

# 硅谷课堂第十二天-公众号消息和微信授权登录

## 硅谷课堂第十二天-公众号消息和微信授权登录

### 一、公众号普通消息

#### ▪1、实现目标

#### 2、消息接入

- 2.1、公众号服务器配置
- 2.2、验证来自微信服务器消息
- 2.3、消息接收

#### 3、配置内网穿透(ngrok)

- 3.1、注册用户
- 3.2、实名认证
- 3.3、开通隧道
- 3.4、启动隧道
- 3.5、测试

#### 4、消息业务实现

- 4.1、service\_vod 模块创建接口
- 4.2、创建模块定义接口
- 4.3、service\_wechat 引入依赖
- 4.4、service\_wechat 模块实现方法
- 4.5、更改 MessageController 方法

#### ▪5、测试公众号消息

### 二、公众号模板消息

#### ▪1、实现目标

#### ▪2、模板消息实现

#### ▪3、申请模板消息

#### ▪4、添加模板消息

#### 5、公众号测试号申请模板消息

##### ▪5.1、新增测试模板

##### ▪5.2、填写信息

#### 6、模板消息接口封装

- 6.1、MessageController
- 6.2、service 接口
- 6.3、service 接口实现
- 6.4、通过 swagger 测试效果

### 三、微信授权登录

- 1、需求描述

#### 2、授权登录

- 2.1、配置授权回调域名
- 2.2、部署公众号前端页面
- 2.3、前端处理

#### 3、授权登录接口

- 3.1、引入微信工具包
- 3.2、添加配置
- 3.3、添加工具类
- 3.4、controller 类
- 3.5、编写 UserInfoService

#### 3.6、使用 token

- 3.6.1、JWT 介绍
- 3.6.2、JWT 的原理
- 3.6.3、整合 JWT

## 一、公众号普通消息

### 1、实现目标

- 1、“硅谷课堂”公众号实现根据关键字搜索相关课程，如：输入“java”，可返回 java 相关的一个课程
- 2、“硅谷课堂”公众号点击菜单“关于我们”，返回关于我们的介绍
- 3、关注或取消关注等

## 2、消息接入

参考文档：[https://developers.weixin.qq.com/doc/offiaccount/Basic\\_Information/Access\\_Overview.html](https://developers.weixin.qq.com/doc/offiaccount/Basic_Information/Access_Overview.html)

接入微信公众平台开发，开发者需要按照如下步骤完成：

- 1、填写服务器配置
- 2、验证服务器地址的有效性
- 3、依据接口文档实现业务逻辑

### 2.1、公众号服务器配置

在测试管理 -> 接口配置信息，点击“修改”按钮，填写服务器地址（URL）和 Token，其中 URL 是开发者用来接收微信消息和事件的接口 URL。Token 可由开发者可以任意填写，用作生成签名（该 Token 会和接口 URL 中包含的 Token 进行比对，从而验证安全性）

说明：本地测试，url 改为内网穿透地址

# 测试号管理

测试号信息

applD

wx09f201e9013e81d8

appsecret

6c999765c12c51850d28055e8b6e2eda

接口配置信息

请填写接口配置信息，此信息需要你有自己的服务器资源，填写的URL需要正确响应微信发送的Token验证，请阅读[消息接口路径，通过内网穿透测试](#)

URL

http://ggkt.vipgz1.91tunnel.com/api/wechaty

Token

atguigu

提交

## 2.2、验证来自微信服务器消息

### (1) 概述

开发者提交信息后，微信服务器将发送 GET 请求到填写的服务器地址 URL 上，GET 请求携带参数如下表所示：

参数	描述
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。
timestamp	时间戳
nonce	随机数
echostr	随机字符串

开发者通过检验 signature 对请求进行校验（下面有校验方式）。若确认此次 GET 请求来自微

信服务器，请原样返回 echostr 参数内容，则接入生效，成为开发者成功，否则接入失败。加密/校验流程如下：

- 1、将 token、timestamp、nonce 三个参数进行字典序排序
- 2、将三个参数字符串拼接成一个字符串进行 sha1 加密
- 3、开发者获得加密后的字符串可与 signature 对比，标识该请求来源于微信

## **(2) 代码实现**

### **创建 MessageController**

```

@RestController
@RequestMapping("/api/wechat/message")
public class MessageController {

    private static final String token = "ggkt";

    /**
     * 服务器有效性验证
     * @param request
     * @return
     */
    @GetMapping
    public String verifyToken(HttpServletRequest request) {
        String signature = request.getParameter("signature");
        String timestamp = request.getParameter("timestamp");
        String nonce = request.getParameter("nonce");
        String echostr = request.getParameter("echostr");
        log.info("signature: {} nonce: {} echostr: {} timestamp: {}", signature, nonce, echostr, timestamp);
        if (this.checkSignature(signature, timestamp, nonce)) {
            log.info("token ok");
            return echostr;
        }
        return echostr;
    }

    private boolean checkSignature(String signature, String timestamp, String nonce) {
        String[] str = new String[]{token, timestamp, nonce};
        //排序
        Arrays.sort(str);
        //拼接字符串
        StringBuffer buffer = new StringBuffer();
        for (int i = 0; i < str.length; i++) {
            buffer.append(str[i]);
        }
        //进行sha1加密
        String temp = SHA1.encode(buffer.toString());
        //与微信提供的signature进行比对
    }
}

```

```
        return signature.equals(temp);
    }
}
```

完成之后，我们的校验接口就算是开发完成了。接下来就可以开发消息接收接口了。

## 2.3、消息接收

接下来我们来开发消息接收接口，消息接收接口和上面的服务器校验接口地址是一样的，都是我们一开始在公众号后台配置的地址。只不过消息接收接口是一个 POST 请求。

在公众号后台配置的时候，消息加解密方式选择了明文模式，这样在后台收到的消息直接就可以处理了。微信服务器给我发来的普通文本消息格式如下：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[text]]></MsgType>
  <Content><![CDATA[this is a test]]></Content>
  <MsgId>1234567890123456</MsgId>
</xml>
```

参数	描述
ToUserName	开发者微信号
FromUserName	发送方帐号（一个 OpenID）
CreateTime	消息创建时间（整型）
MsgType	消息类型，文本为 text
Content	文本消息内容
MsgId	消息 id，64 位整型

看到这里，大家心里大概就有数了，当我们收到微信服务器发来的消息之后，我们就进行 XML 解析，提取出来我们需要的信息，去做相关的查询操作，再将查到的结果返回给微信服务器。

这里我们先来个简单的，我们将收到的消息解析并打印出来：

