1月17日 日报

1. 已完成的内容

# AngularJS 依赖注入

## 什么是依赖注入

wiki 上的解释是：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。

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。与服务定位器模式形成直接对比的是，它允许客户端了解客户端如何使用该系统找到依赖

*一句话 --- 没事你不要来找我，有事我会去找你。*

AngularJS 提供很好的依赖注入机制。以下5个核心组件用来作为依赖注入：

* value
* factory
* service
* provider
* constant

## value

Value 是一个简单的 javascript 对象，用于向控制器传递值（配置阶段）：

// 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);

}});

## factory

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。

// 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);

}});...

## 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。

// 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. 工作成果
2. 未完成内容
3. 未完成原因
4. 遇到的问题及解决方案

一口气听太多定义，关于格式书写这种很容易晕，有时候迟疑一下就跟不上老师讲的了，没有实际的例子，总觉得很空，逻辑也显得特别绕。好久没有一下听这么多内容了，接受的有点慢，需要点时间缓冲。

1. 教师评语：