Web前端开发

第17章：AngularJS

阮晓龙

13938213680 / rxl@hactcm.edu.cn
http://web.book.51xueweb.cn
http://www.51xueweb.cn

河南中医药大学管理科学与工程学科

2018.5

本章主要内容

- AngularJS概述
- AngularJS基本概念
- AngularJS应用
- 案例：使用AngularJS实现即时搜索

1.AngularJS概述

- AngularJS是一个JavaScript类库
  - 一个用来开发动态Web应用的框架
  - 通过<script>标签添加到HTML页面。
  - 以HTML为模板语言并通过扩展的HTML语法使应用组件更加清晰和简洁。
  - AngularJS的创新之处在于，通过数据绑定和依赖注入减少了大量代码，而这些都在浏览器端通过JavaScript实现，能够和任何服务端技术完美结合。

1.AngularJS概述

- 相关网址
  - AngularJS官方下载地址：https://angularjs.org
  - AngularJS中文网：http://www.angularjs.net.cn

1.AngularJS概述

- AngularJS为克服HTML在构建应用上的不足而设计，有着诸多特性，其中最为核心的是：
  - MVC
  - 模块化
  - 自动化双向数据绑定
  - 依赖注入

1.AngularJS概述

- MVC
  - MVC是一种代码结构组织方式，由模型（Model）、视图（View）、控制器（Controller）3部分组成。
    - 模型（Model）：是应用程序中处理数据逻辑的部分，通常负责在数据库中存取数据。
    - 视图（View）：用户看到并进行交互操作的界面。
    - 控制器（Controller）：应用程序中处理用户交互的部分，通常负责从视图读取数据，控制用户输入，并向模型发送数据。
  - 这种开发模式能够合理组织代码，降低代码间耦合度，方便后期维护。
  - 在Angular应用中，视图就是Dom、控制器就是JavaScript，模型数据存储在对象的属性中。

1.AngularJS概述

- 模块化
  - 使用AngularJS构建应用时采用模块化方式组织代码，将整个应用划分为若干个模块，每个模块都有特定的职责。采用模块化的组织方式可以最大程度实现代码复用，使开发像搭积木一样进行。AngularJS中的模块主要分官方提供模块和自定义模块两种：
    - 官方提供的模块包括ng、ngRoute、ngAnimate、ngTouch等
    - 用户自定义的模块通过angular.module('模块名',[ ])创建

1.AngularJS概述

- 自动化双向数据绑定
  - 在传统JS框架中，页面的HTML代码与数据混合在一起，AngularJS则在视图和模型之间建立映射关系，实现数据的自动同步：：
    - 方向一：Model->View，{{Model数据}} 或<XXX ng-xxx="Model数据"> ，Model改变View跟着改。
    - 方向二：View->Model，<表单控件 ng-model="Model数据名"> ，View变化Model跟着变化

1.AngularJS概述

- 依赖注入
  - 依赖注入（Dependency Injection,简称DI）是一种设计模式，一个或更多的依赖（或服务）被注入（或者通过引用传递）到一个独立的对象（或客户端）中，然后成为了该客户端状态的一部分。该模式分离了客户端依赖本身行为的创建，并遵循依赖反转和单一职责原则，这使得程序变得松耦合，可扩展性更强。

1.AngularJS概述

- AngularJS框架由以下三个主要部分组成：
  - ng-app指令，在HTML中定义AngularJS应用程序。
  - ng-model指令，把元素值（比如输入域的值）绑定到应用程序。
  - ng-bind指令，把应用程序数据绑定到HTML视图。

2.AngularJS基本概念

- AngularJS中一些基本概念如下：

表 17-1 AngularJS 概念总览

概念	说明
模板（Template）	带有 Angular 扩展标记的 HTML
指令(Directive)	用于通过自定义属性和元素扩展 HTML 的行为
模型(Model)	用于显示给用户并且与用户互动的数据
作用域(Scope)	用来存储模型(Model)的语境(context)。模型放在这个语境中才能被控制器、指令和表达式等访问到
表达式(Expression)	模板中可以通过它来访问作用域（Scope）中的变量和函数
编译器(Compiler)	用来编译模板(Template)，并且对其中包含的指令(Directive)和表达式(Expression)进行实例化
过滤器(Filter)	负责格式化表达式(Expression)的值，以便呈现给用户
视图(View)	用户看到的内容（即 DOM）
数据绑定(Data Binding)	自动同步模型(Model)中的数据和视图(View)表现
控制器(Controller)	视图(View)背后的业务逻辑
依赖注入(Dependency Injection)	负责创建和自动装载对象或函数
注入器(Injector)	用来实现依赖注入(Injection)的容器
模块(Module)	用来配置注入器
服务(Service)	独立于视图(View)的、可复用的业务逻辑

