



第八讲 模块和服务

教学目标

1. 模块
2. 创建和使用服务
3. 全局对象

1

模块

1

创建模块

```
<script>
```

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

```
</script>
```

```
<div ng-app="myApp">...</div>
```

- ✱ myApp 参数对应执行应用的 HTML 元素。
- ✱ 可以在 AngularJS 应用中添加控制器，指令，过滤器等。

②

使用多个模块

```
<script>
```

```
    var app = angular.module("myApp1", []);
```

```
    var app = angular.module("myApp2", []);
```

```
</script>
```

```
<div ng-app="myApp1">...</div>
```

```
<div ng-app="myApp2">...</div>
```

AngularJS默认在一个html界面中只启动一个 ng-app 模块，而且是界面中第一次出现的那个使用 ng-app 声明的模块，

3

理解模块

- ❁ 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。
- ❁ 模块是一个集合。

可包含：控制器、过滤器、指令、服务

4

使用模块

为模块添加控制器

```
angular.module("exampleApp", [])  
  .controller("defaultCtrl", function ($scope) {  
    $scope.data = {  
      cities: ["伦敦", "约纽", "巴黎"],  
      clicks: 0  
    };  
    $scope.$watch('data.clicks', function (newVal) {  
      //监视数据data.clicks, 当变化时将data.clicks的值传递给newVal  
      console.log("单击的次数: " + newVal);  
    });  
  })
```

Demo:listing01.html

```
.directive("triButton", function () {  
    return {  
        scope: { counter: "=counter" },  
        link: function (scope, element, attrs) {  
            element.on("click", function (event) {  
                //on() 方法在被选元素及子元素上添加一个或多个事件处理程序  
                console.log("单击的按钮是: " + event.target.innerText);  
                //target是jQuery 1.3 新增属性, 返回最初触发事件的DOM元素  
                scope.$apply(function () {  
                    //scope.$apply将代码添加到执行队列  
                    scope.counter++;  
                });  
            });  
        }  
    };  
});
```

Demo:listing01.html

4

使用模块

在视图中应用模块中添加的控制器、指令

```
<body ng-controller="defaultCtrl">
  <div class="well">
    <div tri-button counter="data.clicks">
      <button ng-repeat="city in data.cities">
        {{city}}
      </button>
    </div>
    <h5>单击的次数: {{data.clicks}}</h5>
  </div>
</body>
```

Demo:listing01.html

④

使用模块

案例分析：

- 1、使用ng-repeat指令创建三个按钮
- 2、应用triButton指令处理按钮的单击事件
- 3、 triButton指令监控并更新由控制器定义的计数器
- 4、通过独立作用域使用数据共享
- 5、控制器和指令将信息写入控制台
- 6、总单击数通过HTML标签中行内绑定的表达式被显示出来

Demo:listing01.html

5

模块依赖

理解模块依赖

```
angular.module('mod3', ['mod1']);  
angular.module('mod4', ['mod2']);
```

Demo:ex01.html

```
angular.module('mod3', ['mod2', 'mod1']);  
angular.module('mod4', ['mod1', 'mod2']);
```

Demo:ex02html

5

模块依赖

创建外部模块

```
angular.module("custom", [])  
.directive("triButton", function () {  
    return {  
        scope: { counter: "=counter" },  
        link: function (scope, element, attrs) {  
            element.on("click", function (event) {  
                console.log("单击的按钮是: " +  
                    event.target.innerText);  
                scope.$apply(function () {  
                    scope.counter++;  
                });  
            });  
        }  
    };  
});
```

Demo:listing04.html

5

模块依赖

依赖外部模块：`<script src="Listing 04.js"></script>`

```
angular.module("exampleApp",["custom"])
.controller(" defaultCtrl ", function ($scope) {
    $scope.data = {
        cities: ["伦敦", "纽约", "巴黎"],
        clicks: 0
    };
    $scope.$watch("data.clicks", function (newVal) {
        console.log("单击次数: " + newVal);
    });
});
```

Demo:listing05.html

模块中添加应用

- ✿ 添加控制器
- ✿ 添加过滤器
- ✿ 添加指令
- ✿ 添加服务

2

创建和使用服务

1

服务

- **服务**打包可重用的功能，使之能在应用程序中使用。
- **模块**打包可重用的功能。使之能跨多个应用程序使用。

模块定义了三个方法用于定义服务：

- ◆ **factory**
- ◆ **service**
- ◆ **provider**

②

Factory方法

factory()——工厂函数方式

factory()可以通过返回一个包含service方法和数据的对象来定义一个service。

```
var app=angular.module("custom", []);  
app.factory( "serviceName", function(){});
```

