

微信官方文档：

[https://mp.weixin.qq.com/wiki?t=resource/res\\_main&id=mp144524143](https://mp.weixin.qq.com/wiki?t=resource/res_main&id=mp144524143)

2

微信公众平台测试号：[https://mp.weixin.qq.com/debug/cgi-](https://mp.weixin.qq.com/debug/cgi-bin/sandbox?t=sandbox/login)

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

开发工具：IDEA

实现功能：用户首次关注发送一个图文给用户，用户点击图文之后会让用户勾选适合自己的标签，然后为用户自动打上标签，其中包括后台对用户所选的标签是否已经存在后台，用户是否已经拥有此标签等逻辑处理，然后给用户相应的回复，同时完成后台交互功能，后台用户回复“创建标签”会回复给用户相应的打标签介绍，然后用户选择相应的操作，后台根据用户的选择为用户打上相应的标签，同时包含标签的逻辑判断，用户回复“查看标签”可回复给用户后台创建的标签（这个在开发的时候可以作为获取 token 来使用）

## 1、基于 wx-tools

项目地址：<https://github.com/antgan/wx-tools>

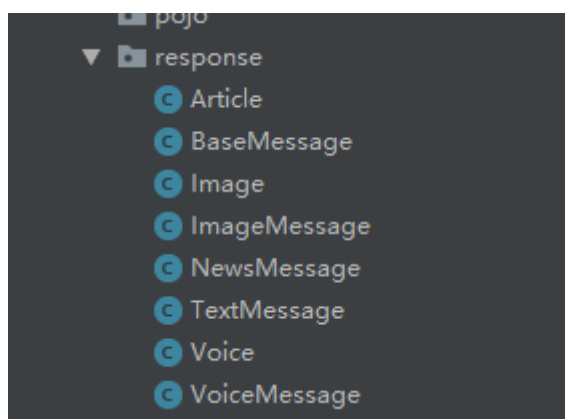
项目开发技术文档：<https://www.w3cschool.cn/wxtools/>

## 2、ngrok 内网穿透，域名映射

为了方便开发能够在本地进行调试我们需要将 127.0.0.1 映射到外网上去，可选的工具很多，这里采用的是 ngrok。

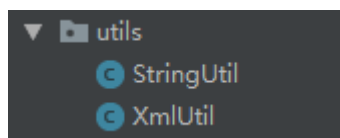
## 3、核心代码

### 3.1



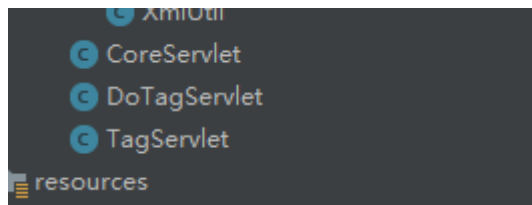
这是对微信公众号后台也就是微信服务器返回给用户的一些消息的封装，这个可以选择性进行使用。

### 3.2

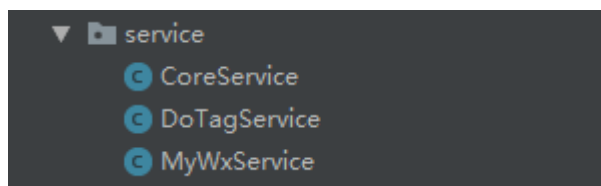


这是用到的两个主要的工具类，一个是对 xml 数据的解析，一个是对字符串的判断

### 3.3



最为核心的三个 servlet 主要用于对数据的处理，但是不直接处理消息，业务逻辑的处理全部交给对应的 service 去处理



都是相互对应的

#### 3.3.1、CoreSevlet

这个是核心的 servlet，主要用来处理微信服务器发送过来的请求

```

/**
 * 核心servlet, 处理微信服务器发送过来的消息
 */
public class CoreServlet extends HttpServlet {

    private IService iService = new MyWxService();

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {...}

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

        // 将请求、响应的编码均设置为UTF-8 (防止中文乱码)
        req.setCharacterEncoding("UTF-8");
        resp.setCharacterEncoding("UTF-8");
        PrintWriter out = resp.getWriter();

        CoreService coreService = new CoreService();
        String s = coreService.parseWxRequest(req);

        // 响应消息, 将相应的xml数据转发给微信服务器
        out.print(s);
        System.out.println("消息: " + s);
        out.close();
    }
}

