1. 简介

AngularJS 是一个 JavaScript 框架。

它可通过 <script> 标签添加到 HTML 页面。

AngularJS 通过 指令 扩展了 HTML，且通过 表达式 绑定数据到 HTML。

AngularJS 是以一个 JavaScript 文件形式发布的，可通过 script 标签添加到网页中：

<script src=”Angular文件名称”></script>

作用：

AngularJS 把应用程序数据绑定到 HTML 元素。

AngularJS 可以克隆和重复 HTML 元素。

AngularJS 可以隐藏和显示 HTML 元素。

AngularJS 可以在 HTML 元素"背后"添加代码。

AngularJS 支持输入验证。

1. 指令 -- AngularJS 指令是以 ng 作为前缀的 HTML 属性。--用来扩展HTML

AngularJS 通过被称为 指令 的新属性来扩展 HTML。

AngularJS 通过内置的指令来为应用添加功能。

AngularJS 允许你自定义指令

<div ng-app="">

<p>名字 : <input type="text" ng-model="name"></p>

<h1>Hello {{name}}</h1>

</div>

**ng-app** 指令定义一个 AngularJS 应用程序。

ng-app 指令告诉 AngularJS，<div> 元素是 AngularJS 应用程序 的"所有者"。

**ng-model** 指令把元素值（比如输入域的值）绑定到应用程序。

ng-model 指令**把输入域的值绑定到应用程序变量 name。**

**ng-bind** 指令把应用程序数据绑定到 HTML 视图。

ng-bind 指令**把应用程序变量 name 绑定到某个段落的 innerHTML。**

1. 数据绑定 – 双向绑定 ng-model将数据同步到scope里，表达式是将scope同步到html中

AngularJS 中的数据绑定，同步了 AngularJS 表达式与 AngularJS 数据。

{{ firstName }} 是通过 ng-model="firstName" 进行同步。

代码：

<div ng-app="" ng-init="quantity=1;price=5">

<h2>价格计算器</h2>

数量： <input type="number" ng-model="quantity">

价格： <input type="number" ng-model="price">

<p><b>总价：</b> {{ quantity \* price }}</p>

</div>

1. 重复HTML元素ng-repeat，关于数组中元素的调用

代码：

<div data-ng-app="" data-ng-init="names=['Jani','Hege','Kai']">

<p>使用 ng-repeat 来循环数组</p>

<ul>

<li **data-ng-repeat="x in names"**>

{{ x }}

</li>

</ul>

</div>

1. 创建自定义指令

可以使用 .directive 函数来添加自定义的指令。

要调用自定义指令，HTML 元素上需要添加自定义指令名。

使用驼峰法来命名一个指令， runoobDirective, 但在使用它时需要以 - 分割, runoob-directive:

代码：

<script>

var app = angular.module("myApp", []);

**app.directive**("runoobDirective", function() {

return {

template : "<h1>自定义指令!</h1>"

};

});

</script>

调用：

元素名 <runoob-directive></runoob-directive>

属性 <div runoob-directive></div>

类名 <div class="runoob-directive"></div>

注释 <!-- directive: runoob-directive -->

1. 限制指令的调用

通过添加 restrict 属性,并设置值为 "A", 来设置指令只能通过属性的方式来调用:

restrict 值可以是以下几种:

E 作为元素名使用

A 作为属性使用

C 作为类名使用

M 作为注释使用

restrict 默认值为 EA, 即可以通过元素名和属性名来调用指令。

1. AngularJS表达式

AngularJS 表达式写在**双大括号**内：{{ expression }}。

AngularJS 表达式把数据绑定到 HTML，这与 ng-bind 指令有异曲同工之妙。

AngularJS 将在表达式书写的位置"输出"数据。

AngularJS 表达式 很像 JavaScript 表达式：它们可以包含文字、运算符和变量。

实例 {{ 5 + 5 }} 或 {{ firstName + " " + lastName }}

1. 数字 -- ng-init=””用来初始化
2. 字符串
3. 对象
4. 数组

* AngularJS表达式和JS表达式区别

类似于 JavaScript 表达式，AngularJS 表达式可以包含字母，操作符，变量。

与 JavaScript 表达式不同，AngularJS 表达式可以写在 HTML 中。

与 JavaScript 表达式不同，AngularJS 表达式**不支持条件判断，循环及异常。**

与 JavaScript 表达式不同，AngularJS 表达式**支持过滤器**。

1. AngularJS应用

在js中：

AngularJS 模块（Module） 定义了 AngularJS 应用。

AngularJS 控制器（Controller） 用于控制 AngularJS 应用。

代码：

<script>

var app = angular.module('myApp', []);//定义模块

app.controller('myCtrl', function($scope) { //定义控制器

$scope.firstName= "John";

$scope.lastName= "Doe";

});

</script>

在html中：

ng-app指令定义了应用, ng-controller 定义了控制器。

代码：

<div ng-app="myApp" ng-controller="myCtrl">

//在html中定义了模块myApp,定义了控制器myCtrl

名: <input type="text" ng-model="firstName"><br>

姓: <input type="text" ng-model="lastName"><br>

<br>

姓名: {{firstName + " " + lastName}} //AngularJS值表达式

</div>

4、ng-model指令

1. 双向数据绑定，ng-model会将输入域的值同步到scope中，实时更新。
2. 验证用户输入

代码：

<form ng-app="" name="myForm">

Email:

<input type="email" name="myAddress" **ng-model="text"**> //model指令

<span **ng-show="myForm.myAddress.$error.email"**>不是一个合法的邮箱地址</span>

</form>

1. 应用状态

vaild：值合法显示true，否则显示false

dirty:值发生改变显示true，否则显示false

error：出现错误显示true，其他不显示

Touched：触屏点击显示true，否则显示false

代码：

<form ng-app="" name="myForm" ng-init="myText = 'test@runoob.com'">

Email:

<input type="email" name="myAddress" **ng-model="myText"** required></p>

<h1>状态</h1>

{{myForm.myAddress.$valid}}

{{myForm.myAddress.$dirty}}

{{myForm.myAddress.$touched}}

</form>

1. 为HTML元素提供CSS类

代码：

<style>

**input.ng-invalid** {

background-color: lightblue;

}

</style>

<body>

<form ng-app="" name="myForm">

输入你的名字:

<input name="myAddress" ng-model="text" **required**>//requred不为空

</form>

添加状态：

ng-model 指令根据表单域的状态添加/移除以下类：

ng-empty

ng-not-empty

ng-touched

ng-untouched

ng-valid

ng-invalid

ng-dirty

ng-pending

ng-pristine

1. Scope作用域

**Scope(作用域) 是应用在 HTML (视图) 和 JavaScript (控制器)之间的纽带。**

Scope 是**一个对象**，有可用的方法和属性。

Scope 可应用在视图和控制器上。

代码：--创建控制器时候将 $scope 对象当做一个参数传递

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.carname = "Volvo";

});

</script>

当在控制器中添加 $scope 对象时，视图 (HTML) 可以获取了这些属性。

视图中，你不需要添加 $scope 前缀, 只需要添加属性名即可，如： {{carname}}。

AngularJS 应用组成如下：

View(视图), 即 HTML。

Model(模型), 当前视图中可用的数据。

**Controller(控制器), 即 JavaScript 函数**，可以添加或修改属性。

scope 是模型。

**scope 是一个 JavaScript 对象**，带有属性和方法，这些属性和方法可以在视图和控制器中使用。

$scope.say 可以是一个函数

scope作用范围：

作用域范围只在某一个控制器，不能作用于其他的控制器。

$rootScope根作用域： -- 和 Angular应用程序相关

所有的应用都有一个 $rootScope，它**可以作用在 ng-app 指令包含的所有 HTML 元素中。**

$rootScope 可作用于整个应用中。是各个 controller 中 scope 的桥梁。**用 rootscope 定义的值，可以在各个 controller 中使用。**

