1、创建普通nodejs项目 采用KOA+vue的模式

2、注册订阅号并登录，进入开发----->开发者工具----->公众平台测试账号

3、nodemon **npm install -g nodemon** 联机调试工具 安装完成进入项目直接nodemon 相当于执行node index.js

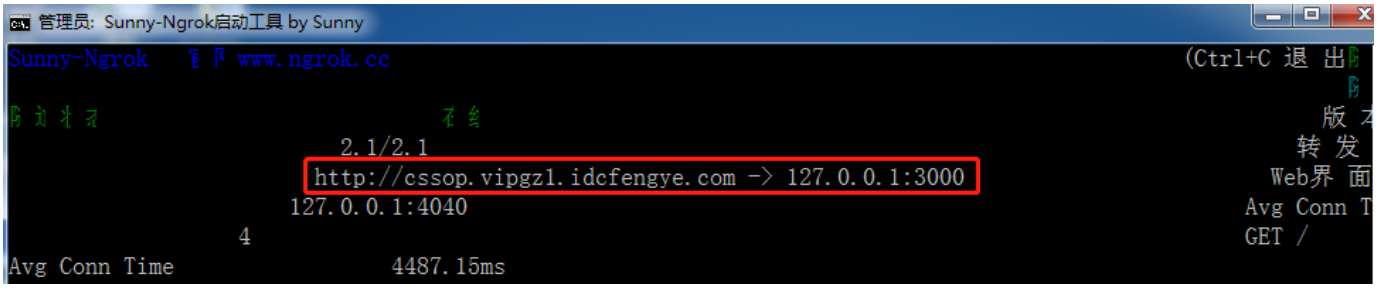
4、Sunny-Ngrok配置 =〉作用：本地IP映射为外网地址

https://www.ngrok.cc/user.html

隧道管理

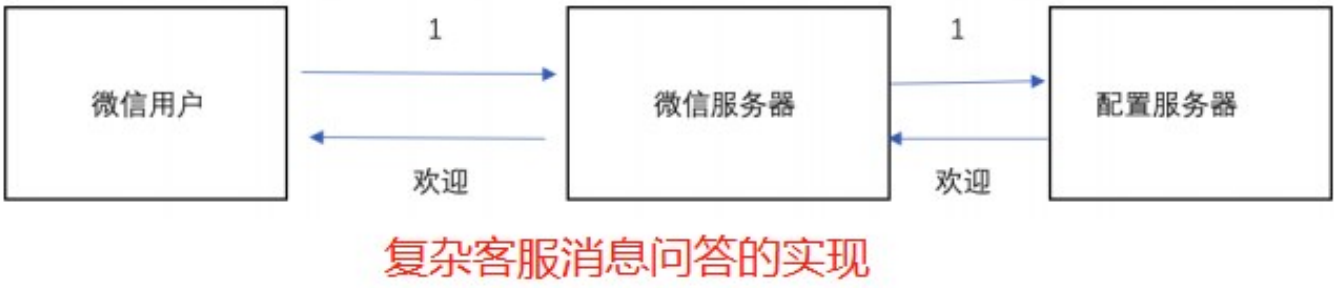
注意：未付款订单将会在一个小时候自动取消

隧道id	隧道名称	隧道协议	本地端口	服务器类型	到期日期	赠送域名	状态	操作
> 151752187577	erp	http	47.93.241.158:8080	Ngrok (客户端下载)	2019-04-01 15:17:52 续费	http://gdgnsto.vipgz1.idcfengye.com	查看状态	编辑
> 151304187577	本地测试2	http	127.0.0.1:3000	Ngrok (客户端下载)	2019-04-01 15:13:04 续费	http://cssop.vipgz1.idcfengye.com	查看状态	编辑
> 150336187577	本地测试	https	127.0.0.1:3000	Ngrok (客户端下载)	2019-04-01 15:03:36 续费	https://gdsh.vipgz1.idcfengye.com	查看状态	编辑



5、公众号的关键词自动回复

6、配置服务器实现多规则问答



复杂客服消息问答的实现

7、配置服务器验证微信服务器，确认是由微信发过来的数据

用token，时间戳，随机数 做一个SHA1的加密算法，生成一个加密自符串

配置服务器对接收到的三个数据进行加密计算，其加密结果如果和微信服务器加密结果

一致，则验证通过

SHA1算法

> 安全哈希算法（Secure Hash Algorithm）主要适用于数字签名标准（Digital Signature Standard DSS）里面定义的数字签名算法（Digital Signature Algorithm DSA）。对于长度小于2^64位的消息，SHA1会产生一个160位的消息摘要。当接收到消息的时候，这个消息摘要可以用来验证数据的完整性。在传输的过程中，数据很可能会发生变化，那么这时候就会产生不同的消息摘要。SHA1有如下特性：不可以从消息摘要中复原信息；两个不同的消息不会产生同样的消息摘要,(但会有1x10 ^ 48分之一的机率出现相同的消息摘要,一般使用时忽略)。

```
// 验证
router.get('/wechat', ctx => {
console.log('微信认证...', ctx.url)
const {
query
} = url.parse(ctx.url, true)
const {
signature, // 微信加密签名，signature结合了开发者填写的token参数和请求中的timestamp参数、nonce参数。（token,时间，随机数）
timestamp, // 时间戳
nonce, // 随机数
echostr // 随机字符串
} = query
console.log('wechat', query)
// 将 token timestamp nonce 三个参数进行字典序排序并用sha1加密
let str = [conf.token, timestamp, nonce].sort().join('');
let strSha1 = crypto.createHash('sha1').update(str).digest('hex');
console.log(`自己加密后的字符串为：${strSha1}`);
console.log(`微信传入的加密字符串为：${signature}`);
console.log(`两者比较结果为：${signature == strSha1}`);
// 签名对比，相同则按照微信要求返回echostr参数值
if (signature == strSha1) {
```

```
ctx.body = echostr
} else {
ctx.body = "你不是微信"
}
})
```

8、微信公众平台相关开发接口（微信服务器端API调用）

https://mp.weixin.qq.com/wiki?t=resource/res_main&id=mp1421140837

要调用**API**首先要先获取**access_token**，有了**access_token**才能调其它接口

```
// 获取零时数据存储
const tokenCache = {
access_token : '',
updateTime:Date.now(),
expires_in:7200
}
// 调用API获取access_token数据
router.get('/getToken',async ctx => {
const url = `https://api.weixin.qq.com/cgi-bin/token?
grant_type=client_credential&appid=${conf.appid}&secret=${conf.appsecret}`
const res = await axios.get(url)
Object.assign(tokenCache,res.data,{ //将获取到的数据存入零时数据tokenCache中
updateTime:Date.now() //更新时间
})
ctx.body = res.data // 返回数据
})
```

9、使用[co-wechat-api]库调**API**接口，这个[co-wechat-api]封装好的方法获取数据

```
// 引入co-wechat-api库
const WechatAPI = require('co-wechat-api')
const api = new WechatAPI(conf.appid,conf.appsecret)
// 路由方法
```

```
router.get('/getFollowers', async ctx => {
let res
// 调用封装好的方法获取数据
res = await api.getFollowers()
res = await api.batchGetUsers(res.data.openid,'zh_CN')
console.log('batchGetUser',res)

ctx.body = res
})
```