**微信公众号平台项目**

# 什么是微信公众平台

微信公众号主要面向名人、政府、媒体、企业等机构推出的合作推广业务。在这里可以通过微信渠道将品牌推广给上亿的微信用户，减少宣传成本，提高品牌知名度，打造更具影响力的品牌形象。

# 初始微信公众平台

## 官方网址

<https://mp.weixin.qq.com/cgi-bin/loginpage?t=wxm2-login&lang=zh_CN>

### 帐号分类



## 微信公众平台介绍

### 消息管理

### 用户管理

### 素材管理

### 基本设置

#### 服务器配置(未启用)

启用并设置服务器配置后，用户发给公众号的消息以及开发者需要的事件推送，将被微信转发到该URL中

### 外网映射工具

#### Ngrok使用

●windows用户:

1,下载windows版本的客户端，解压到你喜欢的目录  
2,在命令行下进入到path/to/windows\_386/下  
3,执行 ngrok -config=ngrok.cfg -subdomain xxx 80 //(xxx 是你自定义的域名前缀)

4,如果开启成功 你就可以使用 xxx.tunnel.qydev.com 来访问你本机的 127.0.0.1:80 的服务啦  
5,如果你自己有顶级域名，想通过自己的域名来访问本机的项目，那么先将自己的顶级域名解析到123.57.165.240(域名需要已备案哦)，然后执行./ngrok -config=ngrok.cfg -hostname xxx.xxx.xxx 80 //(xxx.xxx.xxx是你自定义的顶级域名)  
6，如果开启成功 你就可以使用你的顶级域名来访问你本机的 127.0.0.1:80 的服务啦  
7,如果失败 就加下交流群 反馈下问题 本屌会看看什么原因....吧

#### Natapp使用

 windows ,点击开始->运行->命令行提示符 后进入 natapp.exe的目录  
    运行   natapp -authtoken= 175396706488ac93

## 公众测试平台

因为是非认证公众号，部分权限不足，建议大家测试中使用公众测试平台。

https://mp.weixin.qq.com/debug/cgi-bin/sandbox?t=sandbox/login

# 微信环境搭建

## 相关依赖

## itmayiedu-shopp-parent 新增

|  |
| --- |
| <dependency>  <groupId>dom4j</groupId>  <artifactId>dom4j</artifactId>  </dependency>    <dependency>  <groupId>org.glassfish.jersey.core</groupId>  <artifactId>jersey-server</artifactId>  </dependency> |

## 配置文件

application.yml

|  |
| --- |
| server:  port: 81  # context-path: /web  eureka:  client:  serviceUrl:  defaultZone: http://localhost:8761/eureka/  spring:  application:  name: weixin |

## 微信事件通知验证

https://mp.weixin.qq.com/wiki?t=resource/res\_main&id=mp1421135319

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：

### 工具类

#### CheckUtil

|  |
| --- |
| public class CheckUtil {  public static final String *tooken* = "itmayiedu"; // 开发者自行定义Tooken  public static boolean checkSignature(String signature, String timestamp, String nonce) {  // 1.定义数组存放tooken，timestamp,nonce  String[] arr = { *tooken*, timestamp, nonce };  // 2.对数组进行排序  Arrays.*sort*(arr);  // 3.生成字符串  StringBuffer sb = new StringBuffer();  for (String s : arr) {  sb.append(s);  }  // 4.sha1加密,网上均有现成代码  String temp = *getSha1*(sb.toString());  // 5.将加密后的字符串，与微信传来的加密签名比较，返回结果  return temp.equals(signature);  }  public static String getSha1(String str) {  if (str == null || str.length() == 0) {  return null;  }  char hexDigits[] = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f' };  try {  MessageDigest mdTemp = MessageDigest.*getInstance*("SHA1");  mdTemp.update(str.getBytes("UTF-8"));  byte[] md = mdTemp.digest();  int j = md.length;  char buf[] = new char[j \* 2];  int k = 0;  for (int i = 0; i < j; i++) {  byte byte0 = md[i];  buf[k++] = hexDigits[byte0 >>> 4 & 0xf];  buf[k++] = hexDigits[byte0 & 0xf];  }  return new String(buf);  } catch (Exception e) {  return null;  }  }  } |

