

微信公众帐号开发教程第 1 篇-引言.....	2
微信公众帐号开发教程第 2 篇-微信公众帐号的类型（普通和会议）.....	3
微信公众帐号开发教程第 3 篇-开发模式启用及接口配置.....	7
微信公众帐号开发教程第 4 篇-消息及消息处理工具的封装.....	17
微信公众帐号开发教程第 5 篇-各种消息的接收与响应.....	39
微信公众帐号开发教程第 6 篇-文本消息的内容长度限制揭秘.....	45
微信公众帐号开发教程第 7 篇-文本消息中换行符的使用.....	49
微信公众帐号开发教程第 8 篇-文本消息中使用网页超链接.....	51
微信公众帐号开发教程第 9 篇-QQ 表情的发送与接收.....	53
微信公众帐号开发教程第 10 篇-解析接口中的消息创建时间 CreateTime.....	61
微信公众帐号开发教程第 11 篇-符号表情的发送（上）.....	63
微信公众帐号开发教程第 12 篇-符号表情的发送（下）.....	76
微信公众帐号开发教程第 13 篇-图文消息全攻略.....	85
微信公众帐号开发教程第 14 篇-自定义菜单的创建及菜单事件响应.....	95
微信公众帐号开发教程第 15 篇-自定义菜单的 view 类型（访问网页）.....	117
微信公众帐号开发教程第 16 篇-应用实例之历史上的今天.....	123
微信公众帐号开发教程第 17 篇-应用实例之智能翻译.....	132
微信公众帐号开发教程第 18 篇-应用实例之音乐搜索.....	140
微信公众平台开发教程第 19 篇-应用实例之人脸检测.....	152
微信公众平台开发教程第 20 篇-新手解惑 40 则.....	169
微信公众平台开发教程第 21 篇-“可信网址”白名单.....	175

微信公众帐号开发教程第 1 篇-引言

接触微信公众帐号已经有两个多月的时间了，在这期间，除了陆续完善个人公众帐号 xiaoqrobot 以外，还带领团队为公司开发了两个企业应用：一个是普通类型的公众帐号，另一个是会议类型的公众帐号。经过这 3 个公众帐号的开发，对目前微信公众平台开放的 api 算是比较熟悉了，像文本消息、图文消息、音乐消息、语音消息、位置消息等全部用到过，菜单也使用过。所以，就有了写微信公众帐号开发教程的想法，将学习到的技术经验分享出来，帮助更多需要的朋友，也希望借此认识同行的朋友，相互交流，共同进步！

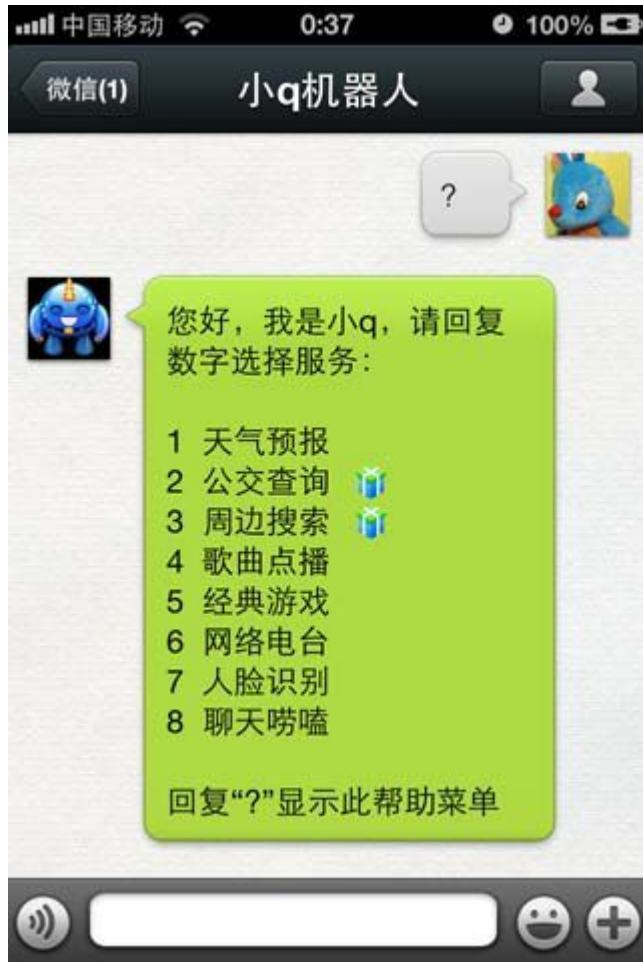
下面将对即将推出的微信公众帐号开发系列连载教程做简单的说明。教程主要是面向有一定 Java 编程基础的朋友，不打算从编程语言开始讲起，一是考虑到自己没有那么多时间和精力（要上班、装修、学车等），二是怕等我把编程语言讲完，微信公众帐号又发生了大变化，这样教程就显得有点过时，没有吸引力了，所以只能是有侧重点的介绍。至于内容方面，大概会涉及到：

- 1) 前沿知识：微信公众帐号的分类、两种模式各自的特点和区别、开发模式的配置使用等；
- 2) API 中各类消息的使用（我已经对 api 进行封装并打成了 jar 包，到时候会考虑分享出来）；
- 3) 微信公众帐号开发中的小技巧（如换行、通过代码发送表情、屏幕飘雪花、表情的接收识别、在 Android 和 iOS 上表现不一致等等）；
- 4) 与业务系统对接的方法（链接、短信等，除了技术讲解还会做一定的分析对比）；
- 5) 微信公众平台上常见功能的开发（如像小黄鸡那样的人机对话、天气预报、精确的定位及百度地图的使用、音乐搜索、语音识别解析等）

当然，具体写出来的内容肯定不止这些，但一定会包含以上介绍的所有内容。

我也不知道多久能写完这些内容，当然是越快越好，我会尽全力的。**希望正在看博文的你通过微信关注 xiaoqrobot 或者在博客留言支持**，给我动力，谢谢！

开发 xiaoqrobot 就是为了学习微信公众帐号开发，将 api 开放出来的各类消息都体验了。虽然现在看来有点大杂烩的意思，但还是比较实用的，一款生活、娱乐的好帮手，目前已有 370 多关注者。**周边搜索功能定位比较准确（解决了纠偏问题，能精确到十米范围），平时出门在外搜美食、ATM 机、厕所、超市等再方便不过了，还提供路线导航；聊天唠嗑功能是我自己开发的，后面的连载教程很多内容都会从中抽取出来**，下面是主界面截图，对系列连载教程有所期待的朋友很建议关注体验下，做的不好的地方也请多提意见，除了技术本身外，体验也是我比较重视关注的。



微信公众帐号开发教程第 2 篇-微信公众帐号的类型（普通和会议）

个人公众帐号与企业公众帐号

记得在两个月前，我在微信官方开发群里问个人公众帐号与企业公众帐号有什么区别的时候，还被人笑话过，没有人愿意告知，也许是这个问题问的太过于简单了吧。我想一定也还有不少朋友在刚接触时，也搞不清楚这一点。其实，在注册微信公众帐号时，是不区分个人帐号与企业帐号的，它们需要填写的注册资料是一样的，这个区别仅仅是帐号申请成功后在使用用途上的区别罢了。然而，在注册公众帐号时的确有个类型可以选择，但并不是选择个人帐号与企业帐号，那有些什么类型可以选择呢？这也正是今天我想讲的主题，请继续往下看。

注册时可选择的两种帐号类型

微信公众帐号注册的最后一步是填写“公众号信息”，最后一个选项是选择“类型”，它有二个值可供选择“普通公众帐号类型”和“公众会议帐号”。当我们选择“公众会议号”时，下方会出现醒目的红字“提醒：会议号是有一定时间限制的公众帐号，过期后将无法登录使用。”，如下图所示。

1 基本信息

2 邮箱激活

3 信息登记

帐号名称

(2-16个字)名称一经设置无法更改。

功能介绍

(4-120个字)介绍此公众帐号功能与特色。

运营地区

国家

语言

简体中文

类型

普通公众帐号类型

公众会议帐号

提醒：会议号是有一定时间限制的公众帐号，过期后将无法登录使用。

按号码查找

详细资料

微信公众号小

用户说明

关注微信官员号/取微信官号问题/官号平台新服务。

关注

那注册时到底应该选择哪个类型呢？这就需要我们对两种类型有一定的了解才好做出判断。下面将主要通过介绍公众会议帐号与普通公众帐号的区别来进行说明。

公众帐号与普通帐号的区别

在注册好的公会会议帐号的“设置”一栏里，可以看到“会议号设置”项，如下图所示：



其实会议帐号与普通帐号的区别在“会议号设置”里就能全部体现出来，它们的区别有以下三点：

1) 有效时间

普通帐号是创建后永久有效的，而会议帐号的有效期只有一个月，一个月后帐号就失效了。帐号失效后登录微信公众平台时，会提示“**该公众会议号已经过期，无法再登录使用**”，如下图所示：



帐号失效后已关注了会议帐号的用户继续使用时，会提示“**该公众帐号已过期，无法下发消息**”，但如果是有菜单权限的会议帐号，仍然可以通过菜单获取信息，帐号过期后菜单的响应没有被禁止，如下图所示：



从上图可以看到，会议帐号过期后，无法再通过文本获取消息，但点击菜单是可以继续使用的，图中的图文消息“峰会概况”就是点击菜单后返回的。

2) 关注权限

普通帐号任何人都可以关注，没有权限限制。会议帐号是可以设置关注权限的，分为两种：任何人都可以关注和需要通过验证才可以关注，不进行此项设置时默认是前者。如果设置为需要通过验证才可以关注，就有点类似于微信添加朋友时的验证一样，只不过这里的验证问题是可以设置的，并且如果你设置的验证消息是类似于询问用户身份的，例如“请问您的真实姓名叫什么？”，你还可以勾选“将验证消息作为备注名”，这样就很好辨认所有关注了会议帐号的人。

3) 参与人相互可见

普通帐号的关注者之间是不可见的，而会议帐号的关注者之间是相互可见的，这是什么意思呢？在会议号设置里，如果勾选了“参与人相互可见”，那么在关注了该会议帐号后，能够在帐号详细资料里看到多了一项“与会者”，点击它将会显示所有关注了该会议帐号的微信号列表，并且点击某个参与人还可以查看详细资料、申请加为好友等。这是会议帐号比较给力的一个功能，方便参加会议的人相互认识。



以上三点是会议帐号的特点，也是与普通帐号的区别。可以看出，会议帐号是在普通帐号功能的基础上增加了帐号有效时间限制（一个月）、关注权限和关注者相互可见三个功能。

其实，微信目前对会议帐号的支持还远远不够。比如像会议主题、时间、地点等会议的常规属性设置都不支持，还有会议通常都会有的签到、互动、投票等环节没有任何体现，更没有考虑到周期性的会议，希望微信后期的版本对这块的支持力度更大。

微信公众帐号开发教程第 3 篇-开发模式启用及接口配置

编辑模式与开发模式

微信公众帐号申请成功后，要想接收处理用户的请求，就必须要在“高级功能”里进行配置，点击“高级功能”，将看到如下界面：



从上图中可以看到，高级功能包含两种模式：编辑模式和开发模式，并且这两种模式是互斥关系，即两种模式不能同时开启。那两种模式有什么区别呢？作为开发人员到底要开启哪一种呢？

编辑模式：主要针对非编程人员及信息发布类公众帐号使用。开启该模式后，可以方便地通过界面配置“自定义菜单”和“自动回复的消息”。

开发模式：主要针对具备开发能力的人使用。开启该模式后，能够使用微信公众平台开放的接口，通过编程方式实现自定义菜单的创建、用户消息的接收/处理/响应。这种模式更加灵活，建议有开发能力的公司或个人都采用该模式。

启用开发模式（上）

微信公众帐号注册完成后，默认开启的是编辑模式。那么该如何开启开发模式呢？操作步骤如下：

1) 点击进入编辑模式，将右上角的编辑模式开关由“开启”切换到“关闭”，如下图所示：

高级功能 > 编辑模式



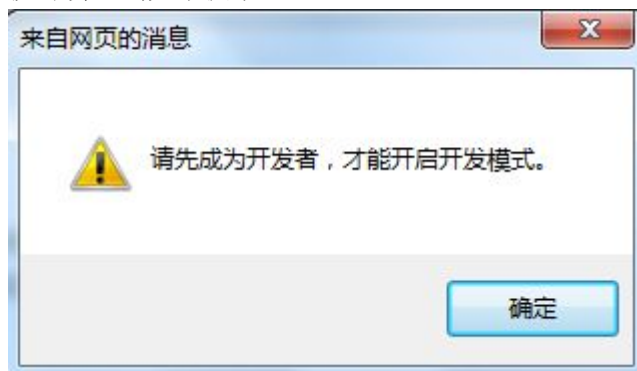
编辑模式已开启

高级功能 > 编辑模式



编辑模式已关闭

2) 点击高级功能进入到开发模式，将右上角的开发模式开关由“关闭”切换到“开启”，但在切换时会遇到如下提示：



提示需要我们先成为开发者，才能开启开发模式。那就先点击下图所示的“成为开发者”按钮：

尚未成为开发者

申请成为开发者后，可以使用公众平台的接口进行开发。

如果提示资料不全，那就先补齐资料再回来继续操作。需要补全的资料有公众帐号头像、描述和运营地区。



资料不全，无法继续

你需要添加公众号头像、描述和运营地区后才能继续下一步。

[现去补全资料](#)

待资料补全后，再次点击“成为开发者”，这时将看到接口配置信息界面，如下图所示：

接口配置信息

请填写接口配置信息，此信息需要你拥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

URL 必须以http://开头，目前支持80端口。

Token 必须为英文或数字，长度为3-32字符。

[什么是Token？](#)

提交

这里需要填写 URL 和 Token 两个值。URL 指的是能够接收处理微信服务器发送的 GET/POST 请求的地址，并且是已经存在的，现在就能够在浏览器访问到的地址，这就要求我们先把公众帐号后台处理程序开发好（至少应该完成了对 GET 请求的处理）并部署在公网服务器上。Token 后面会详细说明。

也就是说要完成接口配置，只需要先完成微信服务器的 GET 请求处理就可以？是的。那这是为什么呢？因为这是微信公众平台接口中定义的。具体请参考 API 文档-消息接口-消息接口指南中的网址接入部分。[点此进入](#)。

网址接入

公众平台用户提交信息后，微信服务器将发送GET请求到填写的URL上，并且带上四个参数：

参数	描述
signature	微信加密签名
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，否则接入失败。

signature结合了开发者填写的token参数和请求中的timestamp参数、nonce参数。

加密/校验流程：

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

上面写的很清楚，其实你只要能理解上面在说什么就 OK 了，至于怎么编写相关代码，我已经帮你完成了，请继续往下看。

创建公众帐号后台接口程序

创建一个 Java Web 工程，并新建一个能够处理请求的 Servlet，命名任意，我在这里将其命名为 org.liufeng.course.servlet.CoreServlet，代码如下：

[java] view plaincopy

```
1. package org.liufeng.course.servlet;
2.
3. import java.io.IOException;
4. import java.io.PrintWriter;
5.
6. import javax.servlet.ServletException;
7. import javax.servlet.http.HttpServlet;
8. import javax.servlet.http.HttpServletRequest;
9. import javax.servlet.http.HttpServletResponse;
10.
```

```
11. import org.liufeng.course.util.SignUtil;
12.
13. /**
14.  * 核心请求处理类
15.  *
16.  * @author liufeng
17.  * @date 2013-05-18
18.  */
19. public class CoreServlet extends HttpServlet {
20.     private static final long serialVersionUID = 4440739483644821986L;
21.
22.     /**
23.      * 确认请求来自微信服务器
24.      */
25.     public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
26.         // 微信加密签名
27.         String signature = request.getParameter("signature");
28.         // 时间戳
29.         String timestamp = request.getParameter("timestamp");
30.         // 随机数
31.         String nonce = request.getParameter("nonce");
32.         // 随机字符串
33.         String echostr = request.getParameter("echostr");
34.
35.         PrintWriter out = response.getWriter();
36.         // 通过检验 signature 对请求进行校验, 若校验成功则原样返回 echostr, 表示接入成功, 否则接入失败
37.         if (SignUtil.checkSignature(signature, timestamp, nonce)) {
38.             out.print(echostr);
39.         }
40.         out.close();
41.         out = null;
42.     }
43.
44.     /**
45.      * 处理微信服务器发来的消息
46.      */
47.     public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
48.         // TODO 消息的接收、处理、响应
49.     }
50.
51. }
```

可以看到，代码中只完成了 **doGet** 方法，它的作用正是确认请求是否来自于微信服务器；而 **doPost** 方法不是我们这次要讲的内容，并且完成接口配置也不需要管 **doPost** 方法，就先空在那里。

在 **doGet** 方法中调用了 **org.liufeng.course.util.SignUtil.checkSignature** 方法，**SignUtil.java** 的实现如下：

[java] view plaincopy

```
1. package org.liufeng.course.util;
2.
3. import java.security.MessageDigest;
4. import java.security.NoSuchAlgorithmException;
5. import java.util.Arrays;
6.
7. /**
8.  * 请求校验工具类
9.  *
10.  * @author liufeng
11.  * @date 2013-05-18
12.  */
13. public class SignUtil {
14.     // 与接口配置信息中的 Token 要一致
15.     private static String token = "weixinCourse";
16.
17.     /**
18.      * 验证签名
19.      *
20.      * @param signature
21.      * @param timestamp
22.      * @param nonce
23.      * @return
24.      */
25.     public static boolean checkSignature(String signature, String timestamp,
        String nonce) {
26.         String[] arr = new String[] { token, timestamp, nonce };
27.         // 将 token、timestamp、nonce 三个参数进行字典序排序
28.         Arrays.sort(arr);
29.         StringBuilder content = new StringBuilder();
30.         for (int i = 0; i < arr.length; i++) {
31.             content.append(arr[i]);
32.         }
33.         MessageDigest md = null;
34.         String tmpStr = null;
35.
```

```

36.         try {
37.             md = MessageDigest.getInstance("SHA-1");
38.             // 将三个参数字符串拼接成一个字符串进行 sha1 加密
39.             byte[] digest = md.digest(content.toString().getBytes());
40.             tmpStr = byteToStr(digest);
41.         } catch (NoSuchAlgorithmException e) {
42.             e.printStackTrace();
43.         }
44.
45.         content = null;
46.         // 将 sha1 加密后的字符串可与 signature 对比, 标识该请求来源于微信
47.         return tmpStr != null ? tmpStr.equals(signature.toUpperCase()) : false;
48.     }
49.
50.     /**
51.      * 将字节数组转换为十六进制字符串
52.      *
53.      * @param byteArray
54.      * @return
55.      */
56.     private static String byteToStr(byte[] byteArray) {
57.         String strDigest = "";
58.         for (int i = 0; i < byteArray.length; i++) {
59.             strDigest += byteToHexStr(byteArray[i]);
60.         }
61.         return strDigest;
62.     }
63.
64.     /**
65.      * 将字节转换为十六进制字符串
66.      *
67.      * @param mByte
68.      * @return
69.      */
70.     private static String byteToHexStr(byte mByte) {
71.         char[] Digit = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };
72.         char[] tempArr = new char[2];
73.         tempArr[0] = Digit[(mByte >> 4) & 0X0F];
74.         tempArr[1] = Digit[mByte & 0X0F];
75.
76.         String s = new String(tempArr);
77.         return s;

```

```
78.     }  
79. }
```

这里唯一需要注意的就是 `SignUtil` 类中的成员变量 `token`，这里赋予什么值，在接口配置信息中的 `Token` 就要填写什么值，两边保持一致即可，没有其他要求，建议用项目名称、公司名称缩写等，我在这里用的是项目名称 `weixinCourse`。

最后再来看一下 `web.xml` 中，`CoreServlet` 是怎么配置的，`web.xml` 中的配置代码如下：

[html] view plaincopy

```
1.  <?xml version="1.0" encoding="UTF-8"?>  
2.  <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"  
3.      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
4.      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
5.      http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">  
6.      <servlet>  
7.          <servlet-name>coreServlet</servlet-name>  
8.          <servlet-class>  
9.              org.liufeng.course.servlet.CoreServlet  
10.         </servlet-class>  
11.     </servlet>  
12.  
13.     <!-- url-pattern 中配置的/coreServlet 用于指定该 Servlet 的访问路径 -->  
14.     <servlet-mapping>  
15.         <servlet-name>coreServlet</servlet-name>  
16.         <url-pattern>/coreServlet</url-pattern>  
17.     </servlet-mapping>  
18.  
19.     <welcome-file-list>  
20.         <welcome-file>index.jsp</welcome-file>  
21.     </welcome-file-list>  
22. </web-app>
```