3.AngularJS应用

- AngularJS通过<script>标签添加到HTML页面进行初始化，主要有自动初始化、手动初始化两种方式：
  - 自动初始化
    - 在以下两种情况下完成自动初始化：
      - DOMContentLoaded事件触发时。
      - angular.js脚本执行时，document.readyState被设置‘complete’。
    - 初始化时，Angular首先找到表示应用开始位置的ng-app指令，然后执行以下过程：
      - 加载ng-app指令所指的模块。
      - 创建应用所需注入器（injector）。
      - 以ng-app所在的节点为根节点，遍历编译DOM树。

现场演示：

- 案例17-01：AngularJS自动初始化

3.AngularJS应用

- AngularJS通过<script>标签添加到HTML页面进行初始化，主要有自动初始化、手动初始化两种方式：
  - 手动初始化
    - 如果希望在初始化阶段拥有更多的控制权，在应用中使用脚本加载器或者想要在Angular编译页面之前执行其它操作，可以使用手动初始化方法。
    - 与自动初始化不同，手动初始化不使用ng-app指令，通过应用程序完成初始化。

3.AngularJS应用

- AngularJS通过<script>标签添加到HTML页面进行初始化，主要有自动初始化、手动初始化两种方式：
  - 手动初始化
    - 代码运行顺序如下：
      - HTML页面以及所有JS脚本加载完毕后，Angular找到文档根节点。
      - 调用angular.module()创建模块。
      - 调用api/angular.bootstrap加载模块，并将文档元素编译成一个可执行双向绑定的应用。

现场演示：

- 案例17-02：AngularJS手动初始化

3.AngularJS应用

- 指令是什么
  - 指令是一些附加在HTML元素上的自定义标记（如：属性、元素、CSS类）。
  - 它告诉AngularJS的HTML编译器（$compile）在元素上附加某些指定的行为，甚至操作DOM、改变DOM元素以及它的各级子节点。
  - AngularJS有一整套内置指令，如ngBind、ngModel和ngView，同时支持用户自定义指令。当Angular启动器引导应用程序时，HTML编译器遍历整个DOM，以匹配DOM元素里的指令。

3.AngularJS应用

- 指令是什么
  - AngularJS常用指令：

表 17-2 AngularJS 常用指令

指令	描述
ng-app	定义应用程序的根元素。
ng-bind	绑定 HTML 元素到应用程序数据。
ng-click	定义元素被单击时的行为。
ng-controller	为应用程序定义控制器对象。
ng-disabled	绑定应用程序数据到 HTML 的 disabled 属性。
ng-init	为应用程序定义初始值。
ng-model	绑定应用程序数据到 HTML 元素。
ng-repeat	为控制器中的每个数据定义一个模板。
ng-show	显示或隐藏 HTML 元素。

3.AngularJS应用

- 指令匹配
  - Angular的HTML编译器通过规范化的元素属性匹配要调用的指令。
  - Angular把一个元素的标签和属性名字进行规范化，来决定哪个元素匹配哪个指令。通常用区分大小写的规范化命名方式(比如ngModel)来识别指令。然而HTML是区分大小的，所以在DOM中使用的指令只能用小写的方式命名，除此之外还会使用破折号隔开(比如：ng-model)。

3.AngularJS应用

- 指令匹配
  - 规范过程
    - 从元素或属性的名字前面去掉“x-”和“data-”。
    - 从“:”、“-”或“_”分隔的形式转换成小驼峰命名法(camelCase)。如下代码，从input元素ng-model属性匹配了ngModel指令

```
<input ng-model="foo">
```

3.AngularJS应用

- Angular模板是一个声明式的视图，它将指定信息从模型、控制器变成用户在浏览器上可以看见的视图。
- 它引导Angular为一个只包含HTML、CSS以及Angular标记和属性的静态DOM加上一些行为和格式转换器，最终变成一个动态的DOM。
- Angular中的指令（Directive）、表达式（Expressions）、过滤器（Filter）和表单控件(Form Control)元素属性可直接在模板中使用。

