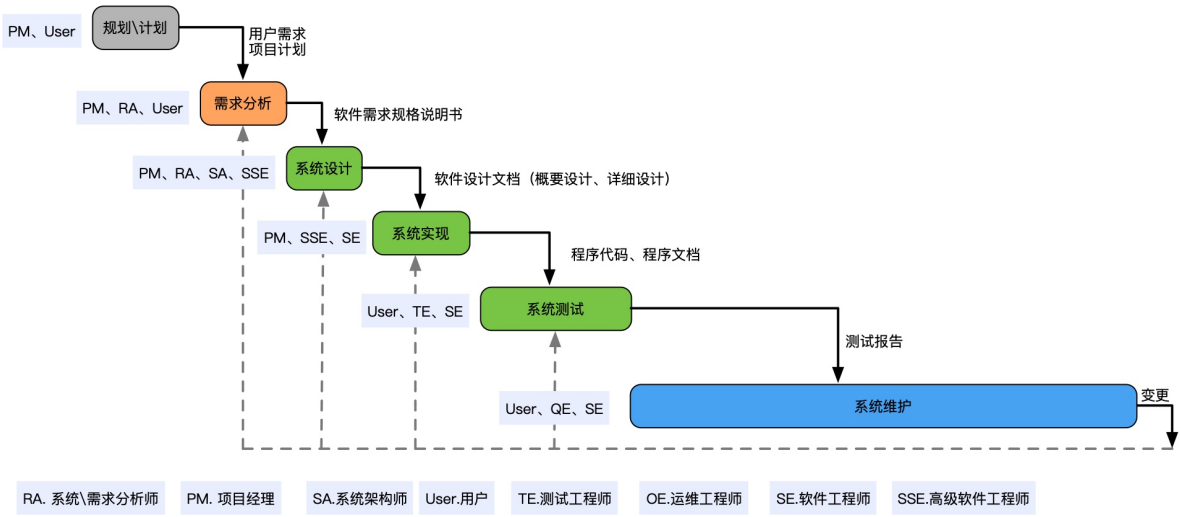
day30 mock数据

软件开发生存周期

- 问题定义
- 可行性分析
- 需求分析 （产品经理（原型设计 axuer 墨刀 ）确定技术选型）
- 概要设计 （文档化 UML图）
- 详细设计 （数据库设计 ...）
- 编码 (后端、前端（前后端联调））
- 测试 （测试人员（白盒测试、黑盒测试、自动化测试） （禅道））
- 运维 （（云计算运维 自动化运维 ）（实施））



前端和后台的联调是最后再进行操作，那么前期的前端的数据应该是模拟的数据（测试数据接口获取数据，自己进行mock）。

前端和后台的比例 1:2 1:3 1:4 (如果按照项目组的划分 一般分前端俩个后端5个测试一个 ui 一个 项目组的统筹称为项目组长，对应的项目组长是受项目经理的调配的（可能俩到三个项目组由一个项目统筹）。）

mock数据

- **使用在线mock （内部采用的是mock.js）**
- **使用测试数据（后端接口）**
- **自己mock**

  mock.js 进行数据mock

  json-server来进行mock

在线mock平台了解

- apipost

- fastmock
- ....