```

并且将请求转给 service 去处理，首先我们需要将这个 servlet 的地址配置在微信公众开发平台的 URL 中，这样微信服务器会往这个 servlet 发送相应的请求，我们在这里接收做相应的处理，需要先对微信服务器发送过来的请求做验证，确定是微信服务器发送过来的请求。

请求从微信服务器发送过来之后，我们将请求交给 CoreService 去处理，我们主要在这个 service 中去做相应的逻辑处理

这里要注意，响应这块要把握一点，你只要返回给微信服务器正确的数据，它就会给你做相应的事情，返回的数据是 xml 数据包

### 3.3.2、对接公众号和服务器

对接步骤：

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

首先是填写服务器的配置

第二步验证服务器地址的有效性是比较重要的，因为我们做公众号开发就是将用户发送的各个请求通过微信服务器转发到我们自己的服务器上，然后进行相应的处理，因此我们做这个有效性的验证就是让我们的服务器知道这个请求是微信服务器发送过来的，那么如何进行验证呢？我们的服务器会接收到微信服务器发过来的一个 get 请求，这其中包含四个参数，分别是 token，timestamp，nonce 和 signature。验证方式如下

- 将 token、timestamp、nonce 三个参数进行字典序排序
- 将三个参数字符串拼接成一个字符串进行 sha1 加密
- 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

```
1 #配置如下
2 wx.appId=wx2ed3...fedf23...
3 wx.appSecret=df...03...b510b1fd1209085
4 wx.token=mytoken
5 wx.aesKey=
6 wx.mchId=
7
```

项目中将对应的信息写在了 wx.properties

## 3.4、TagServlet

这个是主要处理打标签的 servlet，同样是具体的实现交给对应的 service 去处理，对于项目中所有含有 Tag 的都是关于打标签的，大多数原理跟 CoreServlet 差不多

# 4、功能实现流程

## 4.1、对接微信服务器，验证请求

这个是在 CoreServlet 中实现

```
private IService iService = new MyWxService();
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    // 微信加密签名
    String signature = req.getParameter("signature");
    // 时间戳
    String timestamp = req.getParameter("timestamp");
    // 随机数
    String nonce = req.getParameter("nonce");
    // 随机字符串
    String echostr = req.getParameter("echostr");
    PrintWriter out = resp.getWriter();
    if (iService.checkSignature(signature, timestamp, nonce, echostr)) {
        out.print(echostr);
    }
    out.close();
    out = null;
}
```

相关类请参考 wx-tools

## 4.2、解析微信服务器发送的请求

同样是在 CoreServlet 中去接收，然后交给 CoreService 去处理

```
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException {  
    // 将请求、响应的编码均设置为UTF-8（防止中文乱码）  
    req.setCharacterEncoding("UTF-8");  
    resp.setCharacterEncoding("UTF-8");  
    PrintWriter out = resp.getWriter();  
  
    CoreService coreService = new CoreService();  
    String s = coreService.parseWxRequest(req);  
  
    // 响应消息，将相应的xml数据转发给微信服务器  
    out.print(s);  
    System.out.println("消息: " + s);  
    out.close();  
}
```

## 4.3、对用户请求做出响应

请求交给 CoreService 之后，做出响应额处理然后返回给微信服务器数据，这个数据是 xml 数据包格式，只要返回正确的数据包，微信服务器就会响应给用户正确的消息

```
/**  
 * 解析用户请求得到的数据wx  
 */  
wx = XStreamTransformer.fromXml(WxXmlMessage.class, request.getInputStream());  
String content = wx.getContent();  
/**  
 * 根据用户发送的content做出响应  
 */
```