到这里，所有编码都完成了，就是这么简单。接下来就是将工程发布到公网服务器上，如果没有公网服务器环境，可以去了解 `BAE`、`SAE` 或 `阿里云`。发布到服务器上后，我们在浏览器里访问 `CoreServlet`，如果看到如下界面就表示我们的代码没有问题：

HTTP ERROR 500

Problem accessing /coreServlet. Reason:

Server Error

Caused by:

```
java.lang.NullPointerException
    at java.lang.String.compareTo(String.java:1176)
    at java.lang.String.compareTo(String.java:109)
    at java.util.Arrays.mergeSort(Arrays.java:1157)
    at java.util.Arrays.sort(Arrays.java:1092)
    at org.liufeng.course.util.SignUtil.checkSignature(SignUtil.java:28)
    at org.liufeng.course.servlet.CoreServlet.doGet(CoreServlet.java:37)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:734)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:847)
    at org.eclipse.jetty.servlet.ServletHolder.handle(ServletHolder.java:550)
    at org.eclipse.jetty.servlet.ServletHandler.doHandle(ServletHandler.java:489)
    at org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:119)
```

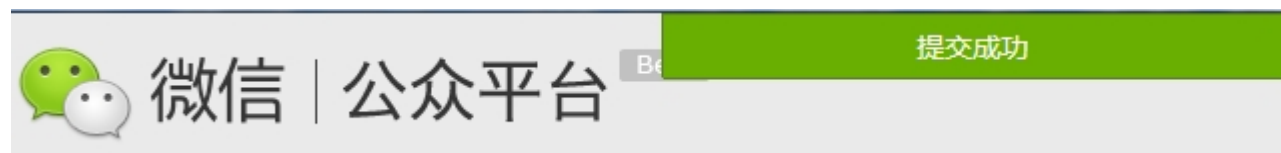
啊，代码都报空指针异常了还说证明没问题？那当然了，因为直接在地址栏访问 `coreServlet`，就相当于提交的是 `GET` 请求，而我们什么参数都没有传，在验证的时候当然会报空指针异常。

接下来，把 `coreServlet` 的访问路径拷贝下来，再回到微信公众平台的接入配置信息界面，将 `coreServlet` 的访问路径粘贴到 `URL` 中，并将 `SignUtil` 类中指定的 `token` 值 `weixinCourse` 填入到 `Token` 中，填写后的结果如下图所示：

URL	<input type="text" value="http://0.xiaoqrobot.duapp.com/coreServlet"/>
Token	<input type="text" value="weixinCourse"/>
	什么是Token?
<input type="button" value="提交"/>	

我在写这篇教程的时候是使用的 `BAE` 环境，如果想学习微信公众帐号开发又没有公网服务器环境的，建议可以试试，注册使用都很方便，如果有问题我们还可以交流。

接着点击“提交”，如果程序写的没问题，并且 `URL`、`Token` 都填写正确，可以在页面最上方看到“提交成功”的提示，并会再次跳转到开发模式设置界面，而且能够看到“你已成为开发者”的提示，如下图所示：





你已成为开发者

接口配置信息 [修改](#)

URL : http://0.xiaoqrobot.duapp.com/coreServlet
Token : weixinCourse

启用开发模式（下）

这个时候就已经成为开发者了，百般周折啊，哈哈，到这里还没有完哦，还有最后一步工作就是将开发模式开启。将右上角的开发模式开关由“关闭”切换到“开启”，如下图所示：



到这里，接口配置、开发模式的开启就都完成了，本章节的内容也就讲到这里。[接下来要章节要讲的就是如何接收、处理、响应由微信服务器转发的用户发送给公众帐号的消息，也就是完成 CoreServlet 中 doPost 方法的编写。](#)

微信公众平台开发教程第 4 篇-消息及消息处理工具的封装

工欲善其事必先利其器！本篇内容主要讲解如何将微信公众平台定义的消息及消息相关的操作封装成工具类，方便后期的使用。这里需要明确的是消息其实是由用户发给你的公众帐号

的，消息先被微信平台接收到，然后微信平台会将该消息转给你在开发模式接口配置中指定的 URL 地址。

微信公众平台消息接口

要接收微信平台发送的消息，我们需要先熟悉微信公众平台 API 中消息接口部分，[点此进入](#)，点击后将进入到消息接口指南部分，如下图所示：



在上图左侧可以看到微信公众平台目前开放的接口有三种：消息接口、通用接口和自定义菜单接口。通用接口和自定义菜单接口只有拿到内测资格才能调用，而内测资格的申请也已经关闭了，我们只有期待将来某一天微信会对大众用户开放吧，所以没有内测资格的用户就不要再浪费时间在这两个接口上，只需要用好消息接口就可以了。

消息推送和消息回复

下面将主要介绍消息接口。对于消息的接收、响应我们只需要关注上图中的“4 消息推送”和“5 消息回复”就足够了。

我们先来了解接口中的“消息推送”指的是什么，点击“4 消息推送”，可以看到接口中的“消息推送”指的是“当普通用户向公众帐号发消息时，微信服务器将 POST 该消息到填写的 URL

上”，即这里定义的是用户能够发送哪些类型的消息、消息有哪些字段、消息被微信服务器以什么方式转发给我们的公众帐号后台。

消息推送

当普通微信用户向公众账号发消息时，微信服务器将POST该消息到填写的URL上。：

消息推送中定义了我们将会接收到的消息类型有 5 种：文本消息、图片消息、地理位置消息、链接消息和事件推送，其实语音消息我们也能够接收到的，只不过拿不到具体的语音文件而以（需要内测资格才能够获取语音文件）。

接口中的“消息回复”定义了我们能回复给用户的消息类型、消息字段和消息格式，微信公众平台的接口指南中是这样描述的：

消息回复

对于每一个POST请求，开发者在响应包中返回特定xml结构，对该消息进行响应（现支持回复文本、图文、语音、视频、音乐和对收到的消息进行星标操作）。

微信服务器在五秒内收不到响应会断掉连接。

上面说到我们能回复给用户的消息有 5 种，但目前在开发模式下能回复的消息只有 3 种：文本消息、音乐消息和图文消息，而语音消息和视频消息目前只能在编辑模式下使用。

消息的封装

接下来要做的就是将消息推送（请求）、消息回复（响应）中定义的消息进行封装，建立与之对应的 Java 类（Java 是一门面向对象的编程语言，封装后使用起来更方便），下面的请求消息是指消息推送中定义的消息，响应消息指消息回复中定义的消息。

请求消息的基类

把消息推送中定义的所有消息都有的字段提取出来，封装成一个基类，这些公有的字段包括：ToUserName（开发者微信号）、FromUserName（发送方帐号，OPEN_ID）、CreateTime（消息的创建时间）、MsgType（消息类型）、MsgId（消息 ID），封装后基类 `org.liufeng.course.message.req.BaseMessage` 的代码如下：

[java] view plaincopy

```
1. package org.liufeng.course.message.req;
2.
3. /**
4.  * 消息基类（普通用户 -> 公众帐号）
5.  *
6.  * @author liufeng
7.  * @date 2013-05-19
```

```
8.  */
9.  public class BaseMessage {
10.     // 开发者微信号
11.     private String ToUserName;
12.     // 发送方帐号（一个 OpenID）
13.     private String FromUserName;
14.     // 消息创建时间 （整型）
15.     private long CreateTime;
16.     // 消息类型（text/image/location/link）
17.     private String MsgType;
18.     // 消息 id, 64 位整型
19.     private long MsgId;
20.
21.     public String getToUserName() {
22.         return ToUserName;
23.     }
24.
25.     public void setToUserName(String toUserName) {
26.         ToUserName = toUserName;
27.     }
28.
29.     public String getFromUserName() {
30.         return FromUserName;
31.     }
32.
33.     public void setFromUserName(String fromUserName) {
34.         FromUserName = fromUserName;
35.     }
36.
37.     public long getCreateTime() {
38.         return CreateTime;
39.     }
40.
41.     public void setCreateTime(long createTime) {
42.         CreateTime = createTime;
43.     }
44.
45.     public String getMsgType() {
46.         return MsgType;
47.     }
48.
49.     public void setMsgType(String msgType) {
50.         MsgType = msgType;
51.     }
```

```

52.
53.     public long getMsgId() {
54.         return MsgId;
55.     }
56.
57.     public void setMsgId(long msgId) {
58.         MsgId = msgId;
59.     }
60. }

```

请求消息之文本消息

[java] view plaincopy

```

1.  package org.liufeng.course.message.req;
2.
3.  /**
4.   * 文本消息
5.   *
6.   * @author liufeng
7.   * @date 2013-05-19
8.   */
9.  public class TextMessage extends BaseMessage {
10.     // 消息内容
11.     private String Content;
12.
13.     public String getContent() {
14.         return Content;
15.     }
16.
17.     public void setContent(String content) {
18.         Content = content;
19.     }
20. }

```

请求消息之图片消息

[java] view plaincopy

```

1.  package org.liufeng.course.message.req;
2.
3.  /**
4.   * 图片消息
5.   *
6.   * @author liufeng
7.   * @date 2013-05-19
8.   */

```

```

9.  public class ImageMessage extends BaseMessage {
10.     // 图片链接
11.     private String PicUrl;
12.
13.     public String getPicUrl() {
14.         return PicUrl;
15.     }
16.
17.     public void setPicUrl(String picUrl) {
18.         PicUrl = picUrl;
19.     }
20. }

```

请求消息之地理位置消息

[java] view plaincopy

```

1.  package org.liufeng.course.message.req;
2.
3.  /**
4.   * 地理位置消息
5.   *
6.   * @author liufeng
7.   * @date 2013-05-19
8.   */
9.  public class LocationMessage extends BaseMessage {
10.     // 地理位置维度
11.     private String Location_X;
12.     // 地理位置经度
13.     private String Location_Y;
14.     // 地图缩放大小
15.     private String Scale;
16.     // 地理位置信息
17.     private String Label;
18.
19.     public String getLocation_X() {
20.         return Location_X;
21.     }
22.
23.     public void setLocation_X(String location_X) {
24.         Location_X = location_X;
25.     }
26.
27.     public String getLocation_Y() {
28.         return Location_Y;
29.     }

```



```

30.
31.     public void setLocation_Y(String location_Y) {
32.         Location_Y = location_Y;
33.     }
34.
35.     public String getScale() {
36.         return Scale;
37.     }
38.
39.     public void setScale(String scale) {
40.         Scale = scale;
41.     }
42.
43.     public String getLabel() {
44.         return Label;
45.     }
46.
47.     public void setLabel(String label) {
48.         Label = label;
49.     }
50. }

```

请求消息之链接消息

[java] view plaincopy

```

1.  package org.liufeng.course.message.req;
2.
3.  /**
4.   * 链接消息
5.   *
6.   * @author liufeng
7.   * @date 2013-05-19
8.   */
9.  public class LinkMessage extends BaseMessage {
10.     // 消息标题
11.     private String Title;
12.     // 消息描述
13.     private String Description;
14.     // 消息链接
15.     private String Url;
16.
17.     public String getTitle() {
18.         return Title;
19.     }
20.

```

```

21.     public void setTitle(String title) {
22.         Title = title;
23.     }
24.
25.     public String getDescription() {
26.         return Description;
27.     }
28.
29.     public void setDescription(String description) {
30.         Description = description;
31.     }
32.
33.     public String getUrl() {
34.         return Url;
35.     }
36.
37.     public void setUrl(String url) {
38.         Url = url;
39.     }
40. }

```

请求消息之语音消息

[java] view plaincopy

```

1.  package org.liufeng.course.message.req;
2.
3.  /**
4.   * 音频消息
5.   *
6.   * @author liufeng
7.   * @date 2013-05-19
8.   */
9.  public class VoiceMessage extends BaseMessage {
10.     // 媒体 ID
11.     private String MediaId;
12.     // 语音格式
13.     private String Format;
14.
15.     public String getMediaId() {
16.         return MediaId;
17.     }
18.
19.     public void setMediaId(String mediaId) {
20.         MediaId = mediaId;
21.     }

```

```

22.
23.     public String getFormat() {
24.         return Format;
25.     }
26.
27.     public void setFormat(String format) {
28.         Format = format;
29.     }
30. }

```

响应消息的基类

同样，把消息回复中定义的所有消息都有的字段提取出来，封装成一个基类，这些公有的字段包括：ToUserName（接收方帐号，用户的 OPEN_ID）、FromUserName（开发者的微信号）、CreateTime（消息的创建时间）、MsgType（消息类型）、FuncFlag（消息的星标标识），封装后基类 org.liufeng.course.message.resp.BaseMessage 的代码如下：

[java] view plaincopy

```

1.  package org.liufeng.course.message.resp;
2.
3.  /**
4.   * 消息基类（公众帐号 -> 普通用户）
5.   *
6.   * @author liufeng
7.   * @date 2013-05-19
8.   */
9.  public class BaseMessage {
10.     // 接收方帐号（收到的 OpenID）
11.     private String ToUserName;
12.     // 开发者微信号
13.     private String FromUserName;
14.     // 消息创建时间 （整型）
15.     private long CreateTime;
16.     // 消息类型（text/music/news）
17.     private String MsgType;
18.     // 位 0x0001 被标志时，星标刚收到的消息
19.     private int FuncFlag;
20.
21.     public String getToUserName() {
22.         return ToUserName;
23.     }
24.
25.     public void setToUserName(String toUserName) {
26.         ToUserName = toUserName;
27.     }

```

```

28.
29.     public String getFromUserName() {
30.         return FromUserName;
31.     }
32.
33.     public void setFromUserName(String fromUserName) {
34.         FromUserName = fromUserName;
35.     }
36.
37.     public long getCreateTime() {
38.         return CreateTime;
39.     }
40.
41.     public void setCreateTime(long createTime) {
42.         CreateTime = createTime;
43.     }
44.
45.     public String getMsgType() {
46.         return MsgType;
47.     }
48.
49.     public void setMsgType(String msgType) {
50.         MsgType = msgType;
51.     }
52.
53.     public int getFuncFlag() {
54.         return FuncFlag;
55.     }
56.
57.     public void setFuncFlag(int funcFlag) {
58.         FuncFlag = funcFlag;
59.     }
60. }

```

响应消息之文本消息

[java] view plaincopy

```

1.  package org.liufeng.course.message.resp;
2.
3.  /**
4.   * 文本消息
5.   *
6.   * @author liufeng
7.   * @date 2013-05-19
8.   */

```

```

9.  public class TextMessage extends BaseMessage {
10.     // 回复的消息内容
11.     private String Content;
12.
13.     public String getContent() {
14.         return Content;
15.     }
16.
17.     public void setContent(String content) {
18.         Content = content;
19.     }
20. }

```

响应消息之音乐消息

[java] view plaincopy

```

1.  package org.liufeng.course.message.resp;
2.
3.  /**
4.   * 音乐消息
5.   *
6.   * @author liufeng
7.   * @date 2013-05-19
8.   */
9.  public class MusicMessage extends BaseMessage {
10.     // 音乐
11.     private Music Music;
12.
13.     public Music getMusic() {
14.         return Music;
15.     }
16.
17.     public void setMusic(Music music) {
18.         Music = music;
19.     }
20. }

```

音乐消息中 Music 类的定义

[java] view plaincopy

```

1.  package org.liufeng.course.message.resp;
2.
3.  /**
4.   * 音乐 model
5.   *

```

```
6.  * @author liufeng
7.  * @date 2013-05-19
8.  */
9.  public class Music {
10.     // 音乐名称
11.     private String Title;
12.     // 音乐描述
13.     private String Description;
14.     // 音乐链接
15.     private String MusicUrl;
16.     // 高质量音乐链接, WIFI 环境优先使用该链接播放音乐
17.     private String HQMusicUrl;
18.
19.     public String getTitle() {
20.         return Title;
21.     }
22.
23.     public void setTitle(String title) {
24.         Title = title;
25.     }
26.
27.     public String getDescription() {
28.         return Description;
29.     }
30.
31.     public void setDescription(String description) {
32.         Description = description;
33.     }
34.
35.     public String getMusicUrl() {
36.         return MusicUrl;
37.     }
38.
39.     public void setMusicUrl(String musicUrl) {
40.         MusicUrl = musicUrl;
41.     }
42.
43.     public String getHQMusicUrl() {
44.         return HQMusicUrl;
45.     }
46.
47.     public void setHQMusicUrl(String musicUrl) {
48.         HQMusicUrl = musicUrl;
49.     }
```

```
50.  
51. }
```

响应消息之图文消息

[java] view plaincopy

```
1. package org.liufeng.course.message.resp;  
2.  
3. import java.util.List;  
4.  
5. /**  
6.  * 文本消息  
7.  *  
8.  * @author liufeng  
9.  * @date 2013-05-19  
10. */  
11. public class NewsMessage extends BaseMessage {  
12.     // 图文消息个数，限制为 10 条以内  
13.     private int ArticleCount;  
14.     // 多条图文消息信息，默认第一个 item 为大图  
15.     private List<Article> Articles;  
16.  
17.     public int getArticleCount() {  
18.         return ArticleCount;  
19.     }  
20.  
21.     public void setArticleCount(int articleCount) {  
22.         ArticleCount = articleCount;  
23.     }  
24.  
25.     public List<Article> getArticles() {  
26.         return Articles;  
27.     }  
28.  
29.     public void setArticles(List<Article> articles) {  
30.         Articles = articles;  
31.     }  
32. }
```

图文消息中 Article 类的定义

[java] view plaincopy

```
1. package org.liufeng.course.message.resp;  
2.  
3. /**
```



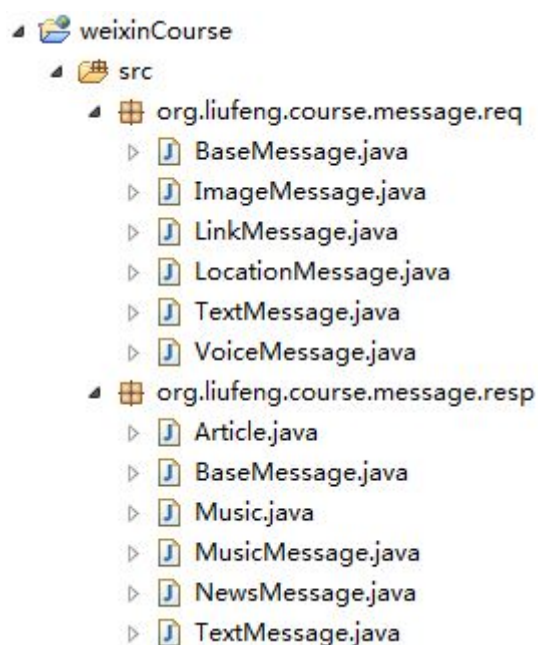
```
4.  * 图文 model
5.  *
6.  * @author liufeng
7.  * @date 2013-05-19
8.  */
9.  public class Article {
10.     // 图文消息名称
11.     private String Title;
12.     // 图文消息描述
13.     private String Description;
14.     // 图片链接, 支持 JPG、PNG 格式, 较好的效果为大图 640*320, 小图 80*80, 限制图片
    链接的域名需要与开发者填写的基本资料中的 Url 一致
15.     private String PicUrl;
16.     // 点击图文消息跳转链接
17.     private String Url;
18.
19.     public String getTitle() {
20.         return Title;
21.     }
22.
23.     public void setTitle(String title) {
24.         Title = title;
25.     }
26.
27.     public String getDescription() {
28.         return null == Description ? "" : Description;
29.     }
30.
31.     public void setDescription(String description) {
32.         Description = description;
33.     }
34.
35.     public String getPicUrl() {
36.         return null == PicUrl ? "" : PicUrl;
37.     }
38.
39.     public void setPicUrl(String picUrl) {
40.         PicUrl = picUrl;
41.     }
42.
43.     public String getUrl() {
44.         return null == Url ? "" : Url;
45.     }
46.
```

```

47.     public void setUrl(String url) {
48.         Url = url;
49.     }
50.
51. }

```

全部消息封装完成后，Eclipse 工程中关于消息部分的结构应该与下图保持一致，如果不一致的（类名、属性名称不一致的）请检查后调整一致，因为后面的章节还要介绍如何将微信开发中通用的类方法、与业务无关的工具类封装打成 jar 包，以后再做微信项目只需要引入该 jar 包即可，这种工作做一次就可以了。



如何解析请求消息？

接下来解决请求消息的解析问题。微信服务器会将用户的请求通过 `doPost` 方法发送给我们，让我们再来回顾下上一章节已经写好的 `doPost` 方法的定义：

[java] view plaincopy

```

1.  /**
2.     * 处理微信服务器发来的消息
3.     */
4.     public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
5.         // TODO 消息的接收、处理、响应
6.     }