```
/**
 * 接收微信服务器发送来的消息
 * @param request
 * @return
 * @throws Exception
 */
@PostMapping
public String receiveMessage(HttpServletRequest request) throws Exception {

    WxMpXmlMessage wxMpXmlMessage = WxMpXmlMessage.fromXml(request.getInputStream());
    System.out.println(JSONObject.toJSONString(wxMpXmlMessage));
    return "success";
}

private Map<String, String> parseXml(HttpServletRequest request) throws Exception {
    Map<String, String> map = new HashMap<String, String>();
    InputStream inputStream = request.getInputStream();
    SAXReader reader = new SAXReader();
    Document document = reader.read(inputStream);
    Element root = document.getRootElement();
    List<Element> elementList = root.elements();
    for (Element e : elementList) {
        map.put(e.getName(), e.getText());
    }
    inputStream.close();
    inputStream = null;
    return map;
}
```

### 3、配置内网穿透(ngrok)

#### 3.1、注册用户

网址：<https://ngrok.cc/login/register>



欢迎注册 Sunny-Ngrok

创建一个新账户


请输入电子邮箱

请输入用户名

请输入密码

请再次输入密码

验证码



注册

已经有账户了? [点此登录](#)

### 3.2、实名认证

(1) 注册成功之后，登录系统，进行实名认证，认证费 2 元，认证通过后才能开通隧道

¥ 主页

📖 教程 (一定要看)

👤 我的信息

¥ 订单管理

☁ 隧道管理

隧道管理

开通隧道

¥ 实名认证

实名认证

实名认证通过  
您已完成实名认证。

重要提示

认证费用

我们采用第三方提供的真人认证接口来验证您提交的信息的真实性。该接口为付费接口，

### 3.3、开通隧道

(1) 选择隧道管理 -> 开通隧道

最后一个是免费服务器，建议选择付费服务器，10 元/月，因为免费服务器使用人数很多，经常掉线

¥ 主页

📖 教程 (一定要看)

👤 我的信息

¥ 订单管理

☁ 隧道管理

隧道管理

开通隧道

¥ 实名认证

广州Ngrok服务器，自定义域名需要腾讯云备案  
如果在阿里云备案的域名，请尝试使用cdn过来

立即购买

VIP年费服务器，仅供年费会员使用  
如果在阿里云备案的域名，请尝试使用cdn过来

立即购买

香港200M Ngrok一号服务器  
原价¥10.00 折扣价¥10

地区：香港

带宽：200M

限速：4M (理论最高速度 512K)

香港大带宽VIP服务器，已经取消流量收费  
如果在阿里云备案的域名，请尝试使用cdn过来

立即购买

美国Ngrok免费服务器  
原价¥0.00 折扣价¥0

地区：香港

带宽：10 M

限速：128k

免费服务器人数比较多，速度较慢，容易掉线  
如果在阿里云备案的域名，请尝试使用cdn过来

立即购买

(2) 点击立即购买 -> 输入相关信息

隧道协议: ☒ http ☐ https ☐ tcp 1、隧道协议选择http

隧道名称:  2、隧道名称 随便填写

前置域名:  3、输入前置域名, 随便填写  
①购买后无法修改, 自定义域名请在开通之后编辑

本地端口:  4、输入本地服务器地址  
①本地映射端口, 如需修改其他端口, 则实际端口, 例如: 127.0.0.1:8000

http验证用户名:   
①进行http映射的时候如需要授权访问请输入账号

http验证密码:   
①进行http映射的时候如需要授权访问请输入密码

价格:  5、如果是收费会显示价格

6、选择确定添加

### (3) 开通成功后, 查看开通的隧道

这里开通了两个隧道, 一个用于后端接口调用, 一个用于公众号前端调用

¥ 主页

📖 教程 (一定要看)

👤 我的信息

¥ 订单管理

☁ 隧道管理

隧道管理

开通隧道

隧道管理

注意: 未付款订单将会在一个小时自动取消

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
> 153709342272	ggkt2	http	127.0.0.1:8080	Ngrok (客户端下载)	2022-04-01 15:37:09 续费	http://ggkt2.vipgz1.91tunnel.com
> 091342342272	ggkt1	http	127.0.0.1:8333	Ngrok (客户端下载)	2022-04-01 09:13:42 续费	http://ggkt.vipgz1.91tunnel.com

## 3.4、启动隧道

### (1) 下载客户端工具

[首页](#)[客户端下载](#)[官方QQ群](#)[使用教程\(一定要看我\)](#)[站长教程](#)

## Ngrok客户端

支持Linux、Windows、路由器等终端

### (2) 选择 windows 版本

Linux	Linux ARM版本
Mac OSX	Mac OSX 64Bit版本
Windows	Windows 32Bit版本
Windows	Win 64Bit版本

### (3) 解压，找到 bat 文件，双击启动

windows_amd64 > windows_amd64			▼	🔄
名称	修改日期	类型		
.DS_Store	2021/11/30 17:59	DS_Store		
sunny.exe	2021/11/30 17:58	应用程序		
Sunny-Ngrok启动工具.bat	2016/7/23 6:34	Window		

### (4) 输入隧道 id，多个使用逗号隔开，最后回车就可以启动

```
=====
Sunny-Ngrok客户端启动工具
作者: Sunny QQ: 327388905
官方QQ群: 532387951 (一号群已满) 276155731 (二号群)
官网: www.ngrok.cc
作者博客: www.sunnyos.com
=====

