

AMP

API管理平台

来历

- 接口的约定和管理
没有统一的工具、没有统一的呈现
- Mock Server
工具众多、与API管理的脱节、模拟难度大

功能

- API的管理
以项目为跟节点，在此项目下创建API，以项目为单位划分成员，做增删改查的权限控制。
- Mock Server
按照一定的规则访问接口，返回选定的虚假数据。

Mock Server 示例

获取用户信息

请求地址

/user

请求参数

Key	Type	Comment	Required
id	String	用户ID	true

返回示例

```
{
  "name": "luoye",
  "email": "luoye@gomeplus.com"
}
```

Mock 地址

http://amp.gomeplus.com/mock/{项目ID}/user?id=123

AMP

退出



luoye
luoye

我的项目

请输入名称进行搜索

新建项目

📁 我的项目 >

📁 全部项目 >

📁 接口测试 >

📁 使用说明 >

📁 使用反馈 >

📁 更换背景 >

技术

- 服务端
 - koa2@^2.0.0
 - Mongodb@^3.0.4
- 前端
 - vue@^1.0.21
- 工具
 - webpack@^1.13.0
 - babel (babel-core@^6.7.7)


```
{  
  "codemirror": "^5.14.2",  
  "comment-regex": "^1.0.0",  
  "https": "^1.0.0",  
  "jquery": "^2.2.3",  
  "js-beautify": "^1.6.2",  
  "koa": "^2.0.0",  
  "koa-better-body": "^2.0.0",  
  "koa-convert": "^1.2.0",  
  "koa-cors": "0.0.16",  
  "koa-generic-session": "^1.10.2",  
  "koa-logger": "^1.3.0",  
  "koa-router": "^7.0.1",  
  "koa-static": "^2.0.0",  
  "koa-views": "^5.0.2",  
  "koa2-cookie-session": "0.0.1",  
  "ldapjs": "^1.0.0",  
  "md5": "^2.1.0",  
  "moment": "^2.13.0",  
  "mongoose": "^4.4.16",  
  "request": "^2.72.0",  
  "vue": "^1.0.21",  
  "vue-form": "^0.3.1",  
  "vue-resource": "^0.7.0",  
  "vue-router": "^0.7.13",  
  "vuex": "^0.6.3",  
  "wangeditor": "^2.1.10"  
}
```

koa2 相关

- babel做入口
- 异步流程控制
回调/观察者模式 > promise > async/await
- 中间件
灵活的执行时机

JS

```
// app.babel.js
require('babel-core/register')({
  presets: ['es2015', 'stage-0']
});
require('babel-polyfill');

require('./app.js');
```

JS

```
// app.js
import Koa from 'koa';
import Router from 'koa-router';

const app = new Koa();
const route = Router();

route.get('/', async (ctx, next) => {
  ctx.body = 'hello world!'
})

app.use(route.routes(), route.allowedMethods());
app.listen(3000);
```

```
User.post('login', async (ctx, next) => {  
  let USER = global.db.getModel('USER');  
  let email = ctx.body.email,  
      pwd = ctx.body.password,  
      remember = ctx.body.remember;  
  if (!email || !pwd) {  
    ctx.fail(400, '请填写邮箱和密码');  
    return;  
  }  
  
  let res = await ldapLogin(ctx);  
  let findUser = await USER.find({email: email});  
  if (!findUser.length) {  
    await USER.create({  
      email: email,  
      pwd: md5(pwd)  
      // ...  
    })  
  }  
  
  ctx.success(res, '登录成功');  
})
```

```
// response middleware
export default async (ctx, next) => {
  ctx.success = (result, msg) => {
    ctx.body = {
      code: 200,
      data: result,
      iserror: 0,
      msg: msg || ''
    }
  }
  ctx.fail = (code, msg) => {
    ctx.body = {
      code: code,
      data: '',
      iserror: 1,
      msg: msg
    }
  }
  await next();
};
```

```
var xResponseTime = function(req, res, next) {  
  var now = new Date();  
  req._startTime = now;  
  
  function log() {  
    console.log(new Date() - req._startTime + 'ms');  
  }  
  
  // 监听res的结束事件  
  res.once('close', log);  
  
  next();  
}  
app.use(xResponseTime);
```

```
const xResponseTime = async (ctx, next) => {  
  let start = new Date();  
  await next();  
  console.log(new Date() - start + 'ms');  
}  
app.use(xResponseTime);
```