```

`doPost` 方法有两个参数，`request` 中封装了请求相关的所有内容，可以从 `request` 中取出微信服务器发来的消息；而通过 `response` 我们可以对接收到的消息进行响应，即发送消息。

那么如何解析请求消息的问题也就转化为如何从 `request` 中得到微信服务器发送给我们的 `xml` 格式的消息了。这里我们借助于开源框架 `dom4j` 去解析 `xml`（这里使用的是 `dom4j-1.6.1.jar`），然后将解析得到的结果存入 `HashMap`，解析请求消息的方法如下：

[java] view plaincopy

```
1.  /**
2.   * 解析微信发来的请求（XML）
3.   *
4.   * @param request
5.   * @return
6.   * @throws Exception
7.   */
8.  @SuppressWarnings("unchecked")
9.  public static Map<String, String> parseXml(HttpServletRequest request) throws Exception {
10.     // 将解析结果存储在 HashMap 中
11.     Map<String, String> map = new HashMap<String, String>();
12.
13.     // 从 request 中取得输入流
14.     InputStream inputStream = request.getInputStream();
15.     // 读取输入流
16.     SAXReader reader = new SAXReader();
17.     Document document = reader.read(inputStream);
18.     // 得到 xml 根元素
19.     Element root = document.getRootElement();
20.     // 得到根元素的所有子节点
21.     List<Element> elementList = root.elements();
22.
23.     // 遍历所有子节点
24.     for (Element e : elementList)
25.         map.put(e.getName(), e.getText());
26.
27.     // 释放资源
28.     inputStream.close();
29.     inputStream = null;
30.
31.     return map;
32. }
```

如何将响应消息转换成 `xml` 返回？

我们先前已经将响应消息封装成了 `Java` 类，方便我们在代码中使用。那么，请求接收成功、处理完成后，该如何将消息返回呢？这里就涉及到如何将响应消息转换成 `xml` 返回的问题，

这里我们将采用开源框架 **xstream** 来实现 **Java** 类到 **xml** 的转换（这里使用的是 **xstream-1.3.1.jar**），代码如下：

[java] view plaincopy

```
1.  /**
2.   * 文本消息对象转换成 xml
3.   *
4.   * @param textMessage 文本消息对象
5.   * @return xml
6.   */
7.  public static String textMessageToXml(TextMessage textMessage) {
8.      xstream.alias("xml", textMessage.getClass());
9.      return xstream.toXML(textMessage);
10. }
11.
12. /**
13.  * 音乐消息对象转换成 xml
14.  *
15.  * @param musicMessage 音乐消息对象
16.  * @return xml
17.  */
18. public static String musicMessageToXml(MusicMessage musicMessage) {
19.     xstream.alias("xml", musicMessage.getClass());
20.     return xstream.toXML(musicMessage);
21. }
22.
23. /**
24.  * 图文消息对象转换成 xml
25.  *
26.  * @param newsMessage 图文消息对象
27.  * @return xml
28.  */
29. public static String newsMessageToXml(NewsMessage newsMessage) {
30.     xstream.alias("xml", newsMessage.getClass());
31.     xstream.alias("item", new Article().getClass());
32.     return xstream.toXML(newsMessage);
33. }
34.
35. /**
36.  * 扩展 xstream，使其支持 CDATA 块
37.  *
38.  * @date 2013-05-19
39.  */
```

```

40. private static XStream xstream = new XStream(new XppDriver() {
41.     public HierarchicalStreamWriter createWriter(Writer out) {
42.         return new PrettyPrintWriter(out) {
43.             // 对所有 xml 节点的转换都增加 CDATA 标记
44.             boolean cdata = true;
45.
46.             @SuppressWarnings("unchecked")
47.             public void startNode(String name, Class clazz) {
48.                 super.startNode(name, clazz);
49.             }
50.
51.             protected void writeText(QuickWriter writer, String text) {
52.                 if (cdata) {
53.                     writer.write("<![CDATA[");
54.                     writer.write(text);
55.                     writer.write("]>");
56.                 } else {
57.                     writer.write(text);
58.                 }
59.             }
60.         };
61.     }
62. });

```

说明：由于 **xstream** 框架本身并不支持 **CDATA** 块的生成，40~62 行代码是对 **xstream** 做了扩展，使其支持在生成 **xml** 各元素值时添加 **CDATA** 块。

消息处理工具的封装

知道怎么解析请求消息，也知道如何将响应消息转化成 **xml** 了，接下来就是将消息相关的处理方法全部封装到工具类 **MessageUtil** 中，该类的完整代码如下：

[java] view plaincopy

```

1. package org.liufeng.course.util;
2.
3. import java.io.InputStream;
4. import java.io.Writer;
5. import java.util.HashMap;
6. import java.util.List;
7. import java.util.Map;
8.
9. import javax.servlet.http.HttpServletRequest;
10.
11. import org.dom4j.Document;
12. import org.dom4j.Element;

```

```
13. import org.dom4j.io.SAXReader;
14. import org.liufeng.course.message.resp.Article;
15. import org.liufeng.course.message.resp.MusicMessage;
16. import org.liufeng.course.message.resp.NewsMessage;
17. import org.liufeng.course.message.resp.TextMessage;
18.
19. import com.thoughtworks.xstream.XStream;
20. import com.thoughtworks.xstream.core.util.QuickWriter;
21. import com.thoughtworks.xstream.io.HierarchicalStreamWriter;
22. import com.thoughtworks.xstream.io.xml.PrettyPrintWriter;
23. import com.thoughtworks.xstream.io.xml.XppDriver;
24.
25. /**
26.  * 消息工具类
27.  *
28.  * @author liufeng
29.  * @date 2013-05-19
30.  */
31. public class MessageUtil {
32.
33.     /**
34.      * 返回消息类型：文本
35.      */
36.     public static final String RESP_MESSAGE_TYPE_TEXT = "text";
37.
38.     /**
39.      * 返回消息类型：音乐
40.      */
41.     public static final String RESP_MESSAGE_TYPE_MUSIC = "music";
42.
43.     /**
44.      * 返回消息类型：图文
45.      */
46.     public static final String RESP_MESSAGE_TYPE_NEWS = "news";
47.
48.     /**
49.      * 请求消息类型：文本
50.      */
51.     public static final String REQ_MESSAGE_TYPE_TEXT = "text";
52.
53.     /**
54.      * 请求消息类型：图片
55.      */
56.     public static final String REQ_MESSAGE_TYPE_IMAGE = "image";
```

```
57.
58.     /**
59.      * 请求消息类型: 链接
60.      */
61.     public static final String REQ_MESSAGE_TYPE_LINK = "link";
62.
63.     /**
64.      * 请求消息类型: 地理位置
65.      */
66.     public static final String REQ_MESSAGE_TYPE_LOCATION = "location";
67.
68.     /**
69.      * 请求消息类型: 音频
70.      */
71.     public static final String REQ_MESSAGE_TYPE_VOICE = "voice";
72.
73.     /**
74.      * 请求消息类型: 推送
75.      */
76.     public static final String REQ_MESSAGE_TYPE_EVENT = "event";
77.
78.     /**
79.      * 事件类型: subscribe(订阅)
80.      */
81.     public static final String EVENT_TYPE_SUBSCRIBE = "subscribe";
82.
83.     /**
84.      * 事件类型: unsubscribe(取消订阅)
85.      */
86.     public static final String EVENT_TYPE_UNSUBSCRIBE = "unsubscribe";
87.
88.     /**
89.      * 事件类型: CLICK(自定义菜单点击事件)
90.      */
91.     public static final String EVENT_TYPE_CLICK = "CLICK";
92.
93.     /**
94.      * 解析微信发来的请求 (XML)
95.      *
96.      * @param request
97.      * @return
98.      * @throws Exception
99.      */
100.    @SuppressWarnings("unchecked")
```



```

101.     public static Map<String, String> parseXml(HttpServletRequest request)
        throws Exception {
102.         // 将解析结果存储在 HashMap 中
103.         Map<String, String> map = new HashMap<String, String>();
104.
105.         // 从 request 中取得输入流
106.         InputStream inputStream = request.getInputStream();
107.         // 读取输入流
108.         SAXReader reader = new SAXReader();
109.         Document document = reader.read(inputStream);
110.         // 得到 xml 根元素
111.         Element root = document.getRootElement();
112.         // 得到根元素的所有子节点
113.         List<Element> elementList = root.elements();
114.
115.         // 遍历所有子节点
116.         for (Element e : elementList)
117.             map.put(e.getName(), e.getText());
118.
119.         // 释放资源
120.         inputStream.close();
121.         inputStream = null;
122.
123.         return map;
124.     }
125.
126.     /**
127.      * 文本消息对象转换成 xml
128.      *
129.      * @param textMessage 文本消息对象
130.      * @return xml
131.      */
132.     public static String textMessageToXml(TextMessage textMessage) {
133.         xstream.alias("xml", textMessage.getClass());
134.         return xstream.toXML(textMessage);
135.     }
136.
137.     /**
138.      * 音乐消息对象转换成 xml
139.      *
140.      * @param musicMessage 音乐消息对象
141.      * @return xml
142.      */
143.     public static String musicMessageToXml(MusicMessage musicMessage) {

```

```

144.     xstream.alias("xml", musicMessage.getClass());
145.     return xstream.toXML(musicMessage);
146. }
147.
148. /**
149.  * 图文消息对象转换成 xml
150.  *
151.  * @param newsMessage 图文消息对象
152.  * @return xml
153.  */
154. public static String newsMessageToXml(NewsMessage newsMessage) {
155.     xstream.alias("xml", newsMessage.getClass());
156.     xstream.alias("item", new Article().getClass());
157.     return xstream.toXML(newsMessage);
158. }
159.
160. /**
161.  * 扩展 xstream, 使其支持 CDATA 块
162.  *
163.  * @date 2013-05-19
164.  */
165. private static XStream xstream = new XStream(new XppDriver() {
166.     public HierarchicalStreamWriter createWriter(Writer out) {
167.         return new PrettyPrintWriter(out) {
168.             // 对所有 xml 节点的转换都增加 CDATA 标记
169.             boolean cdata = true;
170.
171.             @SuppressWarnings("unchecked")
172.             public void startNode(String name, Class clazz) {
173.                 super.startNode(name, clazz);
174.             }
175.
176.             protected void writeText(QuickWriter writer, String text) {
177.
178.                 if (cdata) {
179.                     writer.write("<![CDATA[");
180.                     writer.write(text);
181.                     writer.write("]]>");
182.                 } else {
183.                     writer.write(text);
184.                 }
185.             }
186.         };
187.     }
188. }

```

```
187.     });
188. }
```

OK，到这里关于消息及消息处理工具的封装就讲到这里，其实就是对请求消息/响应消息建立了与之对应的 **Java** 类、对 **xml** 消息进行解析、将响应消息的 **Java** 对象转换成 **xml**。下一篇讲会介绍如何利用上面封装好的工具识别用户发送的消息类型，并做出正确的响应。

微信公众帐号开发教程第 5 篇-各种消息的接收与响应

前一篇文章里我们已经把微信公众平台接口中消息及相关操作都进行了封装，本章节将主要介绍如何接收微信服务器发送的消息并做出响应。

明确在哪接收消息

从微信公众平台接口消息指南中可以了解到，当用户向公众帐号发消息时，微信服务器会将消息通过 **POST** 方式提交给我们在接口配置信息中填写的 **URL**，而我们就需要在 **URL** 所指向的请求处理类 **CoreServlet** 的 **doPost** 方法中接收消息、处理消息和响应消息。

接收、处理、响应消息

下面先来看我已经写好的 **CoreServlet** 的完整代码：

[java] view plaincopy

```
1. package org.liufeng.course.servlet;
2.
3. import java.io.IOException;
4. import java.io.PrintWriter;
5.
6. import javax.servlet.ServletException;
7. import javax.servlet.http.HttpServlet;
8. import javax.servlet.http.HttpServletRequest;
9. import javax.servlet.http.HttpServletResponse;
10.
11. import org.liufeng.course.service.CoreService;
12. import org.liufeng.course.util.SignUtil;
13.
14. /**
15.  * 核心请求处理类
16.  *
17.  * @author liufeng
18.  * @date 2013-05-18
19.  */
```

```
20. public class CoreServlet extends HttpServlet {
21.     private static final long serialVersionUID = 4440739483644821986L;
22.
23.     /**
24.      * 确认请求来自微信服务器
25.      */
26.     public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
27.         // 微信加密签名
28.         String signature = request.getParameter("signature");
29.         // 时间戳
30.         String timestamp = request.getParameter("timestamp");
31.         // 随机数
32.         String nonce = request.getParameter("nonce");
33.         // 随机字符串
34.         String echostr = request.getParameter("echostr");
35.
36.         PrintWriter out = response.getWriter();
37.         // 通过检验 signature 对请求进行校验, 若校验成功则原样返回 echostr, 表示接入成功, 否则接入失败
38.         if (SignUtil.checkSignature(signature, timestamp, nonce)) {
39.             out.print(echostr);
40.         }
41.         out.close();
42.         out = null;
43.     }
44.
45.     /**
46.      * 处理微信服务器发来的消息
47.      */
48.     public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
49.         // 将请求、响应的编码均设置为 UTF-8 (防止中文乱码)
50.         request.setCharacterEncoding("UTF-8");
51.         response.setCharacterEncoding("UTF-8");
52.
53.         // 调用核心业务类接收消息、处理消息
54.         String respMessage = CoreService.processRequest(request);
55.
56.         // 响应消息
57.         PrintWriter out = response.getWriter();
58.         out.print(respMessage);
59.         out.close();
60.     }
```

```
61.  
62. }
```

代码说明：

1) 第 51 行代码：微信服务器 POST 消息时用的是 UTF-8 编码，在接收时也要用同样的编码，否则中文会乱码；

2) 第 52 行代码：在响应消息（回复消息给用户）时，也将编码方式设置为 UTF-8，原理同上；

3) 第 54 行代码：调用 CoreService 类的 processRequest 方法接收、处理消息，并得到处理结果；

4) 第 57~59 行：调用 response.getWriter().write()方法将消息的处理结果返回给用户

从 doPost 方法的实现可以看到，它是通过调用 CoreService 类的 processRequest 方法接收、处理消息的，这样做的目的是为了了解耦，即业务相关的操作都不在 Servlet 里处理，而是完全交由业务核心类 CoreService 去做。下面来看 CoreService 类的代码实现：

[java] view plaincopy

```
1. package org.liufeng.course.service;  
2.  
3. import java.util.Date;  
4. import java.util.Map;  
5. import javax.servlet.http.HttpServletRequest;  
6. import org.liufeng.course.message.resp.TextMessage;  
7. import org.liufeng.course.util.MessageUtil;  
8.  
9. /**  
10.  * 核心服务类  
11.  *  
12.  * @author liufeng  
13.  * @date 2013-05-20  
14.  */  
15. public class CoreService {  
16.     /**  
17.      * 处理微信发来的请求  
18.      *  
19.      * @param request  
20.      * @return  
21.      */  
22.     public static String processRequest(HttpServletRequest request) {  
23.         String respMessage = null;  
24.         try {  
25.             // 默认返回的文本消息内容  
26.             String respContent = "请求处理异常，请稍候尝试！";
```

```
27.
28.      // xml 请求解析
29.      Map<String, String> requestMap = MessageUtil.parseXml(request);
30.
31.      // 发送方帐号 (open_id)
32.      String fromUserName = requestMap.get("FromUserName");
33.      // 公众帐号
34.      String toUserName = requestMap.get("ToUserName");
35.      // 消息类型
36.      String msgType = requestMap.get("MsgType");
37.
38.      // 回复文本消息
39.      TextMessage textMessage = new TextMessage();
40.      textMessage.setToUserName(fromUserName);
41.      textMessage.setFromUserName(toUserName);
42.      textMessage.setCreateTime(new Date().getTime());
43.      textMessage.setMsgType(MessageUtil.RESP_MESSAGE_TYPE_TEXT);
44.      textMessage.setFuncFlag(0);
45.
46.      // 文本消息
47.      if (msgType.equals(MessageUtil.REQ_MESSAGE_TYPE_TEXT)) {
48.          respContent = "您发送的是文本消息！";
49.      }
50.      // 图片消息
51.      else if (msgType.equals(MessageUtil.REQ_MESSAGE_TYPE_IMAGE)) {
52.          respContent = "您发送的是图片消息！";
53.      }
54.      // 地理位置消息
55.      else if (msgType.equals(MessageUtil.REQ_MESSAGE_TYPE_LOCATION))
56.      {
57.          respContent = "您发送的是地理位置消息！";
58.      }
59.      // 链接消息
60.      else if (msgType.equals(MessageUtil.REQ_MESSAGE_TYPE_LINK)) {
61.          respContent = "您发送的是链接消息！";
62.      }
63.      // 音频消息
64.      else if (msgType.equals(MessageUtil.REQ_MESSAGE_TYPE_VOICE)) {
65.          respContent = "您发送的是音频消息！";
66.      }
67.      // 事件推送
68.      else if (msgType.equals(MessageUtil.REQ_MESSAGE_TYPE_EVENT)) {
69.          // 事件类型
```

```

69.         String eventType = requestMap.get("Event");
70.         // 订阅
71.         if (eventType.equals(MessageUtil.EVENT_TYPE_SUBSCRIBE)) {
72.             respContent = "感谢您的关注! ";
73.         }
74.         // 取消订阅
75.         else if (eventType.equals(MessageUtil.EVENT_TYPE_UNSUBSCRIBE)
76. ) {
77.             // TODO 取消订阅后用户再收不到公众号发送的消息, 因此不需要回复
78.             消息
79.         }
80.         // 自定义菜单点击事件
81.         else if (eventType.equals(MessageUtil.EVENT_TYPE_CLICK)) {
82.             // TODO 自定义菜单权没有开放, 暂不处理该类消息
83.         }
84.     }
85.     textMessage.setContent(respContent);
86.     respMessage = MessageUtil.textMessageToXml(textMessage);
87. } catch (Exception e) {
88.     e.printStackTrace();
89. }
90. return respMessage;
91. }
92. }

```

代码说明:

- 1) 第 29 行: 调用消息工具类 `MessageUtil` 解析微信发来的 xml 格式的消息, 解析的结果放在 `HashMap` 里;
- 2) 32~36 行: 从 `HashMap` 中取出消息中的字段;
- 3) 39-44、84 行: 组装要返回的文本消息对象;
- 4) 47~82 行: 演示了如何接收微信发送的各类型的消息, 根据 `MsgType` 判断属于哪种类型的消息;
- 5) 85 行: 调用消息工具类 `MessageUtil` 将要返回的文本消息对象 `TextMessage` 转化成 xml 格式的字符串;

关于事件推送（关注、取消关注、菜单点击）

对于消息类型的判断, 像文本消息、图片消息、地理位置消息、链接消息和语音消息都比较好理解, 有很多刚接触的朋友搞不懂事件推送消息有什么用, 或者不清楚该如何判断用户关注的消息。那我们就专门来看下事件推送, 下图是官方消息接口文档中关于事件推送的说明:

事件推送

事件推送只支持微信4.5版本，目前仅开启自定义菜单接口事件推送。其余功能即将开放，敬请期待。

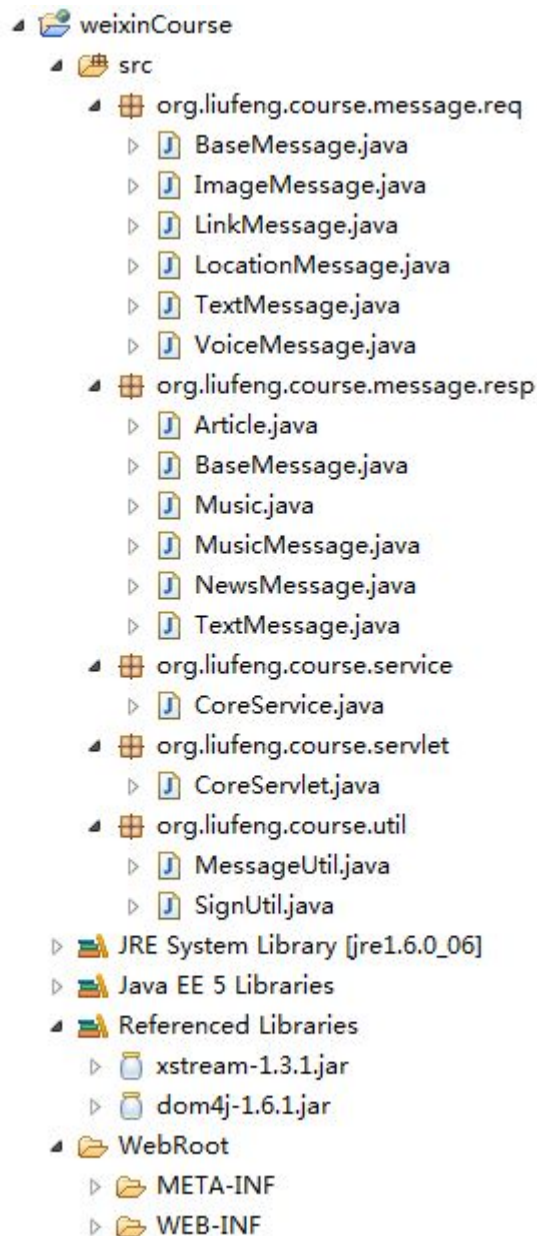
```
<xml> <ToUserName><![CDATA[toUser]]></ToUserName>
<FromUserName><![CDATA[FromUser]]></FromUserName>
<CreateTime>123456789</CreateTime>
<MsgType><![CDATA[event]]></MsgType>
<Event><![CDATA[EVENT]]></Event>
<EventKey><![CDATA[EVENTKEY]]></EventKey>
</xml>
```