自定义服务名称

工厂函数

factory()方法适合于创建的服务，仅仅需要的是一个方法和数据的集合且不需要处理复杂的逻辑。

注意：需要使用.config()来配置service的时候不能使用factory()方法

②

Factory方法

创建服务

```
angular.module("Services", [])  
  .factory("logService", function () {  
    var messageCount = 0;  
    return {  
      log: function (msg) {  
        console.log("(LOG + " +  
          messageCount++ + ") " + msg);  
      }  
    };  
  });
```

Demo:listing06.js

②

Factory方法

在文件中调用服务：

```
<script src="Listing 06.js"></script>  
<script src="Listing 08.js"></script>
```

```
angular.module("exampleApp", ["Directives","Services"])  
.controller("defaultCtrl", function ($scope, logService) {  
    $scope.data = {  
        cities: ["伦敦", "纽约", "巴黎"],  
        clicks: 0  
    };  
    $scope.$watch('data.clicks', function (newVal) {  
        logService.log("单击次数: " + newVal);  
    });  
});
```

Demo:listing07.html

②

Factory方法

在文件中调用服务

```
angular.module("Directives", ["Services"])\n.directive("triButton", function (logService) {\n  return {\n    scope: { counter: "=counter" },\n    link: function (scope, element, attrs) {\n      element.on("click", function (event) {\n        logService.log("单击的按钮是: " +\n                        event.target.innerText);\n        scope.$apply(function () {\n          scope.counter++;\n        });\n      });\n    }\n  });\n});\n});\n});\n});\n});
```

Demo:listing08.js

3

Service方法

service()——构造函数方式

service()方法可以使用原型模式替代JavaScript原始的对象来定义service。

```
var app=angular.module("custom", []);  
app.service( "serviceName", function(){});
```

自定义服务名称

构造函数

service()方法适合于创建功能控制比较多的service

注意：需要使用.config()来配置service的时候不能使用service()方法

③

Service方法

创建构造函数：

```
var baseLogger = function () {  
    this.messageCount = 0;  
    this.log = function (msg) {  
        console.log(this.msgType + ": " +  
            (this.messageCount++) + " " + msg);  
    }  
};
```

Demo:listing10.js

③

Service方法

原型继承：

```
var debugLogger = function () { };  
debugLogger.prototype = new baseLogger();  
debugLogger.prototype.msgType = "Debug";
```

```
var errorLogger = function () { };  
errorLogger.prototype = new baseLogger();  
errorLogger.prototype.msgType = "Error";
```

Demo:listing10.js

③

Service方法

创建服务：

```
angular.module("Services", [])  
    .service("logService", debugLogger)  
    .service("errorService", errorLogger);
```

Demo:listing10.js

③

Service方法

应用服务：
`<script src="directives.js"></script>`
`<script src="Listing 10.js"></script>`

```
angular.module("exampleApp", ["Directives", "Services"])
.controller("defaultCtrl", function (
    $scope, logService, errorService) {
    $scope.$watch('data.clicks', function (newVal) {
        logService.log("单击的次数(log): " + newVal);
        errorService.log("单击的次数(err): " + newVal);
    });
});
```

Demo:listing 09.html

3

Service方法

不使用原型的service方法。

```
angular.module("Services", [])  
.service("logService", function () {  
  return {  
    messageCount: 0,  
    log: function (msg) {  
      console.log("Debug: " +  
        (this.messageCount++) + " " + msg);  
    }  
  });  
});
```

Demo:listing11.js

4

Provider方法

provider()——提供器方式

provider()是创建service最底层的方式，这也是唯一一个可以使用.config()方法配置创建service的方法

```
var app=angular.module("custom", []);  
app.provider("serviceName", function(){});
```

自定义服务名称

工厂函数

provider方法可以更好地控制被创建或被配置的服务对象的方式。

4

Provider方法

provider方法的参数是将被定义服务的名称和工厂函数。工厂函数必须返回**提供器对象**，**\$get方法**——可以返回服务对象。

```
angular.module("Services", [])  
.provider("logService", function() {  
    return {  
        $get: function () {  
            return {  
                messageCount: 0,  
                log: function (msg) {  
                    console.log("(LOG +  
                        "+this.messageCount+++ " ) " + msg);
```

