$parse详解

**一 . 简单使用方法及解析**

(html)

<**div** ng-app="MyApp">

<**div** ng-controller="MyController">

<**h1**>{{ParsedValue}}</**h1**>

</**div**>

</**div**>

(js)

angular.module("MyApp",[])

.controller("MyController", **function**($scope, $parse){

//上下文对象 ，表达式在此对象的上下文执行

**var** context = {

name: "dreamapple"

};

*// 因为这个解析的语句中含有我们想要解析的表达式，*

*// 所以要把不相关的用引号引起来，整体然后用+连接*

**var** expression = "'Hello ' + name"; //a

//用$parse服务封装表达式：

**var** parseFunc = $parse(expression); //b

//在对象context的上下文中执行表达式: "'Hello ' + name"

//在context中，name 值为” dreamapple” ,因此，表达式变为：

//"'Hello ' + ‘dreamapple’" ,最终结果为

// $scope.ParsedValue = “Hello dreamapple”

$scope.ParsedValue = parseFunc(context); //c

});

**二. 分析**

**1.使用步骤: (具体见代码中的abcd..)**

a. 编写要被执行的表达式

b. 用$parse服务封装表达式 ,生成一个函数

c.在指定上下文环境中执行表达式，返回结果

**1.代码等价于：**

**var** context = {

name: "dreamapple",

convert:**function**(value){

**return** value + name;

}

};

$scope.val = context.convert("Hello");

不同的是调用者不是context，而是使用$parse服务

**三. 其它例子**

**1. 例子1**

(html)

<**div** ng-app="MyApp">

<**div** ng-controller="MyController">

<**h1**>{{ParsedValue}}</**h1**>

</**div**>

</**div**>

(js)

angular.module("MyApp",[])

.controller("MyController", **function**($scope, $parse){

*//上下文对象*

$scope.context = {

add: **function**(a, b){**return** a + b;},

mul: **function**(a, b){**return** a \* b}

}

*//要被执行的表达式*

$scope.expression = "mul(a, add(b, c))";

*//参数上下文对象，用户传给表达式，表达式会在这个对象里找到对应的变量*

$scope.data = {

a: 3,

b: 6,

c: 9

};

*//使用$parse服务封装表达式，生成调用函数*

**var** parseFunc = $parse($scope.expression);

*//在context环境，data对象里面的属性作为参数，执行表达式 ：*

*//mul(a, add(b, c)) ，在$scope.data 中，a=3,b=6 ,c= 9*

*// => mul(3,add(6,9)) => 3 \* (6 + 9) => 45*

$scope.ParsedValue = parseFunc($scope.context, $scope.data);

});

**2. 例子2 , 自定义指令，响应标签元素的右击事件**

**（html）**

<**body ng-controller="rightClickMainController"**>

<--如果右击该div标签，则触发rightClickFn事件 !-->

<--并且将event对象传递过去!-->  
<**div right-click="rightClickFn($event)"**>1213123123</**div**>  
</**body**>

**(js controller)**

**angular**.module(**"testRightClickApp"**)  
 .controller(**"rightClickMainController"** , *controllerFun*);  
*controllerFun*.**$injector** = [];  
  
**function** *controllerFun*($scope){  
 $scope.rightClickFn = *rightClickFun*;  
 $scope.**abc** = **"def"**;  
  
 **function** *rightClickFun*($event){  
 alert($event);  
 }  
}

**(js directive)**

**angular**.module(**"commonDirective"**)  
 .**directive**(**"rightClick"** , *directiveFun*);  
*directiveFun*.**$injector** = [**"$parse"**];  
  
**function** *directiveFun*($parse){  
 **var** ret = {};  
 ret.**restrict** = **"A"**;  
 ret.link = *linkFun*;  
 **return** ret;  
  
 **function** *linkFun*(scope ,element , attrs){

//标签传入的要被执行的函数表达式  
 **var** expression = attrs[**"rightClick"**];

//*使用$parse服务封装expression函数表达式，生成调用函数*  
 **var** fn = $parse(expression);

// 用jquery绑定右击事件（mousedown事件，并且event.which==3）  
 element.on(**"mousedown"** ,*clickFun*);  
 **function** *clickFun*(event){  
 event.preventDefault();  
 **if**(event.**which** == 3){  
 **var** *callback* = **function**(){

//在scope的上下文执行该方法，

//=> scope.**rightClickFn($event)**

**//即在controller中执行**  
 fn(scope , {**$event**: event});  
 }

// 在更新view之前执行expression

// 因为是在jquery库的环境中，所以要调用$apply  
 scope.**$apply**(*callback*);  
 }  
  
  
 }  
 }  
}

**四. Api解析**

* $parse  
    
  + 作用：将一个AngularJS表达式转换成一个函数
* Usage  
  $parse(expression)  
    
  + arguments  
      
    - expression:需要被编译的AngularJS语句
  + returnsfunc(context, locals)  
      
    - context[object]:针对你要解析的语句，这个对象中含有你要解析的语句中的表达式(通常是一个scope object)
    - locals[object]: 关于context中变量的本地变量，对于覆盖context中的变量值很有用。
    - 返回的函数还有下面三个特性：  
        
      * literal[boolean]:表达式的顶节点是否是一个javascript字面量
      * constant[boolean]:表达式是否全部是由javascript的常量字面量组成
      * assign[func(context, local)]:可以用来在给定的上下文中修改表达式的值