拿到解析之后的数据，然后就可以取出其中包含的数据，然后我们就可以根据用户发送的请求内容和请求类型做出判断

### 4.3.1、响应事件

```
/**
 * 首次关注，做出响应
 */
if (msgType.equals("event")){

    //第一步：构造URL获取Code
    String oauthUrl = iService.oauth2buildAuthorizationUrl(redirectUrl + "http://dd7d9514.ngrok.io/tagServlet", WxConsts.OAUTH2_S
    System.out.println(oauthUrl);

    //上传封面图
    WxMediaUploadResult result = iService.uploadTempMedia(WxConsts.MEDIA_IMAGE, new File(pathname + "F://test.jpg"));
    System.out.println(result.getMedia_id());

    //回复图文消息
    WxXmlOutNewsMessage.Item item = new WxXmlOutNewsMessage.Item();
    item.setDescription("如鹏网---专注于大学生在线教育!");

    item.setPicUrl("http://i66.tinypic.com/2d1ttl5.png");
    item.setTitle("欢迎关注如鹏网!");
    item.setUrl(oauthUrl);
    //将图文转换成xml
    WxXmlOutNewsMessage build = WxXmlOutMessage.NEWS().addArticle(item).toUser(wx.getFromUserName()).fromUser(wx.getToUserNa
    respXml = build.toXml();
}
```

根据用户的请求消息类型，判断事件类型，如果是用户关注事件，我们就回复一个图文

同样的道理我们可以根据请求解析出来的数据来对文本信息进行判断，然后做出响应，主要的处理代码如下



```

}else
/**
 * 文字消息响应
 */

if (content.equals("测试图文")){
    //回复图文消息
    WxXmlOutNewsMessage.Item item = new WxXmlOutNewsMessage.Item();
    item.setDescription("这是测试图文的一些描述");
    item.setPicUrl("http://i66.tinypic.com/2d1ttl5.png");
    item.setTitle("测试图文标题");
    item.setUrl("https://www.baidu.com/");
    //将图文转换成xml
    WxXmlOutNewsMessage build = WxXmlOutMessage.NEWS().addArticle(item).toUser(wx.getFromUserName()).fromUser(wx.getFromUserName());
    respXml = build.toXml();
}else if (content.equals("创建标签")){
    //创建标签
    content = "欢迎关注如鹏网，为了给你带来更好的服务，我们推出了标签功能，以便为你推荐你更感兴趣的内容请选择你感兴趣的方向：0-ASP.NET、1-PYTHON、2-JAVA、3-C语言、4-C#";
    WxXmlOutTextMessage build = WxXmlOutMessage.TEXT().content(content).toUser(wx.getFromUserName()).fromUser(wx.getFromUserName());
    respXml = build.toXml();
}
//asp.net标签
else if (content.equals("0")){
    String C = CoreService.creatTag("ASP.NET", wx.getContent(),wx.getFromUserName(),wx.getToUserName());
    respXml = C;
}
//python标签
else if (content.equals("1")){
    String python = CoreService.creatTag("python", wx.getContent(),wx.getFromUserName(),wx.getToUserName());
    respXml = python;
}
//java标签
else if (content.equals("2")){
    String Java = CoreService.creatTag("Java", wx.getContent(),wx.getFromUserName(),wx.getToUserName());
    respXml = Java;
}
//C语言标签
else if (content.equals("3")){
    String C = CoreService.creatTag("C语言", wx.getContent(),wx.getFromUserName(),wx.getToUserName());
    respXml = C;
}
//C#标签
else if (content.equals("4")){
    String cc = CoreService.creatTag("C#", wx.getContent(),wx.getFromUserName(),wx.getToUserName());
    respXml = cc;
}

