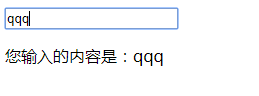
Angular

使用的小例子步骤：  
根作用域（最大的作用域）  
1.导入Angular的script的包  
2.在我们定义的input里面创写上ng-model="username"是一会要获取的值

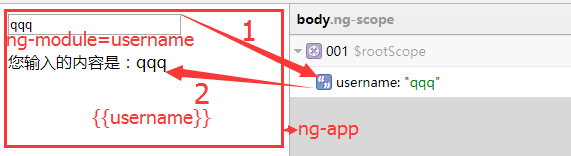
3.在要接收这个值的span中添加{{username}一会接收值}}

**ng-app(指令) :** 告诉angular核心它管理当前标签所包含的整个区域,并且会自动创建$rootScope根作用域对象  
**ng-model :** 将当前输入框的值与谁关联(属性名:属性值), 并作为当前作用域对象($rootScope)的属性  
**{{}} (表达式) :** 显示数据,从作用域对象的指定属性名上取

例如：  
运行效果：后面会再跟着输出的，一起输出



**说明：**



**2.ng-app：**声明所在的整个标签都是由angular来解析管理的，如果没有这个声明就不会产生根作用域  
**3.ng-modul**e（能提供数据用） 当前输入框的值关联到内存$rootScoped的属性上——> ng-module = "xxx"

    <script src="../../js/angular-1.2.29/angular.js"></script>  
    <body ng-app>  
    <input type="text" class="type" ng-model="username">  
    <span>输入的是:{{username}}</span>  
    </body>

xxx 整个带着input里面输入的值，是属性名也是属性值，下面同样的xxx的名字的可以接收到这个input输入的值

**4.{{xxx}}**（只能用来显示数据）接收上面的XXX接收到的值，显示作用域上对应的属性的值  
**5.ng-init**  初始化数据，给个默认值 比如给xxx="kobe"xxx的值，nameinput一上来上来就会是kobe

关键：

* 当页面输入框的数据改变是，angular会将数据更新到域对象的的对应属性
* 当前对象上的属性数据发生改变时，angular会更新其对应的页面显示

**Angular的特点**

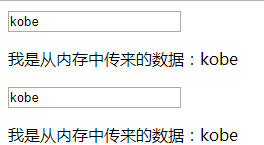
**一、双向数据绑定**

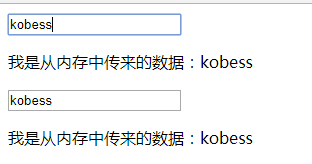
1. 数据绑定: 数据从一个地方A转移(传递)到另一个地方B, 而且这个操作由框架来完成  
2. 双向数据绑定: 数据可以从View(视图层)流向Model（模型）, 也可以从Model流向View  
 \* 视图(View): 也就是我们的页面(主要是Andular指令和表达式)  
 \* 模型(Model) : 作用域对象(当前为$rootScope), 它可以包含一些属性或方法  
 \* 当改变View中的数据, Model对象的对应属性也会随之改变: ng-model指令 数据从View==>Model  
 \* 当Model域对象的属性发生改变时, 页面对应数据随之更新: {{}}表达式 数据从Model==>View  
 \* ng-model是双向数据绑定, 而{{}}是单向数据绑定  
3.ng-init 用来初始化当前作用域变量。

**例子：**

* 在内存和html中可以数据交流，是双向的

body ng-app="" ng-init="name='tom'">  
<input type="text" ng-model="name">  
<p>姓名1：{{name}}</p>  
<input type="text" ng-model="name">  
<p>姓名2：{{name}}</p>

图1.一上来的界面

图2.一起改变的结果

注意：

* 单向(只支持一个方向的数据流向)

ng-init是单向的，是初始化之后，后面改变了，也不会改变

* view<---->module是双向的

**二、声明式依赖注入**

**\* 依赖对象**：完成某个特定的功能需要某个对象才能实现，这个对象就是依赖对象。\* 回调函数的event的就是依赖对象  
**\* 依赖注入**：依赖的对象以形参的形式被注入进来使用，这种方式就是依赖注入。\* 回调函数有形参就是依赖注入

\* angular的 ‘$scope’对象就是依赖对象，并且是依赖注入的形式进行使用。  
 ！！！形参必须是特定的名称, 否则Angular无法注入抛异常

**命令式：**

按照命令一步一步的执行，写清楚每一个步骤

更注重执行过程，像解答题

**声明式：**

是对命令式的局部包装，有些功能可以用命令式包装好的 这里直接用

更注重执行结果，像填空选择

**控制器：**

1. 作用域对象 :  
 \* 一个js实例对象, ng-app指令默认会创建一个根作用域对象($rootScope)  
 \* 它的属性和方法与页面中的指令或表达式是关联的  
2. 控制器:的功能流程  
 \* 用来控制AngularJS应用数据的 实例对象  
 \* **ng-controller :** 指定控制器构造函数, Angular会自动new此函数创建控制器对象  
 \* 同时Angular还有创建一个新的域对象$scope, 它是$rootScope的**子对象**  
 \* 在控制器函数中声明$scope形参, Angular会自动将**$scope**传入

**总结：**就是用一个大的div包住一部分代码，在div中加上**ng-controller="MyController"，**就会成为新的作用域（像函数一样）这个作用域里面的就会成为一个根作用域的子对象（孩子，可遗传，可引用根作用域的值）这时候他是一个实例对象，然后在script中可以操纵这个作用域的值