Mongodb 相关

- 对js开发者友好
原生js语法调用
- 前后端数据结构统一
方便调用、接口灵活
- mongodb 外键
定义：ref
查询：populate

```
Model.create({  
    // 增  
})  
Model.remove({  
    // 删  
})  
Model.update({  
    // 改  
})  
Model.find({  
    // 查  
})
```

```
{
  "_id" : ObjectId("5791c11bfeb79ca5509a99d4"),
  "name" : "美信数据开发平台",
  "desc" : "美信数据开发平台",
  "creator" : ObjectId("573e7b7d796923827d1ab00d"),
  "remark" : "<pre style=\"max-width:100%;overflow-x:auto;\"><code c
  "members" : [
    ObjectId("576bb9a7974f9f147f520cd3"),
    ObjectId("573e7b7d796923827d1ab00d"),
    ObjectId("5796ccb42bf51e0d0a7e8b4d"),
    ObjectId("5796ccb62bf51e0d0a7e8b4e"),
    ObjectId("5796ccc12bf51e0d0a7e8b4f"),
    ObjectId("574cf3f79c78b131703b85bf"),
    ObjectId("5796f5d82bf51e0d0a7e8b50"),
    ObjectId("57a9b0172bf51e0d0a7e8b68")
  ],
  "create_time" : 1469169947560.0,
  "__v" : NumberInt(0)
}
```

[illegible]


```
{
  users: {
    name: String,
    email: String,
    password: String,
    department: String,
    role: String,
    bg: String,
    avatar: String
  },
  projects: {
    name: { type: String, required: true },
    desc: String,
    creator: { type: Schema.Types.ObjectId, required: true, ref: 'users' },
    create_time: { type: Number, required: true, default: Date.now },
    members: [{ type: Schema.Types.ObjectId, required: true, ref: 'users' }],
    remark: String // 备注
  },
  urls: {
    name: { type: String, required: true },
    desc: String,
    creator: { type: Schema.Types.ObjectId, required: true, ref: 'users' },
    create_time: { type: Number, required: true, default: Date.now },
    url: String,
    parent_project: { type: Schema.Types.ObjectId, required: true, ref: 'projects' },
    status: { type: Number, default: 0 }, // 0未完成 1完成
    method: String,
    request_params: Array, // 请求参数说明
    request_example: Array, // 请求示例
    response_params: Array, // 返回的参数说明
    response_example: { // 返回示例
      in_use: { type: Number, default: 0 },
      example_array: { type: Array }
    },
    remark: String // 备注
  },
  feedback: {
    user: { type: Schema.Types.ObjectId, required: true, ref: 'users' },
    feedback: String
  }
}
```

```
Model.find(realQuery)
  .populate('creator')
  .populate('members')
  .populate('parent_project')
```

JS

接口相关

- 统一的接口规范
- RESETful 接口风格
- Mock Server 的实现

```
// response middleware
export default async (ctx, next) => {
  ctx.success = (result, msg) => {
    ctx.body = {
      code: 200,
      data: result,
      iserror: 0,
      msg: msg || ''
    }
  }
  ctx.fail = (code, msg) => {
    ctx.body = {
      code: code,
      data: '',
      iserror: 1,
      msg: msg
    }
  }
  await next();
};
```



```
// 对 project/url 的增删改查
import Router from 'koa-router';

const Api = Router({
  prefix: '/api'
});

Api
  .get('/:model', async (ctx, next) => {

  })
  .post('/:model', async (ctx, next) => {

  })
  .put('/:model', async (ctx, next) => {

  })
  .delete('/:model', async (ctx, next) => {

  })

export default Api;
```