参数	描述
ToUserName	接收方微信号
FromUserName	发送方微信号，若为普通用户，则是一个OpenID
CreateTime	消息创建时间
MsgType	消息类型，event
Event	事件类型，subscribe(订阅)、unsubscribe(取消订阅)、CLICK(自定义菜单点击事件)
EventKey	事件KEY值，与自定义菜单接口中KEY值对应

这里我们只要关心两个参数：**MsgType** 和 **Event**。当 **MsgType=event** 时，就表示这是一条事件推送消息；而 **Event** 表示事件类型，包括订阅、取消订阅和自定义菜单点击事件。也就是说，无论用户是关注了公众帐号、取消对公众帐号的关注，还是在使用公众帐号的菜单，微信服务器都会发送一条 **MsgType=event** 的消息给我们，而至于具体这条消息表示关注、取消关注，还是菜单的点击事件，就需要通过 **Event** 的值来判断了。（注意区分 **Event** 和 **event**）

连载五篇教程总结

经过 5 篇的讲解，已经把开发模式启用，接口配置，消息相关工具类的封装，消息的接收与响应全部讲解完了，而且贴上了完整的源代码，相信有一定 **Java** 基础的朋友可以看的明白，能够通过系列文章基本掌握微信公众平台开发的相关技术知识。下面我把目前项目的完整结构贴出，方便大家对照：



微信公众帐号开发教程第 6 篇-文本消息的内容长度限制揭秘

相信不少朋友都遇到过这样的问题：**当发送的文本消息内容过长时，微信将不做任何响应。**那么到底微信允许的文本消息的最大长度是多少呢？我们又该如何计算文本的长度呢？为什么还有些人反应微信好像支持的文本消息最大长度在 1300 多呢？这篇文章会彻底解除大家的疑问。

[接口文档中对消息长度限制为 2048](#)

参数	描述
ToUserName	接收方帐号（收到的OpenID）
FromUserName	开发者微信号
CreateTime	消息创建时间
MsgType	text
Content	回复的消息内容,长度不超过2048字节
FuncFlag	位0x0001被标志时，星标刚收到的消息。

可以看到，接口文档中写的很明确：回复的消息内容长度不超过 2048 字节。那为什么很多人测试反应消息内容长度在 1300 多字节时，微信就不响应了呢？我想这问题应该在这部分人没有搞清楚到底该如何计算文本的字节数。

如何正确计算文本所占字节数

计算文本（字符串）所占字节数，大家第一个想到的应该就是 String 类的 `getBytes()` 方法，该方法返回的是字符串对应的字节数组，再计算数组的 `length` 就能够得到字符串所占字节数。例如：

[java] view plaincopy

```

1. public static void main(String []args) {
2.     // 运行结果: 4
3.     System.out.println("柳峰".getBytes().length);
4. }
```

上面的示例中计算了两个中文所占的字节数为 4，即一个汉字占 2 个字节。真的是这样吗？其实我们忽略了一个问题：对于不同的编码方式，中文所占的字节数也不一样！这到底要怎么办呢？在上面的例子中，我们并没有指定编码方式，那么会使用操作系统所默认的编码方式。先来看我得出的三条结论：

- 1) 如果上面的例子运行在默认编码方式为 ISO8859-1 的操作系统平台上，计算结果是 2；
- 2) 如果上面的例子运行在默认编码方式为 gb2312 或 gbk 的操作系统平台上，计算结果是 4；
- 3) 如果上面的例子运行在默认编码方式为 utf-8 的操作系统平台上，计算结果是 6；

如果真的是这样，是不是意味着 `String.getBytes()` 方法在我们的系统平台上默认采用的是 gb2312 或 gbk 编码方式呢？我们再来看一个例子：

[java] view plaincopy

```

1. public static void main(String []args) throws UnsupportedEncodingException
   {
2.     // 运行结果: 2
3.     System.out.println("柳峰".getBytes("ISO8859-1").length);
4.     // 运行结果: 4
5.     System.out.println("柳峰".getBytes("GB2312").length);
6.     // 运行结果: 4
7.     System.out.println("柳峰".getBytes("GBK").length);
8.     // 运行结果: 6
9.     System.out.println("柳峰".getBytes("UTF-8").length);
10. }

```

这个例子是不是很好地证明了我上面给出的三条结论呢？也就是说采用 **ISO8859-1** 编码方式时，一个中/英文都只占一个字节；采用 **GB2312** 或 **GBK** 编码方式时，一个中文占两个字节；而采用 **UTF-8** 编码方式时，一个中文占三个字节。

微信平台采用的编码方式及字符串所占字节数的计算

那么，在向微信服务器返回消息时，该采用什么编码方式呢？当然是 **UTF-8**，因为我们已经在 `doPost` 方法里采用了如下代码来避免中文乱码了：

[java] view plaincopy

```

1. // 将请求、响应的编码均设置为 UTF-8（防止中文乱码）
2. request.setCharacterEncoding("UTF-8");
3. response.setCharacterEncoding("UTF-8");

```

为了验证我所说了，我写了个例子来测试：

[java] view plaincopy

```

1. private static String getMsgContent() {
2.     StringBuffer buffer = new StringBuffer();
3.     // 每行 70 个汉字，共 682 个汉字加 1 个英文的感叹号
4.     buffer.append("不知道什么时候开始喜欢这里每个夜里都会来这里看你你长得多么美丽
   叫我不能不看你看不到你我就迷失了自己好想牵你的手走过风风雨雨有什么困难我都陪你");
5.     buffer.append("不知道什么时候开始喜欢这里每个夜里都会来这里看你你长得多么美丽
   叫我不能不看你看不到你我就迷失了自己好想牵你的手走过风风雨雨有什么困难我都陪你");
6.     buffer.append("不知道什么时候开始喜欢这里每个夜里都会来这里看你你长得多么美丽
   叫我不能不看你看不到你我就迷失了自己好想牵你的手走过风风雨雨有什么困难我都陪你");
7.     buffer.append("不知道什么时候开始喜欢这里每个夜里都会来这里看你你长得多么美丽
   叫我不能不看你看不到你我就迷失了自己好想牵你的手走过风风雨雨有什么困难我都陪你");
8.     buffer.append("不知道什么时候开始喜欢这里每个夜里都会来这里看你你长得多么美丽
   叫我不能不看你看不到你我就迷失了自己好想牵你的手走过风风雨雨有什么困难我都陪你");
9.     buffer.append("不知道什么时候开始喜欢这里每个夜里都会来这里看你你长得多么美丽
   叫我不能不看你看不到你我就迷失了自己好想牵你的手走过风风雨雨有什么困难我都陪你");

```

```

10.     buffer.append("不知道什么时候开始喜欢这里每个夜里都会来这里看你你长得多么美丽
        叫我不能不看你看不到你我就迷失了自己好想牵你的手走过风风雨雨有什么困难我都陪你");
11.     buffer.append("不知道什么时候开始喜欢这里每个夜里都会来这里看你你长得多么美丽
        叫我不能不看你看不到你我就迷失了自己好想牵你的手走过风风雨雨有什么困难我都陪你");
12.     buffer.append("不知道什么时候开始喜欢这里每个夜里都会来这里看你你长得多么美丽
        叫我不能不看你看不到你我就迷失了自己好想牵你的手走过风风雨雨有什么困难我都陪你");
13.     buffer.append("不知道什么时候开始喜欢这里每个夜里都会来这里看你你长得多么美丽
        叫我不能不看你看不到你我就迷失了自己好想牵!");
14.     return buffer.toString();
15. }
16.
17. public static void main(String []args) throws Exception {
18.     // 采用 gb2312 编码方式时占 1365 个字节
19.     System.out.println(getMsgContent().getBytes("gb2312").length);
20.     // 采用 utf-8 编码方式时占 2047 个字节
21.     System.out.println(getMsgContent().getBytes("utf-8").length);
22. }

```

getMsgContent()方法返回的内容正是微信的文本消息最长能够支持的，即采用 UTF-8 编码方式时，文本消息内容最多支持 2047 个字节，也就是微信公众平台接口文档里所说的回复的消息内容长度不超过 2048 字节，即使是等于 2048 字节也不行，你可以试着将 getMsgContent()方法里的内容多加一个英文符号，这个时候微信就不响应了。

同时，我们也发现，如果采用 gb2312 编码方式来计算 getMsgContent()方法返回的文本所占字节数的结果是 1365，这就是为什么很多朋友都说微信的文本消息最大长度好像只支持 1300 多字节，并不是接口文档中所说的 2048 字节，其实是忽略了编码方式，只是简单的使用了 String 类的 getBytes()方法而不是 getBytes("utf-8")方法去计算所占字节数。

Java 中 utf-8 编码方式时所占字节数的计算方法封装

[java] view plaincopy

```

1.  /**
2.   * 计算采用 utf-8 编码方式时字符串所占字节数
3.   *
4.   * @param content
5.   * @return
6.   */
7.  public static int getByteSize(String content) {
8.      int size = 0;
9.      if (null != content) {
10.         try {
11.             // 汉字采用 utf-8 编码时占 3 个字节
12.             size = content.getBytes("utf-8").length;

```

```
13.         } catch (UnsupportedEncodingException e) {  
14.             e.printStackTrace();  
15.         }  
16.     }  
17.     return size;  
18. }
```

好了，本章节的内容就讲到这里，我想大家通过这篇文章所学到的应该不仅仅是 2047 这个数字，还应该对字符编码方式有一个新的认识。

微信公众帐号开发教程第 7 篇-文本消息中换行符的使用

本篇文章主要介绍在文本消息中使用换行符的好处以及如何使用换行符。

最近一个月虽然抽不出时间写博客，但却一直在认真答复大家提出的问题。收到这么多的回复、关注和答谢，还是蛮有成就感的，让我觉得做这件事越来越有意义，更加坚定了我继续写下去的决心。经过前面六篇文章的讲解，相信在看文章的你，已经掌握了微信公众帐号的基础开发知识（基于 Java），如框架搭建、API 封装、消息接收与回复等；接下来的系列文章将专注于讲解公众帐号开发中的技巧及实用功能的开发（如天气查询、周边搜索、人机对话等）。

使用换行的好处及示例

使用换行的好处无非就是让信息的呈现更加整齐、美观和直观，适当的在文本消息中使用换行符，会让人看了之后感觉很舒服、清晰、明了。下面是公众帐号 **xiaoqrobot** 的主菜单示例，就是合理地使用了换行符，看上去是不是很直观、清爽呢？（什么？觉得很丑？呃，那就算是我自恋吧...）



你可以试想一下，如果这个文本菜单没有使用一个换行符，那会长什么样？

如何在文本消息中使用换行符？

在微信公众帐号的文本消息中，换行符仍然是“`\n`”，下面通过代码来讲解 **xiaoqrobot** 的文本菜单是如何实现的？

[java] view plaincopy

```
1.  /**
2.   * xiaoqrobot 的主菜单
3.   *
4.   * @return
5.   */
6.  public static String getMainMenu() {
7.      StringBuffer buffer = new StringBuffer();
8.      buffer.append("您好，我是小 q，请回复数字选择服务： ").append("\n\n");
9.      buffer.append("1 天气预报").append("\n");
10.     buffer.append("2 公交查询").append("\n");
11.     buffer.append("3 周边搜索").append("\n");
```

```

12.    buffer.append("4 歌曲点播").append("\n");
13.    buffer.append("5 经典游戏").append("\n");
14.    buffer.append("6 美女电台").append("\n");
15.    buffer.append("7 人脸识别").append("\n");
16.    buffer.append("8 聊天唠嗑").append("\n\n");
17.    buffer.append("回复“?”显示此帮助菜单");
18.    return buffer.toString();
19. }

```

怎么样，实现起来是不是很简单呢？

1) 9-16 行就是菜单项，菜单项之间都是用一个换行符分隔；

2) 第 8 行、第 16 号末尾都使用了两个换行符，这样可以把菜单项与其他内容分隔开，更有层次感，看上去也会舒服、直观一点。

可能细心的朋友已经发现了：在截图上，“周边搜索”和“美女电台”后边都有一个“礼物”表情，而代码中并没有看到，这是我专门去掉的，因为我打算后面专门用一篇文章把 QQ 表情的发送、处理、接收讲清楚。

细节决定成败！

微信公众帐号开发教程第 8 篇-文本消息中使用网页超链接

本文主要介绍网页超链接的作用以及如何在文本消息中使用网页超链接。

网页超链接的作用

我想但凡是熟悉 HTML 的朋友，对超链接一定不会陌生。而今天我们要讨论和使用的只是超链接中的其中一种---网页超链接，即使用 HTML 中的<a>标签将某段文字链接到其他网页上去，示例如下：

[html] view plaincopy

```

1.  <a href="http://blog.csdn.net/lyq8479">柳峰的博客</a>

```

上面是一段标准的 HTML 代码，实现了一个网页超链接，即将“柳峰的博客”5 个字链接到了博客主页 URL，当“柳峰的博客”5 个字时，会打开 <http://blog.csdn.net/lyq8479> 所指向的网页。

如何在文本消息中使用网页超链接

其实，不知道如何在文本消息中使用网页超链接的开发者几乎 100%都熟悉 HTML，特别是对 HTML 中的<a>标签再熟悉不过了。那到底在微信公众帐号的文本消息中使用超链接有什么特别之处呢？为什么如此多的朋友都曾经在这个问题上栽过跟头？我们先来看在微信中两种错误使用超链接的方法：

错误用法 1（a 标签的 href 属性值未被引号引起）：

[html] view plaincopy

1. `柳峰的博客`

错误用法 2（a 标签的 href 属性值被单引号引起）：

[html] view plaincopy

1. `柳峰的博客`

在做 Web 开发时，以上两种写法都是可以的，但是放在微信公众帐号的文本消息中，这两种写法都是错误的，网页超链接并不会起作用，而且在 Android 手机上还会将 HTML 代码原样显示出来，如下图所示：

Android 手机上的效果：



iPhone 手机上的效果：



可以看出，在微信上，HTML 的 a 标签属性值不用引号引起，或者使用单引号引起，都是错误的写法（在 iPhone 上，a 标签属性 href 的值用单引号是正常的）。正确的用法是将 a 标签 href 属性的值用双引号引起，代码如下：

[html] view plaincopy

1. `柳峰的博客`

这样在 Android 和 iPhone 手机上，都可以正确显示超链接，并且点击该超链接，会使用微信内置浏览器打开 <http://blog.csdn.net/lyq8479>。

提示：在测试微信公众帐号时，不要只是在自己的手机上测试通过就认为完全没问题了，因为目前微信公众帐号上有好几处在 Android 和 iOS 平台上表现不一致。

微信公众帐号开发教程第 9 篇-QQ 表情的发送与接收

我想大家对 QQ 表情一定不会陌生，一个个小头像极大丰富了聊天的乐趣，使得聊天不再是简单的文字叙述，还能够配上喜、怒、哀、乐等表达人物心情的小图片。本文重点要介绍的内容就是如何在微信公众平台使用 QQ 表情，即在微信公众帐号开发模式下，如何发送 QQ 表情给用户，以及如何识别用户发来的是 QQ 表情。

QQ 表情代码表



















首先需要明确的是：QQ 表情虽然呈现为一张张动态的表情图片，但在微信公众平台的消息接口中却是属于文本消息；也就是说当用户向公众帐号发送 QQ 表情时，公众帐号后台程序接收到的消息类型 `MsgType` 的值为 `text`。只要上面这点能理解了，下面的工作就好开展了。

对于 QQ 表情，发送的是文本消息，而呈现出来却是表情图片，那么每一个 QQ 表情图片一定会有与之相对应的表情代码。下面是我已经整理好的微信公众帐号中使用的 QQ 表情代码对照表：

qq表情图片	文字代码	符号代码	qq表情图片	文字代码	符号代码
	/微笑	/::)		/抓狂	/::Q
	/撇嘴	/::~~		/吐	/::T
	/色	/::B		/偷笑	/,,@P
	/发呆	/::		/愉快	/,,-D
	/得意	/:8-)		/白眼	/::d
	/流泪	/::<		/傲慢	/,,@o
	/害羞	/::S		/饥饿	/::g
	/闭嘴	/::X		/困	/: ~)
	/睡	/::Z		/惊恐	/::!
	/大哭	/::'		/流汗	/::L
	/尴尬	/::		/憨笑	/::>
	/发怒	/::@		/悠闲	/::,@
	/调皮	/::P		/奋斗	/,,@f

	/呲牙	/::D		/咒骂	/::S
	/惊讶	/::O		/疑问	/:?
	/难过	/::(	/嘘	/:,@x
	/酷	/::+		/晕	/:,@@
	/冷汗	/:-b		/疯了	/::8
	/衰	/:,@!		/篮球	/:basketb
	/骷髅	/:!!!		/乒乓	/:oo
	/敲打	/:xx		/咖啡	/:coffee
	/再见	/:bye		/饭	/:eat
	/擦汗	/:wipe		/猪头	/:pig
	/抠鼻	/:dig		/玫瑰	/:rose
	/鼓掌	/:handclap		/凋谢	/:fade
	/糗大了	/:&-(	/嘴唇	/:showlove

	/坏笔	/:B-)		/爱心	/:heart
	/左哼哼	/:<@		/心碎	/:break
	/右哼哼	/:@>		/蛋糕	/:cake
	/哈欠	/::-O		/闪电	/:li
	/鄙视	/:>-		/炸弹	/:bome
	/委屈	/:P-(	/刀	/:kn
	/快哭了	/::'		/足球	/:footb
	/阴险	/:X-)		/瓢虫	/:ladybug
	/亲亲	/::*		/便便	/:shit
	/吓	/:@x		/月亮	/:moon
	/可怜	/:8*		/太阳	/:sun
	/菜刀	/:pd		/礼物	/:gift
	/西瓜	/:<W>		/拥抱	/:hug

	/啤酒	/:beer		/强	/:strong
	/弱	/:weak		/发抖	/:shake
	/握手	/:share		/恼火	/:<O>
	/胜利	/:v		/转圈	/:circle
	/抱拳	/:@)		/磕头	/:k otow
	/勾引	/:jj		/回头	/:turn
	/拳头	/:@@		/跳绳	/:skip
	/差劲	/:bad		/投降	/:oY
	/爱你	/:lyu		/激动	/:#-0
	/NO	/:no		/乱舞	/:hi phot
	/OK	/:ok		/献吻	/:kiss
	/爱情	/:love		/左太极	/:<&
	/飞吻	/:<L>		/右太极	/:&>
	/跳跳	/:jump			

上面一共列出了 105 个 QQ 表情，每个表情都给出了与之相对应的**文字代码**与**符号代码**（也许这两种叫法并不恰当），至于这两种代码怎么来的以及如何使用，下面马上会讲到。

用户向公众帐号发送 QQ 表情

在微信上使用公众帐号时，如何发送 QQ 表情，我想这个很少有人不会的。在输入框旁边有一个笑脸的图片按钮，点击它将会弹出表情选择界面，可选择的表情依次为“QQ 表情”、“符号表情”和“动画表情”。当我们点击选择了某个 QQ 表情后，发现在输入框中会显示该表情的**文字代码**，这里是用一对中括号引起的，如下图所示：



其实，当我们很熟悉要使用 QQ 表情的**文字代码**时，也可以直接在输入框中输入表情的代码，而不需要弹出表情选择框。如下图所示：



从上图可以看出，在输入框中输入“**[呲牙]**”、“**/呲牙**”和“**/::D**”这三种代码的作用一样，都是发送呲牙的 QQ 表情。这个时候，大家再回过头去看文章最开始的 QQ 表情代码对照表，就明白是怎么回事了。