4

Provider方法

向提供器对象添加函数

```
return {  
  messageCounterEnabled: function (setting) {  
    if (angular.isDefined(setting)) {  
      counter = setting;  
      return this;  
    } else {  
      return counter;  
    }  
  },  
};
```

Demo:listing14.js

4

Provider方法

向提供器对象配置服务

```
.config(function (logServiceProvider) {  
  logServiceProvider.debugEnabled(true).messageCo  
  unterEnabled(false);  
})
```

Demo:listing15.html

5

理解三者关系

factory : 把 service 的方法和数据放在一个对象里，并返回这个对象

service : 通过构造函数方式创建 service，返回一个实例化对象

provider : 创建一个可通过config 配置的 service，\$get 中返回的就是用 factory 创建 service 的内容。

从底层实现上来看，service 调用了 factory，返回其实例；factory 调用了 provider，返回其 \$get 中定义的内容。factory 和 service 功能类似，只不过 factory 是普通 function；service 是构造器；provider 是加强版 factory，返回一个可配置的 factory。

3

全局对象

①

DOM API全局对象

名称	描述
\$window	提供DOM window对象的引用
\$anchorScroll	滚动浏览器窗口到指定锚点
\$document	提供jqLite对象包括DOM window.document对象
\$interval	提供window.setInterval函数的增强封装
\$timeout	提供window.setTimeout函数的增强封装
\$location	提供URL的入口
\$log	提供console对象的封装

①

访问window对象

\$window是对全局window对象的封装

```
angular.module("exampleApp", [])  
  .controller("defaultCtrl", function ($scope,  
    $window) {  
    $scope.displayAlert = function(msg) {  
      $window.alert(msg);  
    }  
  });
```

Demo:listing 16.html

②

访问document对象

\$document是一个包含全局window.document对象的的
jqLite对象

```
angular.module("exampleApp", [])  
  .controller("defaultCtrl", function ($scope,  
    $window, $document) {  
    $document.find("button").on("click",  
function (event) {  
    $window.alert(event.target.innerText);  
});  
});
```

Demo:listing 17.html

3

访问URL

\$location服务是围绕全局window对象的location对象的封装。提供了访问当前URL的入口。

\$location服务操作URL第一个 **#** 号后面的部分

Scheme:	//	Login:password@	Address	:port	/path/to/resource	?query_string	#fragment
协议	URL标识符	身份验证	服务器地址	服务器端口	文件路径	搜索项	散列

URL入口由三部分组成：**路径 (path)**、**搜索项(search term)**、**散列(hash)**

散列(hash) --- #

#后面出现的任何字符，都会被浏览器解读为位置标识符

搜索项(search term)--- ?

\$location服务的方法

`absUrl()`：返回当前文档完整的URL。

`hash()`：获取或设置URL的散列部分。

`host()`：返回URL中的主机名称。

`path()`：获取或设置URL 路径，并返回\$location。

`port()`：返回当前路径的端口号。

`protocol()`：返回当前URL的协议。

`replace()`：如果被调用，就会用改变后的URL直接替换浏览器中的历史记录，而不是在历史记录中新建一条信息，这样可以阻止『后退』。

`search()`：以对象形式返回当前url的搜索部分。

`url()`：当不带参数时，返回url；当带有参数时，返回\$location。

3

访问URL

\$location服务的事件

\$locationChangeStart: URL被改变前触发

\$locationChangeSuccess: URL被改变后触发

```
angular.module("exampleApp", [])  
  .controller("defaultCtrl", function ($scope,  
    $location) {  
    $scope.$on("$locationChangeSuccess",  
    function (event, newUrl) {  
      $scope.url = newUrl;  
    });  
  });
```

Demo:listing 18.html

4

使用\$log服务

\$log来取代console.log

```
$scope.setName = function(){  
    $scope.yourname=$scope.fname+" "+$scope.lname;  
    $log.info("info-----"+$scope.yourname);  
    $log.log("log-----"+$scope.yourname);  
    $log.error("error-----"+$scope.yourname);  
    $log.warn("warn-----"+$scope.yourname);  
}
```

Demo:listing19.html

本课小结

1. 模块
2. 创建和使用服务
3. 全局对象

TNAKS

主讲：王智娟

QQ: 24132228

Email: wangzhijuan@onest.net