```

#### 4.3.1.1、创建标签的封装

因为对于文本消息的响应，我们主要是做打标签的操作，有很多的相似之处，所以这里我们封装了一个创建标签的方法，主要是对标签是否存在的判断以及标签的各种操作方法。

```

//为用户创建标签的方法
public static String creatTag(String codeName,String content,String getFromUserName,String getToUserName){

    String respXml = null;

    IService iService = new MyWxService();

    // 微信服务器推送过来的是XML格式。
    // WxXmlMessage wx = null;
    //为用户创建Java标签
    //1、创建标签
    try {
        //首先需要判断是否已存在该标签,没有则创建
        WxUserTagResult AllUserTagresult = iService.queryAllUserTag();

        List<WxUserTagResult.WxUserTag> tags = AllUserTagresult.getTags();
        //得到标签名称数组
        String[] tagname_list = new String[tags.size()];
        for (int i=0;i<tags.size();i++){
            tagname_list[i] = tags.get(i).getName();
        }
        System.out.println(tagname_list[1]);
        //如果已经存在该标签ArrayUtils.contains(tagname_list,codeName)
        if (StringUtil.isIn(codeName,tagname_list)){
            System.out.println("已经存在");
            //此时表明已存在该标签,所以需要查看用户是否已经拥有该标签,没有拥有则为其设置
            //首先获取用户基本信息,从基本信息中得到他拥有的标签
            WxUserList.WxUser user = iService.getUserInfoByOpenId(new WxUserList.WxUser.WxUserGet(getFromUserN
            //获得该用户拥有的标签列表
            String[] tagid_list = user.getTagid_list();
            int aspID = 0;

            for (WxUserTagResult.WxUserTag tag : tags){
                if (tag.getName().equals(codeName)){

```

以上都是在 CoreService 中完成的,基本关于非图文的标签功能就已经全部完成,这里要结合 wx-tools 一起去参考,因为其中有些逻辑是 wx-tools 帮我们完成的。

## 4.4、图文创建标签

对于图文创建标签不像文本交互创建标签这样,我们需要通过微信的网页授权,也就是通过 Oauth2.0 认证来获取用户的相关消息,通过微信网页授权我们会得到一个网页特有的 token,在微信网页授权这块我们将回调地址写成我们要回复给用户的图文,当用户点击即为默认授权,然后我们得到相对应的 token 和 openid,在这里我们需要将得到的 token 和 openid 进行保存,以便执行图文标签创建的过程

#### 4.4.1、功能流程解析

```

    * 首次关注，做出响应
    */
    if (msgType.equals("event")){

        //第一步：构造URL获取Code
        String oauthUrl = iService.oauth2buildAuthorizationUrl(redirectUrl + "http://dd7d9514.ngrok.io/tagServlet", WxConsts.MEDIA_IMAGE);
        System.out.println(oauthUrl);

        //上传封面图
        WxMediaUploadResult result = iService.uploadTempMedia(WxConsts.MEDIA_IMAGE, new File(pathname + "F://test.jpg"));
        System.out.println(result.getMedia_id());

        //回复图文消息
        WxXmlOutNewsMessage.Item item = new WxXmlOutNewsMessage.Item();
        item.setDescription("如鹏网---专注于大学生在线教育!");

        item.setPicUrl("http://i66.tinypic.com/2d1ttl5.png");
        item.setTitle("欢迎关注如鹏网!");
        item.setUrl(oauthUrl);
        //将图文转换成xml
        WxXmlOutNewsMessage build = WxXmlOutMessage.NEWS().addArticle(item).toUser(wx.getFromUserName()).fromUser(wx.getToUserName());
        respXml = build.toXml();
    }
}
```

