## AngularJS 学习笔记

### 1、控制器

控制器或作用域中访问根作用域（$rootScope）及其他的资源对象，需指明，

格式为：

myapp.controller('PcController' ,['$scope','$rootScope','$filter',**function** ($scope,$rootScope,$filter) {  
 $scope.FirstName = "PengFei";  
 $scope.FirstName= $filter('lowercase')("pengfei");  
 $scope.LastName = "zeng";  
 $scope.Age = 1;  
 $scope.getFullName = **function**(){  
 $rootScope.rootinfo.a = $scope.FirstName + " " + $scope.LastName;  
 **return** $scope.FirstName + " " + $scope.LastName ;  
 }  
}]  
);

“作用域”就是初始化时的函数，所有变量或者函数均可以认定为初始化函数中的局部变量。（想法够新，够劲）访问根作用域或者其他资源对象可以理解为访问形参变量

### 2、表达式

使用{{}}进行标识，可以将表达式与evel函数等同理解。其中两个花括号中的使用的变量和函数可直接使用作用中的。此处不需要加$scope前缀。本人的理解是，此处会将两个花括号中的内容放到对应的作用域（函数）中执行evel函数（未认证过）。

### 3、过滤器

其主要目的是将数据信息格式化以后进行输出 使用方式是$filter(‘*formatstr*’)(TEXT)或者{{TEXT | *formatstr* }}

1. Currency 默认是本地货币不过我的环境貌似都是美元，据说可以调整货币类型，没试过
2. Date 年月日的各种输出，忘记的话，翻书饭API（可带毫秒）
3. filter 根据字符过滤对象级别过滤支持函数级别过滤
4. json 直接将字符串变成json格式数据
5. limitTo 截取，毫不留情的截肢手术
6. lowercase 转小写
7. number 格式化成千分位显示
8. orderBy 排序来了，支持函数级别的操作
9. uppercase 转大写

自定义过滤器：

根据控制器创建局部过滤器：

myapp.controller('PcController' ,['$scope','$rootScope','$filter',**function** ($scope,$rootScope,$filter) {  
  
 $scope.FirstName = "PengFei";  
 $scope.FirstName= $filter('lowercase')("pengfei");  
 $scope.LastName = "zeng";  
 $scope.Age = 1;  
 $scope.getFullName = **function**(){  
 $rootScope.rootinfo.a = $scope.FirstName + " " + $scope.LastName;  
 **return** $scope.FirstName + " " + $scope.LastName ;  
 }  
}]  
).filter('capitalize',**function** (){  
 **return function** (inputStr) {  
 **if**(inputStr){  
 **return** inputStr[0].toUpperCase();  
 }  
 }  
});

表单检查部分：

required 不能为空

ng-minlength={*number*}最小长度.

ng-maxlength={*number*}最大长度.

ng-pattern=”[a-zA-Z]” 正则的干活

type=’email’ 电子邮件此处语法有点不一致

type=’number’ 仅数字此处语法有点不一致

type=’url’ 仅URL此处语法有点不一致

其实正则就够了，学好正则，伴您编程一生-\_-

表单验证变量使用：  
 formName.inputFiledName.*property*

$pristine 未修改过是True

$dirty 修改过是True 和上面这个是相反的业务

$valid 表单内容合法为True

$invalid 表单内容不合法为True，和上面这个相反

$error 内容验证失败为True

表单验证Css样式

ng-pristine、ng-dirty、ng-valid、ng-inalid表单在验证的时候会在对应的标签上添加对应的CSS样式，此处可以添加对应的css样式可以增强交互体验

### 4、自定义指令

ng-app 定义根作用域可以放在html,body,div,一定要赋值，代码还需有规则,注意：使用时需要使用符号隔开驼峰部分“my-Directive”

<div ng-app="MyApp">  
 <h1>Directive Test</h1>  
  
 <div ng-controller="PcController" style="width: 300px;margin: auto;" my-Directive>  
 </div>  
</div>  
<script type="text/javascript">  
 **var** myapp = angular.*module*('MyApp',[]).directive('myDirective',**function** () {  
 **return**{  
 restrict:'EACM',  
 replace:**true**,  
 template:'<button ng-click="showName" class="btn btn-primary">ShowName</button>'  
 };});