* $scope 和 $rootScope 都可以当做参数传进控制器里的函数里面

1. AngularJS控制器

AngularJS 控制器 控制 AngularJS 应用程序的数据。

AngularJS 控制器是常规的 JavaScript 对象。

ng-controller 指令定义了应用程序控制器。

控制器是 JavaScript 对象，由标准的 JavaScript 对象的构造函数 创建。

控制器中也可以定义方法（函数）-- 代码：

<script>

var app = angular.module('myApp', []);

app.controller('personCtrl', function($scope) {

$scope.firstName = "John";

$scope.lastName = "Doe";

$scope.fullName = function() {

return $scope.firstName + " " + $scope.lastName;

}

});

</script>

外部文件的控制器：

在大型的应用程序中，通常是把控制器存储在外部文件中。

只需要把 <script> 标签中的代码复制到名为 personController.js 的外部文件中即可：

<script src="personController.js"></script>

1. AngularJS过滤器 -- 过滤器可以使用一个管道字符（|）添加到表达式和指令中。

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。

表达式中添加过滤器--代码：{{ lastName | lowercase }}

指令中添加过滤器—代码：

<div ng-app="myApp" ng-controller="namesCtrl">

<ul>

<li ng-repeat="**x in names | orderBy:'country**'">//排序

{{ x.name + ', ' + x.country }}

</li>

</ul>

</div>

<div ng-app="myApp" ng-controller="namesCtrl">

<p>输入过滤:</p>

<p><input type="text" ng-model="test"></p>

<ul>

<li **ng-repeat="x in names | filter:test | orderBy:'country'"**>

{{ (x.name | uppercase) + ', ' + x.country }}

</li>

</ul>

</div>

自定义过滤器--.filter方法 – 应用方法

代码：

<div ng-app="myApp" ng-controller="myCtrl">

姓名: {{ msg | reverse }}

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.msg = "Runoob";

});

app.filter('reverse', function() { //可以注入依赖

return function(text) {

return text.split("").reverse().join("");

}

});

</script>

1. AngularJS Service 服务

在 AngularJS 中，服务是**一个函数或对象**，可在你的 AngularJS 应用中使用。AngularJS 内建了30 多个服务。

1. $location 服务，它可以返回当前页面的 URL 地址。

代码：

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope, $location) {

$scope.myUrl = $location.absUrl();

});

</script>

1. $http 是 AngularJS 应用中最常用的服务。 服务向服务器发送请求，应用响应服务器传送过来的数据。

代码：

$http 服务向服务器请求信息，返回的值放入变量 "myWelcome" 中。

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope, $http) {

$http.get("welcome.htm").then(function (response) {

$scope.myWelcome = response.data;

});

});

</script>

1. AngularJS $timeout 访问在规定的毫秒数后执行指定函数。

代码：

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope, $timeout) {

$scope.myHeader = "Hello World!";

**$timeout(function () {**

**$scope.myHeader = "How are you today?";**

**}, 2000);**

});

</script>

1. $interval 服务 -- 访问在指定的周期(以毫秒计)来调用函数或计算表达式。

代码：

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope, $interval) {

$scope.theTime = new Date().toLocaleTimeString();

$interval(function () {

$scope.theTime = new Date().toLocaleTimeString();

}, 1000); //周期为1s，每一秒调用一次显示当前时间的函数

});

</script>

1. 创建自定义服务 – 应用程序的.service方法创建自定义服务

代码：

app.service('hexafy', function() {

this.myFunc = function (x) {

return x.toString(16);

}

});

在控制器中调用服务： -- 自定义服务没有$这个标识

app.controller('myCtrl', function($scope, hexafy) {

$scope.hex = hexafy.myFunc(255);

});

在过滤器中调用服务：

app.filter('myFormat',['hexafy', function(hexafy) {

return function(x) {

return hexafy.myFunc(x);

};

}]);