#### XmlUtils

|  |
| --- |
| **public class XmlUtils {**  **/\*\***  **\* 解析微信发来的请求（XML）**  **\***  **\* @param request**  **\* @return Map<String, String>**  **\* @throws Exception**  **\*/**  **@SuppressWarnings("unchecked")**  **public static Map<String, String> parseXml(HttpServletRequest request) throws Exception {**  **// 将解析结果存储在HashMap中**  **Map<String, String> map = new HashMap<String, String>();**  **// 从request中取得输入流**  **InputStream inputStream = request.getInputStream();**  **// 读取输入流**  **SAXReader reader = new SAXReader();**  **Document document = reader.read(inputStream);**  **// 得到xml根元素**  **Element root = document.getRootElement();**  **// 得到根元素的所有子节点**  **List<Element> elementList = root.elements();**  **// 遍历所有子节点**  **for (Element e : elementList)**  **map.put(e.getName(), e.getText());**  **// 释放资源**  **inputStream.close();**  **inputStream = null;**  **return map;**  **}**  **/\*\***  **\* 文本消息对象转换成xml**  **\***  **\* @param textMessage**  **\* 文本消息对象**  **\* @return xml**  **\*/**  **public static String messageToXml(TextMessage textMessage) {**  ***xstream*.alias("xml", textMessage.getClass());**  **return *xstream*.toXML(textMessage);**  **}**  **/\*\***  **\* 扩展xstream使其支持CDATA**  **\*/**  **private static XStream *xstream* = new XStream();**  **}** |

#### JSONObject

|  |
| --- |
| **// 调用第三方智能接口**  **String resultStr = HttpClientUtil**  **.*doGet*("http://api.qingyunke.com/api.php?key=free&appid=0&msg=" + content);**  **JSONObject jsonObject = new JSONObject().*parseObject*(resultStr);**  **Integer integer = jsonObject.getInteger("result");**  **if (integer == null || integer != 0) {**  **textMessage = setTextMessage("亲,系统出错啦!", toUserName, fromUserName);**  **} else {**  **String result = jsonObject.getString("content");**  **textMessage = setTextMessage(result, toUserName, fromUserName);**  **}** |

#### HttpClientUtil

|  |
| --- |
| **public** **class** HttpClientUtil {  **public** **static** String doGet(String url, Map<String, String> param) {  // 创建Httpclient对象  CloseableHttpClient httpclient = HttpClients.*createDefault*();  String resultString = "";  CloseableHttpResponse response = **null**;  **try** {  // 创建uri  URIBuilder builder = **new** URIBuilder(url);  **if** (param != **null**) {  **for** (String key : param.keySet()) {  builder.addParameter(key, param.get(key));  }  }  URI uri = builder.build();  // 创建http GET请求  HttpGet httpGet = **new** HttpGet(uri);  // 执行请求  response = httpclient.execute(httpGet);  // 判断返回状态是否为200  **if** (response.getStatusLine().getStatusCode() == 200) {  resultString = EntityUtils.*toString*(response.getEntity(), "UTF-8");  }  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {  **if** (response != **null**) {  response.close();  }  httpclient.close();  } **catch** (IOException e) {  e.printStackTrace();  }  }  **return** resultString;  }  **public** **static** String doGet(String url) {  **return** *doGet*(url, **null**);  }  **public** **static** String doPost(String url, Map<String, String> param) {  // 创建Httpclient对象  CloseableHttpClient httpClient = HttpClients.*createDefault*();  CloseableHttpResponse response = **null**;  String resultString = "";  **try** {  // 创建Http Post请求  HttpPost httpPost = **new** HttpPost(url);  // 创建参数列表  **if** (param != **null**) {  List<NameValuePair> paramList = **new** ArrayList<>();  **for** (String key : param.keySet()) {  paramList.add(**new** BasicNameValuePair(key, param.get(key)));  }  // 模拟表单  UrlEncodedFormEntity entity = **new** UrlEncodedFormEntity(paramList);  httpPost.setEntity(entity);  }  // 执行http请求  response = httpClient.execute(httpPost);  resultString = EntityUtils.*toString*(response.getEntity(), "utf-8");  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {  response.close();  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  }  **return** resultString;  }  **public** **static** String doPost(String url) {  **return** *doPost*(url, **null**);  }    **public** **static** String doPostJson(String url, String json) {  // 创建Httpclient对象  CloseableHttpClient httpClient = HttpClients.*createDefault*();  CloseableHttpResponse response = **null**;  String resultString = "";  **try** {  // 创建Http Post请求  HttpPost httpPost = **new** HttpPost(url);  // 创建请求内容  StringEntity entity = **new** StringEntity(json, ContentType.***APPLICATION\_JSON***);  httpPost.setEntity(entity);  // 执行http请求  response = httpClient.execute(httpPost);  resultString = EntityUtils.*toString*(response.getEntity(), "utf-8");  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {  response.close();  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  }  **return** resultString;  }  } |