输入需要启动的客户端id, 多个客户端id请使用英文逗号 (,) 隔开: 153709342272, 091342342272
```

```
Sunny-Ngrok www.ngrok.cc

Tunnel Status      online
Version            2.1.1/2.1.1
Forwarding          http://ggkt.vipgz1.91tunnel.com -> 127.0.0.1:8333
Forwarding          http://ggkt2.vipgz1.91tunnel.com -> 127.0.0.1:8080
Web Interface       127.0.0.1:4040
# Conn              0
Avg Conn Time       0.00ms
```

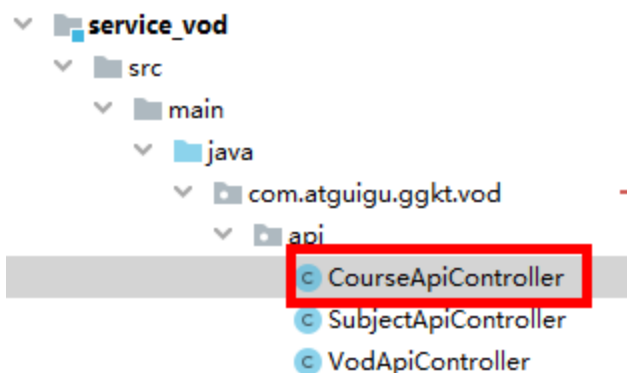
### 3.5、测试

启动服务，在硅谷课堂公众号发送文本消息测试效果。

## 4、消息业务实现

### 4.1、service\_vod 模块创建接口

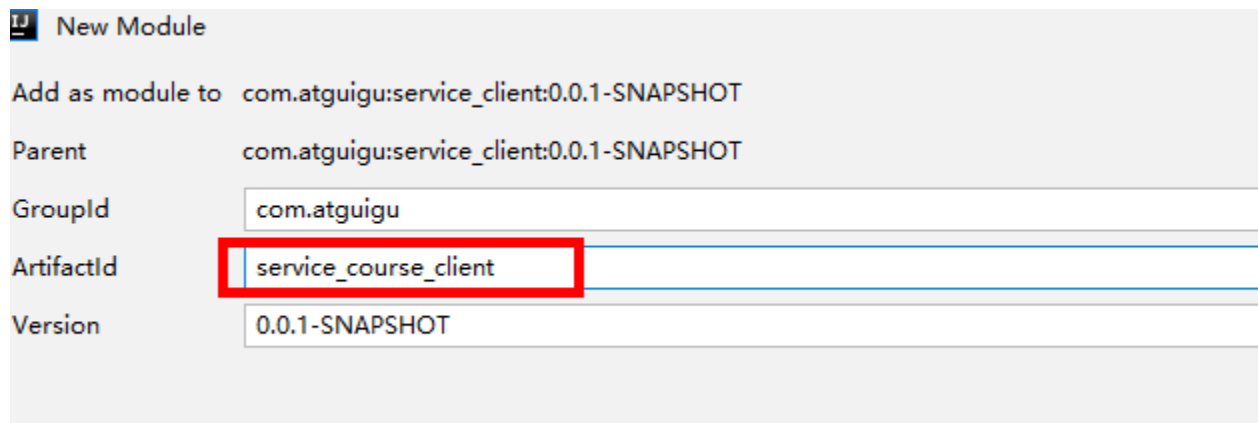
- (1) 创建 CourseApiController 方法，根据课程关键字查询课程信息



```
@ApiOperation("根据关键字查询课程")
@GetMapping("inner/findByKeyword/{keyword}")
public List<Course> findByKeyword(
    @ApiParam(value = "关键字", required = true)
    @PathVariable String keyword){
    QueryWrapper<Course> queryWrapper = new QueryWrapper();
    queryWrapper.like("title", keyword);
    List<Course> list = courseService.list(queryWrapper);
    return list;
}
```

## 4.2、创建模块定义接口

### (1) service\_client 下创建子模块 service\_course\_client



### (2) 定义根据关键字查询课程接口

```
@FeignClient(value = "service-vod")
public interface CourseFeignClient {

    @ApiOperation("根据关键字查询课程")
    @GetMapping("/api/vod/course/inner/findByKeyword/{keyword}")
    List<Course> findByKeyword(@PathVariable String keyword);
}
```

### 4.3、service\_wechat 引入依赖

```
<dependency>
    <groupId>com.atguigu</groupId>
    <artifactId>service_course_client</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</dependency>
```

### 4.4、service\_wechat 模块实现方法

#### (1) MessageService

```
public interface MessageService {
    //接收消息
    String receiveMessage(Map<String, String> param);
}
```

#### (2) MessageServiceImpl

```
@Service
public class MessageServiceImpl implements MessageService {

    @Autowired
    private CourseFeignClient courseFeignClient;

    @Autowired
    private WxMpService wxMpService;

    //接收消息
    @Override
    public String receiveMessage(Map<String, String> param) {
        String content = "";
        try {
            String msgType = param.get("MsgType");
            switch(msgType){
                case "text" :
                    content = this.search(param);
                    break;
                case "event" :
                    String event = param.get("Event");
                    String eventKey = param.get("EventKey");
                    if("subscribe".equals(event)) { //关注公众号
                        content = this.subscribe(param);
                    } else if("unsubscribe".equals(event)) { //取消关注公众号
                        content = this.unsubscribe(param);
                    } else if("CLICK".equals(event) && "aboutUs".equals(eventKey)){
                        content = this.aboutUs(param);
                    } else {
                        content = "success";
                    }
                    break;
                default:
                    content = "success";
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```



```

        content = this.text(param, "请重新输入关键字，没有匹配到相关视频课程").toString();
    }
    return content;
}

/**
 * 关于我们
 * @param param
 * @return
 */
private String aboutUs(Map<String, String> param) {
    return this.text(param, "硅谷课堂现开设Java、HTML5前端+全栈、大数据、全链路UI/UE设计、人工智能");
}

/**
 * 处理关注事件
 * @param param
 * @return
 */
private String subscribe(Map<String, String> param) {
    //处理业务
    return this.text(param, "感谢你关注“硅谷课堂”，可以根据关键字搜索您想看的视频教程，如：JAVA基础");
}

/**
 * 处理取消关注事件
 * @param param
 * @return
 */
private String unsubscribe(Map<String, String> param) {
    //处理业务
    return "success";
}

/**
 * 处理关键字搜索事件
 * 图文消息个数；当用户发送文本、图片、语音、视频、图文、地理位置这六种消息时，开发者只能回复1条图文消息。

```

```

    * @param param
    * @return
    */
    private String search(Map<String, String> param) {
        String fromusername = param.get("FromUserName");
        String tousername = param.get("ToUserName");
        String content = param.get("Content");
        //单位为秒，不是毫秒
        Long createTime = new Date().getTime() / 1000;
        StringBuffer text = new StringBuffer();
        List<Course> courseList = courseFeignClient.findByKeyword(content);
        if(CollectionUtils.isEmpty(courseList)) {
            text = this.text(param, "请重新输入关键字，没有匹配到相关视频课程");
        } else {
            //一次只能返回一个
            Random random = new Random();
            int num = random.nextInt(courseList.size());
            Course course = courseList.get(num);
            StringBuffer articles = new StringBuffer();
            articles.append("<item>");
            articles.append("<Title><![CDATA["+course.getTitle()+"]]></Title>");
            articles.append("<Description><![CDATA["+course.getTitle()+"]]></Description>");
            articles.append("<PicUrl><![CDATA["+course.getCover()+"]]></PicUrl>");
            articles.append("<Url><![CDATA[http://glkt.atguigu.cn/#/liveInfo/"+course.getId()+"]]");
            articles.append("</item>");

            text.append("<xml>");
            text.append("<ToUserName><![CDATA["+fromusername+"]></ToUserName>");
            text.append("<FromUserName><![CDATA["+tousername+"]></FromUserName>");
            text.append("<CreateTime><![CDATA["+createTime+"]></CreateTime>");
            text.append("<MsgType><![CDATA[news]]></MsgType>");
            text.append("<ArticleCount><![CDATA[1]]></ArticleCount>");
            text.append("<Articles>");
            text.append(articles);
            text.append("</Articles>");
            text.append("</xml>");
        }
    }