我们在 CoreService 中对解析请求得到的数据进行处理，判断用户的请求是否为关注事件，如果发现是关注事件我们就回复给用户一条图文，此图文链接就是我们微信网页授权的回调地址，回复图文之后，用户打开是一个简单的网页展示，采用 form 表单的数据进行提交数据

```

8      <%@ page contentType="text/html; charset=UTF-8" language="java" %>
9      <html>
10     <head>
11         <meta charset="UTF-8">
12         <meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0">
13         <meta name="apple-touch-fullscreen" content="YES">
14         <meta name="apple-mobile-web-app-capable" content="yes">
15         <title>如鹏网--专注大学生在线教育</title>
16     </head>
17     <body>
18         
19
20         <form action="/dotag" method="post">
21             <h2>1 请问你对以下哪方面最感兴趣呢? </h2>
22
23             <input type="radio" name="code" value="Java" id="1"/>
24             <label for="1">Java</label>
25             <input type="radio" name="code" value="python" id="2"/>
26             <label for="2">python</label>
27             <input type="radio" name="code" value=".net" id="3"/>
28             <label for="3">python</label>
29             <h2>2 请问您还对以下哪些技术感兴趣呢? (多选) </h2>
30
31             <input type="checkbox" name="hobby" /> C
32             <input type="checkbox" name="hobby" /> C++
33             <input type="checkbox" name="hobby" /> PHP
34             <input type="checkbox" name="hobby" /> Android
35             <input type="checkbox" name="hobby" /> 其他
36
37             <input type="submit" value="提交" />
38         </form>
39     </body>
40 </html>
41

```

然后我们进行 post 提交给 DoTagServlet

```

* 首次关注，做出响应
*/
if (msgType.equals("event")){
    //第一步：构造URL获取Code
    String oauthUrl = iService.oauth2buildAuthorizationUrl("http://dd7d9514.ngrok.io/tagServlet", WxConsts.APP_ID, WxConsts.SECRET);
    System.out.println(oauthUrl);

    //上传封面图
    WxMediaUploadResult result = iService.uploadTempMedia(WxConsts.MEDIA_IMAGE_NEW, File("F://test.jpg"));
}

```

这里要注意我们微信网页的回调地址其实是 tagServlet，微信网页授权的目的就是活的 token 和 openid，然后将其保存下来

```

@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    //解决乱码问题
    resp.setContentType("text/html;charset=UTF-8");
    resp.setCharacterEncoding("UTF-8");
    req.setCharacterEncoding("UTF-8");

    PrintWriter writer = resp.getWriter();

    IService iService = new MyWxService();
    //拿code换token和openid
    WxOAuth2AccessTokenResult result = null;
    try {
        result = iService.oauth2ToGetAccessToken(req.getParameter("code"));

        //通过网页授权获取到的openid和token
        String access_token = result.getAccess_token();
        String openid = result.getOpenid();
        System.out.println(access_token);
        System.out.println(openid);

    } catch (WxErrorException e) {
        e.printStackTrace();
    }
    String state = req.getParameter("state");
    System.out.println("state"+state);

    HttpSession session = req.getSession();
    session.setAttribute("token", result.getAccess_token());
    session.setAttribute("openid", result.getOpenid());

    req.getRequestDispatcher("/index.jsp").forward(req, resp);
}

```

在 TagServlet 中我们需要得到 token 和 openid 然后将其保存起来，这里将其保存在 session 中，以便我们用到的时候能够取出（这里杨老师提到的另一个做法是可以将 token 和 openid 通过 form 表单进行提交）

```

import java.util.List;

public class DoTagServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        doPost(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        resp.setContentType("text/html;charset=UTF-8");

        HttpSession session = req.getSession();
        String token = (String) session.getAttribute("token");
        String openid = (String) session.getAttribute("openid");
        String code = req.getParameter("code");
        IService iService = new MyWxService();
        PrintWriter writer = resp.getWriter();

        String s = DoTagService.doTag(openid, token, code);

        writer.print(s);
    }
}

```


form 表单的提交时提交到了 DoTagServlet 中，我们在这里拿到用户提交的数据以及我们存在 session 中的 openid 和 token，然后我们就可以执行打标签的操作，同样是交给对应的 DoTagService 去处理

```
resp.setContentType("text/html;charset=UTF-8");

HttpSession session = req.getSession();
String token = (String) session.getAttribute(name: "token");
String openid = (String) session.getAttribute(name: "openid");
String code = req.getParameter(name: "code");
IService iService = new MyWxService();
PrintWriter writer = resp.getWriter();

String s = DoTagService.doTag(openid, token, code);

writer.print(s);
}
```