### BaseMessage

|  |
| --- |
| @Getter  @Setter  **public** **class** BaseMessage {  /\*\*  \* 开发者微信  \*/  **private** String ToUserName;  /\*\*  \* 发送方openid  \*/  **private** String FromUserName;  /\*\*  \* 创建时间  \*/  **private** **long** CreateTime;  /\*\*  \* 内容类型  \*/  **private** String MsgType;  // /\*\*  // \* 消息id  // \*/  // private long MsgId ;  } |

### TextMessage

|  |
| --- |
| @Getter  @Setter  **public** **class** TextMessage **extends** BaseMessage {  **private** String Content;  } |

### DispatCherServlet

|  |
| --- |
| @Slf4j  @RestController  **public** **class** DispatCherServlet {  **private** **static** **final** String ***REQEST\_HTTP*** = "http://api.qingyunke.com/api.php?key=free&appid=0&msg=";  @RequestMapping(value = "/dispatCherServlet", method = RequestMethod.***GET***)  **public** String dispatCherServletGet(String signature, String timestamp, String nonce, String echostr) {  // 1.验证是否微信来源  **boolean** checkSignature = CheckUtil.*checkSignature*(signature, timestamp, nonce);  // 2.如果是微信来源 返回 随机数echostr  **if** (!checkSignature) {  **return** **null**;  }  **return** echostr;  }  @RequestMapping(value = "/dispatCherServlet", method = RequestMethod.***POST***)  **public** **void** dispatCherServletPost(HttpServletRequest reqest, HttpServletResponse response) **throws** Exception {  reqest.setCharacterEncoding("UTF-8");  response.setCharacterEncoding("UTF-8");  Map<String, String> mapResult = XmlUtils.*parseXml*(reqest);  **if** (mapResult == **null**) {  **return**;  }  String msgType = mapResult.get("MsgType");  PrintWriter writer = response.getWriter();  **switch** (msgType) {  **case** "text":  // 获取消息内容  String content = mapResult.get("Content");  // 发送消息  String toUserName = mapResult.get("ToUserName");  // 来自消息  String fromUserName = mapResult.get("FromUserName");  // 調用智能机器人接口  String requestResultJson = HttpClientUtil.*doGet*(***REQEST\_HTTP*** + content);  JSONObject jsonObject = **new** JSONObject().*parseObject*(requestResultJson);  String result = jsonObject.getString("result");  String msg = **null**;  **if** (result.equals("0")) {  msg = jsonObject.getString("content");  } **else** {  msg = "我也不知道回答什么！";  }  String resultTestMsg = setTextMess(msg, toUserName, fromUserName);  writer.print(resultTestMsg);  **break**;  **default**:  **break**;  }  writer.close();  }  **public** String setTextMess(String content, String fromUserName, String toUserName) {  TextMessage textMessage = **new** TextMessage();  textMessage.setFromUserName(fromUserName);  textMessage.setToUserName(toUserName);  textMessage.setContent(content);  textMessage.setMsgType("text");  textMessage.setCreateTime(**new** Date().getTime());  String messageToXml = XmlUtils.*messageToXml*(textMessage);  ***log***.info("####setTextMess()###messageToXml:" + messageToXml);  **return** messageToXml;  }  } |

