MockJs模拟数据接口

随着技术的不断变化，前端项目的工程结构设计和部署方式也在不断的变化中，为了适应企业开发的需求和提高开发效率，我们在大麦网项目中引入了分布式部署，实现前后端环境的分离开发，这样不仅能提高我们的开发效率还能减少服务器之间的相互影响，同时还能保证数据安全等；但是随之而来的问题比较明显，前端如何进行快速的开发，保证后期与后端进行交互的时候不受影响呢？或在跟后端进行调试时，不需要修改太多的代码又能实现与后端联调？在引入mock.js模拟数据接口之前还必须介绍一个前端工具connect-mock-middleware。

## connect-mock-middleware

connect-mock-middleware是一个非常方便、实用的mock模拟工具。

特点：

* 支持mockJs语法
* 支持json、jsonp
* 修改mock数据的时候不需要重新加载

### 1.1.1添加middleware

在config的index文件中添加如下代码，在本地开启一个地址固定的服务。（实际项目中和后端联调的时候会换成实际后端提供的接口服务地址）

const path = require('path')

module.exports = {

dev: {

// Paths

assetsSubDirectory: 'static',

assetsPublicPath: '/',

proxyTable: {

'/api': {

target: 'http://127.0.0.1:3721',

changeOrigin: true,

secure: false

}

},

### 1.1.2 写mock文件

支持两种请求：

1） get /api/xxx

2） post /api/<id>/123

<id> 链接路由表达式 /api/:id/123，id值会发生改变。具体的文件结构如下所示。

mock

└─get

├─api\_xxx.js

post

└─api\_@id\_xxx.js

以项目中的 “猜你喜欢接口”api\_list\_guesslike.js为例：

//猜你喜欢接口

module.exports = function (param) {

let paid = parseInt(param.body.itemTypeId) ;

// let paid = 1;

if (paid == 0) {

return {

"success": "string",

"errorCode": "string",

"msg": "",

"data|1-4": [

{

"id": '@string("number", 2)',

"itemName": '@csentence(5)',

"imgUrl": 'https://pimg.dmcdn.cn/perform/project/1381/138121\_n.jpg',

"areaId": '@string("number", 3)',

"areaName": '@province',

"address": '@county(true)',

"startDate": '@datetime("yyyy-MM-dd")',

"endDate": '@datetime("yyyy-MM-dd")',

"minPrice": '@integer(60, 100)'

}

]

}

}else if (paid == 1) {

return {

"success": "string",

"errorCode": "string",

"msg": "",

"data|1-4":

[

{

"id": '@string("number", 2)',

"itemName": '@csentence(10)',

"imgUrl": 'https://pimg.dmcdn.cn/perform/project/1381/138121\_n.jpg',

"areaId": '@string("number", 3)',

"areaName": '@province',

"address": '@county(true)',

"startDate": '@datetime("yyyy-MM-dd")',

"endDate": '@datetime("yyyy-MM-dd")',

"minPrice": '@integer(60, 100)'

}

]

}

}else {

return {

"success": "string",

"errorCode": "string",

"msg": "",

"data": {

"currentPage": param,

"pageCount": 1,

"pageSize": 10,

"total": 6

}

}

}

}

## mockJS语法

从上面的“猜你喜欢接口”中可能你会发现返回值data里面这些数据值的写法有些奇怪，其实这是mockJS的语法。下面就来介绍一下mockJS。

### 1.2.1 mockJS概述

Mock.js是一个模拟数据生成器，可以让前端独立于后端进行开发。如果你正在开发一个前端页面，可是后台还没有完成供你调用的Api，并且数据格式已经确定，那么你就可以使用Mock.js模拟相关的接口，生成假数据来查看页面效果。MockJS的功能：

* 根据数据模板生成模拟数据
* 模拟 Ajax 请求，生成并返回模拟数据
* 基于 HTML 模板生成模拟数据

### 1.2.2语法规范

Mock.js 的语法规范包括两部分：

1. 数据模板定义规范（Data Template Definition，DTD）
2. 数据占位符定义规范（Data Placeholder Definition，DPD）

#### 1）数据模板定义规范 DTD

**数据模板中的每个属性由 3 部分构成：属性名、生成规则、属性值：**

// 属性名 name

// 生成规则 rule

// 属性值 value

'name|rule': value

**注意：**

* 属性名 和 生成规则 之间用竖线 | 分隔。
* 生成规则 是可选的。
* 生成规则 有 7 种格式：
  1. 'name|min-max': value
  2. 'name|count': value
  3. 'name|min-max.dmin-dmax': value
  4. 'name|min-max.dcount': value
  5. 'name|count.dmin-dmax': value
  6. 'name|count.dcount': value
  7. 'name|+step': value
* **生成规则 的 含义 需要依赖 属性值的类型 才能确定。**
* 属性值 中可以含有 @占位符。
* 属性值 还指定了最终值的初始值和类型。

