



AngularJS 高级(一)

讲师：文元



泛IT职业教育机构

信·机构

- 慧科教育集团旗下泛IT职业教育培训机构。
- 与找座儿形成国内第一家O2O混合式教学落地基地。
- 良心教学，可信赖大机构！

新·模式

- 同时采用最合理的TS教学模式和最合理教学时间安排。
- 全程手把手教学、边学边练，7*12小时答疑指导，保证学员即学即会。

心·服务

- 多年教学经验资深讲师倾心教学
- 配备专门的生活导师，提供入学接送、精选宿舍、生活指导、学习督促等一条龙服务。
- 提供专业就业指导服务，简历指导+模拟面试，保证快速就业。

薪·未来

- 无限互联学生毕业平均薪资12755元，比同行业机构平均高出12%。
- 98%的无限互联学生在毕业4周内成功就业！

- 依赖注入与双向数据绑定
- AngularJS 表单(一)
- AngularJS 表单(二)
- AngularJS 的服务
- 模块中的服务注册方法
- 自定义指令

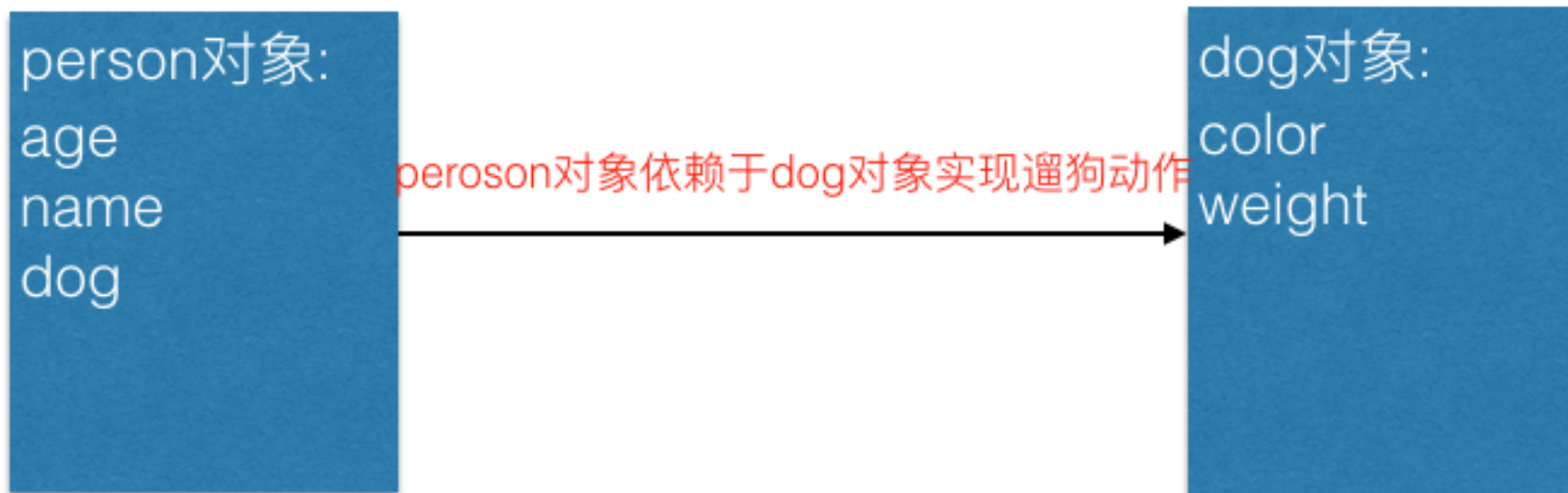


一、依赖注入与双向数据绑定

1. 依赖注入的介绍
2. 依赖注入的原理
3. 双向数据绑定

- 依赖注入(DI)源于 Java 里面的 spring 框架,对象在被创建的时候,由一个调控系统内所有对象的外界实体将其所依赖的对象的引用传递给它。也可以说,依赖被注入到对象中(例如: `new Array()`,创建的这个数组对象必须依赖于 `Array` 对象)
- 在 AngularJS 中,代码间的依赖处理则非常轻松,通过 AngularJS 中,代码间的依赖处理则非常轻松,通过 AngularJS 中特有的依赖注入方式,将依赖的对象轻松注入任意需要的地方,而且不必关注被注入对象本身的逻辑,这种方式减轻了代码开发量,并且提高了工作效率
- 依赖注入是 AngularJS 特性之一

