第13讲 供应与注入服务

教学目标

1. 依赖注入
2. 管理注入
3. 声明依赖

1 依赖注入

① 什么是依赖注入

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。

为什么使用依赖注入？

处理代码之间的依赖关系，减少组件间的耦合。

② 依赖注入

对于一个DI容器来说，必须具备3个要素：

注册服务、获取对象、声明依赖关系。

- **注册服务：**module和**$provide**服务；
- **获取对象：$injector服务**用来获取对象
- **声明依赖关系：**在angular中有3种方式——

  推断式注入，标记式注入，内联式注入

自定义服务

```
angular.module("exampleApp", [])
.factory('$log',function(){……})
.service('$log1',function(){……})
.provider("$log2", function() {……})
.controller("defaultCtrl",
      function ($scope, $log,$log1,$log2) {……});
```

Demo:listing01.html

④ 注册服务组件

$provide服务用于注册AngularJS组件，并注入它们解决依赖的服务。

名称	描述
factory(name , service)	定义服务
service(name , service)	定义服务
provider(name , service)	定义服务
decorator(name , service)	定义服务修饰器
value(name , value)	定义变量
constant(name , value)	定义常量

使用$provide服务

```
angular.module("exampleApp", [])
.config(function($provide){
    $provide.factory('$log',function(){
        return{
            log: function (msg) {
                console.log("输出信息: "+ msg);
            }
        };
    })
})
.controller("defaultCtrl", function ($scope, $log) {
    ……
});
```

Demo:listing02.html

⑤ 使用$provide服务

```
angular.module("exampleApp", [])
.config(function($provide){
    $provide.service('$log',function(){
        return{
            log: function (msg) {
                console.log("输出信息: "+ msg);
            }
        };
    })
})
.controller("defaultCtrl", function ($scope, $log) {
    ……
});
```

Demo:listing03.html

⑤ 使用$provide服务

```
angular.module("exampleApp", [])
.config(function($provide){
    $provide.provider('$log',function(){
        return{
            $get:function(){
                return {
                    log: function (msg) {
                     console.log("输出信息: "+ msg);
                    }
```

```
.controller("defaultCtrl", function ($scope, $log){}
```

Demo:listing04.html

⑤ 使用$provide服务

```
.factory('$loger',function(){
    return{
        log: function (msg) {console.log("输出信息"+msg);}
    };
})
.config(function($provide) {
    $provide.decorator("$loger",function($delegate){
        $delegate.oldLog = $delegate.log;
        $delegate.newlog= function (msg) {
              $delegate.oldLog("loger输出 " + msg);
         };
         return $delegate;
     });
})
```

Demo:listing05.html

⑤ 使用$provide服务

```
.config(function($provide){
    $provide.value('num',5);
    $provide.value('num1','hi');
    $provide.value('num2',[10,20,30]);
    $provide.value('num3',{name:'lily'});
})
```

```
.controller("defaultCtrl", function
($scope,num,num1,num2,num3) {
    $scope.clicked = function () {
        console.log('value值为: '+num+','
                +num1+','+num2+','+num3.name);
    };
});
```

Demo:listing06.html

⑤ 使用$provide服务

```
.config(function($provide){
    $provide.constant('num',5);
    $provide.constant('num1','hi');
    $provide.constant('num2',[10,20,30]);
    $provide.constant('num3',{name:'lily'});
})
```

```
.controller("defaultCtrl", function
($scope,num,num1,num2,num3) {
    $scope.clicked = function () {
        console.log('constant输出值为: '+num+','
                +num1+','+num2+','+num3.name);
    };
});
```

Demo:listing07.html

2 管理注入

$injector服务

$injector服务负责确定函数声明的依赖，并解决这些依赖。

名称	描述
annotate(fn)	获取指定函数的参数，包括那些不相应服务的
get(name)	获取指定服务名称的服务对象
has(name)	如果指定名称的服务存在，则返回true
invoke(fn,self,locals)	调用指定函数，使用指定的值作为该函数的this，并使用指定的非服务参数值