```

```

        return text.toString();
    }

    /**
     * 回复文本
     * @param param
     * @param content
     * @return
     */
    private StringBuffer text(Map<String, String> param, String content) {
        String fromusername = param.get("FromUserName");
        String tousername = param.get("ToUserName");
        //单位为秒，不是毫秒
        Long createTime = new Date().getTime() / 1000;
        StringBuffer text = new StringBuffer();
        text.append("<xml>");
        text.append("<ToUserName><![CDATA["+fromusername+"]]></ToUserName>");
        text.append("<FromUserName><![CDATA["+tousername+"]]></FromUserName>");
        text.append("<CreateTime><![CDATA["+createTime+"]]></CreateTime>");
        text.append("<MsgType><![CDATA[text]]></MsgType>");
        text.append("<Content><![CDATA["+content+"]]></Content>");
        text.append("</xml>");
        return text;
    }
}

```

#### 4.5、更改 MessageController 方法

```
/**
 * 接收微信服务器发送来的消息
 * @param request
 * @return
 * @throws Exception
 */
@PostMapping
public String receiveMessage(HttpServletRequest request) throws Exception {
    Map<String, String> param = this.parseXml(request);
    return messageService.receiveMessage(param);
}
```

### 5、测试公众号消息

(1) 点击个人 -> 关于我们，返回关于我们的介绍



(2) 在公众号输入关键字，返回搜索的课程信息



## 二、公众号模板消息

### 1、实现目标

购买课程支付成功微信推送消息

### 2、模板消息实现

接口文档：[https://developers.weixin.qq.com/doc/offiaccount/Message\\_Management/Template\\_Message\\_Interface.html](https://developers.weixin.qq.com/doc/offiaccount/Message_Management/Template_Message_Interface.html)

### 3、申请模板消息

首先我们需要知道，模板消息是需要申请的。

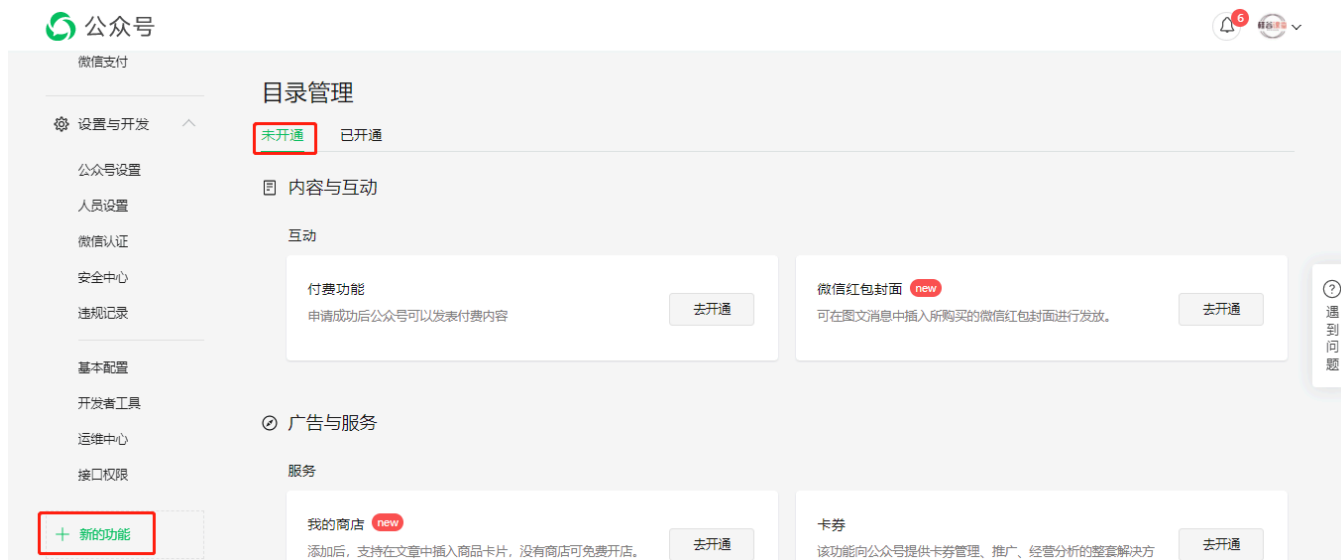
但是我们在申请时还是有一些东西要注意，这个在官方的文档有非常详细的说明。

[https://developers.weixin.qq.com/doc/offiaccount/Message\\_Management/Template\\_Message\\_Operation\\_Specifications.html](https://developers.weixin.qq.com/doc/offiaccount/Message_Management/Template_Message_Operation_Specifications.html)



这个大家好好看看。选择行业的时候要谨慎些，因为这个一个月只可以修改一次。

下面看看在哪里申请，硅谷课堂已经申请过，忽略



申请之后就耐心等待，审核通过之后就会出现“广告与服务”模板消息的菜单。



## 4、添加模板消息

审核通过之后，我们就可以添加模板消息，进行开发了。

我们点击模板消息进入后，直接在模板库中选择你需要的消息模板添加就可以了，添加之后就会在我的模板中。会有一个模板 id，这个模板 id 在我们发送消息的时候会用到。

模板消息如下：

### 模板消息

[我的模板](#)[模板库](#)

[我的模板](#) > [模板详情](#)

亲爱的用户：您有一笔订单支付成功。  
订单编号：0410311909194996  
商品名称：哺乳各种问题  
支付时间：2019年9月19日 16  
支付金额：999  
如需服务请尽快预约，避免老师档期无法安排。电话4000282887。

详情 >

内容示例

模板ID	86wjNWYPF8hy3YFppkqlhbyw7jwmnh_Nvz_Dw8kURk4 开发者调用模板消息接口时需提供模板ID
标题	订单支付成功通知
行业	教育 - 培训
详细内容	{{first.DATA}} 订单编号：{{keyword1.DATA}} 商品名称：{{keyword2.DATA}} 支付时间：{{keyword3.DATA}} 支付金额：{{keyword4.DATA}} {{remark.DATA}}