Restrict配置信息：及初始值

E 元素 <my-directive=”*initValue*”></ my-directive >

A 属性 <div my-directive=”*initValue*”></div>

C 类 <div class=”my-directive:*initValue*”></div>

M注释 <!-- directive:mydirective :*initValue* -->

推荐使用 属性模式，各种好处不一一列举

如何在模板里面增加动态字段绑定还需研究研究

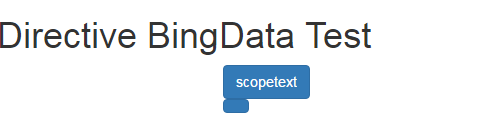
2016年8月22日22:12:43

绑定字段的方式继续研究

<div ng-app="MyApp">  
 <h1>Directive BingData Test</h1>  
 <div ng-controller="PcController" style="width: 300px;margin: auto;" my-Directive button-Text="aaaa">  
 </div>  
 <div style="width: 300px;margin: auto;" my-Directive button-Text="aaaa">  
 </div>  
</div>  
<script type="text/javascript">  
 **var** myapp = angular.*module*('MyApp',[]).directive('myDirective',**function** () {  
 **return**{  
 restrict:'EACM',  
 template:'<div style="width: 300px;margin: auto;"> \  
 <button ng-click="showName" class="btn btn-primary">{{ buttonText }}</button>\  
 </div>'  
 };});  
 myapp.controller('PcController' ,['$scope','$rootScope','$filter',**function** ($scope,$rootScope,$filter) {  
 $scope.FirstName = "PengFei";  
 $scope.FirstName= $filter('lowercase')("pengfei");  
 $scope.LastName = "zeng";  
 $scope.Age = 1;  
 $scope.buttonText="scopetext";  
 $scope.getFullName = **function**(){  
 $rootScope.rootinfo.a = $scope.FirstName + " " + $scope.LastName;  
 **return** $scope.FirstName + " " + $scope.LastName ;  
 }  
 }]  
 ).filter('capitalize',**function** (){  
 **return function** (inputStr) {  
 **if**(inputStr){  
 **return** inputStr[0].toUpperCase();  
 }  
 }  
 });  
</script>

以上的编写方式可以将业务与模板混淆，此处需谨慎小心使用，补课堕入无间道（意外收获）

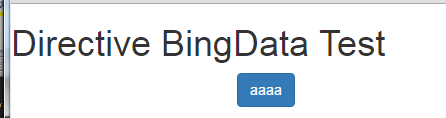
效果如下：



初始化方式实现：

div ng-app="MyApp">  
 <h1>Directive BingData Test</h1>  
 <div style="width: 300px;margin: auto;" my-Directive button-Text="aaaa">  
 </div>  
</div>  
<script type="text/javascript">  
 **var** myapp = angular.*module*('MyApp',[]).directive('myDirective',**function** () {  
 **return**{  
 restrict:'EACM',  
 replace:**true**,  
 scope:{buttonText:'@'},  
 template:'<div style="width: 300px;margin: auto;"> \  
 <button ng-click="showName" class="btn btn-primary">{{ buttonText }}</button>\  
 </div>'  
 };});  
 </script>