fastmock的使用

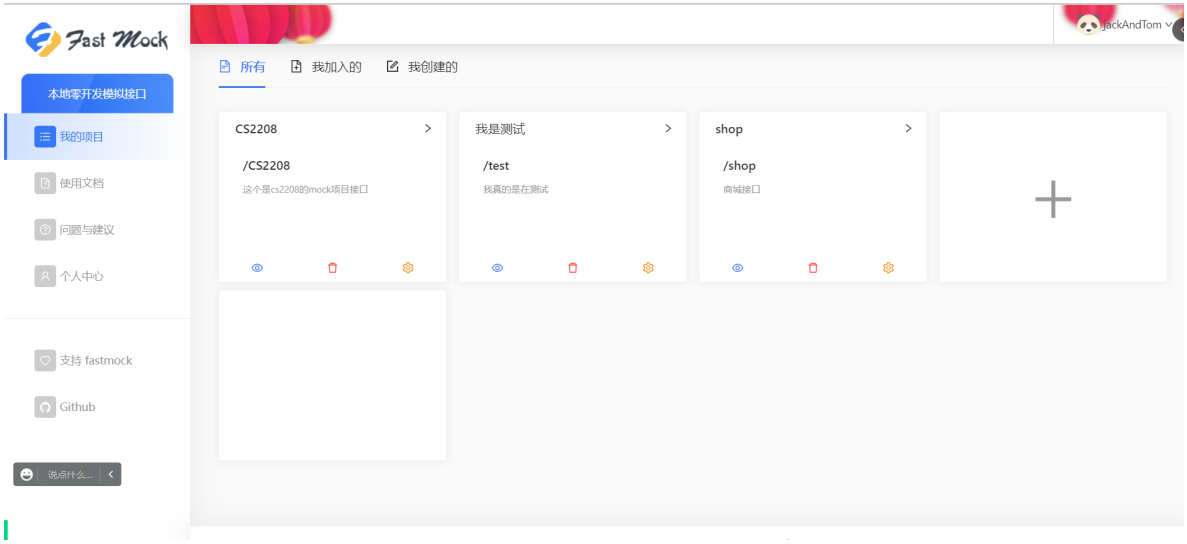
[fastmock地址]