1. 可以在控制器，指令，过滤器或其他服务中使用自定义服务。

9、$http服务 -- $http 是 AngularJS 中的一个核心服务，用于读取远程服务器的数据。

1. POST 与 GET 简写方法格式：-- 还有其他一些简写方法

$http.get('/someUrl', config).then(successCallback, errorCallback);

$http.post('/someUrl', data, config).then(successCallback, errorCallback);

具体代码：

// 简单的 GET 请求，可以改为 POST

$http({

method: 'GET',

url: '/someUrl'

}).then(function successCallback(response) {

// 请求成功执行代码

}, function errorCallback(response) {

// 请求失败执行代码

});

1. $http.get(url) 是用于读取服务器数据的函数。

代码：

<script>

var app = angular.module('myApp', []);//定义一个应用

app.controller('siteCtrl', function($scope, $http) {//http服务当做参数传入

$http({

method: 'GET',

url: 'http://www.runoob.com/try/angularjs/data/sites.php'

}).then(function successCallback(response) {

$scope.names = response.data.sites;//请求成功执行代码

}, function errorCallback(response) {

// 请求失败执行代码

});

});

</script>

1. AngularJS select – 使用ng-options指令创建一个下拉选择框

可以使用 ng-option 指令来创建一个下拉列表，列表项通过对象和数组循环输出，

使用 ng-options 的选项的一个对象，ng-repeat 是一个字符串。

* ng语法：

ng-options="option.id as option.title for option in curOptionArray"

ption.id是select**提交的值** option.title是select**显示的值** option代表数组curOptionArray里面的当前元素

代码：使用ng-optiongs创建

<div ng-app="myApp" ng-controller="myCtrl">

<select ng-init="selectedName = names[0]" ng-model="selectedName" **ng-options="x for x in names"**>

</select>

</div> //ng-intit设置初始值

代码：使用ng-repeat指令创建

<div ng-app="myApp" ng-controller="myCtrl">

<select>

**<option ng-repeat="x in names">{{x}}</option>**

</select>

</div>

使用数组作为数据源—可以使用两种方式创建

$scope.sites = [

{site : "Google", url : "http://www.google.com"},

{site : "Runoob", url : "http://www.runoob.com"},

{site : "Taobao", url : "http://www.taobao.com"}

];

使用数据对象作为数据源（key-value）—只能使用options创建

$scope.sites = {

site01 : "Google",

site02 : "Runoob",

site03 : "Taobao"

};

代码：<select ng-model="selectedSite" ng-options="x for (x, y) in sites">

</select>

10、AngularJS ng-repeat显示表格

代码：

<div ng-app="myApp" ng-controller="customersCtrl">

<table>

<tr ng-repeat="x in names">

<td>{{ x.Name }}</td>

<td>{{ x.Country }}</td>

</tr> //循环创建行

</table>

</div>

1. 使用CSS样式

<style>

table, th , td {

border: 1px solid grey;

border-collapse: collapse;

padding: 5px;

}

table tr:nth-child(odd) {

background-color: #f1f1f1;

}

table tr:nth-child(even) {

background-color: #ffffff;