- 依赖的例子: 比如要实现一个人遛狗的功能,那么首先得有 person 对象, person 对象里面有自己的属性方法,然后,要实现遛狗这个行为,必须得有 dog 对象, 而 dog 对象又有自己的属性方法;这个时候,要实现人遛狗这个动作的话就必须 person 对象依赖于 dog 对象来实现,不然就无法完成此动作



- 依赖注入,从字面上来说,它分为两个部分: 一是依赖,另一部分是注入
- 当一个对象在建立时,需要依赖于另一个对象,这是代码层的一种依赖关系,当代码中声明了依赖关系之后, AngularJS 通过 injector 注入器将所依赖的对象进行注入操作

- 在 AngularJS 中,每一个应用都有一个 injector 注入器来处理依赖的创建,注入器实际上是一个负责查找和创建依赖的服务定位器,所以声明的依赖注入对象都是由它来进行处理;此外,当获取 injector 注入器对象后,还可以调用该对象的 get 函数来获得任何一个已经被定义过的服务的实例
- ng-bind-html 是用来解析 html 代码的,但是有个条件,必须得让模块依赖 ngSanitize,不然就无法完成 html 代码的解析
- 控制器里面的 \$scope,拥有许多的属性方法,这些属性方法都是 AngularJS 中定义好的,实际上是通过 injector 注入器完成了此对象的注入
- 后期会具体地讲述 injector 的实现原理

- 双向数据绑定,指的是可以通过修改数据来改变视图的显示,也可以通过操作视图改变数据
- 双向数据绑定是 AngularJS 特性之一
- 双向数据绑定最经典的场景----表单
- 通过表单案例来演示双向数据绑定

二、AngularJS 表单

1. 表单基本验证功能
2. 表单中的 checkbox 和 radio 控件

- 表单是各类控件(如 input 、 select 、 textarea)的集合体,这些控件依附于表单,形成 DOM 元素中最为重要的数据交互元素,AngularJS 也对表单中的控件做了专门的包装,其中最重要的一项就是控件自我验证功能

注册帐号

表单验证

昵称

❗ 昵称不可以为空

密码

ℹ 长度为6-16个字符

ℹ 不能包含空格

ℹ 不能是9位以下纯数字

确认密码

❗ 请再次输入密码

- 在 AngularJS 中,专门针对表单和表单中的控件提供了下列属性,用于验证控件交互值的状态
 - \$pristine 表示表单或控件内容是否未输入过
 - \$dirty 表示表单或控件内容是否已输入过
 - \$valid 表示表单或控件内容已通过验证
 - \$invalid 表示表单或控件内容未通过验证
 - \$error 表示表单或控件内容验证时的错误提示信息
- 前四项属性均返回布尔类型的值,最后一项属性返回一个错误对象,包含全部表单控件验证时的返回的错误信息

- 可以用 AngularJS 做一个简单的表单验证功能,效果如图所示:

姓名:

密码:

提交

姓名: 姓名不能为空

密码: 邮件不能为空

提交

姓名: 姓名不能为空

密码: 邮件格式不对

提交

- required : 是一个布尔属性, required 规定必需在提交表单之前填写输入字段
- ng-minlength : 最小长度
- ng-maxlength : 最大长度
- ng-change : 内容改变时触发
- ng-pattern: 正则匹配

- 验证显示的格式为: 表单的 name 值.标签的 name 值.\$error.ng 后面的单词

```
<span ng-show="test_form.t_name.$error.required">  
    姓名不能为空  
</span>  
<span ng-show="test_form.t_name.$error.minlength">  
    长度不能小于3  
</span>  
<span ng-show="test_form.t_name.$error.pattern">  
    数字开头  
</span>
```

- 在表单控件中,checkbox 控件和 radio 控件与 `<input>` 元素的其他类型控件不同,这两个控件不具有 AngularJS 的控件验证功能
- checked 有选中和非选中两种状态, radio 只有一种选中状态
- checkedbox 控件和 radio 控件都可以通过 `ng-model` 指令绑定控制器的属性,一旦绑定完成,在首次加载时,将以绑定的属性值作为控件初始化的状态

表单中的 checkbox 和 radio 控件

- 在 checkbox 中,添加两个指令: ng-true-value,ng-false-value
- ng-true-value: 表示选中时返回的值
- ng-false-value: 表示未选中时返回的值
- 设置的值必须是表达式

```
input type="checkbox" ng-model="a" ng-true-value="'同意'" ng-false-value="'不同意'" 
```