**生成规则和示例：**

##### 1. 属性值是字符串 ****String****

1. 'name|min-max': string

通过重复 string 生成一个字符串，重复次数大于等于 min，小于等于 max。

1. 'name|count': string

通过重复 string 生成一个字符串，重复次数等于 count。

##### 2. 属性值是数字 ****Number****

1. 'name|+1': number

属性值自动加 1，初始值为 number。

1. 'name|min-max': number

生成一个大于等于 min、小于等于 max 的整数，属性值 number 只是用来确定类型。

1. 'name|min-max.dmin-dmax': number

生成一个浮点数，整数部分大于等于 min、小于等于 max，小数部分保留 dmin 到 dmax位。

Mock.mock({

'number1|1-100.1-10': 1,

'number2|123.1-10': 1,

'number3|123.3': 1,

'number4|123.10': 1.123

})

// =>

{

"number1": 12.92,

"number2": 123.51,

"number3": 123.777,

"number4": 123.1231091814

}

##### 3. 属性值是布尔型 ****Boolean****

1. 'name|1': boolean

随机生成一个布尔值，值为 true 的概率是 1/2，值为 false 的概率同样是 1/2。

1. 'name|min-max': value

随机生成一个布尔值，值为 value 的概率是 min / (min + max)，值为 !value 的概率是 max / (min + max)。

##### 4. 属性值是对象 ****Object****

1. 'name|count': object

从属性值 object 中随机选取 count 个属性。

1. 'name|min-max': object

从属性值 object 中随机选取 min 到 max 个属性。

##### 5. 属性值是数组 ****Array****

1. 'name|1': array

从属性值 array 中随机选取 1 个元素，作为最终值。

1. 'name|+1': array

从属性值 array 中顺序选取 1 个元素，作为最终值。

1. 'name|min-max': array

通过重复属性值 array 生成一个新数组，重复次数大于等于 min，小于等于 max。

1. 'name|count': array

通过重复属性值 array 生成一个新数组，重复次数为 count。

##### 6. 属性值是函数 ****Function****

1. 'name': function

执行函数 function，取其返回值作为最终的属性值，函数的上下文为属性 'name' 所在的对象。

##### 7. 属性值是正则表达式 ****RegExp****

1. 'name': regexp

根据正则表达式 regexp 反向生成可以匹配它的字符串。用于生成自定义格式的字符串。

Mock.mock({

'regexp1': /[a-z][A-Z][0-9]/,

'regexp2': /\w\W\s\S\d\D/,

'regexp3': /\d{5,10}/

})

// =>

{

"regexp1": "pJ7",

"regexp2": "F)\fp1G",

"regexp3": "561659409"

}

#### 2）数据占位符定义规范 DPD

占位符 只是在属性值字符串中占个位置，并不出现在最终的属性值中。

占位符 的格式为：

@占位符

@占位符(参数 [, 参数])

**注意：**

1. 用 @ 来标识其后的字符串是 占位符。
2. 占位符 引用的是 Mock.Random 中的方法。
3. 通过 Mock.Random.extend() 来扩展自定义占位符。
4. 占位符 也可以引用 数据模板 中的属性。
5. 占位符 会优先引用 数据模板 中的属性。
6. 占位符 支持 相对路径 和 绝对路径。

Mock.mock({

name: {

first: '@FIRST',

middle: '@FIRST',

last: '@LAST',

full: '@first @middle @last'

}

})

// =>

{

"name": {

"first": "Charles",

"middle": "Brenda",

"last": "Lopez",

"full": "Charles Brenda Lopez"

}

}

注意：更多的mock数据示例可以参考<http://mockjs.com/examples.html> 网站进行配置。

### 1.3 Snail mock

通过上面的步骤，基本上可以实现后端接口的模拟，那么目前的问题是这些数据如何能被我们调用呢？这就是接下来要介绍的内容snail mock。它能够模拟服务器的功能，生成接口的url服务地址供我们调用。

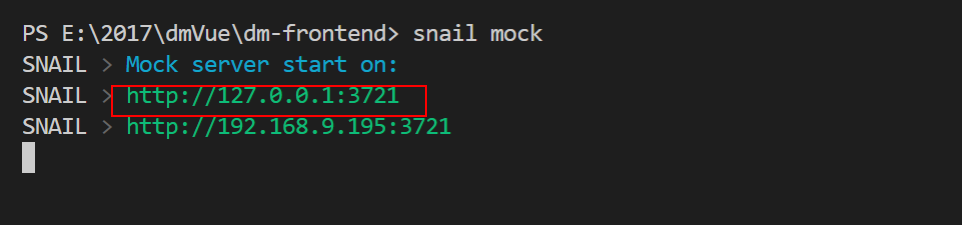
要使用这个前端工具首先需要初始化。在命令窗口执行如下命令：

$ npm install -g snail-cline

这样便全局安装了这个工具，接下来要开启mock服务也是需要再命令窗口执行如下命令：

$ snail mock

这时就可以启动一个模拟的服务了。具体如图1所示。<http://127.0.0.1:3721>这个地址就是前面在添加middleware的时候配置的地址。



然后我们在另起一个命令窗口，使用npm start命令去打开项目即可。

注意：初学者比较容易放的错误就是使用snail mock开启了一个服务，接下来要去打开项目的时候会中断这个服务，在本命令窗口下去执行npm start打开项目，这样请求会失败。

辅助工具：格式化json数据的工具

https://www.baidufe.com/fehelper