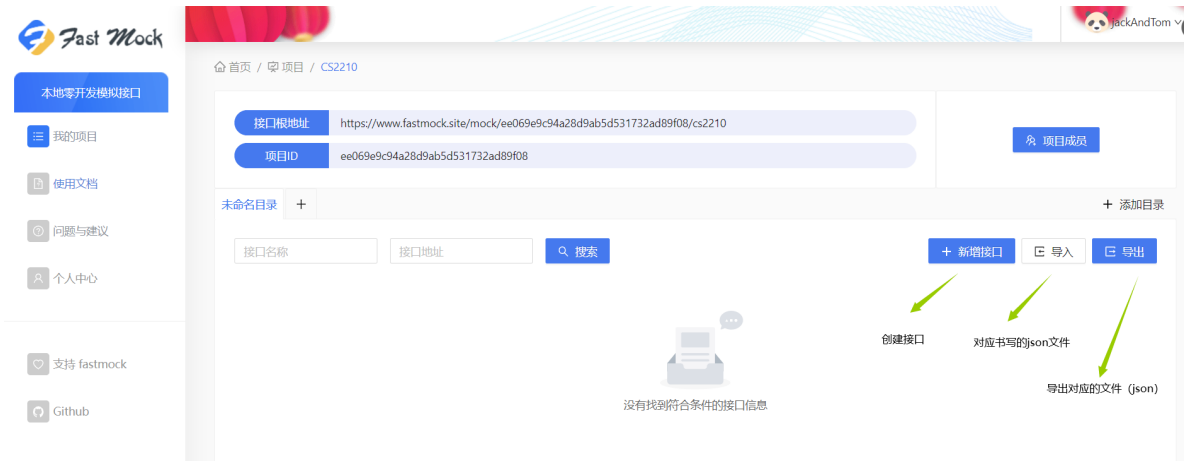
使用步骤

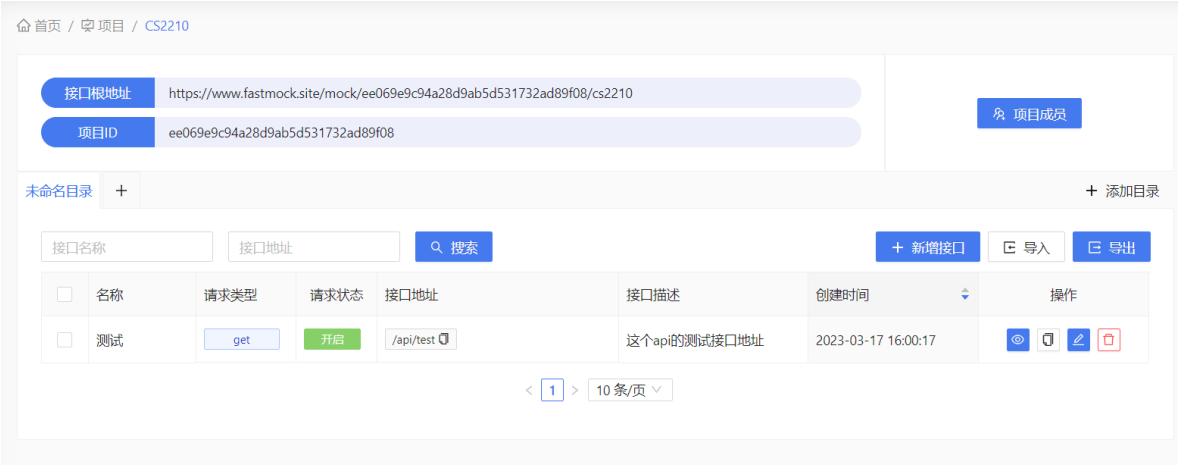
1、注册，登录


Fast Mock
用户登录
用户名或者邮箱地址
密码
请输入右侧计算结果
登 录
还没有账号? 注册账号
忘记密码
其他登录方式:

2、新建对应的接口


Fast Mock
本地零开发模拟接口
我的项目
使用文档
问题与建议
个人中心
支持 fastmock
Github
所有
我加入的
我创建的
CS2208
/CS2208
这个是cs2208的mock项目接口
我是测试
/test
我真的是在测试
shop
/shop
商城接口


Fast Mock
本地零开发模拟接口
我的项目
使用文档
问题与建议
个人中心
支持 fastmock
Github
首页 / 项目 / CS2210
接口根地址
https://www.fastmock.site/mock/ee069e9c94a28d9ab5d531732ad89f08/cs2210
项目ID
ee069e9c94a28d9ab5d531732ad89f08
项目成员
未命名目录
添加目录
接口名称
接口地址
搜索
新增接口
导入
导出
创建接口
对应书写的json文件
导出对应的文件（json）
没有找到符合条件的接口信息




```
{
  "code":200,
  "message":"成功",
  "data":{
    "city":"@city(true)",
    "address":"@zip()",
    "username":"@cname()",
    "age":"@integer(1, 100)"
  }
}
```

3、访问

```
<button>点击请求</button>
<script>
    document.querySelector('button').onclick = function () {
        let xhr = new XMLHttpRequest()
        xhr.open('get',
'https://www.fastmock.site/mock/ee069e9c94a28d9ab5d531732ad89f08/cs2210/api/test
')
        xhr.send()
        xhr.onreadystatechange = () => {
            if (xhr.readyState == 4 && /^2\d{2}$/.test(xhr.status)) {
                console.log(JSON.parse(xhr.responseText))
            }
        }
    }
</script>
```

mock.js文档相关内容

随机生成的内容

[mock.js的地址]

- Mock.Random
  - Basic
    boolean, natural, integer, float, character, string, range
  - Date
    date, time, datetime, now
  - Image
    img, dataImage
  - Color
    color, hex, rgb, rgba, hsl
  - Text
    paragraph, sentence, word, title, cparagraph, csentence, cword, ctitle
  - Name
    first, last, name, cfirst, clast, cname
  - Web
    url, domain, protocol, tld, email, ip
  - Address
    region, province, city, county, zip
  - Helper
    capitalize, upper, lower, pick, shuffle
  - Miscellaneous
    guid, id, increment

token的构成

token是一个令牌，它是在jwt（json-web-token）中，它是为了保证安全性所存储的一个存储少了数据的一个容器。

token的构成

- 密钥 （token需要被加密）
- 数据 （一般存储的用户的id）
- 过期时间 （如果过期了应该没有用）

JWT简介

- JWT：Json Web Token，是基于Json的一个公开规范，这个规范允许我们使用JWT在用户和服务器之间传递安全可靠的信息，他的两大使用场景是：认证和数据交换
- 使用起来就是，由服务端根据规范生成一个令牌（token），并且发放给客户端。此时客户端请求服务端的时候就可以携带者令牌，以令牌来证明自己的身份信息。
- 作用：类似session保持登录状态 的办法，通过token来代表用户身份。