- 在 radio 中,只要将多个控件的 ng-model 指令值设置为相同,这些 radio 类型控件就只有一个选中时的值,并且,当一个选中时,其他控件自动变成非选中的状态

性别:

```
<input type="radio" ng-model="b" value="男">男  
<input type="radio" ng-model="b" value="女">女
```

三、AngularJS 表单(二)

1. 表单中的 select 控件
2. 绑定数据的几种方式

- 在 AngularJS 中,与其他表单中的控件元素相比,select 控件的功能要强大很多,它可以借助 ng-options 指令属性,将数字、对象类型的数据添加到 <option> 元素中,同时还能在添加数据时进行分组显示

- 绑定简单的数组数据: `$scope.data = ['a','b','c'];`

```
<select ng-model="d" ng-options="txt for txt in data">
  <option value="">
    --请选择--
  </option>
</select>
```

- 在 `<select>` 控件中,通过 `ng-options` 指令设置属性值,采用 `...for...in...` 格式将数组与 `<select>` 控件绑定
- 必须添加 `ng-model` 属性,否则无法绑定控制器中的数组,并且 `ng-model` 的属性值就是 `<select>` 类型控件的选中值,它们之间是双向绑定关系

绑定数据的几种方式

- 绑定简单的数组对象数据:

```
$scope.data2 = [  
  {id: '1', name: '张三'},  
  {id: '2', name: '李四'},  
  {id: '3', name: '王五'},  
  {id: '4', name: '赵六'}  
];
```



采用..as..for..in..的格式绑定

```
<select ng-model="d" ng-options="txt.id as txt.name for txt in data2">  
  <option value="">  
    --请选择--  
  </option>  
</select>
```

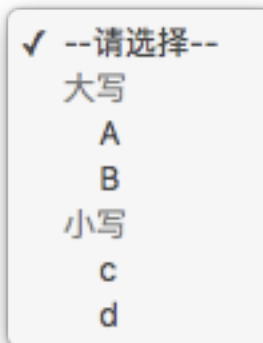
选取时获取的值

显示的值

绑定数据的几种方式

- 以分组的形式绑定对象数据:

```
$scope.data3 = [  
  {id: '1', name: 'A', key: '大写'},  
  {id: '2', name: 'B', key: '大写'},  
  {id: '3', name: 'c', key: '小写'},  
  {id: '4', name: 'd', key: '小写'}  
];
```



采用..as..group by..for..in..的格式绑定

```
<select ng-model="d" ng-options="txt.id as txt.name group by txt.key  
  for txt in data3">  
  <option value="">  
    --请选择--  
  </option>  
</select>
```

选中值 显示值 分组值

四、AngularJS 服务(service)

1. 服务的介绍
2. 内置服务的简单介绍
3. 自定义服务
4. 使用 `$provide` 自定义服务

- 到目前为止,我们只关心视图是如何同 `$scope` 绑定在一起,以及控制器是如何管理数据的,出于内存占用和性能的考虑,控制器只会在需要时被实例化,并且不需要就会被销毁,这意味着每次切换路由或重新加载视图的时候,当前的控制器会被 AngularJS 清除掉
- 服务提供了一种能在应用的整个生命周期内保持数据的方法,它能够在控制器之间进行通信,并能保证数据的一致性
- 服务是一个单例对象,在每个应用中只会被实例化一次(被 `$injector` 实例化),并且是延迟加载的(需要时才会被创建)
- 服务提供了把与特定功能相关联的方法集中在一起的接口

- 应用里面大部分的业务逻辑和持久化数据都应该放在服务里面
- 按照功能的不同,它又可以分为内置服务和自定义服务
- 内置服务的命名以 \$ 开头,自定义服务应该避免

- AngularJS 提供了许多内置的服务,例如常用的 `$scope`, `$http` 和 `$location` 等
- 可以在控制器中直接调用内置服务,而无需访问服务所涉及的底层代码,从而确保整个应用的结构不被污染
- 内置服务在应用中任何地方调用的方法都是统一的,通过直接调用这些服务,也可以将复杂的应用功能进行简化和分块话,从而提升代码开发的效率

- 在下面的代码中,通过依赖注入的方式向控制器注入了一个名为 `$location` 的内置服务,此代码中,没有通过数组进行声明,这种简单的注入称之为隐性注入服务
- 一旦在控制器中注入了服务,就可以直接访问该服务对象中包含的属性和方法,本例的 `absUrl` 方法,功能是返回当前地址栏中的 URL 地址
- `$location` 服务除了包含 `absUrl` 方法外,还有其他的一些方法,在后期将会详细介绍此些内置服务

```
app1.controller('firstCtrl',function ($scope,$location) {  
    $scope.url = function () {  
        $scope.url = $location.absUrl();  
    }  
});
```