3.AngularJS应用

- AngularJS表达式写在双大括号内“{{expression}}”，把数据绑定到HTML。AngularJS表达式可以包含文字、运算符和变量。
- 以下示例展示了数字、对象和数组在AngularJS表达式中的应用。

现场演示：

- 案例17-03：AngularJS表达式

3.AngularJS应用

- 作用域是一个存储应用数据模型的对象，其层级结构对应DOM树结构，为表达式提供了一个执行上下文，同时可以监听表达式的变化并传播事件。

3.AngularJS应用

- AngularJS应用由View（视图）、Model（模型）、Controller（控制器）组成。作用域是应用在视图和控制器之间的纽带。作用域中的属性方法可以在视图和控制器中使用，示例如下。

```
<div ng-app="demoApp" ng-controller="demoCtrl">
        <div><label>技术：<input type="text" ng-model="name" ></label></div>
        <div>内容输出：<span>{{show}}</span></div>
        <input type="button" ng-click="showTech()" value="确定" >
    </div>
    <script type="text/javascript" src="script/angular.min.js"></script>
    <script type="text/javascript">
        var app = angular.module('demoApp', []);
        app.controller('demoCtrl', function ($scope) {
            $scope.name = "Web 前端开发";
            $scope.showTech = function () {
                $scope.show = "我要学" + $scope.name + "!";
            }
        });
    </script>
```

现场演示：

- 案例17-04：AngularJS作用域

3.AngularJS应用

- AngularJS应用由View（视图）、Model（模型）、Controller组成
  - 示例包含内容：
    - 控制器：demoCtrl引用$scope对象并注册了两个属性和一个方法。
    - $scope对象：持有本例所需数据模型，包括name属性、show属性和showTech()方法。
    - 视图：拥有一个输入框、一个按钮以及一个利用双向数据绑定来显示数据的内容块。

3.AngularJS应用

- AngularJS应用由View（视图）、Model（模型）、Controller组成
  - 从控制器发起的角度看，示例有两个流程：
    - 控制器向作用域写属性：input因ng-model指令实现了双向数据绑定，给作用域name属性赋值后，input发现name属性值已变更，进而在视图中渲染出改变的值，即“Web前端开发”。
    - 控制器向作用域中写方法：“确定”按钮因ng-click指令绑定了showTech()方法，因此点击“确定”按钮时调用作用域中的showTech()方法，该方法读取作用域中的name属性，并加前缀“我要学”然后赋值给作用域中创建的show属性。

3.AngularJS应用

- AngularJS应用由View（视图）、Model（模型）、Controller组成
  - 从视图角度来看，分三部分内容：
    - input文本框中的渲染逻辑：ng-model指令对作用域和视图元素进行双向数据绑定，一方面根据ng-model中的name去作用域取值，如果存在则在输入框中显示，另一方面接受用户输入，并将用户输入的字符串传递给name，作用域中该属性的值实时更新为用户输入的值。
    - input按钮中的逻辑：接受用户单击，调用作用域中的showTech()方法。
    - {{show}}的渲染逻辑：在用户未点击按钮时不显示内容；取值阶段：“确定”按钮被点击后，表达式向$scope中取已存在的show属性；计算阶段：在当前作用域下计算show表达式，然后渲染视图显示“我要学Web前端开发！”。

3.AngularJS应用

- AngularJS控制器是控制AngularJS应用程序数据的JavaScript对象，通过ng-controller指令定义，就像JavaScript中的构造函数一般，是用来增强Angular作用域（scope）的。
- 示例17-04中ng-app定义了AngularJS应用程序demoApp，ng-controller指令定义了控制器demoCtrl。
- 控制器demoCtrl是一个JavaScript函数，其作用域对象$scope用来保存AngularJS Model（模型）的对象，在作用域中创建了属性name、属性show和方法showTech()。
- ng-model指令在input文本框和name属性间建立数据绑定。

3.AngularJS应用

- {{show}}表达式将show属性绑定到页面视图。
- ng-click指令将showTech方法绑定到页面视图，当用户点击input按钮时，show属性被赋值并显示到页面视图。

3.AngularJS应用

- 在视图模板（templates）、控制器（controllers）或者服务（services）中，过滤器通过管道字符（|）添加到表达式和指令中，用来格式化表达式中的值。
- AngularJS内置的过滤器如下：

表 17-2 AngularJS 内置的过滤器

过滤器	描述
currency	格式化数字为货币格式。
filter	从数组项中选择一个子集。
lowercase	格式化字符串为小写。
orderBy	根据某个表达式排列数组。
uppercase	格式化字符串为大写。