在发送时，需要将内容中的参数（{{.DATA}}内为参数）赋值替换为需要的信息

我们需要模板消息：

1、订单支付成功通知。

模板库中没有的模板，可以自定义模板，审核通过后可以使用。

## 5、公众号测试号申请模板消息

### 5.1、新增测试模板

模板消息接口				
新增测试模板 最多10个，接受模板消息需要关注测试号				
序号	模板ID(用于接口调用)	模板标题	模板内容	操作
1	V-x2o4oTIW4rXwGzy M-YprNBQV9XmKxwp k_rQpXeGCE	订单成功	{{first.DATA}} 订单编号: {{keyword1.DATA}} 商品名称: {{keyword2.DATA}} 支付时间: {{keyword3.DATA}} 支付金额: {{keyword4.DATA}} {{remark.DATA}}	删除

### 5.2、填写信息

#### (1) 下载示例参考

下载地址：[https://developers.weixin.qq.com/doc/offiaccount/Message\\_Management/Template\\_Message\\_Operation\\_Specifications.html](https://developers.weixin.qq.com/doc/offiaccount/Message_Management/Template_Message_Operation_Specifications.html)

基础消息能力

接收普通消息

接收事件推送

被动回复用户消息

模板消息接口

公众号一次性订阅消息

模板消息运营规范

消息加解密说明

群发接口和原创校验

接口调用频次限制说明

获取公众号的自动回复规则

强调到期提醒类要注意频率和使用场景；待办任务提醒类的内

2.3一般延时性通知

此类模板消息举例如下：审核结果类通知、退款结果类通知、通知、反馈类通知等一些具有延时性的对用户很重要的消息

附目前允许发的模板示例下载：点击下载

二、目前不允许发的模板消息

1、模板内容与服务场景（含标题、关键词）不一致的模板

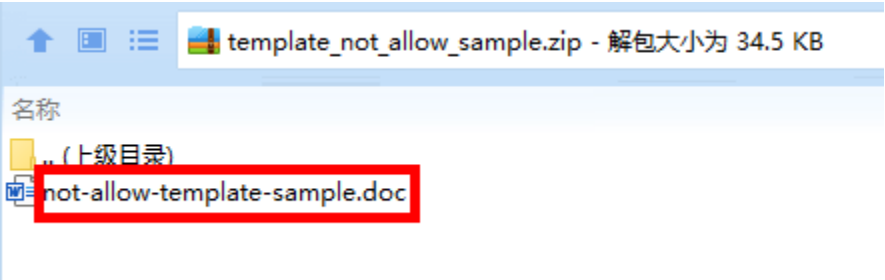
此类模板消息的内容与服务场景不一致，属于滥用模板消息，

2、涉嫌广告营销类消息模板

此类模板消息具有广告营销性质，有诱导用户消费的意图，一

理。此类模板消息举例如下：

消费优惠类通知、购物返利类通知、商品降价类通知、商品





## (2) 填写模板标题和模板内容

请注意：

- 1、测试模板的模板ID仅用于测试，不能用来给正式帐号发送模板消息
- 2、为方便测试，测试模板可任意指定内容，但实际上正式帐号的模板消息，只能从模板库中获得
- 3、需为正式帐号申请新增符合要求的模板，需使用正式号登录公众平台，按指引申请
- 4、模板内容可设置参数(模板标题不可)，供接口调用时使用，参数需以{{开头，以.DATA}}结尾

模板标题

订单支付成功

模板内容

{{first.DATA}}  
订单编号: {{keyword1.DATA}}  
商品名称: {{keyword2.DATA}}  
支付时间: {{keyword3.DATA}}  
支付金额: {{keyword4.DATA}}  
.....

提交 取消

## 6、模板消息接口封装

### 6.1、MessageController

添加方法

```
@GetMapping("/pushPayMessage")
public Result pushPayMessage() throws WxErrorException {
    messageService.pushPayMessage(1L);
    return Result.ok();
}
```

6.2、service 接口

MessageService

```
void pushPayMessage(Long orderId);
```

6.3、service 接口实现

- (1) MessageServiceImpl 类
- (2) openid 值

测试号二维码



请用微信扫码关注测试公众号

用户列表 (最多100个)

序号	昵称	微信号
1	大爱	oepf36SawvvS8Rdqva-Cy4flFFtg

(3) 模板 id 值

模板消息接口

新增测试模板 最多10个，接受模板消息需要关注测试号

序号	模板ID(用于接口调用)	模板标题	模板内容
1	V-x2o4oTIW4rXwGzy M-YprNBQV9XmKxwp k_rQpXeGCE	订单成功	{{first.DATA}} 订单编号: {{keyword1.DATA}} 商品名称: {{keyword2.DA1 3.DATA}} 支付金额: {{keyword4.DATA}} {{remark.D.

```

@Autowired
private WxMpService wxMpService;

//TODO 暂时写成固定值测试，后续完善
@SneakyThrows
@Override
public void pushPayMessage(long orderId) {
    String openid = "oepf36SawvvS8RdqvA-Cy4f1FFtg";
    WxMpTemplateMessage templateMessage = WxMpTemplateMessage.builder()
        .toUser(openid)//要推送的用户openid
        .templateId("V-x2o4oTIW4rXwGzyM-YprNBQV9XmKxwPk_rQpXeGCE")//模板id
        .url("http://ggkt2.vipgz1.91tunnel.com/#/pay/"+orderId)//点击模板消息要访问的网址
        .build();
    //3,如果是正式版发送消息，这里需要配置你的信息
    templateMessage.addData(new WxMpTemplateData("first", "亲爱的用户：您有一笔订单支付成功。", "#272727"));
    templateMessage.addData(new WxMpTemplateData("keyword1", "1314520", "#272727"));
    templateMessage.addData(new WxMpTemplateData("keyword2", "java基础课程", "#272727"));
    templateMessage.addData(new WxMpTemplateData("keyword3", "2022-01-11", "#272727"));
    templateMessage.addData(new WxMpTemplateData("keyword4", "100", "#272727"));
    templateMessage.addData(new WxMpTemplateData("remark", "感谢你购买课程，如有疑问，随时咨询！", "#272727"));
    String msg = wxMpService.getTemplateMsgService().sendTemplateMsg(templateMessage);
    System.out.println(msg);
}