简单登录接口

- 检索对应的用户名和密码
- 正确的话发送对应的token到达对应的浏览器
- 浏览器接收就需要存储对应的token （cookie localstroage sessionstroage）
- 跳转主页（提示登录成功）

```
{
  "data":function({_req,Mock}){
    if(_req.body.username == "admin" && _req.body.password == "123456"){
      return Mock.mock({
        "username":"@cname",
        "userId":"@id",
        "token":"@word(32)"
      })
    }else{
      return {}
    }
  }
}
```

获取内容的接口

- token在请求发送的时候放在请求头（读取对应本地存储的token）
- 检索token （是否过期 以及是否正确）
- 正确返回数据 不正确重定向到登录页面

```
{
    "data":function({_req,Mock}){
        if(_req.headers.token){
            return Mock.mock({
                "username":"@clast() @cfirst()",
                "address":"@county()",
                "email":"@email",
                "phone":"@phone",
                "nickname":"@firt @last",
                "keyword":"@word(3)",
                "imgUrl":"@image(200x100,#ffcc33,#FFF,png,@word)",
                "id":"@id",
                "time":"@datetime",
                "lastTime":"@now"
            })
        }
    }
}
```

Json-Server

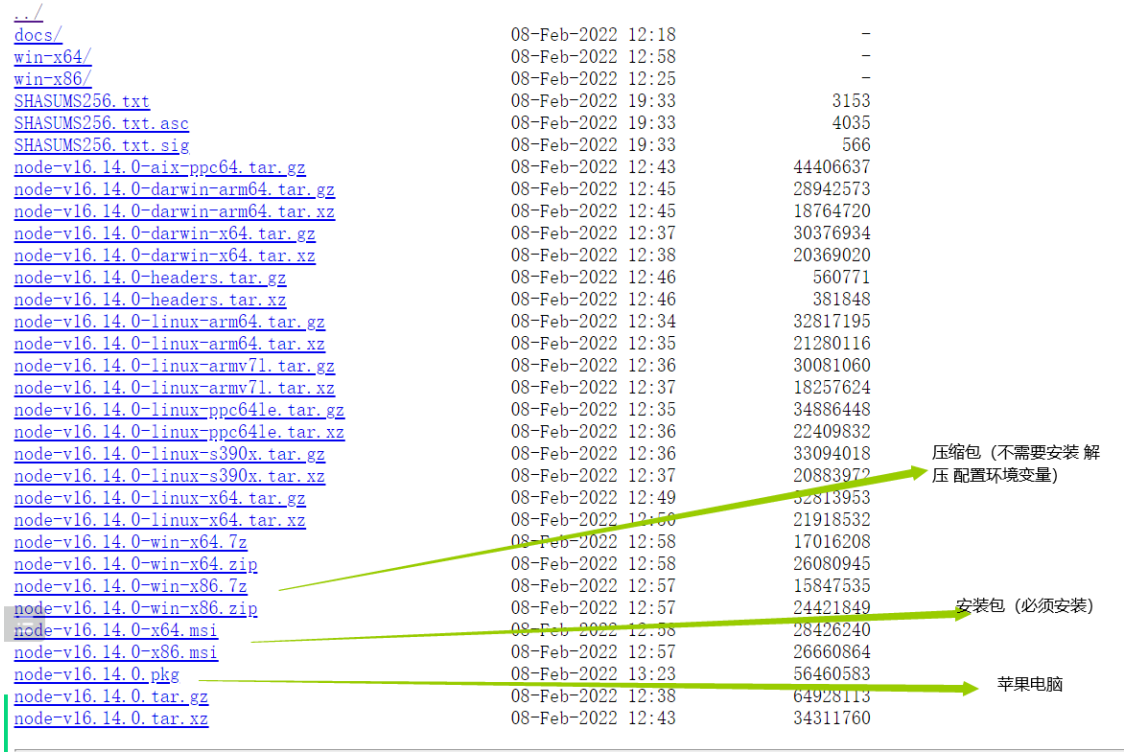
概述：

Json-Server它是一个第三方插件，它是通过json文件来模拟对应的接口主要利用了node的express来进行相关服务搭建。如果需要使用先需要安装node环境。