现场演示：

- 案例17-05：AngularJS内置过滤器

3.AngularJS应用

- 自定义过滤器
  - 创建自定义过滤器过程很简单，仅仅需要在模块中注册一个新的过滤器方法即可。这个方法将返回一个以输入值为第一个参数的新的过滤方法，过滤器中的参数作为附加参数传递给它。以下示例使用过滤器将字符串中的第一个“_”替换为“-”。

现场演示：

- 案例17-06：AngularJS自定义过滤器

3.AngularJS应用

- AngularJS表单是输入控件（input、select、textarea等）的集合。
- 在视图中通过基本的数据绑定形式可以访问到表单和控件的内部状态，因此可以在应用中通过ng-model指令在表单数据模型和表单视图间建立双向数据绑定，记录表单状态，在数据改变时更新状态。

现场演示：

- 案例17-07：AngularJS表单绑定与状态控制

3.AngularJS应用

- 大部分应用都有一个主方法用来实例化、组织、启动应用。AngularJS应用没有主方法，而是使用模块来声明应用应该如何启动。
- AngularJS通过angular.module()方法创建模块，模块中可添加控制器、指令、过滤器等。

3.AngularJS应用

- 以下示例是（17-04）一个简单的模块应用：在页面视图中，ng-app指令绑定应用demoApp，通过angular.module()声明应用模块；ng-controller绑定控制器，app.controller()函数定义控制器及其属性和方法。

```
<div ng-app="demoApp" ng-controller="demoCtrl">
        <div><label>技术：<input type="text" ng-model="name" ></label></div>
        <div>内容输出：<span>{{show}}</span></div>
        <input type="button" ng-click="showTech()" value="确定" >
    </div>
    <script type="text/javascript" src="script/angular.min.js"></script>
    <script type="text/javascript">
        var app = angular.module('demoApp', []);
        app.controller('demoCtrl', function ($scope) {
            $scope.name = "Web 前端开发";
            $scope.showTech = function () {
                $scope.show = "我要学" + $scope.name + "!";
            }
        });
    </script>
```

3.AngularJS应用

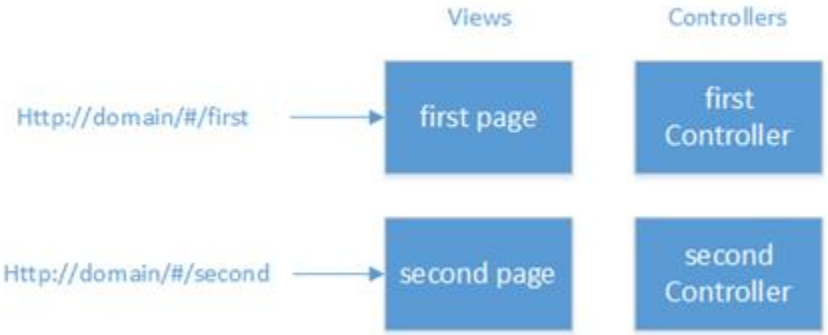
- AngularJS路由允许我们通过不同的URL访问不同的内容，可以实现多视图的单页Web应用（single page web application，SPA）。
- 通常URL形式为：http://domain/first/page，但在单页Web应用中AngularJS通过“#”+“标记”实现，如下所示：

```
http://domain/#/first
http://domain/#/second
http://domain/#/third
```

- 当点击以上任一链接时，向服务端发送的请求都是一样的（http://domain/），#号后的内容被浏览器忽略。

3.AngularJS应用

- 如图所示，每个URL都有对应的视图和控制器



3.AngularJS应用

- 简单路由示例如下：

```
<div ng-app="demoApp">
    <ul>
        <li><a href="#/">首页</a></li>
        <li><a href="#/jiadian">家电</a></li>
        <li><a href="#/shouji">手机</a></li>
        <li><a href="#/tushu">图书</a></li>
    </ul>
    <div ng-view></div>
</div>
<script type="text/javascript" src="script/angular.min.js"></script>
<script type="text/javascript" src="script/angular-route.min.js"></script>
<script type="text/javascript">
    var app = angular.module('demoApp', ['ngRoute']);
    app.config(['$routeProvider', function ($routeProvider) {
        $routeProvider
            .when('/', {template: '首页'})
            .when('/jiadian', {template: '家电商品页'})
            .when('/shouji', {template: '品牌手机页'})
            .when('/tushu', {template: '分类图书页'})
            .otherwise({redirectTo: '/'});
    }]);
</script>
```