② 确定函数依赖---annotate( )

```
var logClick = function ($log, $hander, message) {
    ……
};
$scope.clicked = function () {
    //获取指定函数的参数
    var deps = $injector.annotate(logClick);
    for (var i = 0; i < deps.length; i++) {
        console.log("logClick函数的参数: " + deps[i]);
    }
};
```

Demo:listing08.html

③ 过滤函数参数---has( )

```
var logClick = function ($log, $handler, message) {
    ……
};
$scope.clicked = function () {
    var deps = $injector.annotate(logClick);
    for (var i = 0; i < deps.length; i++) {
        if ($injector.has(deps[i])) {
            console.log("logClick函数的参数: " + deps[i]);
        }
    }
};
```

Demo:listing09.html

④ 获取服务实例---get( )

可以通过**$injector.get方法**获得需要的服务对象。

```
$scope.clicked = function () {
    var deps = $injector.annotate(logClick);
    var arr = [];
    for (var i = 0; i < deps.length; i++) {
        if ($injector.has(deps[i])) {
            arr.push($injector.get(deps[i]));
        } else if (deps[i] == "message") {
            arr.push("message obj");
        }
    }
    console.log(arr);
};
```

Demo:listing10.html

⑤ 简化调用过程---invoke( )

$injector.invoke方法将找到服务，并管理需要为其提供的值。传入的invoke方法的参数依次是将被调用的函数、this值和与函数参数一致的属性的对象。

```
var logClick = function ($log, $handler, message) {
……
};
$scope.clicked = function () {
    var localVars = {$handler:'handler',
                     message: "message"};
    $injector.invoke(logClick, null, localVars);
};
```

Demo:listing11.html

⑤ $rootElement服务

从根元素中获取$injector服务。

$rootElement服务提供了访问应用了ng-app指令的HTML元素的方法。$rootElement服务对象有个额外的方法injector，它返回$injector服务对象。

```
$scope.clicked = function () {
    var localVars = { $handler:'handler',
                      message: "message" };
    //从根元素中获取$injector服务
    $rootElement.injector().invoke(logClick, null,
localVars);
};
```

Demo:listing12.html

3 声明依赖

声明依赖

- 推断式注入
- 标记式注入
- 内联式注入

① 推断式注入

推断式注入方式**需要保证参数名称与服务名称相同**。

```
.config(function($provide){
    $provide.factory('hello1',function(){……});
    $provide.factory('hello2',function(){……})
})

.controller("defaultCtrl", function($scope,hello1,hello2){
    $scope.hello = function(){
        hello1.hello();
        hello2.hello();
    }
});
```

Demo:listing13.html

② 标记式注入

标记式注入方式需要设置一个依赖数组，数组内是依赖的服务名字，在函数参数中，可以随意设置参数名称，但是**必须保证顺序的一致性。**

```
var myCtrl = function($scope,hello1,hello2){
    $scope.hello = function(){
        hello1.hello();
        hello2.hello();
    }
};
myCtrl.$injector = ['hello1','hello2'];
app.controller("defaultCtrl", myCtrl);
```

Demo:listing14.html

③ 内联式注入

内联式注入方式直接传入两个参数，一个是名字，另一个是一个数组。这个数组的最后一个参数是真正的方法体，其他的都是依赖的目标，但是要保证与方法体的参数顺序一致（与标记注入一样）。

```
.controller("defaultCtrl",['$scope','hello1','he
llo2',function($scope,hello1,hello2){
    $scope.hello = function(){
        hello1.hello();
        hello2.hello();
    }
}]);
```

Demo:listing15.html

本课小结

1. 依赖注入
2. 管理注入
3. 声明依赖

TNAKS

主讲：王智娟
QQ：24132228
Email:wangzhijuan@onest.net