}

</style>

1. 使用过滤器

<table>

<tr ng-repeat="x in names | **orderBy : 'Country'**">

<td>{{ x.Name }}</td>

<td>{{ x.Country }}</td>

//转换大写的过滤器

<td>{{ x.Country | uppercase }}</td>

</tr>

</table>

1. 显示序号--$index

<table>

<tr ng-repeat="x in names">

<td>{{ $index + 1 }}</td>

<td>{{ x.Name }}</td>

<td>{{ x.Country }}</td>

</tr>

</table>

1. 奇$odd 偶$even

<table>

<tr ng-repeat="x in names">

<td ng-if="$odd" style="background-color:#f1f1f1">

{{ x.Name }}</td>

<td ng-if="$even">

{{ x.Name }}</td>

<td ng-if="$odd" style="background-color:#f1f1f1">

{{ x.Country }}</td>

<td ng-if="$even">

{{ x.Country }}</td>

</tr>

</table>

11、AngularJS SQL

12、HTML DOM -- AngularJS 为HTML DOM元素的属性提供了绑定应用数据的指令

1. ng-disabled 指令, 如果 ng-disabled 中的表达式返回 true 则表单字段将被禁用。

代码：

<div ng-app="" ng-init="mySwitch=true">

<button **ng-disabled="mySwitch"**>点我!</button>

<input type="checkbox" **ng-model="mySwitch"**/>按钮

{{ mySwitch }}

</div>

ng-disabled 指令绑定应用程序数据 "mySwitch" 到 HTML 的 disabled 属性。

ng-model 指令绑定 "mySwitch" 到 HTML input checkbox 元素的内容（value）。

1. ng-show 指令隐藏或显示一个 HTML 元素。

代码：

<div ng-app="">

<p ng-show="true">我是可见的。</p>

<p ng-show="false">我是不可见的。</p>

</div>

1. ng-hide 指令用于隐藏或显示 HTML 元素。

代码：

<div ng-app="">

<p ng-hide="true">我是不可见的。</p>

<p ng-hide="false">我是可见的。</p>

</div>

13、AngularJS 有自己的 HTML 事件指令 – JS都可以读取到HTML中的事件的

onclick事件，代码：

<div ng-app="myApp" ng-controller="myCtrl">

<button ng-click="count = count + 1">点我！</button>

<p>{{ count }}</p>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.count = 0;

});

</script>

ng-hide可以用于标签属性结合onclick事件，代码：

<div ng-app="myApp" ng-controller="personCtrl">

<button ng-click="toggle()">隐藏/显示</button> //调用scope中的方法

<p ng-hide="myVar"> //利用hide指令进行判断

名: <input type=text ng-model="firstName"><br>

姓: <input type=text ng-model="lastName"><br><br>

姓名: {{firstName + " " + lastName}}

</p>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('personCtrl', function($scope) {

$scope.firstName = "John";

$scope.lastName = "Doe";

$scope.myVar = false;

$scope.toggle = function() {

$scope.myVar = !$scope.myVar;

}

});

</script>

14、AngularJS模块

模块定义了一个应用程序。

模块是应用程序中不同部分的容器。（有控制器，指令，容器）

模块是应用控制器的容器。

控制器通常属于一个模块

1. 在模块定义中 [] 参数用于定义模块的依赖关系。中括号[]表示该模块没有依赖，如果有依赖的话会在中括号写上依赖的模块名字。
2. 添加控制器 app.controller,控制器一般就是一个方法
3. 添加自定义指令 app.directive

<script>

var app = angular.module("myApp", []);

app.directive("runoobDirective", function() {

return {

template : "我在指令构造器中创建!"

};

});

</script>

1. 对于 HTML 应用程序，通常建议把所有的脚本都放置在 <body> 元素的最底部。

这会提高网页加载速度，因为 HTML 加载不受制于脚本加载。

在 <body> 元素中加载 AngularJS 库，但是必须放置在您的 AngularJS 脚本前面。

15、AngularJS表单

HTML控件：input元素、select元素、button元素、textrea元素

1. 数据绑定

Input元素使用ng-model实现数据绑定，将输入框的值绑定到scope当中

1. Checkbox（复选框）-- 选中为true

checkbox 的值为 true 或 false，可以使用 ng-model 指令绑定。

1. 单选框 – 选中为value中的值

代码：

<form>

选择一个选项:

<input type="radio" ng-model="myVar" value="dogs">Dogs

<input type="radio" ng-model="myVar" value="tuts">Tutorials

<input type="radio" ng-model="myVar" value="cars">Cars

</form>

//switch指令

<div ng-switch="myVar">

<div ng-switch-when="dogs">

<h1>Dogs</h1>

<p>Welcome to a world of dogs.</p>

</div>

<div ng-switch-when="tuts">

<h1>Tutorials</h1>

<p>Learn from examples.</p>

</div>

<div ng-switch-when="cars">

<h1>Cars</h1>

<p>Read about cars.</p>

</div>

</div>

1. 下拉菜单option，再结合switch。

<form>

选择一个选项:

<select ng-model="myVar">

<option value="">

<option value="dogs">Dogs

<option value="tuts">Tutorials

<option value="cars">Cars

</select>

</form>

1. 表单初始化

<script>

var app = angular.module('myApp', []);

app.controller('formCtrl', function($scope) {

$scope.master = {firstName: "John", lastName: "Doe"};

$scope.reset = function() {

$scope.user = angular.copy($scope.master);

};

$scope.reset();//直接调用方法初始化，使user等于master

});

</script>

16、输入验证

AngularJS 表单和控件可以提供验证功能，并对用户输入的非法数据进行警告。

<form ng-app="myApp" ng-controller="validateCtrl"

name="myForm" novalidate>

<p>用户名:<br>

<input type="text" name="user" **ng-model="user"** required><!--非空-->

<span style="color:red" **ng-show="myForm.user.$dirty && myForm.user.$invalid**"><!--值发生改变且非法-->//利用属性判断

<span ng-show="myForm.user.$error.required">用户名是必须的。</span><!--当出现错误-->

</span>

</p>

<p>邮箱:<br>

<input type="email" name="email" ng-model="email" required><!--非空-->

<span style="color:red" ng-show="myForm.email.$dirty && myForm.email.$invalid"><!--值发生改变且非法-->

<span ng-show="myForm.email.$error.required">邮箱是必须的。</span><!--当出现错误为true-->

<span ng-show="myForm.email.$error.email">非法的邮箱地址。</span><!--当邮箱地址错误为true-->

</span>

</p>

<p>

<input type="submit"

ng-disabled="myForm.user.$dirty && myForm.user.$invalid || <!--用户名值发生改变且合法 或者 邮箱值改变且合法-->

myForm.email.$dirty && myForm.email.$invalid">

</p></form>

属性 描述

$dirty 表单有填写记录

$valid 字段内容合法的

$invalid 字段内容是非法的

$pristine 表单没有填写记录

17、AngularJS API -- API 意为 Application Programming Interface（应用程序编程接口）

AngularJS 全局 API 用于执行常见任务的 JavaScript 函数集合，如：

比较对象

迭代对象

转换对象

全局 API 函数使用 angular 对象进行访问。

1. 常用函数：

API 描述

angular.lowercase() 转换字符串为小写

angular.uppercase() 转换字符串为大写

angular.isString() 判断给定的对象是否为字符串，如果是返回 true。

angular.isNumber() 判断给定的对象是否为数字，如果是返回 true。

将angular看做一个对象，直接调用其方法就可以，只能在js中使用。

18、AngularJS Bootstrap -- 引用别的框架

19、AngularJS 包含指令 – ng-include—在HTML中使用这个指令

同域名包含：

<div ng-app="myApp" ng-controller="sitesCtrl">

包含的内容为HTML文件，里面也有一些AngularJS的代码。

跨域名包含：-- 涉及到php的使用

20、AngularJS动画

AngularJS 提供了动画效果，可以配合 CSS 使用。

AngularJS 使用动画需要引入 angular-animate.min.js 库。

引入源代码文件：

<script src="http://cdn.static.runoob.com/libs/angular.js/1.4.6/angular-animate.min.js"></script>

应用程序固定：

<body ng-app="ngAnimate">

代码：

<script src="http://cdn.static.runoob.com/libs/angular.js/1.4.6/angular.min.js"></script>

<script src="http://cdn.static.runoob.com/libs/angular.js/1.4.6/angular-animate.min.js">

</script>

</head>

<body ng-app="ngAnimate">

<h1>隐藏 DIV: <input type="checkbox" ng-model="myCheck"></h1>

<div ng-hide="myCheck"></div>

</body>

</html>

设置关联模型：代码：

<body ng-app="myApp">

<h1>隐藏 DIV: <input type="checkbox" ng-model="myCheck"></h1>

<div ng-hide="myCheck"></div>

<script>

var app = angular.module('myApp', ['ngAnimate']);