公众帐号向用户发送 QQ 表情

与用户向公众帐号发送 QQ 表情一样，在开发模式下，公众帐号也可以用同样的表情代码（文字代码或符号代码）向用户回复 QQ 表情。代码片段如下：

[java] view plaincopy

```
1. // 文本消息
2. if (msgType.equals(MessageUtil.REQ_MESSAGE_TYPE_TEXT)) {
3.     // 回复文本消息
4.     TextMessage textMessage = new TextMessage();
5.     textMessage.setToUserName(fromUserName);
6.     textMessage.setFromUserName(toUserName);
7.     textMessage.setCreateTime(new Date().getTime());
8.     textMessage.setMsgType(MessageUtil.RESP_MESSAGE_TYPE_TEXT);
9.     textMessage.setFuncFlag(0);
10.    textMessage.setContent("[难过] /难过 /::(");
11.
12.    // 文本消息对象转换成 xml 字符串
13.    respMessage = MessageUtil.textMessageToXml(textMessage);
14. }
```

上面代码片段的作用是：判断发送的消息类型，如果是文本消息（MsgType=text），则回复三个难过的 QQ 表情给用户。可以看出，不管是用户发给公众帐号，还是公众帐号发给用户，都可以使用 QQ 表情的文字代码（如：[难过] /难过）和符号代码（如 /::()）。

公众帐号识别用户发送的 QQ 表情

在掌握了如何发送 QQ 表情后，我们再来看看公众帐号如何识别用户发送的是 QQ 表情。这是什么意思呢？当用户向公众帐号发送一个 QQ 表情，在后台程序中接收到的会是什么值，我们又怎么知道这个值就是一个 QQ 表情。

其实，只要做个简单的测试，比如：将接收到的文本消息输出到日志中（可以用 log4j 或者 System.out.print），不难发现：向公众帐号发送一个 QQ 表情，在后台程序中接收到的是 QQ 表情的符号代码。

下面是我简单封装的一个方法，通过正则表达式实现的，用于判断用户发送的是否是单个 QQ 表情。

[java] view plaincopy

```
1. /**
2.  * 判断是否是 QQ 表情
3.  *
4.  * @param content
5.  * @return
6.  */
```



```

7. public static boolean isQqFace(String content) {
8.     boolean result = false;
9.
10.    // 判断 QQ 表情的正则表达式
11.    String qqfaceRegex = "/:~|/:B|/:8-\\)|/:<|/:$|/:X|/:Z|/:'\\"/>

```

下面是方法的使用，实现了这样一个简单的功能：用户发什么 QQ 表情给公众帐号，公众帐号就回复什么 QQ 表情给用户（xiaoqrobot 就是这么做的）。实现代码如下：

[java] view plaincopy

```

1. // 文本消息
2. if (msgType.equals(MessageUtil.REQ_MESSAGE_TYPE_TEXT)) {
3.     // 文本消息内容
4.     String content = requestMap.get("Content");
5.
6.     // 判断用户发送的是否是单个 QQ 表情
7.     if(XiaoqUtil.isQqFace(content)) {
8.         // 回复文本消息
9.         TextMessage textMessage = new TextMessage();
10.        textMessage.setToUserName(fromUserName);
11.        textMessage.setFromUserName(toUserName);
12.        textMessage.setCreateTime(new Date().getTime());
13.        textMessage.setMsgType(MessageUtil.RESP_MESSAGE_TYPE_TEXT);
14.        textMessage.setFuncFlag(0);
15.        // 用户发什么 QQ 表情，就返回什么 QQ 表情
16.        textMessage.setContent(content);
17.
18.        // 将文本消息对象转换成 xml 字符串
19.        respMessage = MessageUtil.textMessageToXml(textMessage);

```



```
20.    }
21. }
```

好了，关于微信公众帐号中 QQ 表情的使用就介绍这么多。其实，我并不希望初学者上来只是简单拷贝我贴出的代码，实现了自己想要的功能就完事了，更希望初学的朋友能够通过此文章学会一种思考问题和解决问题的方法。

微信公众帐号开发教程第 10 篇-解析接口中的消息创建时间

CreateTime

从微信公众平台的[消息接口指南](#)中可以看出，每种类型的消息定义中，都包含有 **CreateTime** 参数，它表示消息的创建时间，如下图所示：

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	text 
Content	文本消息内容
MsgId	消息id，64位整型 http://blog.csdn.net/lyq8479

上图是消息接口指南中 4.1-文本消息的定义。注意 **CreateTime** 的描述：消息创建时间（**整型**），重点在于这是一个整型的时间，而不是我们大家所熟悉的类似于"yyyy-MM-dd HH:mm:ss"的标准格式时间。**本文主要想介绍的就是微信消息接口中定义的整型消息创建时间 CreateTime 的含义，以及如何将 CreateTime 转换成我们所熟悉的时间格式。**

整型 CreateTime 的含义

消息接口中定义的消息创建时间 **CreateTime**，它表示 **1970 年 1 月 1 日 0 时 0 分 0 秒至消息创建时所间隔的秒数**，注意是间隔的秒数，不是毫秒数！

整型 CreateTime 的转换

在 Java 中，我们也经常会通过下面两种方式获取 long 类型的时间，先上代码：

```
[java] view plaincopy
```

```
1. /**
```

```

2.  * 演示 Java 中常用的获取 long 类型时间的两种方式
3.  */
4.  public static void main(String[] args) {
5.      long longTime1 = System.currentTimeMillis();
6.      // 1373206143378
7.      System.out.println(longTime1);
8.
9.      long longTime2 = new java.util.Date().getTime();
10.     // 1373206143381
11.     System.out.println(longTime2);
12. }

```

上面两种获取 long 类型时间的方法是等价的，获取到的结果表示当时时间距离 1970 年 1 月 1 日 0 时 0 分 0 秒 0 毫秒的**毫秒数**，注意这里是毫秒数！那么这里获取到的 long 类型的时间如何转换成标准格式的时间呢？方法如下：

[java] view plaincopy

```

1.  /**
2.   * 演示 Java 中常用的获取 long 类型时间的两种方式
3.   */
4.   public static void main(String[] args) {
5.       // 当前时间（距离 1970 年 1 月 1 日 0 时 0 分 0 秒 0 毫秒的毫秒数）
6.       long longTime = 1373206143378L;
7.
8.       String stdFormatTime = formatTime(longTime);
9.       // 输出: 2013-07-07 22:09:03
10.      System.out.println(stdFormatTime);
11.  }
12.
13. /**
14.  * 将 long 类型的时间转换成标准格式（yyyy-MM-dd HH:mm:ss）
15.  *
16.  * @param longTime
17.  * @return
18.  */
19. public static String formatTime(long longTime) {
20.     DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
21.     return format.format(new Date(longTime));
22. }

```

上面演示了将一个 long 类型的时间转换成标准格式的时间，只是简单的运用了 SimpleDateFormat 类，比较好懂的。那么再回到今天的主题上来，如何将 CreateTime 转换成标准格式的时间。

微信消息接口中的 **CreateTime** 表示距离 1970 年的秒数，而 **System.currentTimeMillis()** 表示距离 1970 年的毫秒数，它们之间的换算就相当于：1 秒=1000 毫秒，即将 **CreateTime** 乘以 1000，就变成了距离 1970 年的毫秒数了，就可以使用上面的 **formatTime()**方法来处理了，是不是很简单呢？

下面，我还是单另封装一个方法，用于将微信消息中的整型的消息创建时间 **CreateTime** 转换成标准格式的时间，如下：

[java] view plaincopy

```
1.  /**
2.   * 将微信消息中的 CreateTime 转换成标准格式的时间（yyyy-MM-dd HH:mm:ss）
3.   *
4.   * @param createTime 消息创建时间
5.   * @return
6.   */
7.  public static String formatTime(String createTime) {
8.      // 将微信传入的 CreateTime 转换成 long 类型，再乘以 1000
9.      long msgCreateTime = Long.parseLong(createTime) * 1000L;
10.     DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
11.     return format.format(new Date(msgCreateTime));
12. }
```

微信公众帐号开发教程第 11 篇-符号表情的发送（上）

题外话（可以略过）

相信这篇文章已经让大家等的太久了，不是我故弄玄虚、吊大家胃口，而是写一篇文章真的需要花太多的时间。也许一篇文章，你们花 3-5 分钟就看完了、就学会掌握了，而我却要花 2-3 个小时的时间来完成，也许只有用心写过文章的人才能体会，希望大家能够相互体谅！

也曾经有人对我说，我写的东西太初级，都是入门级的东西。好吧，我承认众口难调，很难满足所有的读者，再加上我自己也只是个新手，一个 4 月前才听说微信公众平台这个词的初学者，谢谢你们以不同方式对我的激励，我会更加努力的！

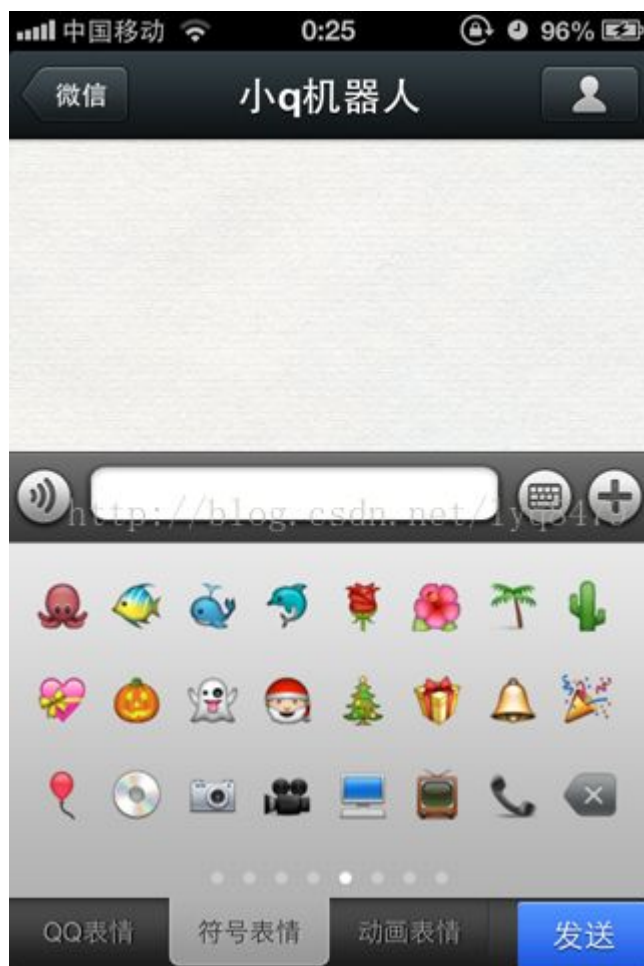
第 9 篇文章介绍了 **QQ 表情的发送与接收**。在此之后，很多朋友问我如何发 emoji 表情（微信上叫符号表情），也就让我有了写这篇文章的决心。在此之前，我在网上进行了大量的搜索，发现根本没有介绍这方面的文章，并且在微信公众帐号开发官方交流群里提问，也少有人知道该如何发送 emoji 表情。今天，就让我们一起来揭开它的神秘面纱！

文章概要

本文重点介绍如何在微信公众帐号开发模式下，通过程序代码向用户发送符号表情。至于如何识别用户发送的是符号表情，就不在此讲解了，留给大家一点学习思考的空间。我只是给大家一个提示：用户向公众帐号发送符号表情，其实也是一条文本消息，这与 QQ 表现是一样的，即便是文本消息，将接收的符号表情内容打印到日志，不就知道每个表情对应的文本了吗？呵呵，当然也没有这么简单，并不是像其他文本消息，这里需要对接收到符号表情消息先做编码的转换。好了，就提示这么多。

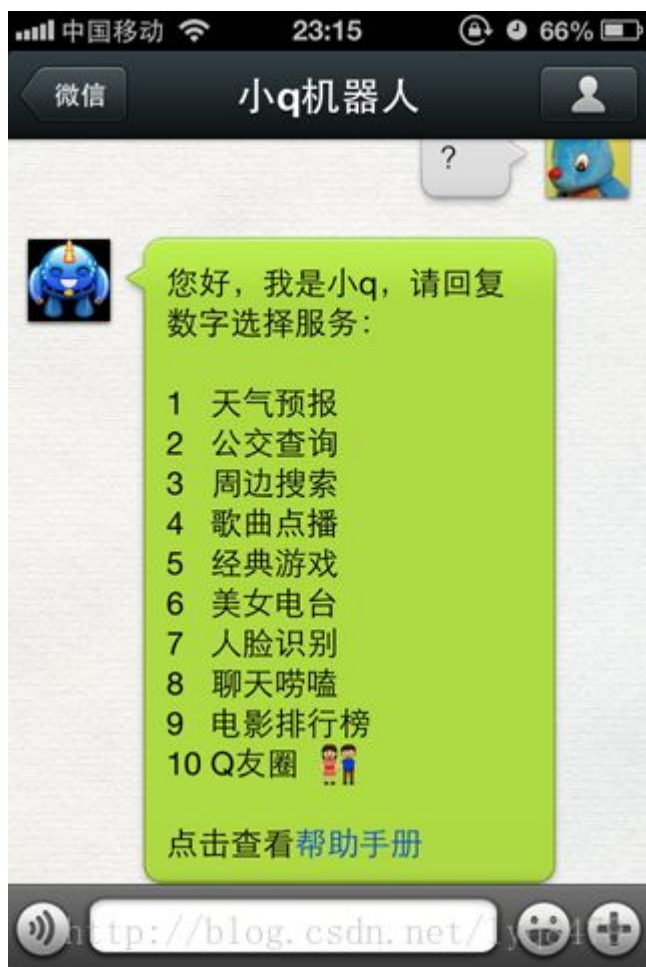
认识符号表情

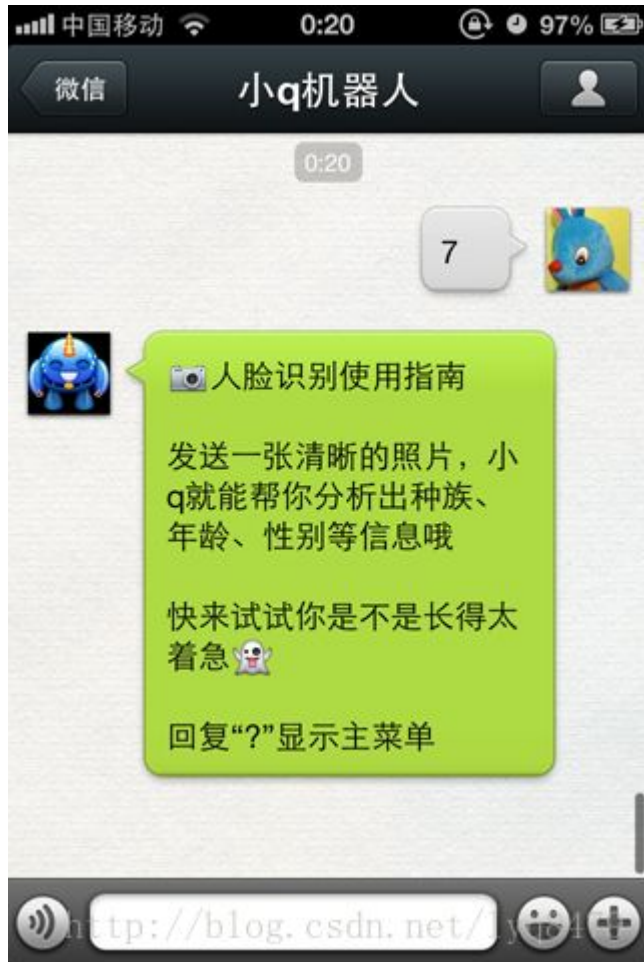
在公众帐号的主交互界面，窗口底部的输入框旁边有一个笑脸的图片按钮，点击它将会弹出表情选择界面，可选择表情依次为“QQ 表情”、“符号表情”和“动画表情”，我们选择“符号表情”，将会看到如下图所示界面：



可以看出，相比 QQ 表情，符号表情要更加实用。为什么这么说呢？因为 QQ 表情大都是脸部表情，而符号表情除了脸部表情外，还有很多与生活息息相关的表情，例如：动物、花朵、树木、电视、电话、电脑、吉它、球类、交通工具等等。如果能在消息中使用符号表情，会不会显得更加生动、有趣呢？

再来看看小 q 机器人中使用符号表情的效果，先上两张图：





左边截图是小 q 机器人的主菜单，在 Q 友圈文字旁边的那个表情就是符号表情，是一女一男两人小朋友，示意着在 Q 友圈里可以结识到更多的朋友，不要想歪了，^_^。右边截图是人脸识别功能的使用指南，里面的“相机”、“鬼脸”也是符号表情，这样看上去是不是更加有趣味性呢？如果是纯文本，一定会显得太单调、太枯燥了。





























Emoji 表情的分类

Emoji 表情有很多种版本，包括 Unified、DoCoMo、KDDI、Softbank 和 Google，而且不同版本的表情代码也不一样，更可恶的是：不同的手机操作系统、甚至是同一操作系统的不同版本所支持的 emoji 表情又不一样。所以，完美主义者可以止步了，因为目前 emoji 表情并不能保证在所有终端上都能正常使用。

















庆幸的是，我已经在超过 10 余部终端上测试过 emoji 表情的使用，这其中包括 iPhone 4S、iPhone 5、Android 2.2、Android 4.0+、Win8、iPad2，只有极个别终端上显示不出来或显示为一个小方格，所以并没有什么太大的影响，也就可以放心使用了！

Emoji 表情代码表之 Unified 版本























上面介绍的几种版本的 emoji 表情，都是通过 **unicode** 编码来表示的。换言之，不同版本的 emoji 表情对应的 **unicode** 编码值也不一样。本篇文章，我先给出 **Unified** 版本 emoji 表情的代码表，如下图所示：

Emoji 表情	Unicode 编码	Emoji 表情	Unicode 编码
	U+1F604		U+1F60A
	U+1F603		U+263A
	U+1F609		U+1F60D
	U+1F618		U+1F61A
	U+1F633		U+1F601
	U+1F60C		U+1F61C
	U+1F61D		U+1F612
	U+1F60F		U+1F613
	U+1F614		U+1F61E
	U+1F616		U+1F625
	U+1F630		U+1F628
	U+1F623		U+1F622
	U+1F62D		U+1F602
	U+1F632		U+1F631








<http://blog.csdn.net/yq8479>

	U+1F620		U+1F621
	U+1F62A		U+1F637
	U+1F47F		U+1F47D
	U+2764		U+1F494
	U+1F498		U+2728
	U+1F31F		U+2755
	U+2754		U+1F4A4
	U+1F4A6		U+1F3B5
	U+1F525		U+1F4A9
	U+1F44D		U+1F44E
	U+1F44A		U+270C
	U+1F446		U+1F447
	U+1F449		U+1F448
	U+261D		U+1F4AA
	U+1F48F		U+1F491
	U+1F466		U+1F467

<http://blog.csdn.net/lyq8479>

	U+1F469		U+1F468
	U+1F47C		U+1F480
	U+1F48B		U+2600
	U+2614		U+2601
	U+26C4		U+1F319
	U+26A1		U+1F30A
	U+1F431		U+1F436
	U+1F42D		U+1F439
	U+1F430		U+1F43A
	U+1F438		U+1F42F
	U+1F428		U+1F43B
	U+1F437		U+1F42E
	U+1F417		U+1F435
	U+1F434		U+1F40D
	U+1F426		U+1F414
	U+1F427		U+1F41B
	U+1F419		U+1F420
	U+1F433		U+1F42C
	U+1F339		U+1F33A
	U+1F334		U+1F335

<http://blog.csdn.net/lyq8479>

	U+1F49D		U+1F383
	U+1F47B		U+1F385
	U+1F384		U+1F381
	U+1F514		U+1F389
	U+1F388		U+1F48F
	U+1F4F7		U+1F3A5
	U+1F48B		U+1F4FA
	U+1F4DE		U+1F513
	U+1F512		U+1F511
	U+1F528		U+1F4A1
	U+1F4EB		U+1F6C0
	U+1F4B0		U+1F4A3
	U+1F52B		U+1F48A
	U+1F3C8		U+1F3C0
	U+26BD		U+26BE
	U+26F3		U+1F3C6
	U+1F47E		U+1F3A4
	U+1F3B8		U+1F459

<http://b1qg.csdn.net/tyq8479>

	U+1F451		U+1F302
	U+1F45C		U+1F484
	U+1F48D		U+1F48E
	U+2615		U+1F37A
	U+1F37B		U+1F378
	U+1F354		U+1F35F
	U+1F35D		U+1F363
	U+1F35C		U+1F373
	U+1F366		U+1F382
	U+1F34E		U+2708
	U+1F680		U+1F6B2
	U+1F684		U+26A0
	U+1F3C1		U+1F6B9
	U+1F6BA		U+2B55
	U+274C		U+00A9
	U+00AE		U+2122

公众帐号如何向用户发送 emoji 表情

上面已经给出了 emoji 表情的 unified unicode 代码对照表，那么这些代码要如何使用，才能发送出对应的 emoji 表情呢？如果你只是简单的像使用 QQ 表情代码那样，直接在文本消息的 Content 里写 emoji 表情代码，一定是会原样显示的。