- 内置服务的方法使用相当简单,只需要将服务注入需要服务的容器中(如控制器、指令或其他自定义的服务),就可以采用对象的方式调用服务中包含的各个属性和方法,操作简单
- 内置服务功能强大,但是都是一些通用的功能,当开发 AngularJS 应用的时候,大部分的逻辑和功能都需要自定义
- 为了更好的运用 AngularJS 服务功能,开发者可以自定义符合应用本身业务逻辑的服务

- 定义服务的方法主要包含以下两种
 - 使用内置的 `$provide` 服务
 - 调用模块中的服务注册方法

使用 \$provide 自定义服务

```
$provide.factory('output',function () {  
    console.log('被实例化一次');  
    var stu = {  
        name: '张三',  
        sex: '男',  
        score: 60  
    };  
    return stu;  
});
```

服务名称

服务返回的对象

- 在以上代码中,当定义模块时,以依赖注入的方式添加了一个名为 \$provide 的内置服务,调用该服务对象的工厂函数 factory,自定义了一个名为 output 的服务,为了避免与其他对象或服务冲突,自定义服务的名称前缀通常为一个 不加 \$ 符号
- 在 \$output 服务中,定义了一个名为 stu 的 json 对象,并通过 return 语句返回该对象,即该项目定义的服务功能是返回一个 json 格式的对象,用于控制器的调用

- 在控制器代码中,无论是内置还是自定义的服务,调用方式都是一样的,都是通过依赖注入的方式向控制器添加服务
- 由于自定义服务和内置服务都只是在注入时被实例化一次,因此,服务被注入的动作实际上是 Angular 编译器引入实例化对象的过程,所以,当通过依赖注入的方式向控制器添加自定义服务时,这个服务已经是一个实例化后的服务对象

```
app1.controller('firstCtrl',function ($scope,output) {  
    $scope.info = $output;  
});
```

五、模块中的服务注册方法(一)

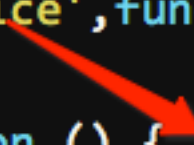
1. `provider()`
2. `constant()` 和 `value()`
3. `factory()`
4. `service()`

- 在 AngularJS 应用中, `factory()` 方法是用来注册服务的最常规方式, 同时还有其他一些 API 可以在特定情况下帮助开发者减少代码量, 共有五种方法来创建服务
 - `constant`
 - `value()`
 - `provider()`
 - `factory()`
 - `service()`

- `$provide` 服务负责告诉 Angular 如何创建一个新的可注入的东西：即服务。服务会被叫做供应商的东西来定义，你可以使用 `$provide` 来创建一个供应商。你需要使用 `$provide` 中的 `provider()` 方法来定义一个供应商，同时你也可以通过要求 `$provide` 被注入到一个应用的 `config()` 函数中来获得 `$provide` 服务，注意在 `config()` 函数中，只有 `provider()` 才能被注入
- `provider`, `value`, `constant`, `service`, `factory`, 本质上都是 `provider`, 只是以上方式用了不同简单的快捷写法, 让人看的更舒服一点
- `provider` 的基本原则就是通过实现 `$get` 方法来在应用中注入单例
- `provider` 里面必须得有 `$get` 方法

- provider() 接受两个参数,name 和 fn
- name 是需要注册的服务的名字
- fn 是实现的函数
- fn 函数 可以返回简单类型、函数乃至对象等任意类型的数据
- 调用此服务的时候,得到的结果则为 \$get 方法里面的返回值

```
//创建服务
app1.provider('ageService',function () {
  this.start = 18;
  this.$get = function () {
    return this.start + 2;
  }
});
```



服务名称

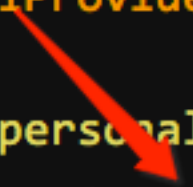
```
app1.provider('personal',function () {  
  this.name = '';  
  this.$get = function (){  
    var service = {};  
    var that = this;  
    service.setName = function (name) {  
      this.name = name;  
    };  
    service.getName = function () {  
      return '返回对象的名字为:' + this.name + ',服务的名字为' + that.name;  
    }  
  
    return service;  
  }  
});
```