此处需注意的地方是 scope属性值的设置，DOM节点中初始化的数据如果要传递进来的话，就需要使用buttonText:'@'的方式声明

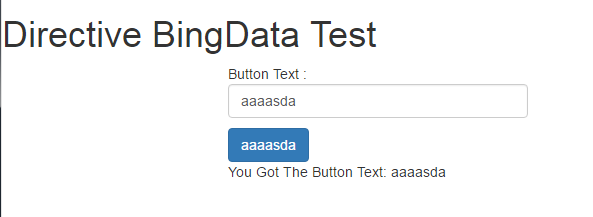


与控制器数据绑定操作：

<div ng-app="MyApp">  
 <h1>Directive BingData Test</h1>  
 <div ng-controller="PcController" style="width: 300px;margin: auto;" >  
 <p> Button Text :<input type="text" class="form-control" ng-model="btnText" placeholder="Enter Button Text "> </p>  
 <div style="width: 300px;margin: auto;" my-Directive btn-Text="btnText">  
 </div>  
 <p>You Got The Button Text: {{ btnText }}</p>  
 </div>  
</div>  
<script type="text/javascript">  
 **var** myapp = angular.*module*('MyApp',[]).directive('myDirective',**function** () {  
 **return**{  
 restrict:'EACM',  
 replace:**true**,  
 scope:{buttonText:'=btnText'},  
 template:'<div style="width: 300px;margin: auto;"> \  
 <button ng-click="showName" class="btn btn-primary">{{ buttonText }}</button>\  
 </div>'  
 };});  
 myapp.controller('PcController' ,['$scope','$rootScope','$filter',**function** ($scope,$rootScope,$filter) {  
 $scope.FirstName = "PengFei";  
 $scope.FirstName= $filter('lowercase')("pengfei");  
 $scope.LastName = "zeng";  
 $scope.Age = 1;  
 $scope.buttonText="scopetext";  
 $scope.getFullName = **function**(){  
 $rootScope.rootinfo.a = $scope.FirstName + " " + $scope.LastName;  
 **return** $scope.FirstName + " " + $scope.LastName ;  
 }  
 }]  
 ).filter('capitalize',**function** (){  
 **return function** (inputStr) {  
 **if**(inputStr){  
 **return** inputStr[0].toUpperCase();  
 }  
 }  
 });  
</script>

此处需要注意的问题是DOM节点配置的时候模板处btn-Text="btnText"其中btnText是作用域里面的字段名称，模板里面的scope配置不可以直接引用DOM处的btn-Text去对应地方填充值，而是需要转换一下：buttonText:'=btnText' 外面使用 btnText 模板内部使用buttonText，很奇怪的赋值表达式。

效果如下



此处目前学到这里，其他高级用法还需后续进行研究

### 5、内部指令

布尔型： 取值除了布尔型变量，普通变量当有值时理解为True，无值时理解为False

ng-disabled 直接绑定到DOM标签的属性上 效果是禁用。适用范围：<input>(text、checkbox、redio、number、url、email、submit),<textarea> ,<select> ,<button>

ng-switch case… when …. Then … DB Edition

ng-switch-when DOM中赋值的内容为常量 可预置多条供选择，可以是代码块，也可以是常量，同样还可以是一个变量的表达式，总之符合AngularJS规范的代码段都可。

<div class="highlight">  
 <h2>ng-switch</h2>  
 <p> Switch Text :<input type="text" class="form-control" ng-model="switchData.switchKey"  
 placeholder="Enter Switch Text "/>  
 </p>  
 <div ng-switch on="switchData.switchKey">  
 <p ng-switch-default>{{switchData.switchDefaultValue}}</p>  
 <h3 ng-switch-when="switchok">{{switchData.switchShowInfo}}</h3>  
 </div>  
</div>

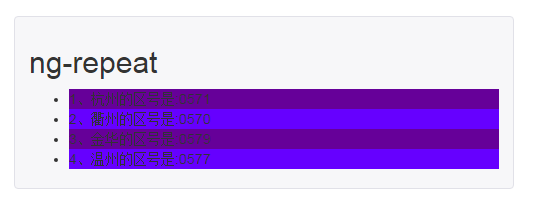
ng-if同ng-show和ng-hide不单独举例说明，其中最本质的区别就是ng-if是删除或生成节点。

ng-if模式下对于在Dom上做的一些操作需要慎重，改模式下是拿模板重新生成的。

ng-repeat顾名思义是重复的意思，讲的是根据模板重复生成多段代码块放置到对应的DOM中去

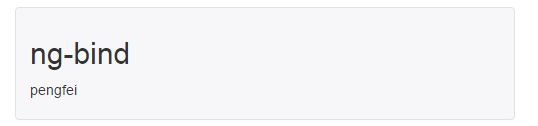
此处举例使用ul==>li

<!-- ng-repeat -->  
<div class="highlight">  
 <h2>ng-repeat</h2>  
 <ul>  
 <li ng-repeat="tmpdict in listData" ng-class="{even : !$even ,odd : !$odd}">  
 {{$index + 1}}、{{tmpdict.dictName}}的区号是:{{tmpdict.dictID}}  
 </li>  
 </ul>  
</div>



ng-bind使用效果与{{}}类似

<!-- ng-bind -->  
<div class="highlight">  
 <h2>ng-bind</h2>  
 <P ng-bind="FirstName"></p>  
</div>



ng-change使用手法是值发生变化时的一个通知事件的接口，不单独举例说明

ng-click 使用点击时的一个通知事件的接口，不单独举例说明

ng-class 包装了默认的class属性，此处的优势在于支持表达式，例如ng-class=”{red: x> 5}”此处X>5是一个表达式，如果表达式返回True,那就给所在DOM元素加上red的样式属性

ng-select 此示例中 selectedValue 是选中后的数据，listData是选项列表，tmpdict是中间临时变量，不需要再作用域当中声明，后面展示选中的值时，如果表单中需要可以做内部绑定。（精髓在于配置就可以完成功能，还需再研究研究）