```

## 6.4、通过 swagger 测试效果

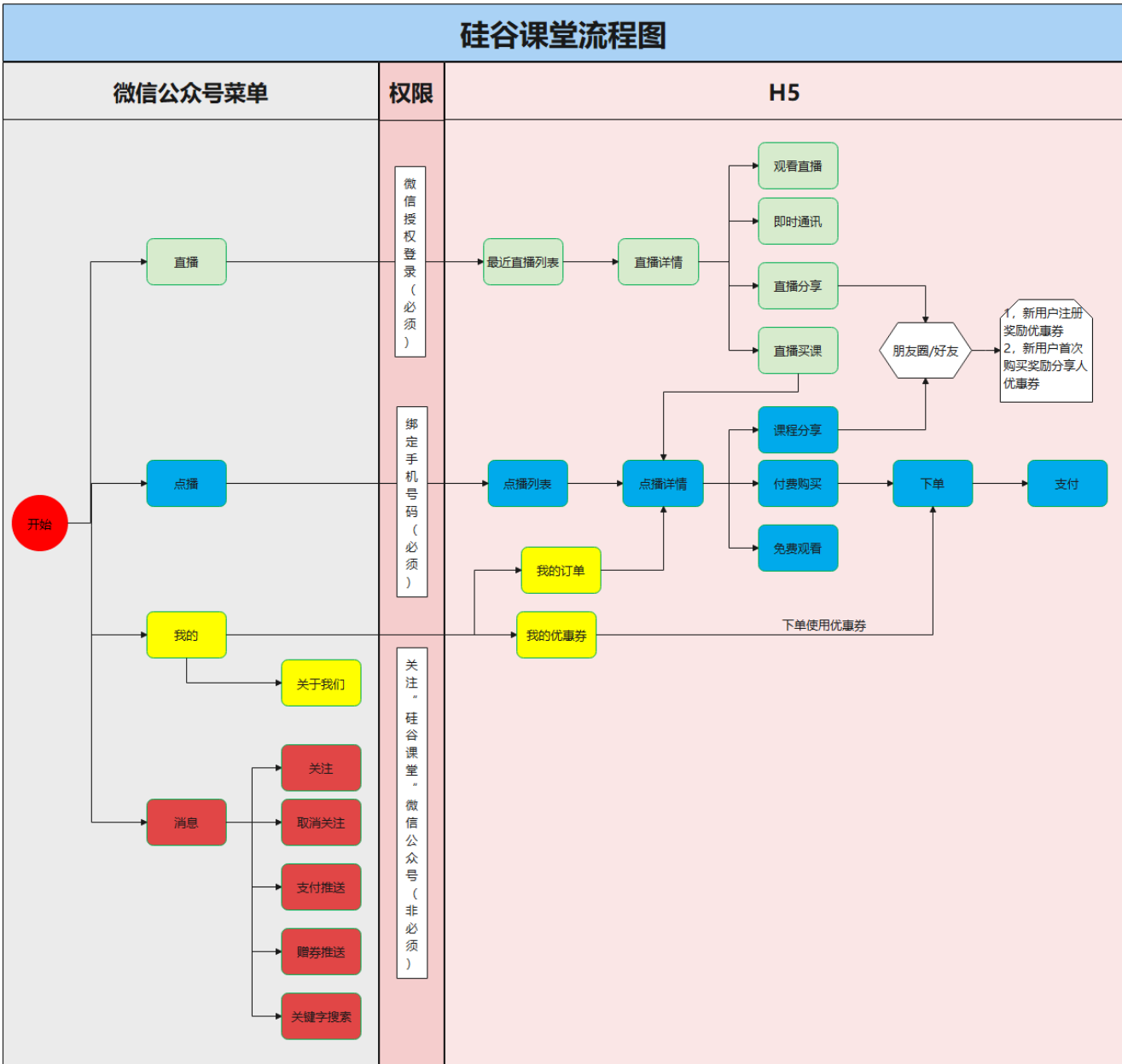
### (1) 在公众号可以看到发送的模板消息



### 三、微信授权登录

#### 1、需求描述

根据流程图通过菜单进入的页面都要授权登录



## 2、授权登录

接口文档：[https://developers.weixin.qq.com/doc/offiaccount/OA\\_Web\\_Apps/Wechat\\_webpage\\_authorization.html](https://developers.weixin.qq.com/doc/offiaccount/OA_Web_Apps/Wechat_webpage_authorization.html)

说明：

- 1、严格按照接口文档实现
- 2、应用授权作用域 scope：scope 为 snsapi\_userinfo

## 2.1、配置授权回调域名

### (1) 在公众号正式号配置

在微信公众号请求用户网页授权之前，开发者需要先到公众平台官网中的“设置与开发 - 接口权限 - 网页服务 - 网页帐号 - 网页授权获取用户基本信息”的配置选项中，修改授权回调域名。请注意，这里填写的是域名（是一个字符串），而不是 URL，因此请勿加 http:// 等协议头。

本地测试配置内网穿透地址



### (2) 在公众号测试号配置

	网页帐号	网页授权获取用户基本信息	无上限	修改
	基础接口	判断当前客户端版本是否支持指定JS接口	无上限	
	分享接口	获取“分享到朋友圈”按钮点击状态及自定义分享内容接口	无上限	
		获取“分享给朋友”按钮点击状态及自定义分享内容接口	无上限	
		获取“分享到QQ”按钮点击状态及自定义分享内容接口	无上限	
		获取“分享到腾讯微博”按钮点击状态及自定义分享内容接口	无上限	

## OAuth2.0网页授权

授权回调页面域名:

ggkt.vipgz1.91tunnel.com

用户在网页授权页同意授权给公众号后，微信会将授权数据传给一个回调页面，回调页面需在此域名下，以确保安全可靠。沙盒号回调地址支持域名和ip，正式公众号回调地址只支持域名。

## 2.2、部署公众号前端页面

(1) 公众号前端页面已经开发完成，直接部署使用即可

Data (D:) > vscode > ggkt > vue-mobile >

名称

- \_\_MACOSX
- node\_modules
- public
- src
- .eslintignore
- .gitignore
- babel.config.js
- MP\_verify\_W6s0xK64jTB5QkC8.txt
- package.json
- package-lock.json
- README.md
- README1.md
- vue.config.js

## (2) 启动公众号页面项目

使用命令：`npm run serve`

```
D:\vscode\ggkt\vue-mobile>npm run serve
```

```
> vue-mobile@0.1.0 serve D:\vscode\ggkt\vue-mobile
> vue-cli-service serve
```

```
INFO Starting development server...
40% building 133/137 modules 4 active D:\vscode\ggkt\vue-mobile\node
Warning: The fs.promises API is experimental
98% after emitting CopyPlugin
```

```
DONE Compiled successfully in 6833ms
```

```
App running at:
```

```
- Local: http://localhost:8080/
- Network: http://192.168.2.134:8080/
```

