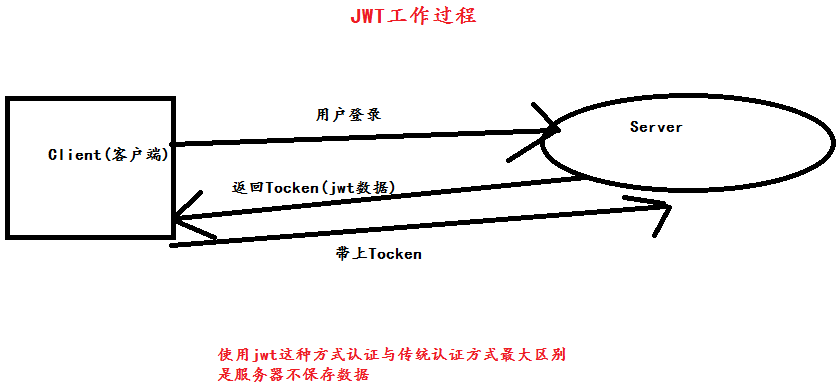
1. 什么是JWT：

json web token: JWT是一种紧凑的、自包含的，用于在多方传递JSON对象的技术，传递的数据可以使用数字签名增加其安全行，JTW一般用于处理用户身份验证[前后端分离/微信小程序/app开发]或数据信息交换。



1. JWT数据结构：

1、JWT的数据结构是： A.B.C 由字符点 ’.’ 来分隔的三部分数据。

A ---header 头信息

B ---payload 有效载荷

C ---signature 签名

1)header:

数据结构： ｛“alg”:”加密算法名称”,”typ”:”JWT”｝

alg是加密算法定义，如： HMAC SHA256 RSA

typ是token类型，这里固定为JWT

2)payload

在payload中一般用于记录实体(通常为用户信息)或其他数据

3)signature

签名信息，这个由开发者提供的信息，是服务器验证验证传递的数据是否有效的安全标准。在生成JWT最终数据之前，先使用header中定义的加密算法，将header和payload进行加密，并使用点进行连接，如：加密后的header.加密后的payload，再使用相同的加密算法对加密后的数据和签名信息进行加密，等到最终结果。

//密钥

let { keys } = require('../model/config');

class Auth {

    static base64(str) { //base64加密方法

        if (typeof str !== 'string') {

            str = JSON.stringify(str);

        }

        return Buffer.from(str).toString("base64");

    }

    static base64decode(str = "") { // base64解密

        return Buffer.from(str, "base64").toString('utf-8');

    }

    static   getToken(uid) {

        let header = {

            "alg": "Base64",

            "typ": "JWT"

        };

        let payload = {

            'curt':new Date().getTime(),

            uid,

            "lastTime": 60 \* 1000 \* 60, // 过期时间60分钟, 对应的毫秒数,

        }

        //分别生成头部及载体的base64编码

        let headerBase64 = this.base64(header);

        let payloadBase64 = this.base64(payload);

        // console.log(headerBase64, payloadBase64);

        //生成签名：

        //this.base64(headerBase64.payloadBase64.密钥)

        let jwt = this.base64(headerBase64 + '.' + payloadBase64 + '.' + keys);

        // console.log(jwt);

        return headerBase64+’.’+payloadBase64 +’.’+jwt;

}

    static async checkTokenAndGetUid  (req) { // 检验jwt的token是否过期， 和是否合法

        let arr = req.headers.authorization.split(".");

        let jwtObj;

        if (arr.length === 3 && this.base64decode(arr[1]) !== "") {

            jwtObj = JSON.parse(this.base64decode(arr[1]));

            req.uid = jwtObj['uid'];

            if (new Date().getTime() - jwtObj[''curt''] < jwtObj['lastTime']) { // 解密拿出过期时间, 判断是否过期

                return null; // 代表正确, 返回空的错误信息对象

            } else {

                return "登录状态已过期, 请重新登录";

            }

        } else {

            return "jwt格式非法, 请确认请求头Authorization";

        }

    }

}

module.exports = Auth;

1. 客户端可以将JWT数据保存在localStorage中:

localStorage['jwt'] = resultObj['jwt'];

localStorage['nickname'] = resultObj['username'];

1. 根据商品id获取商品相关数据接口：

1、根据商品id获取商品相关数据接口地址：

get [http://localhost:3000/getgoodsinfo](http://localhost:3000/renqigoods)

2、参数：

gid：商品id

3、用到的表：

goods\_list: 商品表

category\_first： 商品一级类别表

category\_second：商品二级类别表

category\_thired： 商品三级类别表

goods\_image： 商品图片表

goods\_style: 商品规格表

goods\_eval: 商品评论表

member: 用户表

4、返回值(json)：

1. 加入购物车接口：

1、加入购物车接口地址：

get [http://localhost:3000/shopadd](http://localhost:3000/renqigoods)

2、参数：

goods\_id： 商品id

goods\_num: 商品数量

goods\_style：商品规格

3、用到的表：

goods\_list: 商品表

shopcar： 购物车表

4、返回值(json)：

1. 查看购物车接口：

1、查看购物车接口地址：

get [http://localhost:3000/showshop](http://localhost:3000/renqigoods)

2、参数：

goods\_id： 商品id

goods\_num: 商品数量

goods\_style：商品规格

3、用到的表：

goods\_list: 商品表

shopcar： 购物车表

4、返回值(json)：

1. 根据用户输入的关键词查询商品信息接口：

1、根据用户输入的关键词查询商品信息接口地址：

get [http://localhost:3000/findgoods](http://localhost:3000/renqigoods)

2、参数：

keys： 用户输入的要查询的关键词

starts: 游标位置

limits：每页要显示的条数

orderval：按什么排序

3、用到的表：

search: 搜索的关键表

goods\_list： 商品表

goods\_eval: 商品评论表

4、返回值(json)：

1. 搜索的热门关键接口：

1、搜索的热门关键接口地址：

get [http://localhost:3000/hotkeyword](http://localhost:3000/renqigoods)

2、参数：无

3、用到的表：

search: 搜索的关键表

4、返回值(json)：