</script>

原理：

ngAnimate 模型可以添加或移除 class 。

ngAnimate 模型并不能使 HTML 元素产生动画，但是 ngAnimate 会监测事件，类似隐藏显示 HTML 元素 ，如果事件发生 ngAnimate 就会使用预定义的 class 来设置 HTML 元素的动画。

21、AngularJS 依赖注入

**定义**：依赖注入（Dependency Injection，简称DI）是一种软件设计模式，在这种模式下，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。

**作用**：该模式分离了客户端依赖本身行为的创建，这使得程序设计变得松耦合，并遵循了依赖反转和单一职责原则。

**相关组件**：5个核心组件用来作为依赖注入

value、factory、service、provider、constant

1. Value，value是一个简单的JS对象，用于向控制器传递值

实例：

// 定义一个模块

var mainApp = angular.module("mainApp", []);

// 创建 value 对象 "defaultInput" 并传递数据

mainApp.value("defaultInput", 5);

...

// 将 "defaultInput" 注入到控制器

mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {

$scope.number = defaultInput;

$scope.result = CalcService.square($scope.number);

$scope.square = function() {

$scope.result = CalcService.square($scope.number);

}});

1. Factory – 不太理解

factory 是一个函数用于返回值。在 service 和 controller 需要时创建。

通常我们使用 factory 函数来计算或返回值。

实例：

// 定义一个模块

var mainApp = angular.module("mainApp", []);

// 创建 factory "MathService" 用于两数的乘积 provides a method multiply to return multiplication of two numbers

mainApp.factory('MathService', function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b

}

return factory;

});

// 在 service 中注入 factory "MathService"

mainApp.service('CalcService', function(MathService){

this.square = function(a) {

return MathService.multiply(a,a);

}

});

1. **provider**

AngularJS 中通过 provider 创建一个 service、factory等(配置阶段)。

Provider 中提供了一个 factory 方法 get()，它用于返回 value/service/factory。

实例：

// 定义一个模块

var mainApp = angular.module("mainApp", []);

// 使用 provider 创建 service 定义一个方法用于计算两数乘积

mainApp.**config**(function($provide) {

$provide.provider('MathService', function() {

this.$get = function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b;

}

return factory;

};});

});

1. constant

constant(常量)用来在配置阶段传递数值，注意这个常量在配置阶段是不可用的。

mainApp.constant("configParam", "constant value");

21、AngularJS 路由 --- 点击会转发到不同页面，相当于内部转发，但是网址不变，只有网页内容发生改变。

作用：AngularJS 路由允许我们通过不同的 URL 访问不同的内容。

通过 AngularJS 可以实现多视图的单页Web应用（single page web application，SPA）。

http://runoob.com/#/first

http://runoob.com/#/second

http://runoob.com/#/third

**原理：**当我们点击以上的任意一个链接时，向服务端请的地址都是一样的 (http://runoob.com/)。 因为 # 号之后的内容在向服务端请求时会被浏览器忽略掉。 所以我们就需要在客户端实现 # 号后面内容的功能实现。 AngularJS 路由 就通过 # + 标记 帮助我们区分不同的逻辑页面并将不同的页面绑定到对应的控制器上。

**实例：**

<body ng-app='routingDemoApp'><!--路由相关的模块就是应用程序-->

<h2>AngularJS 路由应用</h2>

<ul>

<li><a href="#/">首页</a></li>

<li><a href="#/computers">电脑</a></li>

<li><a href="#/printers">打印机</a></li>

<li><a href="#/blabla">其他</a></li>

</ul>

<div ng-view></div><!--使用view命令会使HTML中的内容根据路由的变化而变化-->

<script src="http://cdn.static.runoob.com/libs/angular.js/1.4.6/angular.min.js"></script>

<script src="https://apps.bdimg.com/libs/angular-route/1.3.13/angular-route.js"></script><!--引入路由相关js-->

<script>

angular.module('routingDemoApp',['ngRoute'])//定义一个应用程序

.config(['$routeProvider', function($routeProvider){

$routeProvider

.when('/',{template:'这是首页页面'})