这里需要用到一个 Java 方法做转换处理，方法的代码如下：

[java] view plaincopy

```

1.  /**
2.   * emoji 表情转换(hex -> utf-16)
3.   *
4.   * @param hexEmoji
5.   * @return
6.   */
7.  public static String emoji(int hexEmoji) {

```

```
8.     return String.valueOf(Character.toChars(hexEmoji));
9. }
```

方法说明：例如，“自行车”的 unicode 编码值为 U+1F6B2，如果我们要在程序代码中使用“自行车”这个 emoji 表情，需要这样使用：

[java] view plaincopy

```
1. String bike = String.valueOf(Character.toChars(0x1F6B2));
```

其实前面那个 emoji() 方法就是对上面这行代码做了个简单的封装而已。现在知道如何使用 emoji 表情代码了吧，其实就是将代码表中的 U+ 替换为 0x，再调用 emoji 方法进行转换，将转换后的结果放在文本消息的 Content 中，返回给用户就会显示 emoji 表情了。

下面，我给出一个使用 emoji 表情的完整示例，如下：

[java] view plaincopy

```
1. package org.liufeng.course.service;
2.
3. import java.util.Date;
4. import java.util.Map;
5.
6. import javax.servlet.http.HttpServletRequest;
7.
8. import org.liufeng.course.message.resp.TextMessage;
9. import org.liufeng.course.util.MessageUtil;
10.
11. /**
12.  * 核心服务类
13.  *
14.  * @author liufeng
15.  * @date 2013-05-20
16.  */
17. public class CoreService {
18.     /**
19.      * 处理微信发来的请求
20.      *
21.      * @param request
22.      * @return
23.      */
24.     public static String processRequest(HttpServletRequest request) {
25.         String respMessage = null;
26.         try {
27.             // xml 请求解析
```

```

28.         Map<String, String> requestMap = MessageUtil.parseXml(request);
29.
30.         // 发送方帐号 (open_id)
31.         String fromUserName = requestMap.get("FromUserName");
32.         // 公众帐号
33.         String toUserName = requestMap.get("ToUserName");
34.
35.         // 回复文本消息
36.         TextMessage textMessage = new TextMessage();
37.         textMessage.setToUserName(fromUserName);
38.         textMessage.setFromUserName(toUserName);
39.         textMessage.setCreateTime(new Date().getTime());
40.         textMessage.setMsgType(MessageUtil.RESP_MESSAGE_TYPE_TEXT);
41.         textMessage.setFuncFlag(0);
42.         textMessage.setContent("自行车" + emoji(0x1F6B2) + " 男性"
    " + emoji(0x1F6B9) + " 钱袋" + emoji(0x1F4B0));
43.         respMessage = MessageUtil.textMessageToXml(textMessage);
44.     } catch (Exception e) {
45.         e.printStackTrace();
46.     }
47.
48.     return respMessage;
49. }
50.
51. /**
52.  * emoji 表情转换(hex -> utf-16)
53.  *
54.  * @param hexEmoji
55.  * @return
56.  */
57. public static String emoji(int hexEmoji) {
58.     return String.valueOf(Character.toChars(hexEmoji));
59. }
60. }

```

上面代码的作用是：不管用户发送什么类型的消息，都返回包含三个 emoji 表情的文本消息。如果不明白 **CoreService** 类怎么回事，请查看本系列教程的第 5 篇，或者你只需要认真看第 42 行代码，就知道怎么样把 emoji 表情代码放在文本消息的 **Content** 中了。最后再来看下运行效果截图：

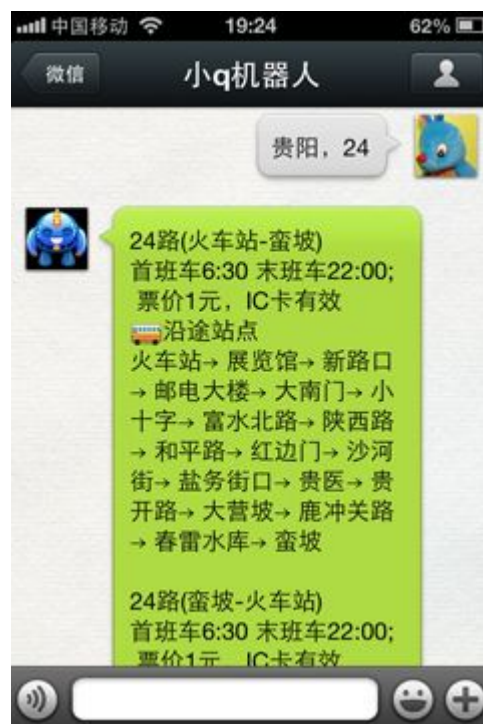
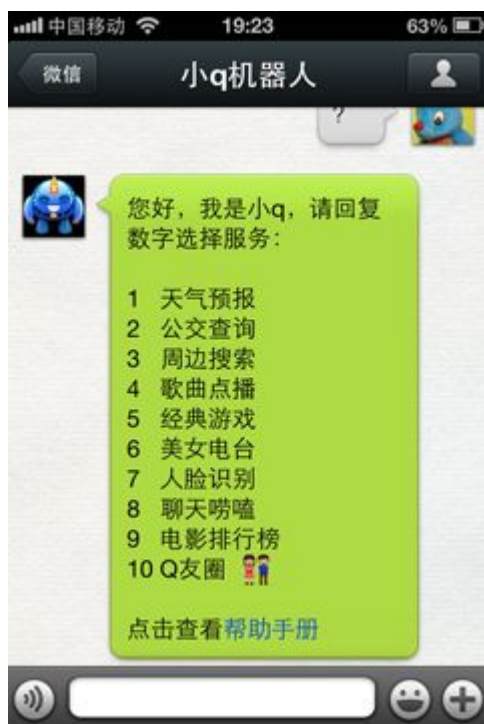


本篇文章要讲的内容就至此结束了，但关于 emoji 表情的讲解还没有结束，为什么这么说呢？请仔细看本篇文章的第二张截图，也就是小 q 机器人的文本菜单，里面用到的 emoji 表情在本文给出的 emoji 代码表里根本找不到（微信上的 emoji 表情与代码表中完全一致），那这个 emoji 表情又是如何发送的呢，请听下回分解！

微信公众帐号开发教程第 12 篇-符号表情的发送（下）

引言及文章概要

第 11 篇文章给出了 Unified 版本的符号表情（emoji 表情）代码表，并且介绍了如何在微信公众帐号开发模式下发送 emoji 表情，还在文章结尾出，卖了个关子：“小 q 机器人中使用的一些符号表情，在微信的符号表情选择栏里根本找不到，并且在上篇文章给出的符号表情代码表（Unified 版）中也没有，那这些表情是如何发送的呢？”如下面两张图所示的符号表情“情侣”和“公共汽车”。



本文主要介绍以下内容：1）如何在微信上使用更多的符号表情（即如何发送在微信符号表情选择栏中不存在的 emoji 表情）；2）给出 SoftBank 版符号表情的代码对照表；3）介绍及演示如何发送 SoftBank 版本的符号表情。让大家彻底玩转微信公众帐号的 emoji 表情！

如何在微信上使用更多的符号表情

我们先来看下，作为一个微信用户，如何向好友或微信公众帐号发送一些微信符号表情选择栏中没有列出的符号表情。例如：小 q 机器人中使用的“情侣”、“公共汽车”两个符号表情，如果我想在与朋友微信聊天时使用，该怎么办呢？请先看下面的两张截图：



可以看出，当我们在输入框中输入“情侣”的全拼“qinglv”、“公共汽车”的全拼“gonggongqiche”时，输入法的文本提示列表中就会自动显示对应的符号表情，怎么样，是不是很容易呢？这类表情还有很多，例如：马桶、厕所、取款机等。

说明：笔者使用的是 iPhone 4S 手机系统自带的输入法做的测试，如果你用的是安卓、或者是第三方输入法，那就另当别论了。

Emoji 表情代码表之 SoftBank 版本

上篇文章讲过，emoji 表情有很多种版本，其中包括 Unified、DoCoMo、KDDI、Softbank 和 Google，并且不同版本用于表示同一符号表情的 Unicode 代码也不相同。本篇文章，给出 SoftBank（日本软银集团）版本的 emoji 表情代码表（网上一般称之为 SB Unicode，指的就是它），如下图所示：

😊\ue415 😊\ue056 😊\ue057 😊\ue414 😊\ue405 😍\ue106 🤔\ue418
😏\ue417 😬\ue40d 😊\ue40a 😬\ue404 😊\ue105 😬\ue409 😊\ue40e
😊\ue402 😓\ue108 😊\ue403 😊\ue058 😬\ue407 😊\ue401 😬\ue40f
😬\ue40b 😊\ue406 😬\ue413 🤔\ue411 😬\ue412 😊\ue410 😬\ue107
😊\ue059 😡\ue416 😊\ue408 😊\ue40c 🐾\ue11a 🐼\ue10c 🧡\ue32c
💙\ue32a 💜\ue32d 💖\ue328 💚\ue32b ❤️\ue022 💔\ue023 💖\ue327
💖\ue329 ✨\ue32e ⭐\ue32f ⭐\ue335 💥\ue334 !\ue021 !\ue337
? \ue020 ? \ue336 zzz\ue13c ☁️\ue330 💧\ue331 🎵\ue326 🎵\ue03e
🔥\ue11d 💩\ue05a 👍\ue00e 👎\ue421 👉\ue420 👊\ue00d 👊\ue010
✌️\ue011 🖐️\ue41e 🖐️\ue012 🤝\ue422 👆\ue22e 🖐️\ue22f 👉\ue231
👈\ue230 🙌\ue427 🙌\ue41d 👆\ue00f 🙌\ue41f 💪\ue14c 🧑\ue201
🧑\ue115 🧑\ue428 🧑\ue51f 🧑\ue429 🧑\ue424 🧑\ue423 🧑\ue253
🧑\ue426 🧑\ue111 🧑\ue425 🧑\ue31e 🧑\ue31f 🧑\ue31d 🧑\ue001

👤\ue002 👩\ue005 👨\ue004 👶\ue51a 👧\ue519 👦\ue518 👦\ue515

👨\ue516 👩\ue517 🧑\ue51b 👮\ue152 👮\ue04e 👩\ue51c 👨\ue51e

💀\ue11c 🦶\ue536 👄\ue003 👄\ue41c 👂\ue41b 👁\ue419 🖐\ue41a

☀️\ue04a 🌂\ue04b ☁️\ue049 🧊\ue048 🌙\ue04c ⚡️\ue13d 🌀\ue443

🐬\ue43e 🐱\ue04f 🐶\ue052 🐻\ue053 🐼\ue524 🐰\ue52c 🐾\ue52a

🐸\ue531 🐯\ue050 🐨\ue527 🐻\ue051 🐼\ue10b 🐮\ue52b 🐴\ue52f

🐾\ue528 🐾\ue01a 🐾\ue134 🐾\ue530 🐾\ue529 🐾\ue526 🐾\ue52d

🐼\ue521 🐼\ue523 🐼\ue52e 🐼\ue055 🐼\ue525 🐼\ue10a 🐼\ue109

🐼\ue522 🐼\ue019 🐼\ue054 🐼\ue520 🐼\ue306 🐼\ue030 🐼\ue304

🍀\ue110 🌹\ue032 🌻\ue305 🌺\ue303 🍁\ue118 🌿\ue447 🍂\ue119

🌴\ue307 🌵\ue308 🌿\ue444 🐚\ue441

🏰\ue436 💖\ue437 👪\ue438 📖\ue43a 📱\ue439 🏠\ue43b 🌌\ue117

🌌\ue440 🌌\ue442 🌌\ue446 🍊\ue445 🧛\ue11b 🎅\ue448 🌲\ue033

🎁\ue112 🔔\ue325 🎊\ue312 🎈\ue310 💿\ue126 📀\ue127 📷\ue008

👥\ue03d 💻\ue00c 📺\ue12a 📱\ue00a 🖨️\ue00b 📞\ue009 📀\ue316

📼\ue129 📡\ue141 📢\ue142 🔊\ue317 📺\ue128 📡\ue14b 📺\ue211

🔍\ue114 🔓\ue145 🔒\ue144 🔑\ue03f ✂️\ue313 🛠️\ue116 💡\ue10f

📞\ue104 ✉️\ue103 📠\ue101 📮\ue102 🚮\ue13f 🚽\ue140 🚰\ue11f

💰\ue12f 🏆\ue031 🚬\ue30e 💣\ue311 🔫\ue113 💊\ue30f 🩺\ue13b

🏈\ue42b 🏀\ue42a ⚽\ue018 ⚾\ue016 🏏\ue015 ⚾\ue014 ⚾\ue42c

🏊\ue42d 🏄\ue017 🏊\ue013 ♠️\ue20e ❤️\ue20c ♣️\ue20f ♦️\ue20d

🏆\ue131 🧩\ue12b 🎯\ue130 🇨🇳\ue12d 🎬\ue324 📝\ue301 📖\ue148

🎨\ue502 🎤\ue03c 🎵\ue30a 🎷\ue042 🎸\ue040 🎸\ue041 🎷\ue12c

👞\ue007 👠\ue31a 👠\ue13e 👢\ue31b 👕\ue006 👔\ue302 👗\ue319

👛\ue321 👙\ue322 🎀\ue314 🎩\ue503 🏰\ue10e 🌐\ue318 🌂\ue43c

 \ue11e  \ue323  \ue31c  \ue034  \ue035  \ue045  \ue338
 \ue047  \ue30c  \ue044  \ue30b  \ue043  \ue120  \ue33b
 \ue33f  \ue341  \ue34c  \ue344  \ue342  \ue33d  \ue33e
 \ue340  \ue34d  \ue339  \ue147  \ue343  \ue33c  \ue33a
 \ue43f  \ue34b  \ue046  \ue345  \ue346  \ue348  \ue347
 \ue34a  \ue349

<http://blog.csdn.net/lyo8479>
 \ue036  \ue157  \ue038  \ue153  \ue155  \ue14d  \ue156
 \ue501  \ue158  \ue43d  \ue037  \ue504  \ue44a  \ue146
 \ue50a  \ue505  \ue506  \ue122  \ue508  \ue509  \ue03b
 \ue04d  \ue449  \ue44b  \ue51d  \ue44c  \ue124  \ue121
 \ue433  \ue202  \ue135  \ue01c  \ue01d  \ue10d  \ue136
 \ue42e  \ue01b  \ue15a  \ue159  \ue432  \ue430  \ue431

🚚\ue42f 🚆\ue01e 🚊\ue039 🚢\ue435 🚝\ue01f 🚇\ue125 🚦\ue03a
 🚨\ue14e ⚠️\ue252 🚧\ue137 🗺️\ue209 🏧\ue154 🚉\ue133 🕒\ue150
 🏰\ue320 🔥\ue123 🏁\ue132 🚩\ue143 🇯🇵\ue50b 🇰🇷\ue514 🇨🇳\ue513
 🇺🇸\ue50c 🇫🇷\ue50d 🇪🇸\ue511 🇮🇹\ue50f 🇷🇺\ue512 🇬🇧\ue510 🇩🇪\ue50e

1\ue21c 2\ue21d 3\ue21e 4\ue21f 5\ue220 6\ue221 7\ue222
 8\ue223 9\ue224 0\ue225 #\ue210 ⬆️\ue232 ⬇️\ue233 ⬅️\ue235
 ➡️\ue234 ↗️\ue236 ↶️\ue237 ↷️\ue238 ↵️\ue239 ⬅️\ue23b ➡️\ue23a
 ⬅️\ue23d ➡️\ue23c OK\ue24d NEW\ue212 TOP\ue24c UP\ue213 COOL\ue214
 🏠\ue507 🏡\ue203 📶\ue20b 📶\ue22a 📶\ue22b 📶\ue226 📶\ue227
 📶\ue22c 📶\ue22d 📶\ue215 📶\ue216 📶\ue217 📶\ue218 📶\ue228
 👤\ue151 👤\ue138 👤\ue139 📶\ue13a 📶\ue208 P\ue14f ♿\ue20a
 🚻\ue434 🚻\ue309 🚻\ue315 🚻\ue30d 🚻\ue207 ID\ue229 🌟\ue206
 🏆\ue205 💖\ue204 vs\ue12e 🏆\ue250 🏆\ue251 🏆\ue14a 🏆\ue149
 ♈\ue23f ♉\ue240 ♊\ue241 ♋\ue242 ♌\ue243 ♍\ue244 ♎\ue245
 ♏\ue246 ♐\ue247 ♑\ue248 ♒\ue249 ♓\ue24a ♊\ue24b ♊\ue23e
 A\ue532 B\ue533 AB\ue534 O\ue535 🟢\ue21a 🟠\ue219 🟡\ue21b
 ⌚\ue02f ⌚\ue024 ⌚\ue025 ⌚\ue026 ⌚\ue027 ⌚\ue028 ⌚\ue029
 ⌚\ue02a ⌚\ue02b ⌚\ue02c ⌚\ue02d ⌚\ue02e ⌚\ue332 ✖️\ue333
 ©\ue24e ®\ue24f ™\ue537

公众帐号如何向用户发送 SoftBank 版本的符号表情

在微信公众帐号开发模式下，发送 **SoftBank** 版的符号表情要比发送 **Unified** 版的符号表情简单的多，直接将符号表情对应的 **SoftBank Unicode** 值写在程序代码中返回给用户即可，无需做任何处理。

下面，我给出一个发送 **SoftBank** 版符号表情的示例，代码如下：

[java] view plaincopy

```
1. package org.liufeng.course.service;
2.
3. import java.util.Date;
4. import java.util.Map;
5.
6. import javax.servlet.http.HttpServletRequest;
7.
8. import org.liufeng.course.message.resp.TextMessage;
9. import org.liufeng.course.util.MessageUtil;
10.
11. /**
12.  * 核心服务类
13.  *
14.  * @author liufeng
15.  * @date 2013-07-21
16.  */
17. public class CoreService {
18.     /**
19.      * 处理微信发来的请求
20.      *
21.      * @param request
22.      * @return
23.      */
24.     public static String processRequest(HttpServletRequest request) {
25.         String respMessage = null;
26.         try {
27.             // xml 请求解析
28.             Map<String, String> requestMap = MessageUtil.parseXml(request);
29.
30.             // 发送方帐号 (open_id)
31.             String fromUserName = requestMap.get("FromUserName");
32.             // 公众帐号
33.             String toUserName = requestMap.get("ToUserName");
34.
35.             // 回复文本消息
36.             TextMessage textMessage = new TextMessage();
```

```

37.         textMessage.setToUserName(fromUserName);
38.         textMessage.setFromUserName(toUserName);
39.         textMessage.setCreateTime(new Date().getTime());
40.         textMessage.setMsgType(MessageUtil.RESP_MESSAGE_TYPE_TEXT);
41.         textMessage.setFuncFlag(0);
42.         textMessage.setContent("自行车\u136 男人\u138 钱袋\u12f 情侣
    \u428 公共汽车\u159");
43.         respMessage = MessageUtil.textMessageToXml(textMessage);
44.     } catch (Exception e) {
45.         e.printStackTrace();
46.     }
47.
48.     return respMessage;
49. }
50. }

```

上面代码的作用是：不管用户发送什么类型的消息，都返回包含 5 个 emoji 表情的文本消息。如果不明白 **CoreService** 类怎么回事，请查看[本系列教程的第 5 篇](#)，或者你只需要认真看第 42 行代码，就知道怎么样把 **SoftBank** 版 emoji 表情代码放在文本消息的 **Content** 中了。最后再来看下运行效果截图：



说明：每一个符号表情都有与之对应的 **Unified unicode**、**Softbank unicode** 代码，并不是说“情侣”、“公共汽车”这类在微信的符号表情栏中找不到的 emoji 表情只能通过本文的方式发送，只要你拿到与之对应的 **Unified unicode** 代码，一样可以使用[上篇文章](#)所讲的方法发送这类符号表情。

好了，关于微信公众帐号向用户发送符号表情的讲解就此结束了，相信有些朋友看完教程已经开始在帐号中使用符号表情了。其实，我更希望大家在拷贝我粘出的 **Unified 版**、**SoftBank 版** 符号表情代码表的同时，也能去了解符号表情各种版本、**Unicode** 编码及增补码的相关

知识，不断拓展自己的知识面，触类旁通，这样才能真正地把我讲解的知识变成你自己的，才能做到以不变应万变。

微信公众帐号开发教程第 13 篇-图文消息全攻略

引言及内容概要

已经有几位读者抱怨“柳峰只用到文本消息作为示例，从来不提图文消息，都不知道图文消息如何使用”，好吧，我错了，原本以为把基础 API 封装完、框架搭建好，再给出一个文本消息的使用示例，大家就能够照猫画虎的，或许是因为我的绘画功底太差，画出的那只猫本来就不像猫吧.....