---

## 2.3、前端处理

### (1) 全局处理授权登录，处理页面：`/src/App.vue`

**说明 1：**访问页面时首先判断是否有 token 信息，如果没有跳转到授权登录接口

**说明 2：**通过 `localStorage` 存储 token 信息

在 HTML5 中，加入了一个 **localStorage** 特性，这个特性主要是用来作为本地存储来使用的，解决了 cookie 存储空间不足的问题(cookie 中每条 cookie 的存储空间很小，只有几 K)，localStorage 中一般浏览器支持的是 5M 大小，这个在不同的浏览器中 localStorage 会有所不同。它只能存储字符串格式的数据，所以最好在每次存储时把数据转换成 json 格式，取出的时候再转换回来。

### (2) 前端代码实现



```

wechatLogin() {
  // 处理微信授权登录
  let token = this.getQueryString('token') || '';
  if(token !== '') {
    window.localStorage.setItem('token', token);
  }

  // 所有页面都必须登录，两次调整登录，这里与接口返回208状态
  token = window.localStorage.getItem('token') || '';
  if (token === '') {
    let url = window.location.href.replace('#', 'guiguketan')
    window.location = 'http://glkt.atguigu.cn/api/user/wechat/authorize?returnUrl=' + url
  }
  console.log('token2: ' + window.localStorage.getItem('token'));
},

```

### 3、授权登录接口

操作模块：service-user

### 3.1、引入微信工具包

```
<dependencies>
  <dependency>
    <groupId>com.github.binarywang</groupId>
    <artifactId>weixin-java-mp</artifactId>
    <version>2.7.0</version>
  </dependency>

  <dependency>
    <groupId>dom4j</groupId>
    <artifactId>dom4j</artifactId>
    <version>1.1</version>
  </dependency>

  <dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
  </dependency>
</dependencies>
```

### 3.2、添加配置

```
# 公众号id和密钥
# 硅谷课堂微信公众平台appId
wechat.mpAppId: wx09f201e9013e81d8
# 硅谷课堂微信公众平台api密钥
wechat.mpAppSecret: 6c999765c12c51850d28055e8b6e2eda

# 授权回调获取用户信息接口地址
wechat.userInfoUrl: http://ggkt.vipgz1.91tunnel.com/api/user/wechat/userInfo
```

### 3.3、添加工具类

```
@Component
public class ConstantPropertiesUtil implements InitializingBean {

    @Value("${wechat.mpAppId}")
    private String appid;

    @Value("${wechat.mpAppSecret}")
    private String appsecret;

    public static String ACCESS_KEY_ID;
    public static String ACCESS_KEY_SECRET;

    @Override
    public void afterPropertiesSet() throws Exception {
        ACCESS_KEY_ID = appid;
        ACCESS_KEY_SECRET = appsecret;
    }
}
```

```
@Component
public class WeChatMpConfig {

    @Autowired
    private ConstantPropertiesUtil constantPropertiesUtil;

    @Bean
    public WxMpService wxMpService(){
        WxMpService wxMpService = new WxMpServiceImpl();
        wxMpService.setWxMpConfigStorage(wxMpConfigStorage());
        return wxMpService;
    }

    @Bean
    public WxMpConfigStorage wxMpConfigStorage(){
        WxMpInMemoryConfigStorage wxMpConfigStorage = new WxMpInMemoryConfigStorage();
        wxMpConfigStorage.setAppId(ConstantPropertiesUtil.ACCESS_KEY_ID);
        wxMpConfigStorage.setSecret(ConstantPropertiesUtil.ACCESS_KEY_SECRET);
        return wxMpConfigStorage;
    }
}
```

### 3.4、controller 类

```

@Controller
@RequestMapping("/api/user/wechat")
public class WechatController {

    @Autowired
    private UserInfoService userInfoService;

    @Autowired
    private WxMpService wxMpService;

    @Value("${wechat.userInfoUrl}")
    private String userInfoUrl;

    @GetMapping("/authorize")
    public String authorize(@RequestParam("returnUrl") String returnUrl, HttpServletRequest request,
        String redirectURL = wxMpService.oauth2buildAuthorizationUrl(userInfoUrl,
            WxConsts.OAUTH2_SCOPE_USER_INFO,
            URLEncoder.encode(returnUrl.replace("guiguketan", "#"))));
    return "redirect:" + redirectURL;
}

@GetMapping("/userInfo")
public String userInfo(@RequestParam("code") String code,
    @RequestParam("state") String returnUrl) throws Exception {
    WxMpOAuth2AccessToken wxMpOAuth2AccessToken = this.wxMpService.oauth2getAccessToken(code);
    String openId = wxMpOAuth2AccessToken.getOpenId();

    System.out.println("【微信网页授权】openId={}"+openId);

    WxMpUser wxMpUser = wxMpService.oauth2getUserInfo(wxMpOAuth2AccessToken, null);
    System.out.println("【微信网页授权】wxMpUser={}"+JSON.toJSONString(wxMpUser));

    UserInfo userInfo = userInfoService.getByOpenid(openId);
    if(null == userInfo) {
        userInfo = new UserInfo();
        userInfo.setOpenId(openId);
        userInfo.setUnionId(wxMpUser.getUnionId());
    }
}

```

```

        userInfo.setNickName(wxMpUser.getNickname());
        userInfo.setAvatar(wxMpUser.getHeadImgUrl());
        userInfo.setSex(wxMpUser.getSexId());
        userInfo.setProvince(wxMpUser.getProvince());

        userInfoService.save(userInfo);
    }
    //生成token
    String token = JwtHelper.createToken(userInfo.getId(), userInfo.getNickName());
    if(returnUrl.indexOf("?") == -1) {
        return "redirect:" + returnUrl + "?token=" + token;
    } else {
        return "redirect:" + returnUrl + "&token=" + token;
    }
}
}

```

### 3.5、编写 UserInfoService

```

@Service
public class UserInfoServiceImpl extends ServiceImpl<UserInfoMapper, UserInfo> implements UserInfoService {

    @Override
    public UserInfo getByOpenid(String openid) {
        QueryWrapper<UserInfo> wrapper = new QueryWrapper<>();
        wrapper.eq("open_id", openid);
        UserInfo userInfo = baseMapper.selectOne(wrapper);
        return userInfo;
    }
}