.when('/computers',{template:'这是电脑分类页面'})

.when('/printers',{template:'这是打印机页面'})

.otherwise({redirectTo:'/'});

}]);

</script><!--配置 $routeProvider，AngularJS $routeProvider 用来定义路由规则。不懂！-->

</body>

1. 路由设置对象

AngularJS 路由也可以通过不同的模板来实现。

$routeProvider.when 函数的第一个参数是 URL 或者 URL 正则规则，第二个参数为路由配置对象。

路由配置对象语法规则如下：

$routeProvider.when(url, {

template: string,

templateUrl: string,

controller: string, function 或 array,

controllerAs: string,

redirectTo: string, function,

resolve: object<key, function>

});

**template:**

如果我们只需要在 ng-view 中插入简单的 HTML 内容，则使用该参数：

**templateUrl:**

如果我们只需要在 ng-view 中插入 HTML 模板文件，则使用该参数：

$routeProvider.when('/computers', {

templateUrl: 'views/computers.html',

});

以上代码会从服务端获取 views/computers.html 文件内容插入到 ng-view 中。

**controller:**

function、string或数组类型，在当前模板上执行的controller函数，生成新的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。

22、AngularJS工作原理

1. **ng-app** 指令位于应用的根元素下。
2. 对于单页Web应用（single page web application，SPA），应用的根通常为 <html> 元素。
3. 一个或多个 ng-controller 指令定义了应用的控制器。每个控制器有他自己的作用域：: 定义的 HTML 元素。
4. AngularJS 在 HTML DOMContentLoaded 事件中自动开始。如果找到 ng-app 指令 ， AngularJS 载入指令中的模块，并将 ng-app 作为应用的根进行编译。
5. 应用的根可以是整个页面，或者页面的一小部分，如果是一小部分会更快编译和执行。

备忘录程序：

<!DOCTYPE html>

<html>

<script src= "http://cdn.bootcss.com/angular.js/1.4.6/angular.min.js"></script>

<body ng-app="myApp" ng-controller="todoCtrl">

<h2>我的备忘录</h2>

<form ng-submit="todoAdd()"><!--提交后调用todoAdd()方法-->

<input type="text" ng-model="todoInput" size="50" placeholder="新增"><!--文本框绑定到应用里todoInput变量-->

<input type="submit" value="新增"><!--提交按钮-->

</form>

<br>

<div ng-repeat="x in todoList"><!--循环读取todoList中的元素--><!--在网页中显示-->

<input type="checkbox" ng-model="x.done"> <span ng-bind="x.todoText"></span>

<!--复选框绑定到x.done变量值-->

<!--利用bind指令读取应用程序内todoText中的值-->

</div>

<p><button ng-click="remove()">删除记录</button></p><!--调用remove()方法删除-->

<script>

var app = angular.module('myApp', []); //定义一个模型，也就是应用程序

app.controller('todoCtrl', function($scope) {//定义一个控制器，也就是方法

$scope.todoList = [{todoText:'Clean House', done:false}];//定义一个数组

//todoInput变量是绑定到输入文本框

//done为false，表示复选框默认是不勾选的

$scope.todoAdd = function() {//用于新增的函数

$scope.todoList.push({todoText:$scope.todoInput, done:false});//将todoInput中的值赋给todoText，往数组里面新增一个元素

$scope.todoInput = "";//将todoInput也就是输入文本框清空

};

//先清空todoList，再判断原有数组中的元素，如果为false也就是不勾选，再添加到todoList中

$scope.remove = function() {//定义remove删除方法

var oldList = $scope.todoList;//将todoList赋值给oldList

$scope.todoList = [];//清空todoList，也就是文本框输入值数组

angular.forEach(oldList, function(x) {//循环读取oldList中的元素，并且定义一个新函数

if (!x.done) $scope.todoList.push(x);//如果为false，也就是复选框不勾选的元素，添加到todoList

});

};

});

</script>

</body>

</html>