使用步骤

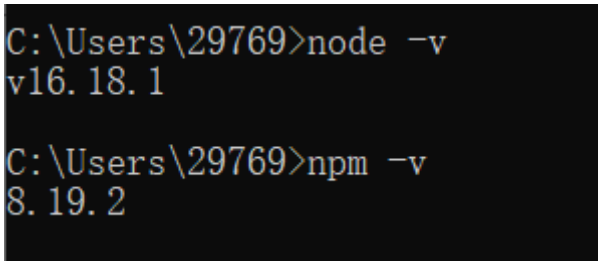
- 安装node

  下载地址

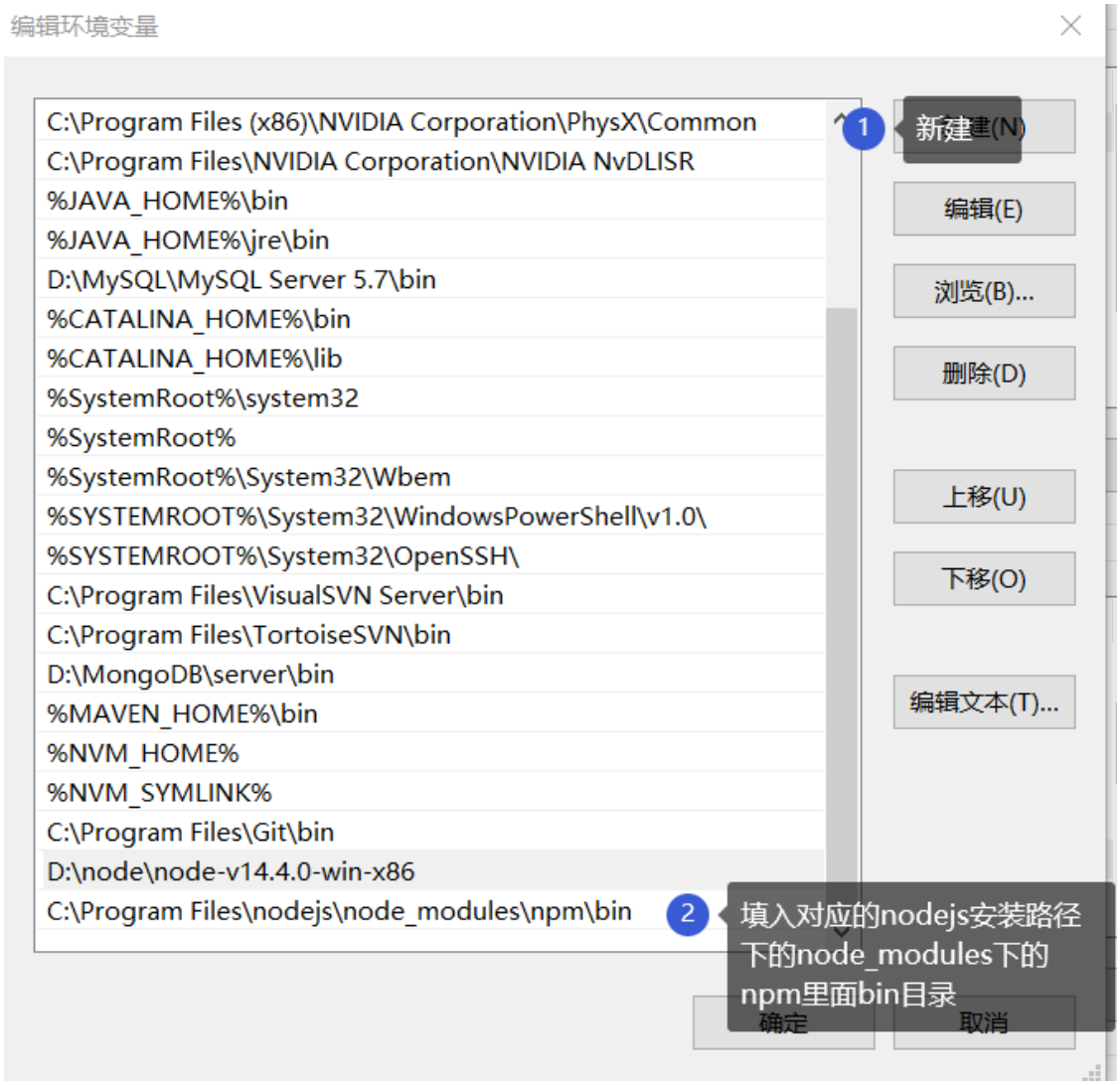


如果是安装版 无脑安装

```
node -v
npm -v
```

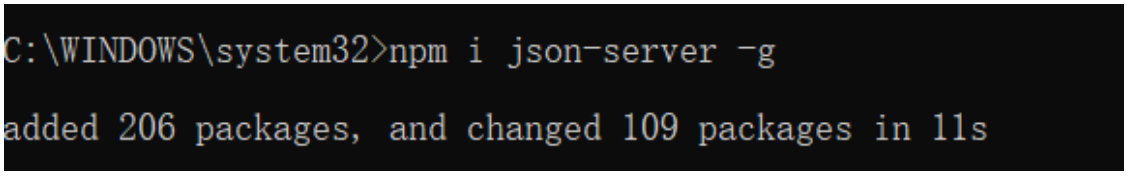


如果node -v没有 或 npm -v没有那么需要配置环境变量



- 通过node下载json-server

```
npm i json-server -g #全局安装对应的json-server
```



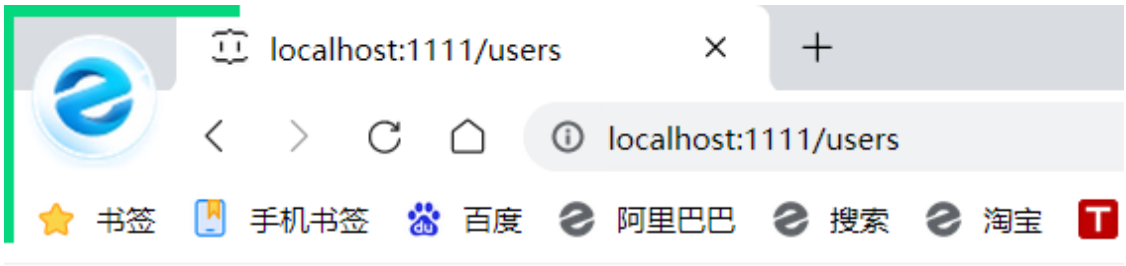
- 新建json文件

```
{
    "users":[
        {
            "id":1,
            "username":"jack",
            "email":"123q@111.com"
        },
        {
            "id":2,
            "username":"tom",
            "email":"123q@111.com"
        }
    ]
}
```

- 通过json-server来监听json文件形成服务

```
D:\CSH5\CSH5-2210\day30\code>json-server -w db.json -p 1111

  \{^_^}/ hi!

  Loading db.json
  Done

  Resources
  http://localhost:1111/users

  Home
  http://localhost:1111

  Type s + enter at any time to create a snapshot of the database
  Watching...
