMultiView中，我们需要给指令传递一个参数：

`<div ng-multi-view="secondaryContent"></div>`

在这个指令中，这个属性被称为”panel”。我们在此不绑定”$routeChangeSuccess”事件，而是去绑 定”$locationChangeSuccess”事件，来确保我们的指令完全的独立于ngRoute。ngMultiView将会根据下面的方式运 行：

1. 一个url的变化会触发’$locationChangeSuccess’事件，它反过来会掉哦那个update()函数；
2. 在update函数内部：获取URL#符号后面的部分
3. 使用这个URL变量，以及panel，我们从MultiViewPaths中查找相应的控制器和模板
4. 如果我们找到了控制器和模板，ngMultiView几乎就和ngView一样了

var app = angular.module('app', []);

app.value('MultiViewPaths',

{'/' : {

content : {

template : '<h1>Home Page</h1><p>More Cats!</p>'

},

secondaryContent : {

template : '<h2>Visitors Online</h2><ul><li ng-repeat="user in users">{{user}}</li></ul>',

controller : 'ListUsersCtrl'

}

},

'/cats' : {

content: {

template : '<h1>All Cats</h1><ul><li ng-repeat="cat in cats">{{cat}}</li></ul>',

controller : 'ListCatsCtrl'

},

secondaryContent : {

template : '<h2>Cat of the Minute: {{cat}}</h2>',

controller : 'CatOfTheMinuteCtrl'

}

}

});

app.directive("ngMultiView", ['$rootScope', '$compile', '$controller', '$location', 'MultiViewPaths', function($rootScope, $compile, $controller, $location, MultiViewPaths){

return {

terminal: true,

priority: 400,

transclude: 'element',

compile : function(element, attr, linker){

return function(scope, $element, attr) {

var currentElement,

panel = attr.ngMultiView;

$rootScope.$on('$locationChangeSuccess', update);

update();

// update view

function update(evt, newUrl, oldUrl){

if(!newUrl){ return }

var url = newUrl.match(/#(\/.\*)/),

match, template, controller;

match = url ? MultiViewPaths[url[1]] : MultiViewPaths['/'];

template = match[panel].template;

controller = match[panel].controller;

if(template){

var newScope = scope.$new(),

locals = {},

newController = controller;

linker(newScope, function(clone){

clone.html(template);

$element.parent().append(clone);

if(currentElement){

currentElement.remove();

}

var link = $compile(clone.contents());

currentElement = clone;

if (newController) {

locals.$scope = newScope;

var controller = $controller(newController, locals);

clone.data('$ngControllerController', newController);

clone.children().data('$ngControllerController', newController);

}

link(newScope);

newScope.$emit('$viewContentLoaded');

});

}else{

//cleanup last view

}

}

}

}

}

}]);

/\* creating the controllers and their data \*/

app.controller('ListUsersCtrl', ['$scope', function($scope){

$scope.users = ['Lord Nikon', 'Acid Burn', 'Crash Override'];

}]);

app.value('cats', ['Toonces','Stache','Americat','Cassiopeia','Puck','Dica','Vivian','Shosh','Gray','Bashful','Querida','Ignatowski','Aenias','Ramsay','Ishcabible','Guinness','Roux','Gefahr']);

app.controller('ListCatsCtrl', ['$scope', 'cats', function($scope, cats){

$scope.cats = cats;

}]);

app.controller('CatOfTheMinuteCtrl', ['$scope', 'cats', function($scope, cats){

var randIndex = Math.floor(Math.random() \* cats.length);

$scope.cat = cats[randIndex];

}]);

我们的ngMultiView指令非常的基本，它不能接受任何由urls传递的参数，也不会处理作用域清除，或者动画。如果你需要更多的功能，你可以修改$route service来让它满足多视图要求。

**总结**

创建自定义指令一开始会很吓人。有太多的术语等着我们去学习。然而，只要你学习编写了一个指令，后面的学习就会简单很多。