本篇主要介绍微信公众帐号开发中图文消息的使用，以及图文消息的几种表现形式。标题取名为“图文消息全攻略”，这绝对不是标题党，是想借此机会把大家对图文消息相关的问题、疑虑、障碍全部清除掉。

图文消息的主要参数说明

通过微信官方的[消息接口指南](#)，可以看到对图文消息的参数介绍，如下图所示：

参数	描述
ToUserName	接收方帐号（收到的OpenID）
FromUserName	开发者微信号
CreateTime	消息创建时间
MsgType	news
ArticleCount	图文消息个数，限制为10条以内
Articles	多条图文消息信息，默认第一个item为大图
Title	图文消息标题
Description	图文消息描述
PicUrl	图片链接，支持JPG、PNG格式，较好的效果为大图640*320，小图80*80。
Url	点击图文消息跳转链接

从图中可以了解到：

- 1) 图文消息的个数限制为 10，也就是图中 **ArticleCount** 的值（图文消息的个数，限制在 10 条以内）；
- 2) 对于多图文消息，第一条图文的图片显示为大图，其他图文的图片显示为小图；
- 3) 第一条图文的图片大小建议为 **640*320**，其他图文的图片大小建议为 **80*80**；

好了，了解这些，再结合第 4 篇文章所讲的[消息及消息处理工具的封装](#)，想要回复图文消息给用户也就不是什么难事了。

图文消息的多种表现形式

下面直接通过代码演示图文消息最主要的五种表现形式的用法，源代码如下：

[java] view plaincopy

```
1. package org.liufeng.course.service;
2.
3. import java.util.ArrayList;
4. import java.util.Date;
5. import java.util.List;
6. import java.util.Map;
7.
8. import javax.servlet.http.HttpServletRequest;
9.
10. import org.liufeng.course.message.resp.Article;
11. import org.liufeng.course.message.resp.NewsMessage;
12. import org.liufeng.course.message.resp.TextMessage;
13. import org.liufeng.course.util.MessageUtil;
14.
15. /**
16.  * 核心服务类
17.  *
18.  * @author liufeng
19.  * @date 2013-07-25
20.  */
21. public class CoreService {
22.     /**
23.      * 处理微信发来的请求
24.      *
25.      * @param request
26.      * @return
27.      */
28.     public static String processRequest(HttpServletRequest request) {
29.         String respMessage = null;
30.         try {
31.             // xml 请求解析
32.             Map<String, String> requestMap = MessageUtil.parseXml(request);
33.
34.             // 发送方帐号 (open_id)
```

```

35.         String fromUserName = requestMap.get("FromUserName");
36.         // 公众帐号
37.         String toUserName = requestMap.get("ToUserName");
38.         // 消息类型
39.         String msgType = requestMap.get("MsgType");
40.
41.         // 默认回复此文本消息
42.         TextMessage textMessage = new TextMessage();
43.         textMessage.setToUserName(fromUserName);
44.         textMessage.setFromUserName(toUserName);
45.         textMessage.setCreateTime(new Date().getTime());
46.         textMessage.setMsgType(MessageUtil.RESP_MESSAGE_TYPE_TEXT);
47.         textMessage.setFuncFlag(0);
48.         // 由于 href 属性值必须用双引号引起，这与字符串本身的双引号冲突，所以要
           转义
49.         textMessage.setContent("欢迎访问
           <a href=\"http://blog.csdn.net/lyq8479\">柳峰的博客</a>!");
50.         // 将文本消息对象转换成 xml 字符串
51.         respMessage = MessageUtil.textMessageToXml(textMessage);
52.
53.         // 文本消息
54.         if (msgType.equals(MessageUtil.REQ_MESSAGE_TYPE_TEXT)) {
55.             // 接收用户发送的文本消息内容
56.             String content = requestMap.get("Content");
57.
58.             // 创建图文消息
59.             NewsMessage newsMessage = new NewsMessage();
60.             newsMessage.setToUserName(fromUserName);
61.             newsMessage.setFromUserName(toUserName);
62.             newsMessage.setCreateTime(new Date().getTime());
63.             newsMessage.setMsgType(MessageUtil.RESP_MESSAGE_TYPE_NEWS);
64.
65.             newsMessage.setFuncFlag(0);
66.
67.             List<Article> articleList = new ArrayList<Article>();
68.             // 单图文消息
69.             if ("1".equals(content)) {
70.                 Article article = new Article();
71.                 article.setTitle("微信公众帐号开发教程 Java 版");
72.                 article.setDescription("柳峰，80 后，微信公众帐号开发经验 4
           个月。为帮助初学者入门，特推出此系列教程，也希望借此机会认识更多同行！");
73.                 article.setPicUrl("http://0.xiaoqrobot.duapp.com/images/
           avatar_liufeng.jpg");
74.                 article.setUrl("http://blog.csdn.net/lyq8479");

```

```

74.         articleList.add(article);
75.         // 设置图文消息个数
76.         newsMessage.setArticleCount(articleList.size());
77.         // 设置图文消息包含的图文集合
78.         newsMessage.setArticles(articleList);
79.         // 将图文消息对象转换成 xml 字符串
80.         respMessage = MessageUtil.newsMessageToXml(newsMessage);

81.     }
82.     // 单图文消息---不含图片
83.     else if ("2".equals(content)) {
84.         Article article = new Article();
85.         article.setTitle("微信公众帐号开发教程 Java 版");
86.         // 图文消息中可以使用 QQ 表情、符号表情
87.         article.setDescription("柳峰, 80 后, " + emoji(0x1F6B9)
88.                                + ", 微信公众帐号开发经验 4 个月。为帮助初学者入门, 特
            推出此系列连载教程, 也希望借此机会认识更多同行! \n\n 目前已推出教程共 12 篇, 包括接口
            配置、消息封装、框架搭建、QQ 表情发送、符号表情发送等。 \n\n 后期还计划推出一些实用功能
            的开发讲解, 例如: 天气预报、周边搜索、聊天功能等。");
89.         // 将图片置为空
90.         article.setPicUrl("");
91.         article.setUrl("http://blog.csdn.net/lyq8479");
92.         articleList.add(article);
93.         newsMessage.setArticleCount(articleList.size());
94.         newsMessage.setArticles(articleList);
95.         respMessage = MessageUtil.newsMessageToXml(newsMessage);

96.     }
97.     // 多图文消息
98.     else if ("3".equals(content)) {
99.         Article article1 = new Article();
100.        article1.setTitle("微信公众帐号开发教程\n 引言");
101.        article1.setDescription("");
102.        article1.setPicUrl("http://0.xiaoqrobot.duapp.com/images/avatar_liufeng.jpg");
103.        article1.setUrl("http://blog.csdn.net/lyq8479/article/details/8937622");
104.
105.        Article article2 = new Article();
106.        article2.setTitle("第 2 篇\n 微信公众帐号的类型");
107.        article2.setDescription("");
108.        article2.setPicUrl("http://avatar.csdn.net/1/4/A/1_lyq8479.jpg");

```

```
109.         article2.setUrl("http://blog.csdn.net/lyq8479/article/d
           etails/8941577");
110.
111.         Article article3 = new Article();
112.         article3.setTitle("第 3 篇\n 开发模式启用及接口配置");
113.         article3.setDescription("");
114.         article3.setPicUrl("http://avatar.csdn.net/1/4/A/1_lyq8
           479.jpg");
115.         article3.setUrl("http://blog.csdn.net/lyq8479/article/d
           etails/8944988");
116.
117.         articleList.add(article1);
118.         articleList.add(article2);
119.         articleList.add(article3);
120.         newsMessage.setArticleCount(articleList.size());
121.         newsMessage.setArticles(articleList);
122.         respMessage = MessageUtil.newsMessageToXml(newsMessage);
123.     }
124.     // 多图文消息---首条消息不含图片
125.     else if ("4".equals(content)) {
126.         Article article1 = new Article();
127.         article1.setTitle("微信公众帐号开发教程 Java 版");
128.         article1.setDescription("");
129.         // 将图片置为空
130.         article1.setPicUrl("");
131.         article1.setUrl("http://blog.csdn.net/lyq8479");
132.
133.         Article article2 = new Article();
134.         article2.setTitle("第 4 篇\n 消息及消息处理工具的封装");
135.         article2.setDescription("");
136.         article2.setPicUrl("http://avatar.csdn.net/1/4/A/1_lyq8
           479.jpg");
137.         article2.setUrl("http://blog.csdn.net/lyq8479/article/d
           etails/8949088");
138.
139.         Article article3 = new Article();
140.         article3.setTitle("第 5 篇\n 各种消息的接收与响应");
141.         article3.setDescription("");
142.         article3.setPicUrl("http://avatar.csdn.net/1/4/A/1_lyq8
           479.jpg");
143.         article3.setUrl("http://blog.csdn.net/lyq8479/article/d
           etails/8952173");
144.
```

```
145.         Article article4 = new Article();
146.         article4.setTitle("第 6 篇\n 文本消息的内容长度限制揭秘");
147.         article4.setDescription("");
148.         article4.setPicUrl("http://avatar.csdn.net/1/4/A/1_lyq8
479.jpg");
149.         article4.setUrl("http://blog.csdn.net/lyq8479/article/d
etails/8967824");
150.
151.         articleList.add(article1);
152.         articleList.add(article2);
153.         articleList.add(article3);
154.         articleList.add(article4);
155.         newsMessage.setArticleCount(articleList.size());
156.         newsMessage.setArticles(articleList);
157.         respMessage = MessageUtil.newsMessageToXml(newsMessage);
158.     }
159.     // 多图文消息---最后一条消息不含图片
160.     else if ("5".equals(content)) {
161.         Article article1 = new Article();
162.         article1.setTitle("第 7 篇\n 文本消息中换行符的使用");
163.         article1.setDescription("");
164.         article1.setPicUrl("http://0.xiaoqrobot.duapp.com/image
s/avatar_liufeng.jpg");
165.         article1.setUrl("http://blog.csdn.net/lyq8479/article/d
etails/9141467");
166.
167.         Article article2 = new Article();
168.         article2.setTitle("第 8 篇\n 文本消息中使用网页超链接");
169.         article2.setDescription("");
170.         article2.setPicUrl("http://avatar.csdn.net/1/4/A/1_lyq8
479.jpg");
171.         article2.setUrl("http://blog.csdn.net/lyq8479/article/d
etails/9157455");
172.
173.         Article article3 = new Article();
174.         article3.setTitle("如果觉得文章对你有所帮助，请通过博客留言
或关注微信公众帐号 xiaoqrobot 来支持柳峰！");
175.         article3.setDescription("");
176.         // 将图片置为空
177.         article3.setPicUrl("");
178.         article3.setUrl("http://blog.csdn.net/lyq8479");
179.
180.         articleList.add(article1);
```

```

181.            articleList.add(article2);
182.            articleList.add(article3);
183.            newsMessage.setArticleCount(articleList.size());
184.            newsMessage.setArticles(articleList);
185.            respMessage = MessageUtil.newsMessageToXml(newsMessage);

186.        }
187.    }
188.    } catch (Exception e) {
189.        e.printStackTrace();
190.    }
191.    return respMessage;
192. }
193.
194. /**
195.  * emoji 表情转换(hex -> utf-16)
196.  *
197.  * @param hexEmoji
198.  * @return
199.  */
200. public static String emoji(int hexEmoji) {
201.     return String.valueOf(Character.toChars(hexEmoji));
202. }
203. }

```

如果不明白 **CoreService** 类放在什么位置，该如何使用，请查看[本系列教程的第 5 篇](#)。上面代码实现的功能是当用户发送数字 1-5 时，分别回复五种不同表现形式的图文消息给用户，如下：

a) 用户发送 1，回复**单图文消息**。参考代码 68~81 行，运行效果如下：



b) 用户发送 2，回复单图文消息---不含图片。参考代码 82~96 行，运行效果如下：



说明：图文消息的标题、描述是可以包含 QQ 表情、符号表情的。

c) 用户发送 3，回复多图文消息。参考代码 97~123 行，运行效果如下：



说明：对于多图文消息，描述不会被显示，可以在标题使用换行符，使得显示更加美观。

d) 用户发送 4，回复多图文消息---首条消息不含图片。参考代码 124~158 行，运行效果如下：



e) 用户发送 5, 回复**多图文消息---最后一条消息不含图片**。参考代码 159~186 行, 运行效果如下:



可以看出, 图文消息有着丰富的内容及多样化的表现形式, 希望大家能够根据各自的特点及实际使用需要, 合理地运用。

最后, **根据实践经验, 我对图文消息做一个使用总结:**

- 1) **一定要给图文消息的 `Url` 属性赋值。**不管是单图文, 还是多图文, 或者是不含图片的图文, 都有可能被用户点击。如果 `Url` 为空, 用户点击后将会打开一个空白页面, 这给用户的体验是非常差的;
- 2) **只有单图文的描述才会显示, 多图文的描述不会被显示;**
- 3) **图文消息的标题、描述中可以使用 QQ 表情和符号表情。**合理地运用表情符号, 会使得消息更加生动;
- 4) **图文消息的标题、描述中可以使用换行符。**合理地使用换行符, 会使得内容结构更加清晰;
- 5) **图文消息的标题、描述中不支持超文本链接 (html 的 `<a>` 标签)。**不只是技术上实现不了, 就连逻辑上也说不通, 因为一条图文消息的任何位置被点击, 都将调用微信内置的浏览器打开 `Url`, 如果标题、描述里再放几个超链接, 不知道点击该打开哪个页面。真搞不懂为什么有好几个同学都在问这个问题, 难道设计成多图文不好吗?
- 6) **图文消息的链接、图片链接可以使用外部域名下的资源,**如本例中: 柳峰的头像、博文的链接, 都是指向 **CSDN** 网站的资源。在网上, 甚至是微信官方交流群里, 认为图文消息

的 `Url`、`PicUrl` 不可以使用外链的大有人在，不知道这谣言从哪开始的，实践是检验真理的唯一标准！

7) 使用指定大小的图片。第一条图文的图片大小建议为 **640*320**，其他图文的图片大小建议为 **80*80**。如果使用的图片太大，加载慢，而且耗流量；如果使用的图片太小，显示后会被拉伸，失真了很难看。

8) 每条图文消息的图文建议控制在 **1-4** 条。这样在绝大多数终端上一屏能够显示完，用户扫一眼就能大概了解消息的主要内容，这样最有可能促使用户去点击并阅读。

微信公众帐号开发教程第 14 篇-自定义菜单的创建及菜单事件响应

微信 5.0 发布

2013 年 8 月 5 日，伴随着微信 5.0 iPhone 版的发布，公众平台也进行了重要的更新，主要包括：

- 1) 运营主体为组织，可选择成为服务号或者订阅号；
- 2) 服务号可以申请自定义菜单；
- 3) 使用 QQ 登录的公众号，可以升级为邮箱登录；
- 4) 使用邮箱登录的公众号，可以修改登录邮箱；
- 5) 编辑图文消息可选填作者；
- 6) 群发消息可以同步到腾讯微博。

其中，大家议论最多的当属前两条，就是关于帐号类型和自定义菜单的更新，我这里做几点补充说明：

- 1) 目前公众号类型分为两种：服务号和订阅号，8 月 5 日平台更新后所有的帐号默认为订阅号，有一次转换成服务号的机会；
- 2) 服务号主要面向企业、政府和其他组织，而订阅号主要面向媒体和个人；
- 3) 只有服务号可以申请自定义菜单，订阅号不能申请；
- 4) 服务号每月只能群发一条消息，而订阅号每天能群发一条消息。

平台更新后，让很多人纠结的是自定义菜单和每天群发一条消息不可兼得，对此，我不想过多评论。

引言及内容概要

在微信 5.0 以前，自定义菜单是作为一种内测资格使用的，只有少数公众帐号拥有菜单，因此出现很多企业为了弄到菜单不惜重金求购。现如今，一大批帐号从订阅号转为服务号，很多都是奔着自定义菜单去的。而且，经测试发现，微信最近的审核放松很多，只要申请服务

号、自定义菜单的基本都成功了，根本不管填写的资料真伪。不知道以后微信会不会翻脸，要求补全企业资料，那将会是一种给小孩一颗糖吃再把他打哭的感觉。。。

自定义菜单是申请到了，到底该怎么创建、怎么使用呢？最近几天不管是微信官方交流群，还是在我博客留言里，都能够看到不少开发者都在为这个发愁。本篇文章就为大家解决这个难题。

自定义菜单的创建步骤

- 1、找到 AppId 和 AppSecret。自定义菜单申请成功后，在“高级功能”-“开发模式”-“接口配置信息”的最后两项就是；
- 2、根据 AppId 和 AppSecret，以 https get 方式获取访问特殊接口所必须的凭证 access_token；
- 3、根据 access_token，将 json 格式的菜单数据通过 https post 方式提交。

分析创建菜单的难点

原来创建菜单这么简单，三步就能搞定？跟把大象放冰箱差不多。呵呵，当然没有这么简单，那我们一步步来看，到底难在哪里？

首先，第 1 步肯定都没有问题，只要成功申请了自定义菜单，一定能拿到 AppId 和 AppSecret 这两个值。

再来看第 2 步，由于是 get 方式获取 access_token，很多人直接把拼好的 url 放在浏览器里执行，access_token 就拿到了。抛开是不是用编程方式实现的来说，这真是个好办法，显然大家在第二步上也没有问题。

最后再看第 3 步，拼装 json 格式的菜单数据，虽然繁琐一点，但基本上也都没有什么问题的，因为官方给了个例子，照猫画虎就行了。那问题一定就出现在 https post 提交上了。

结论：不知道如何创建自定义菜单的朋友，大都可以归为以下三种情况：

- 1) 根本不看或者没看懂公众平台 API 文档中关于“通用接口”、“自定义菜单接口”和“使用限制”部分的说明；
- 2) 不知道如何发起 HTTPS 请求（平时的 http 请求，直接使用 HttpURLConnection 就可以轻松搞定，但 https 请求要复杂一点）；
- 3) 不知道如何通过 POST 方式提交 json 格式的菜单数据。

正在看文章的你，不知道是属于哪一种，或者几种情况都有，不妨留言说出来，也可以做个调查。不管属于哪一种情况，既然看到了这篇文章，相信一定会让你弄明白的。

解读通用接口文档---凭证的获取

我们先来看通用接口文档的简介部分，如下图所示。

通用接口文档

简介

通用接口是使用HTTP请求，让开发者直接与微信服务器交互，实现媒体文件上传、媒体文件获取等功能，达视频等媒体文件的目的。

调用接口所需要的access_token必须通过获取凭证接口获取。

通俗点讲，这段简介可以这么理解：公众平台还有很多特殊的接口，像自定义菜单的创建、语音文件的获取、主动发送消息等，如果开发者想通过 HTTP 请求访问这些特殊接口，就必须要有访问凭证，也就是 access_token。

那么，又该如何获取接口访问凭证 access_token 呢？让我们继续往下看。

http请求方式: GET

https://api.weixin.qq.com/cgi-bin/token?grant_type=client_credential&appid=APPID&secret=APPSECRET

参数说明

参数	是否必须	说明
grant_type	是	获取access_token填写client_credential
appid	是	第三方用户唯一凭证/1yq8479
secret	是	第三方用户唯一凭证密钥，既appsecret

返回说明

正确的Json返回结果:

```
{"access_token":"ACCESS_TOKEN","expires_in":7200}
```

图中已经表达的很清楚了，获取 access_token 是通过 GET 方式访问如下链接：

[java] view plain copy

1. https://api.weixin.qq.com/cgi-bin/token?grant_type=client_credential&appid=APPID&secret=APPSECRET

链接中有三个参数，分别是 grant_type、appid 和 secret。根据图中的参数说明，grant_type 传固定值 client_credential，而 appid 和 secret 就是申请完自定义菜单后微信分配给我们的。请求发送成功后，微信服务器会返回一个 json 串，包含 access_token 和 expires_in 两个元素。其中，access_token 就是我们最终需要的凭证，而 expires_in 是凭证的有效期，单位

是秒，7200 秒也就是 2 个小时。这就意味着，不是每次访问特殊接口，都需要重新获取一次 `access_token`，只要 `access_token` 还在有效期内，就一直可以使用。