```

### 3.6、使用 token

#### 通过 token 传递用户信息

#### 3.6.1、JWT 介绍

##### JWT 工具

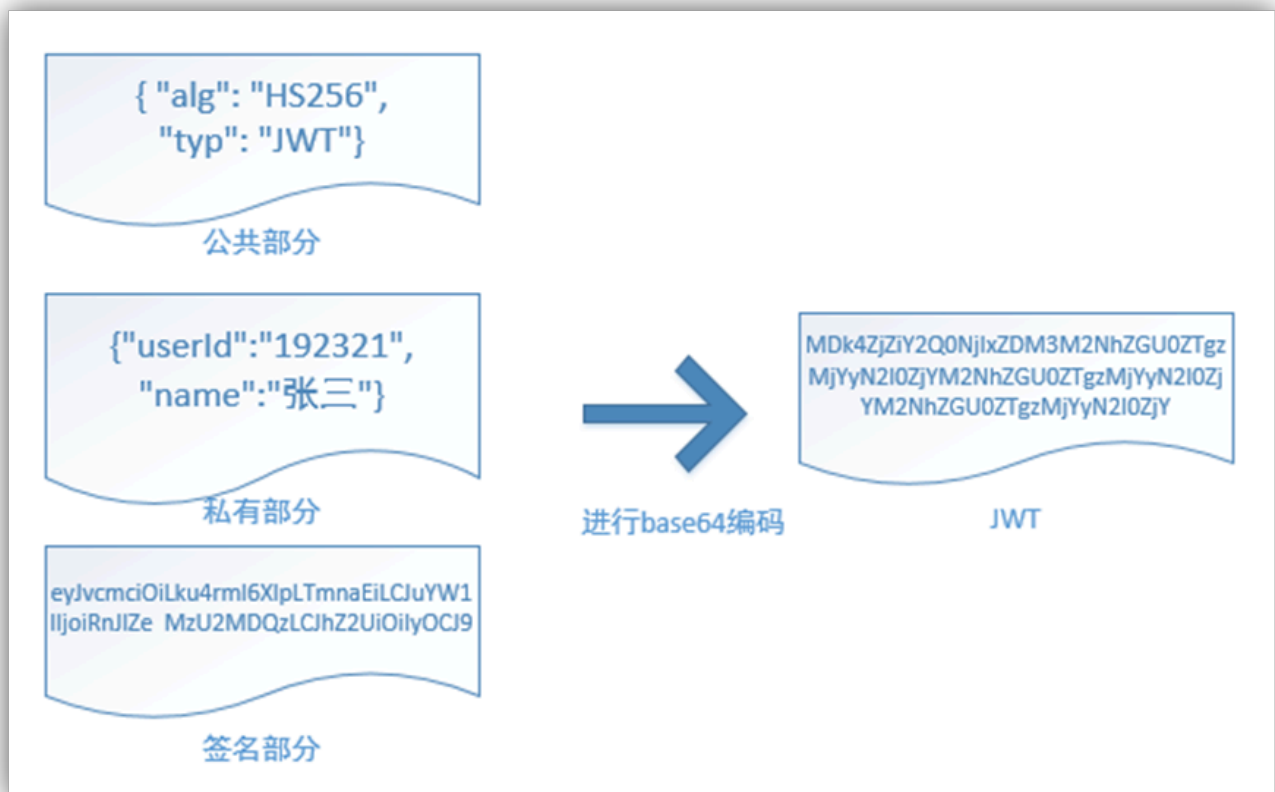
JWT（Json Web Token）是为了在网络应用环境间传递声明而执行的一种基于 JSON 的开放标准。

JWT 的声明一般被用来在身份提供者和服务提供者间传递被认证的用户身份信息，以便于从资源服务器获取资源。比如用在用户登录上

JWT 最重要的作用就是对 token 信息的**防伪**作用。

### 3.6.2、JWT 的原理

一个 JWT 由三个部分**组成**：公共部分、私有部分、**签名部分**。最后由这三者组合进行 base64 编码得到 JWT。



#### (1) 公共部分

主要是该 JWT 的相关配置参数，比如签名的加密算法、格式类型、过期时间等等。

#### (2) 私有部分

用户自定义的内容，根据实际需要真正要封装的信息。



userInfo{用户的 Id, 用户的昵称 nickName}。

### (3) 签名部分

SaltIP: 当前服务器的 IP 地址{linux 中配置代理服务器的 ip}。

主要用户对 JWT 生成字符串的时候, 进行加密{盐值}。

base64 编码, 并不是加密, 只是把明文信息变成了不可见的字符串。但是其实只要用一些工具就可以把 base64 编码解成明文, 所以不要在 JWT 中放入涉及私密的信息。

### 3.6.3、整合 JWT

#### (1) 在 service\_utils 模块添加依赖

```
<dependencies>
  <dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
  </dependency>
  <dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
  </dependency>
  <dependency>
    <groupId>joda-time</groupId>
    <artifactId>joda-time</artifactId>
  </dependency>
</dependencies>
```

#### (2) 添加 JWT 工具类

```

public class JwtHelper {
    //token字符串有效时间
    private static long tokenExpiration = 24*60*60*1000;
    //加密编码密钥
    private static String tokenSignKey = "123456";

    //根据userid 和 username 生成token字符串
    public static String createToken(Long userId, String userName) {
        String token = Jwts.builder()
            //设置token分类
            .setSubject("GGKT-USER")
            //token字符串有效时长
            .setExpiration(new Date(System.currentTimeMillis() + tokenExpiration))
            //私有部分 (用户信息)
            .claim("userId", userId)
            .claim("userName", userName)
            //根据密钥使用加密编码方式进行加密, 对字符串压缩
            .signWith(SignatureAlgorithm.HS512, tokenSignKey)
            .compressWith(CompressionCodecs.GZIP)
            .compact();

        return token;
    }

    //从token字符串获取userid
    public static Long getUserId(String token) {
        if(StringUtils.isEmpty(token)) return null;
        Jws<Claims> claimsJws = Jwts.parser().setSigningKey(tokenSignKey).parseClaimsJws(token);
        Claims claims = claimsJws.getBody();
        Integer userId = (Integer)claims.get("userId");
        return userId.longValue();
    }

    //从token字符串获取getUserName
    public static String getUserName(String token) {
        if(StringUtils.isEmpty(token)) return "";
        Jws<Claims> claimsJws
            = Jwts.parser().setSigningKey(tokenSignKey).parseClaimsJws(token);
    }

```

```
        Claims claims = claimsJws.getBody();  
        return (String)claims.get("userName");  
    }  
  
    public static void main(String[] args) {  
        String token = JwtHelper.createToken(1L, "lucy");  
        System.out.println(token);  
        System.out.println(JwtHelper.getUserId(token));  
        System.out.println(JwtHelper.getUserName(token));  
    }  
}
```