<!-- ng-select -->  
<div class="highlight">  
 <h2>ng-select</h2>  
 <select ng-model="selectedValue" ng-options="tmpdict.dictName for tmpdict in listData">  
 <option value="" >请选择...</option>  
 </select>  
 <h4 ng-show="selectedValue.dictID">您选择的区号是:{{selectedValue.dictID}} 地区是:{{selectedValue.dictName}}</h4>  
</div>

ng-href 和 ng-src 有相似之处，需要等数据到位以后才能生效，提升体验之处。

div class="highlight">  
 <h2>ng-href</h2>  
 <a ng-href="{{ myhref }}">此处延迟两秒加载</a>  
</div>

**var** myapp = angular.*module*('MyApp', []).run(**function**($rootScope,$timeout){  
 $timeout(**function** () {  
 $rootScope.myhref="www.g.cn";  
 },2000);  
});

2016年8月25日21:49:17

ng-form 此处表单主要任务就是实现各种验证，表单的值一开始就已经被存储在$scope当中了

Form的各级元素需填写name属性。此属性中可标识各种维度的检查

{"$error":{"minlength":[{"$viewValue":"aa","$validators":{},"$asyncValidators":{},"$parsers":[],"$formatters":[null],"$viewChangeListeners":[],"$untouched":false,"$touched":true,"$pristine":false,"$dirty":true,"$valid":false,"$invalid":true,"$error":{"minlength":true},"$name":"email","$options":null}]},"$name":"test\_form","$dirty":true,"$pristine":false,"$valid":false,"$invalid":true,"$submitted":false,"email":{"$viewValue":"aa","$validators":{},"$asyncValidators":{},"$parsers":[],"$formatters":[null],"$viewChangeListeners":[],"$untouched":false,"$touched":true,"$pristine":false,"$dirty":true,"$valid":false,"$invalid":true,"$error":{"minlength":true},"$name":"email","$options":null}}

Form元素在使用检查数据和ng-class改变元素的展现方式时，需要使用页面中直接定义的class（尝试10分钟+，想将BootStrap中的alert-warning在检查不通过时添加上去，但是就是不行！（谁能告诉我为什么）

各种验证如果忘记需要查看数的7.2章节：表单验证，代码量比较多，能写进笔记的少。很多东西一眼看过去就能清楚

代码如下：

<div class="highlight">  
 <h2>ng-form</h2>  
 <form name="test\_form" ng-submit="submitForm()" novalidate name="testform">  
 <div class="form-group">  
 <label for="city" >City</label>  
 <select id="city" class="form-control" ng-model="sigFormData.city" ng-options="tmpdict.dictName for tmpdict in listData">  
 <option value="" >请选择...</option>  
 </select>  
 </div>  
 <div class="form-group">  
 <label for="email" >Email</label>  
 <input id="email" type="text" name="email" class="form-control" ng-minlength="5" ng-class="{sigwarning : test\_form.email.$invalid && test\_form.email.$dirty}" ng-model="sigFormData.email" placeholder="Enter Your Email" ng-required="true" />  
 <div> </div>  
 </div>  
 <div class="form-group">  
 <label for="password">Password</label>  
 <input id="password" type="password" name="password" class="form-control" ng-minlength="6" ng-model="sigFormData.password" placeholder="Enter Your Password"/>  
 <div class="error" ng-show="test\_form.password.$dirty && test\_form.password.$invalid">  
 <small class="alert-warning" ng-show="test\_form.password.$error.minlength">您输入的密码少于六位！</small>  
 </div>  
 </div>  
 <div class="form-group">  
 <label for="savepass">Save Password：</label>  
 <input id="savepass" type="checkbox" name="issaved" class="checkbox-inline" ng-model="sigFormData.issaved"/>  
 </div>  
 <div class="btn-group" data-toggle="buttons">  
 <label class="btn btn-primary" ng-class="{active : sigFormData.isneed }">  
 <input type="checkbox" name="isneed" ng-model="sigFormData.isneed" autocomplete="off" checked> Need  
 <span class="glyphicon glyphicon-star" ng-show="sigFormData.isneed" aria-hidden="true"></span>  
 </label>  
 </div>  
 <p>{{sigFormData}}</p>  
 <p>{{test\_form}}</p>  
 </form>  
</div>

2016年8月26日21:27:17