获取用户信息

请求地址

/user

请求参数

Key	Type	Comment	Required
id	String	用户ID	true

返回示例

```
{
  "name": "luoye",
  "email": "luoye@gomeplus.com"
}
```

Mock 地址

http://amp.gomeplus.com/mock/{项目ID}/user?id=123

```
// Mock server 的实现
// 1. 从 request 中解析 { 项目ID }、{ 请求地址 }、{ 请求方式 }、{ 请求参数 }
// 2. 根据上一步获取的条件从库中查找相应数据
// 3. 校验：此接口是否存在、请求参数是否符合要求等
// 4. 返回选定的虚假数据
```

```
import Router from 'koa-router';
const Mock = Router({ ①
  prefix: '/mock'
});
// 获取相关的信息
const formatRequestUrl = function(ctx) { ③
  let url = ctx.request.url;
  let result = {};
  // 获取项目ID
  result.parent_project = url.match(/mock\/(.*)\/)/[1];
  // 获取 mock api 的 url
  result.url = url.match(/mock\/.*\/(.*)/)[1].replace(/?.*/, '')
  // 获取 mock api 的 method
  result.method = ctx.method.toLocaleLowerCase();
  return result;
}
// 错误类型
const errMap = {
  0: '参数类型有误', 1: '缺少必要参数', 2: '没有符合要求的返回示例'
}
// 对 mock api 进行合法性验证，如必需的参数有没有带上等(简单的对比，不具体写)
const checkParams = function(ctx) { ⑥
  // ....
  return {
    err: 1,
    map: 0
  }
}
Mock.all('*', async (ctx, next) => {
  let Url = global.dbHandle.getModel('urls');
  let apiDetail = formatRequestUrl(ctx); ②
  let result = await Url.find(apiDetail); ④
  let mockCheck = checkParams(ctx); ⑤
  if (mockCheck.err) {
    ctx.fail(404, errMap[mockCheck.map]);
    return;
  }
  ctx.success(200, JSON.parse(result[0])); ⑦
});
export default Mock;
```

前端相关

- vuex的使用
- 结合AMP的前端开发流程


```
▼ state: Object
  ▼ alertConfig: Object
    delay: 2500
    msg: ""
    show: false
    type: "info"
  ▼ confirmConfig: Object
    apply: "func"
    cancel: "func"
    msg: "提示信息"
    show: false
    title: "弹出对话框"
  isLogin: true
  loading: false
  ▼ userInfo: Object
    __v: 0
    _id: "573e7b7d796923827d1ab00d"
    avatar: "upload_37e21b36d0394e36ae73a8aed70b254b.jpg"
    bg: "upload_9c77573579f09fa9d00aa102a96583ec.jpg"
    department: "技术中心"
    email: "luoye@gomeplus.com"
    name: "骆也"
    role: "前端业务开发部"
```

JS

```
// package.json
"scripts": {
  "localdev": "node build/app.babel.server.js --env=localdev",
  // ...
  "production": "node build/app.babel.build.js --env=production"
}
```

JS

```
// webpack.config.js
// env => 从命令行中解析得到的环境参数
plugins: [
  new webpack.DefinePlugin({
    'process.env': JSON.stringify(env);
  })
]
```

JS

```
// http.config.js
export default {
  localdev: 'http://amp.gomeplus.com/mock/5791c11bfef79ca5509a99d4',
  // ...
  production: 'https://gomeplus.com/api/v2'
};
```

JS

```
// app.js (前端应用的入口文件)
import httpConfig from './config/http.config.js';
Vue.http.options.root = httpConfig[process.env];
```

webpack/babel

- webpack
规范化的前端开发流程、进阶方法
- babel
入口引导、语法转换

<http://luoye.iask.in:9090>