<div ng-controller="**MyController**">  
 <input type="text" placeholder="姓" ng-model="firstName">  
 <input type="text" placeholder="名" ng-model="lastName">  
 <p>输入的姓名为: {{firstName+'-'+lastName}}</p>  
 <p>输入的姓名2为: {{getName()}}</p>  
</div>

<script type="text/javascript">  
 function **MyController** ($scope) {//**必须是$scope**  
 // alert(this instanceof MyController);//说明是new调用  
 $scope.firstName = 'KB';  
 $scope.lastName = 'Brent';  
  
 //给域对象指定方法  
 $scope.getName = function() {  
 return $scope.firstName + " " + $scope.lastName;  
 };  
  
 console.log($scope.age);  
 }  
</script>

注意：

* 这里的**MyController要和下面的MyController一样**
* 参数必须是$scope,它是这个构造器对象的实例，是根作用域的子对象
* $scope新填的属性，要是这个div中的ng-module的值，在里面初始化，也可以定义新的方法
* 在div要是要调用方法要写完整.getName()得到返回值

**angular.module**可在全局位置，创建，注册，获取angular模块，左右模块，核心或者第三方都必须使用这个机制注册才能在应用中生效

要两个参数

第一个参数：”当前模块对象的名字也是他的标识=ng-app的值”

1. 创建模块对象，angular.modele(“app”,[一个数组或者对象都可以])返回的是一个模块对象，可以用一个变量接收，然后在后面链式调用
2. 生成作用域对象

<body ng-app="myapp">  
<div ng-controller="MyCtrl">  
 <input type="text" ng-model="empName">  
 <p>员工名:{{empName}}</p>  
</div>  
  
<div ng-controller="MyCtrl1">  
 <input type="text" ng-model="empName">  
 <p>员工:{{empName}}</p>  
</div>  
<scripttype="text/javascript" src="../../js/angular-1.5.5/angular.js"></script>  
<script type="text/javascript">

**angular.module("myapp",[])  
 .controller("MyCtrl",function ($scope) {  
 $scope.empName = "wxx"  
  
 })  
 .controller("MyCtrl1",function ($scope) {  
 $scope.empName = 'wxx'  
 })**  
  
</script>

重点是加粗这一段

上面写法的问题：  
 1、形参只能写固定的变量名$scope;  
 2、一旦文件压缩，将不能使用，会报错。

所以以后要换一种写法，把加粗改成如下：

**angular**.*module*(**'MyApp'**,[])  
 .**controller**(**'MyCtrl'**,[**'$scope'**,**function** (a) {  
 a.**empName** = **'tom'** }])  
 .controller(**'MyCtrl1'**,[**'$scope'**,**function** (b) {  
 b.**empName** = **'Jack'**;  
 }])

表达式：

1.创造模块对象

angular.module(“myapp”,[])

.controller(“myctrl”,[“$scope”,function(a){

}] )

2.在body创建新的作用域，

<div ng-controller=” myctrl”>

<p>{{123}}</p>

</div>

3.使用Angular表达式:  
 \* 语法: {{expression}}  
 \* 作用: 显示表达式的结果数据  
 \* 注意: 表达式中引用的变量必须是当前域对象有的属性(包括其原型属性)  
2. 操作的数据  
 \* 基本类型数据: number/string/boolean  
 \* undefined, Infinity, NaN, null解析为空串: "", 不显示任何效果  
 \* 对象的属性或方法  
 \* 数组

**例子：**

写表达式------->{{}}

表达式能操作属性和方法

<**div ng-controller="my"**> ----这个的my和下面的没有对应  
 <**p**>{{123}}</**p**> ---->数字  
 <**p**>{{**abc**}}</**p**>---->这样不行，会找变量undefined就没值了  
 <**p**>{{"abc"}}</**p**>---->这样就可以，字符串  
 <**p**>{{true}}</**p**>---->能显示—>true  
 <**p**>==={{**undefined**}}----></**p**>就什么都不会输出，空白  
 <**p**>{{**abc**.split("")}}</**p**>下面已经说明值是----> ["N","B","A"]  
 <**p**>{{**abc**.split("").reverse("")}}</**p**>----> ["A","B","N"]  
 <**p**>{{**abc**.split("").reverse("").join()}}----></**p**>A,B,N  
</**div**>

<**script**>  
 **angular**.*module*(**"myapp"**,[])  
 .**controller**(**"my"**,[**"$scope"**,**function** (a) {  
 a.**abc**=**"NBA"**;  
  
 }])  
</**script**>

\* \* ng-app: 指定模块名，angular管理的区域  
**\* ng-model：** 双向绑定，输入相关标签  
**\* ng-init：** 初始化数据  
**\* ng-click：** 调用作用域对象的方法（点击时）  
**\* ng-controller:** 指定控制器构造函数名，内部会自动创建一个新 的子作用域（外部的）  
**\* ng-class**: 动态引用定义的样式 {aClass:true, bClass:false}  
**\* ng-styl**e: 动态引用通过js指定的样式对象 {color:'red', background:'blue'}  
**\* ng-click:** 点击监听, 值为函数调用, 可以传$event  
**\* ng-mouseenter:** 鼠标移入监听, 值为函数调用, 可以传$event  
**\* ng-mouseleave:** 鼠标移出监听, 值为函数调用, 可以传$event