服务名称

- 如果希望在 config() 函数中可以对服务进行配置,必须用 provider() 来定义服务
- 如图所示,在 config() 里面的函数中,注入了一个 provider ,此 provider 是 personal 服务的提供者,可以通过服务的提供者对服务进行一些配置
- 服务的提供者: 服务名称 + Provider

```
//配置服务
app1.config(function (personalProvider) {
    personalProvider.name = 'personal';
});
```



服务提供者

- 可以将一个已经存在的变量值注册为服务,并将其注入到应用的其他部分当中,例如,假设需要给后端服务一个 apiKey, 可以用 constant() 将其当做常量保存下来
- constant ()函数可以接受两个参数
- name 服务的名字
- value 需要注册的常量的值 (值或者对象)
- 常量是不能更改的
- 常量服务可以像其他服务一样被注入到配置函数中

```
//常量服务
```

```
app1.constant('apiKey',123456);
```


- 如果服务的 \$get 方法返回的是一个常量,那就没必要定义一个包含复杂功能的完整服务,可以通过 value() 函数方便地注册服务
- value() 函数可以接受两个参数
- name 服务的名字
- value 需要返回的值
- 值是可以改变的
- 值服务不能注入到配置函数中

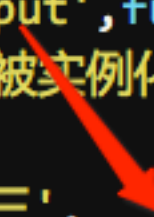
```
//值服务  
app1.value('Fbi',22222);
```

6、模块中的服务注册方法(二)

1. `factory()`
2. `service()`
3. `service`、`factory`、`provider` 三者区别

- factory() 可以接受两个参数,name 和 fn
- name 是需要注册的服务的名字
- fn 是函数,此函数会在 AngularJS 创建服务实例时被调用

```
//创建服务
app1.factory('output',function(){
    console.log('被实例化一次');
    var stu = {
        name: '张三',
        sex: '男',
        score: 60
    };
    return stu;
});
```



服务名称

- 服务是单例对象,fn 函数在应用的声明周期内只会被调用一次
- fn 函数可以返回简单类型、函数乃至对象等任意类型的数据
- fn 函数里面也可以注入其他服务

```
app1.factory('address',function($location){  
    var location = $location.absUrl();  
    return location;  
});
```

- 使用 service() 可以注册一个支持构造函数的服务,允许开发者为服务对象注册一个构造函数
- service() 接受两个参数
- name 需要注册的服务的服务名称
- constructor 构造函数,调用它来实例化服务对象
- service() 会在创建实例时通过 new 关键字来实例化服务对象

- 在此服务中,定义了一个方法 getName(),在服务调用的时候,只需用服务对象调用此方法就行
- 注意: service() 方法不能返回字符串,如果返回字符串的话,则结果为此构造函数

```
app1.service('personService',function(){  
    this.getName = function(){  
        return '张三';  
    }  
    return 'zs'; //不能返回字符串  
});
```



服务名称

- 用 factory 就是创建一个对象,为它添加属性方法,然后把这个对象返回出来,把服务传进 controller 之后,在 controller 里这个对象里面的属性就可以通过 factory 来使用了
- service 相当于创建了一个构造函数,直接给 this 添加属性方法就可以
- provider 是唯一一种可以传进配置函数的服务,当你想要在服务对象启动之前,先进行模块范围的配置,那就应该考虑使用 provider
- factory/service 是第一个注入时才实例化,而 provider 不是,它是在 config 之前就已实例化好

- 依赖注入与双向数据绑定是 AngularJS 的特征
- ng-bind-html 是用来解析 html 代码的,但是有个条件,必须得让模块依赖 ngSanitize,不然就无法完成 html 代码的解析
- ng-true-value 与 ng-false-value 指令使用的时候要注意设置的值一定得是表达式
- 服务提供了一种能在应用的整个生命周期内保持数据的方法,是一个单例对象,在每个应用中只会被实例化一次(被 \$injector实例化),并且是延迟加载的(需要时才会被创建)

- provider, value, constant, service, factory 都可以用来自定义服务,本质上都是 provider
- 使用 provider 定义服务的时候必须得有 \$get 方法
- value 里面的值是可以改变的, constant 里面的值是不能改变的
- factory 是创建一个对象,为其添加属性方法,再返回此对象
- service 相当于创建了一个构造函数,直接往 this 里面添加属性方法就行



慧科集团旗下企业



官方微信

未经允许不得将视频用于商业用途，否则将追究其法律责任！

官方网站： <http://www.wuxianedu.com/>

无限智造 互联精英

版权所有：无限互联