```

- 访问对应的接口




```
[
  {
    "id": 1,
    "username": "jack",
    "email": "123q@111.com"
  },
  {
    "id": 2,
    "username": "tom",
    "email": "123q@111.com"
  }
]
```


RESTFUL

restful是对应的遵从rest规范的一种接口风格，主要用于前后端分离。它以返回json格式的数据以对应的请求方式来区分对应的操作。

对应的请求方式

- get 主要用于获取
- post 主要用于添加
- put 请求用于修改 （修改一个）
- patch 请求用于修改 （修改多个）
- delete 请求用于删除

json-server生成的接口就是rest风格的接口。

使用 _limit 和 _page来进行分页 （返回的是数组）

```
http://localhost:1111/users?_limit=1&_page=2
```

可以使用/id的方式直接获取对应id的内容 （返回的是对象）

```
http://localhost:1111/users/2
```

支持?传递参数进行查询 （返回的是数组）

```
http://localhost:1111/users?username=jack
```

delete请求（删除 必须以/来传递id）

```
http://localhost:1111/users/1
```

put请求 （修改 必须以/来传递id 以body来传递数据）返回修改的对象

```
http://localhost:1111/users/2
```

post请求 （添加 里面id会自动递增） 返回新增的对象

```
http://localhost:1111/users
```

原始的xhr 某些浏览器不支持对应的put请求及delete请求 （可以使用jQuery的ajax来进行请求）