我们将得到的 token 和 openid 还有得到的而用户提交的数据传给

DoTagService, 然后在 DoTagService 做相应的逻辑处理

```
try {
    //拿token换用户信息
    WxUserList.WxUser user = iService.oauth2ToGetUserInfo(token, new WxUserList.WxUser.WxUserGet(openid, WxConsts.LANG_CHINA));

    String s = DoTagService.creatTag(openid, token, code, user);
    resp = s;
} catch (WxErrorException e) {
    e.printStackTrace();
}
```

然后在 DoTagService 我们可以通过得到的 openid 获取到用户信息

```

//为用户创建标签的方法
public static String creatTag(String openid,String token,String code,WxUserList.WxUser user){

    String resp = null;

    IService iService = new MyWxService();

    System.out.println("看这里: ====="+openid+token+code+user.toString());

    //1、创建标签
    try {
        //首先需要判断是否已存在该标签,没有则创建
        WxUserTagResult AllUserTagresult = iService.queryAllUserTag();

        List<WxUserTagResult.WxUserTag> tags = AllUserTagresult.getTags();
        //得到标签名称数组
        String[] tagname_list = new String[tags.size()];
        for (int i=0;i<tags.size();i++){
            tagname_list[i] = tags.get(i).getName();
        }
        System.out.println(tagname_list[1]);
        //如果已经存在该标签ArrayUtils.contains(tagname_list,codeName)
        if (StringUtil.isIn(code,tagname_list)){
            System.out.println("已经存在code:"+code);
            //此时表明已存在该标签,所以需要查看用户是否已经拥有该标签,没有拥有则为其设置
            //首先获取用户基本信息,从基本信息中得到他拥有的标签
            //获得该用户拥有的标签列表
            String[] tagid_list = user.getTagid_list();
            int aspID = 0;

            for (WxUserTagResult.WxUserTag tag : tags){
                if (tag.getName().equals(code)){
                    System.out.println("已经存在code:"+code);
                    System.out.println(tag.getName());
                    aspID = tag.getId();
                    System.out.println(aspID);
                }
            }
            System.out.println(aspID);
            //如果用户拥有该标签ArrayUtils.contains(tagid_list,String.valueOf(aspID))
            if (StringUtil.isIn(String.valueOf(aspID),tagid_list)){

```

接下来就是对用户打标签的操作了，这里的操作和文本创建标签的额外逻辑处理的思想是一致的。

## 5、其他说明

### 5.1、关于 token 的过期问题

对于 token 我们一天的调取上限时 2000 次，但是随着我们本地演示需要不断的重启服务器导致 token 会不断的刷新，从而会达到调取上限，因此，这里采用的一个方法是将得到的 token 进行硬编码，因为一个 token 的使用时长是

两小时，这样就可以有效避免达到调取上限的问题

```
/**
 * 继承自WxService以便修改响应方法
 * 比如对token的存储
 */
public class MyWxService extends WxService {
    @Override
    public String getAccessToken() throws WxErrorException {
        /**
         * token每天调取上限2000次
         * 为避免达到上限
         * 将获得的token以硬编码方式存储
         * 一个token的使用时长有两个小时左右
         */
        //开发测试采用,两小时过后需要调用上线采用的方法刷新获取新的token
        return "12_2sztWZDG-06oyyQ50bBFwW1TiN3LibX7nJ1FCoxLfVpj0T0du0YLfwQHpCvZTIgi0mavu0B9D0wPQ3DA98e2lT0";
        //正常上线采用
        // return super.getAccessToken();
    }
}
```

主要是新创建一个 service，然后将得到的 token 进行 hardcode。

## 5.2、使用 postman 进行调试

因为我们开发是基于微信公众测试号进行的，没有后台直观的看到后台标签的相关信息，所以我们可以借用 postman 来进行借口借口调用查看后台标签相关信息，非常方便，可以实时看到我们的标签是否真的创建成功。

## 5.3、关于图片上传问题

如果遇到图片无法显示的问题，试着将图片链接的域名转换成腾讯的域名。



## 6、最后

微信公众号的开发，只要你能够返回给微信服务器正确的消息格式，微信服务器就会做你想让他做的事情！