### 青云客智能聊天机器人API

<http://api.qingyunke.com/>

**智能机器人API接口说明**  
支持功能：天气、翻译、藏头诗、笑话、歌词、计算、域名信息/备案/收录查询、IP查询、手机号码归属、人工智能聊天  
接口地址：http://api.qingyunke.com/api.php?key=free&appid=0&msg=关键词  
　　　　　key　固定参数free  
　　　　　appid 设置为0，表示智能识别，可忽略此参数  
　　　　　msg　关键词，请参考下方参数示例，该参数可智能识别，该值请经过 urlencode 处理后再提交  
返回结果：{"result":0,"content":"内容"}  
　　　　　result　状态，0表示正常，其它数字表示错误  
　　　　　content　信息内容

# 使用第三方框架开发微信公众号

## [weixin-java-tools](https://github.com/Wechat-Group/weixin-java-tools)

https://github.com/wechat-group/weixin-java-tools

## 微信消息模板接口

### 概述

模板消息仅用于公众号向用户发送重要的服务通知，只能用于符合其要求的服务场景中，如信用卡刷卡通知，商品购买成功通知等。不支持广告等营销类消息以及其它所有可能对用户造成骚扰的消息。

参考网址: https://mp.weixin.qq.com/wiki?t=resource/res\_main&id=mp1433751277

### 代码

|  |
| --- |
| @RestController  @RequestMapping("/weiXin")  **public** **class** WeiXinCatController {  @Autowired  **private** WxMpService wxService;  @RequestMapping("/sendTemplate")  **public** String createWeiXinCat(@RequestBody WxMpTemplateMessage wxMpTemplateMessage) **throws** WxErrorException {  WxMpTemplateMsgService templateMsgService = wxService.getTemplateMsgService();  **return** templateMsgService.sendTemplateMsg(wxMpTemplateMessage);  }  } |

### 参数

|  |
| --- |
| 请求地址http://127.0.0.1:81/weiXin/sendTemplate  {  "toUser": "okYSmt0cItTc4dZXFn43BbmYW8Rw",  "templateId": "eLozPgRNCPMEgOWGHwNW6cXkfrVqtsyDsyEwVabu0ww",  "url": "http://www.itmayiedu.com",  "data": [  {  "name": "first",  "value": "2017年11月06日 23:32",  "color": "#173177"  },  {  "name": "keyword1",  "value": "30",  "color": "#173177"  },  {  "name": "keyword2",  "value": "201410515111522",  "color": "#173177"  }  ]  } |

## 网页授权

参考文章: https://mp.weixin.qq.com/wiki?t=resource/res\_main&id=mp1421140842

## Http协议转码

## UrlEnCode与UrlDeCode

有些符号在URL中是不能直接传递的，如果要在URL中传递这些特殊符号，那么就要使用他们的编码了。编码的格式为：%加字符的ASCII码，即一个百分号%，后面跟对应字符的ASCII（16进制）码值。例如 空格的编码值是"%20"。  
下表中列出了一些URL特殊符号及编码  
:替换为%3A　  
1. + URL 中+号表示空格 %2B   
2. 空格 URL中的空格可以用+号或者编码 %20   
3. / 分隔目录和子目录 %2F   
4. ? 分隔实际的 URL 和参数 %3F   
5. % 指定特殊字符 %25   
6. # 表示书签 %23   
7. & URL 中指定的参数间的分隔符 %26   
8. = URL 中指定参数的值 %3D

|  |
| --- |
| @RequestMapping("/test")  **public** String test(String name) {  System.***out***.println("name:" + name);  **return** "test" + name;  } |

|  |
| --- |
| **public** **static** **void** main(String[] args) {  String userNameEncode = URLEncoder.~~encode~~("yusheng+jun");  System.***out***.println("userNameEncode:" + userNameEncode);  String userNameDecode = URLDecoder.~~decode~~(userNameEncode);  System.***out***.println("userNameDecode:" + userNameDecode);  } |