3.AngularJS应用

- 路由配置对象规则如图：

```
$routeProvider.when(url, {
    template: string,
    templateUrl: string,
    controller: string, function 或 array,
    controllerAs: string,
    redirectTo: string, function,
    resolve: object<key, function>
});
```

  - template：可在ng-view中插入简单HTML内容。
  - templateUrl：可在ng-view中插入HTML模板文件。
  - controller：function、string或数组类型，在当前模板上执行controller函数，生成scope。
  - controllerAs：string类型，为controller指定别名。
  - redirectTo：重定向的地址。
  - resolve：指定当前controller所依赖的其他模块。

3.AngularJS应用

- 路由配置对象示例如下，使用templateUrl向ng-view中插入HTML模板，HTML模板中使用了控制器和表达式，点击链接调用不同的模板链接和控制器显示相应的内容。

现场演示：

- 案例17-08：AngularJS路由
- 案例17-09：AngularJS路由配置对象

3.AngularJS应用

- AngularJS中服务是一个函数或对象，可以在AngularJS应用中使用。
- 常用的AngularJS内建服务如下：
  - $location服务
    - $location服务是一个对象，作为一个参数传递到controller中，使用时需在controller中定义。
    - 下面代码实现效果为：页面视图输出$location内容。

```
var app = angular.module('demoApp', []);
app.controller('showLocation', function($scope, $location){
$scope.strLocation = JSON.stringify($location);
$scope.port = $location.port();
});
```

3.AngularJS应用

- 常用的AngularJS内建服务如下：
  - $http服务
    - $http是AngularJS中最常用的服务，服务向服务器发送请求，应用响应服务器返回的数据。
    - 下面代码实现效果为：页面视图显示请求的页面内容。

```
app.controller('httpService', function($scope, $http){
    $http.get('view/httppage.html').then(function(response){
        $scope.data = response.data;
    })
})
```

3.AngularJS应用

- 常用的AngularJS内建服务如下：
  - $timeout服务和$interval 服务
    - AngularJS中$timeout服务对应了JS window.setTimeout函数，$interval服务对应了JS window.setInterval函数。
    - 下面代码实现效果为：time1和time2时间相差2秒；intervalStr时间持续刷新。

```
app.controller('timeoutService', function($scope, $timeout){
        $scope.time1=new Date();
        $timeout(function(){
            $scope.time2=new Date();
        }, 2000)
})
app.controller('intervalService', function($scope, $interval){
        $interval(function(){
            $scope.intervalStr = new Date().toLocaleTimeString();
        }, 1000);
 })
```

3.AngularJS应用

- 自定义服务
  - 自定义服务通过service方法创建服务对象，对象内可以定义方法，调用方法与内置服务相同。
    - 自定义服务创建如下所示：

```
app.service('myservice', function(){
    this.checkLength = function(str, min, max){
        return str.length >= min && str.length <= max;
    }
})
```

    - 服务调用如下所示：

```
app.controller('cusService', function($scope, myService){
    $scope.check = function(str){
        $scope.showCheck = myservice.checkLength(str, 1, 6) ? '验证通过' : '字符长度不超过 6';
    }
})
```

现场演示：

- 案例17-10：AngularJS服务

4.案例：使用AngularJS实现即时搜索

- 本例综合应用AngularJS指令、表达式、作用于、过滤器、服务等知识点实现预警信息即时搜索应用。
- 预期效果如图：

4.案例：使用AngularJS实现即时搜索

- 案例结构
  - 页面视图
    - 页面视图绑定liveSearch应用、mySearch控制器，文本框绑定searchValue属性、下拉菜单绑定curType属性，切换下拉菜单更新列表数据；列表绑定预警信息，遍历输出序号、设备名称、监控类别、设备类型、设备型号、预警内容、开始时间、恢复时间。
    - 遍历输出预警信息时使用filter过滤器、orderBy过滤器，匹配输入设备名称，按预警开始时间排序。

4.案例：使用AngularJS实现即时搜索

- 案例结构
  - 程序结构
    - 创建liveSearch模块，自定义myService服务对象及getList方法，创建mySearch控制器，并在控制器中调用getList方法。
  - 类型筛选
    - getList方法根据data、type参数过滤并预警信息，返回值更新list属性。由于list属性与页面视图建立双向数据绑定，当list属性更新时，页面视图信息自动变化

现场演示：

- 案例17-11：使用AngularJS实现即时搜索

Thanks.