解读自定义菜单接口文档

还是一样，先来看看自定义菜单接口的简介部分，如下图所示。

简介

开发者获取使用凭证（如何获取凭证）后，可以使用该凭证对公众账号的自定义菜单进行创建、查询和删除等接口可实现以下类型按钮：

click（点击事件）：

用户点击click类型按钮后，微信服务器会通过消息接口(event类型)推送点击事件给开发者，并且带上按钮中开发者填写的key值进行消息回复。

创建自定义菜单后，由于微信客户端缓存，需要24小时微信客户端才会展现出来。建议测试时可以尝试取消关注，则可以查看到创建后的效果。

从图中我们能够获取到以下信息：

- 1) 拿到凭证 `access_token` 后，我们能对菜单执行三种操作：创建、查询和删除；
- 2) 自定义菜单目前只支持 `click` 一种事件，即用户点击后回复某种类型的消息；不能够实现点击菜单项直接打开页面（`type=view` 未开放，目前只是微生活有）；
- 3) 由于微信客户端缓存的原因，菜单创建后并不会立即在微信上显示出来，需要过 24 小时。在测试菜单创建时，可以通过取消关注后，再关注的方式达到立即看菜单的目的。

继续往下看，就是关于菜单怎么创建的介绍，如下图所示。

菜单创建

接口说明

通过POST一个特定结构体，实现在微信客户端创建自定义菜单。

请求说明

http请求方式：POST

`https://api.weixin.qq.com/cgi-bin/menu/create?access_token=ACCESS_TOKEN`

其实就是向地址

https://api.weixin.qq.com/cgi-bin/menu/create?access_token=ACCESS_TOKEN 以 POST 方式提交一个 JSON 格式的菜单字符串。

后面，关于参数说明的部分我就不一一贴图说明了，把重点说一下：

- 1) 自定义菜单是一个 3x5 结构的，即菜单最多只能有二级，一级菜单最多只能有 3 个，每个一级菜单下最多可以有 5 个二级菜单项；
- 2) 菜单项都有一个 key 值。当用户点击某个菜单项时，微信会将该菜单项的 key 值以事件推送的方式发送给我们的后台处理程序。

关于菜单的查询、创建我就不提了，这两个接口使用的频率非常小，一般都用不上。如果需要，再按照我上面提供的思路也不难理解。

解读 API 文档之使用限制

默认每个公众帐号都不能超过下面的频率限制。当超出调用接口频率限制，调用对应接口将会收到如下错误：

```
{"errcode":45009,"errmsg":"api freq out of limit"}
```

接口调用频率限制

接口名称	频率限制
获取凭证接口	200 (次/天)
自定义菜单创建接口	100 (次/天)
自定义菜单查询接口	1000 (次/天)
自定义菜单删除接口	100 (次/天)

很多小伙伴看到这张图就开始疑惑了：怎么菜单还限制使用次数，用户量越来越大的时候，根本不够用啊。看清楚，这个限制是针对接口调用的，也就是针对开发者的，和用户数、使用次数半点关系也没有。

就先拿获取凭证接口来说吧，限制一天只能调用 200 次。还记得前面提到过 access_token 是有有效期的，并且有效期为两小时，也就是获取一次 access_token 后的两小时内，都可以继续使用，那么理想情况一天 24 小时内，是不是只需要获取 12 次就够了？难道 200 次还不够用？

再来看下菜单创建接口限制一天只能调用 100 次。我就这么解释吧，菜单创建一次后，只要你不切换模式（指的是在编辑模式和开发模式间切换）、不调用删除接口，这个菜单会永远存在的。谁没事干，一天要创建 100 次菜单，就算是测试，测个 10 次 8 次足够了吧？

菜单的查询和删除接口的限制我就不解释了，至今为止这二个接口我都没使用过一次。就算有这样的使用需求，一天这么多次的调用，完全足够了。

封装通用的请求方法

读到这里，就默认大家已经掌握了上面讲到的所有关于自定义菜单的理论知识，下面就进入代码实战讲解的部分。

先前我们了解到，创建菜单需要调用二个接口，并且都是 **https** 请求，而非 **http**。如果要封装一个通用的请求方法，该方法至少需要具备以下能力：

- 1) 支持 **HTTPS** 请求；
- 2) 支持 **GET**、**POST** 两种方式；
- 3) 支持参数提交，也支持无参数的情况；

对于 **https** 请求，我们需要一个证书信任管理器，这个管理器类需要自己定义，但需要实现 **X509TrustManager** 接口，代码如下：

[java] view plaincopy

```
1. package org.liufeng.weixin.util;
2.
3. import java.security.cert.CertificateException;
4. import java.security.cert.X509Certificate;
5.
6. import javax.net.ssl.X509TrustManager;
7.
8. /**
9.  * 证书信任管理器（用于 https 请求）
10.  *
11.  * @author liufeng
12.  * @date 2013-08-08
13.  */
14. public class MyX509TrustManager implements X509TrustManager {
15.
16.     public void checkClientTrusted(X509Certificate[] chain, String authType)
17.         throws CertificateException {
18.     }
19.
20.     public void checkServerTrusted(X509Certificate[] chain, String authType)
21.         throws CertificateException {
22.     }
23.
24.     public X509Certificate[] getAcceptedIssuers() {
25.         return null;
26.     }
27. }
```



```
24.    }  
25. }
```

这个证书管理器的作用就是让它信任我们指定的证书，上面的代码意味着信任所有证书，不管是否权威机构颁发。

证书有了，通用的 **https** 请求方法就不难实现了，实现代码如下：

[java] view plaincopy

```
1.  package org.liufeng.weixin.util;  
2.  
3.  import java.io.BufferedReader;  
4.  import java.io.InputStream;  
5.  import java.io.InputStreamReader;  
6.  import java.io.OutputStream;  
7.  import java.net.ConnectException;  
8.  import java.net.URL;  
9.  
10. import javax.net.ssl.HttpsURLConnection;  
11. import javax.net.ssl.SSLContext;  
12. import javax.net.ssl.SSLSocketFactory;  
13. import javax.net.ssl.TrustManager;  
14.  
15. import net.sf.json.JSONObject;  
16.  
17. import org.slf4j.Logger;  
18. import org.slf4j.LoggerFactory;  
19.  
20. /**  
21.  * 公众平台通用接口工具类  
22.  *  
23.  * @author liuyq  
24.  * @date 2013-08-09  
25.  */  
26. public class WeixinUtil {  
27.     private static Logger log = LoggerFactory.getLogger(WeixinUtil.class);  
28.  
29.     /**  
30.      * 发起 https 请求并获取结果  
31.      *  
32.      * @param requestUrl 请求地址  
33.      * @param requestMethod 请求方式（GET、POST）  
34.      * @param outputStr 提交的数据  
35.      * @return JSONObject(通过 JSONObject.get(key)的方式获取 json 对象的属性值)  
36.      */
```

```
37.     public static JSONObject httpRequest(String requestUrl, String requestMe
thod, String outputStr) {
38.         JSONObject jsonObject = null;
39.         StringBuffer buffer = new StringBuffer();
40.         try {
41.             // 创建 SSLContext 对象, 并使用我们指定的信任管理器初始化
42.             TrustManager[] tm = { new MyX509TrustManager() };
43.             SSLContext sslContext = SSLContext.getInstance("SSL", "SunJSSE");
44.             sslContext.init(null, tm, new java.security.SecureRandom());
45.             // 从上述 SSLContext 对象中得到 SSLSocketFactory 对象
46.             SSLSocketFactory ssf = sslContext.getSocketFactory();
47.
48.             URL url = new URL(requestUrl);
49.             HttpURLConnection httpUrlConn = (HttpURLConnection) url.openCo
nnection();
50.             httpUrlConn.setSSLSocketFactory(ssf);
51.
52.             httpUrlConn.setDoOutput(true);
53.             httpUrlConn.setDoInput(true);
54.             httpUrlConn.setUseCaches(false);
55.             // 设置请求方式 (GET/POST)
56.             httpUrlConn.setRequestMethod(requestMethod);
57.
58.             if ("GET".equalsIgnoreCase(requestMethod))
59.                 httpUrlConn.connect();
60.
61.             // 当有数据需要提交时
62.             if (null != outputStr) {
63.                 OutputStream outputStream = httpUrlConn.getOutputStream();
64.                 // 注意编码格式, 防止中文乱码
65.                 outputStream.write(outputStr.getBytes("UTF-8"));
66.                 outputStream.close();
67.             }
68.
69.             // 将返回的输入流转换成字符串
70.             InputStream inputStream = httpUrlConn.getInputStream();
71.             InputStreamReader inputStreamReader = new InputStreamReader(input
tStream, "utf-8");
72.             BufferedReader bufferedReader = new BufferedReader(inputStreamRe
ader);
73.
74.             String str = null;
75.             while ((str = bufferedReader.readLine()) != null) {
```

```

76.         buffer.append(str);
77.     }
78.     bufferedReader.close();
79.     inputStreamReader.close();
80.     // 释放资源
81.     inputStream.close();
82.     inputStream = null;
83.     httpUrlConn.disconnect();
84.     jsonObject = JSONObject.fromObject(buffer.toString());
85. } catch (ConnectException ce) {
86.     log.error("Weixin server connection timed out.");
87. } catch (Exception e) {
88.     log.error("https request error:{", e);
89. }
90.     return jsonObject;
91. }
92. }

```

代码说明：

- 1) 41~50 行：解决 https 请求的问题，很多人问题就出在这里；
- 2) 55~59 行：兼容 GET、POST 两种方式；
- 3) 61~67 行：兼容有数据提交、无数据提交两种情况，也有相当一部分人不知道如何 POST 提交数据；

Pojo 类的封装

在获取凭证创建菜单前，我们还需要封装一些 pojo，这会让我们代码更美观，有条理。

首先是调用获取凭证接口后，微信服务器会返回 json 格式的数据：

{"access_token":"ACCESS_TOKEN","expires_in":7200}，我们将其封装为一个 AccessToken 对象，对象有二个属性：token 和 expiresIn，代码如下：

[java] view plaincopy

```

1. package org.liufeng.weixin.pojo;
2.
3. /**
4.  * 微信通用接口凭证
5.  *
6.  * @author liufeng
7.  * @date 2013-08-08
8.  */
9. public class AccessToken {
10.     // 获取到的凭证
11.     private String token;

```

```

12. // 凭证有效时间, 单位: 秒
13. private int expiresIn;
14.
15. public String getToken() {
16.     return token;
17. }
18.
19. public void setToken(String token) {
20.     this.token = token;
21. }
22.
23. public int getExpiresIn() {
24.     return expiresIn;
25. }
26.
27. public void setExpiresIn(int expiresIn) {
28.     this.expiresIn = expiresIn;
29. }
30. }

```

接下来是对菜单结构的封装。因为我们是采用面向对象的编程方式，最终提交的 **json** 格式菜单数据就应该是由对象直接转换得到，而不是在程序代码中拼一大堆 **json** 数据。菜单结构封装的依据是公众平台 API 文档中给出的那一段 **json** 格式的菜单结构，如下所示：

[java] view plaincopy

```

1. {
2.     "button":[
3.         {
4.             "type":"click",
5.             "name":"今日歌曲",
6.             "key":"V1001_TODAY_MUSIC"
7.         },
8.         {
9.             "type":"click",
10.            "name":"歌手简介",
11.            "key":"V1001_TODAY_SINGER"
12.        },
13.        {
14.            "name":"菜单",
15.            "sub_button":[
16.                {
17.                    "type":"click",
18.                    "name":"hello word",
19.                    "key":"V1001_HELLO_WORLD"

```

```

20.         },
21.         {
22.             "type": "click",
23.             "name": "赞一下我们",
24.             "key": "V1001_GOOD"
25.         }]
26.     }]
27. }

```

首先是**菜单项的基类**，所有一级菜单、二级菜单都共有一个相同的属性，那就是 **name**。菜单项基类的封装代码如下：

[java] view plaincopy

```

1. package org.liufeng.weixin.pojo;
2.
3. /**
4.  * 按钮的基类
5.  *
6.  * @author liufeng
7.  * @date 2013-08-08
8.  */
9. public class Button {
10.     private String name;
11.
12.     public String getName() {
13.         return name;
14.     }
15.
16.     public void setName(String name) {
17.         this.name = name;
18.     }
19. }

```

接着是**子菜单项**的封装。这里对子菜单是这样定义的：没有子菜单的菜单项，有可能是二级菜单项，也有可能是不含二级菜单的一级菜单。这类子菜单项一定会包含三个属性：**type**、**name** 和 **key**，封装的代码如下：

[java] view plaincopy

```

1. package org.liufeng.weixin.pojo;
2.
3. /**
4.  * 普通按钮（子按钮）
5.  *
6.  * @author liufeng

```

```

7.  * @date 2013-08-08
8.  */
9.  public class CommonButton extends Button {
10.     private String type;
11.     private String key;
12.
13.     public String getType() {
14.         return type;
15.     }
16.
17.     public void setType(String type) {
18.         this.type = type;
19.     }
20.
21.     public String getKey() {
22.         return key;
23.     }
24.
25.     public void setKey(String key) {
26.         this.key = key;
27.     }
28. }

```

再往下是父菜单项的封装。对父菜单项的定义：包含有二级菜单项的一级菜单。这类菜单项包含有二个属性：name 和 sub_button，而 sub_button 以是一个子菜单项数组。父菜单项的封装代码如下：

[java] view plaincopy

```

1.  package org.liufeng.weixin.pojo;
2.
3.  /**
4.   * 复杂按钮（父按钮）
5.   *
6.   * @author liufeng
7.   * @date 2013-08-08
8.   */
9.  public class ComplexButton extends Button {
10.     private Button[] sub_button;
11.
12.     public Button[] getSub_button() {
13.         return sub_button;
14.     }
15.
16.     public void setSub_button(Button[] sub_button) {

```

```
17.         this.sub_button = sub_button;
18.     }
19. }
```

最后是整个**菜单对象**的封装，菜单对象包含多个菜单项（最多只能有 3 个），这些菜单项即可以是子菜单项（不含二级菜单的一级菜单），也可以是父菜单项（包含二级菜单的菜单项），如果能明白上面所讲的，再来看封装后的代码就很容易理解了：

[java] view plaincopy

```
1. package org.liufeng.weixin.pojo;
2.
3. /**
4.  * 菜单
5.  *
6.  * @author liufeng
7.  * @date 2013-08-08
8.  */
9. public class Menu {
10.     private Button[] button;
11.
12.     public Button[] getButton() {
13.         return button;
14.     }
15.
16.     public void setButton(Button[] button) {
17.         this.button = button;
18.     }
19. }
```

关于 POJO 类的封装就介绍完了。

凭证 access_token 的获取方法

继续在先前通用请求方法的类 WeixinUtil.java 中加入以下代码，用于获取接口访问凭证：

[java] view plaincopy

```
1. // 获取 access_token 的接口地址（GET） 限 200（次/天）
2. public final static String access_token_url = "https://api.weixin.qq.com/cgi-bin/token?grant_type=client_credential&appid=APPID&secret=APPSECRET";
3.
4. /**
5.  * 获取 access_token
6.  *
7.  * @param appid 凭证
```

```

8.  * @param appsecret 密钥
9.  * @return
10. */
11. public static AccessToken getAccessToken(String appid, String appsecret) {
12.     AccessToken accessToken = null;
13.
14.     String requestUrl = access_token_url.replace("APPID", appid).replace("APPSECRET", appsecret);
15.     JSONObject jsonObject = httpRequest(requestUrl, "GET", null);
16.     // 如果请求成功
17.     if (null != jsonObject) {
18.         try {
19.             accessToken = new AccessToken();
20.             accessToken.setToken(jsonObject.getString("access_token"));
21.             accessToken.setExpiresIn(jsonObject.getInt("expires_in"));
22.         } catch (JSONException e) {
23.             accessToken = null;
24.             // 获取 token 失败
25.             log.error("获取 token 失败 errcode:{} errmsg:{}", jsonObject.getInt("errcode"), jsonObject.getString("errmsg"));
26.         }
27.     }
28.     return accessToken;
29. }

```

自定义菜单的创建方法

继续在先前通用请求方法的类 `WeixinUtil.java` 中加入以下代码，用于创建自定义菜单：

[java] view plaincopy

```

1.  // 菜单创建（POST） 限 100（次/天）
2.  public static String menu_create_url = "https://api.weixin.qq.com/cgi-bin/menu/create?access_token=ACCESS_TOKEN";
3.
4.  /**
5.   * 创建菜单
6.   *
7.   * @param menu 菜单实例
8.   * @param accessToken 有效的 access_token
9.   * @return 0 表示成功，其他值表示失败
10. */
11. public static int createMenu(Menu menu, String accessToken) {
12.     int result = 0;

```



```

13.
14.     // 拼装创建菜单的 url
15.     String url = menu_create_url.replace("ACCESS_TOKEN", accessToken);
16.     // 将菜单对象转换成 json 字符串
17.     String jsonMenu = JSONObject.fromObject(menu).toString();
18.     // 调用接口创建菜单
19.     JSONObject jsonObject = httpRequest(url, "POST", jsonMenu);
20.
21.     if (null != jsonObject) {
22.         if (0 != jsonObject.getInt("errcode")) {
23.             result = jsonObject.getInt("errcode");
24.             log.error("创建菜单失败 errcode:{} errmsg:{}", jsonObject.getInt("errcode"), jsonObject.getString("errmsg"));
25.         }
26.     }
27.
28.     return result;
29. }

```

调用封装的方法创建自定义菜单

[java] view plaincopy

```

1. package org.liufeng.weixin.main;
2.
3. import org.liufeng.weixin.pojo.AccessToken;
4. import org.liufeng.weixin.pojo.Button;
5. import org.liufeng.weixin.pojo.CommonButton;
6. import org.liufeng.weixin.pojo.ComplexButton;
7. import org.liufeng.weixin.pojo.Menu;
8. import org.liufeng.weixin.util.WeixinUtil;
9. import org.slf4j.Logger;
10. import org.slf4j.LoggerFactory;
11.
12. /**
13.  * 菜单管理器类
14.  *
15.  * @author liufeng
16.  * @date 2013-08-08
17.  */
18. public class MenuManager {
19.     private static Logger log = LoggerFactory.getLogger(MenuManager.class);

```

```
20.
21.     public static void main(String[] args) {
22.         // 第三方用户唯一凭证
23.         String appId = "0000000000000000";
24.         // 第三方用户唯一凭证密钥
25.         String appSecret = "00000000000000000000000000000000";
26.
27.         // 调用接口获取 access_token
28.         AccessToken at = WeixinUtil.getAccessToken(appId, appSecret);
29.
30.         if (null != at) {
31.             // 调用接口创建菜单
32.             int result = WeixinUtil.createMenu(getMenu(), at.getToken());
33.
34.             // 判断菜单创建结果
35.             if (0 == result)
36.                 log.info("菜单创建成功!");
37.             else
38.                 log.info("菜单创建失败, 错误码: " + result);
39.         }
40.     }
41.
42.     /**
43.      * 组装菜单数据
44.      *
45.      * @return
46.      */
47.     private static Menu getMenu() {
48.         CommonButton btn11 = new CommonButton();
49.         btn11.setName("天气预报");
50.         btn11.setType("click");
51.         btn11.setKey("11");
52.
53.         CommonButton btn12 = new CommonButton();
54.         btn12.setName("公交查询");
55.         btn12.setType("click");
56.         btn12.setKey("12");
57.
58.         CommonButton btn13 = new CommonButton();
59.         btn13.setName("周边搜索");
60.         btn13.setType("click");
61.         btn13.setKey("13");
62.
63.         CommonButton btn14 = new CommonButton();
```

```
64.         btn14.setName("历史上的今天");
65.         btn14.setType("click");
66.         btn14.setKey("14");
67.
68.         CommonButton btn21 = new CommonButton();
69.         btn21.setName("歌曲点播");
70.         btn21.setType("click");
71.         btn21.setKey("21");
72.
73.         CommonButton btn22 = new CommonButton();
74.         btn22.setName("经典游戏");
75.         btn22.setType("click");
76.         btn22.setKey("22");
77.
78.         CommonButton btn23 = new CommonButton();
79.         btn23.setName("美女电台");
80.         btn23.setType("click");
81.         btn23.setKey("23");
82.
83.         CommonButton btn24 = new CommonButton();
84.         btn24.setName("人脸识别");
85.         btn24.setType("click");
86.         btn24.setKey("24");
87.
88.         CommonButton btn25 = new CommonButton();
89.         btn25.setName("聊天唠嗑");
90.         btn25.setType("click");
91.         btn25.setKey("25");
92.
93.         CommonButton btn31 = new CommonButton();
94.         btn31.setName("Q 友圈");
95.         btn31.setType("click");
96.         btn31.setKey("31");
97.
98.         CommonButton btn32 = new CommonButton();
99.         btn32.setName("电影排行榜");
100.         btn32.setType("click");
101.         btn32.setKey("32");
102.
103.         CommonButton btn33 = new CommonButton();
104.         btn33.setName("幽默笑话");
105.         btn33.setType("click");
106.         btn33.setKey("33");
107.
```

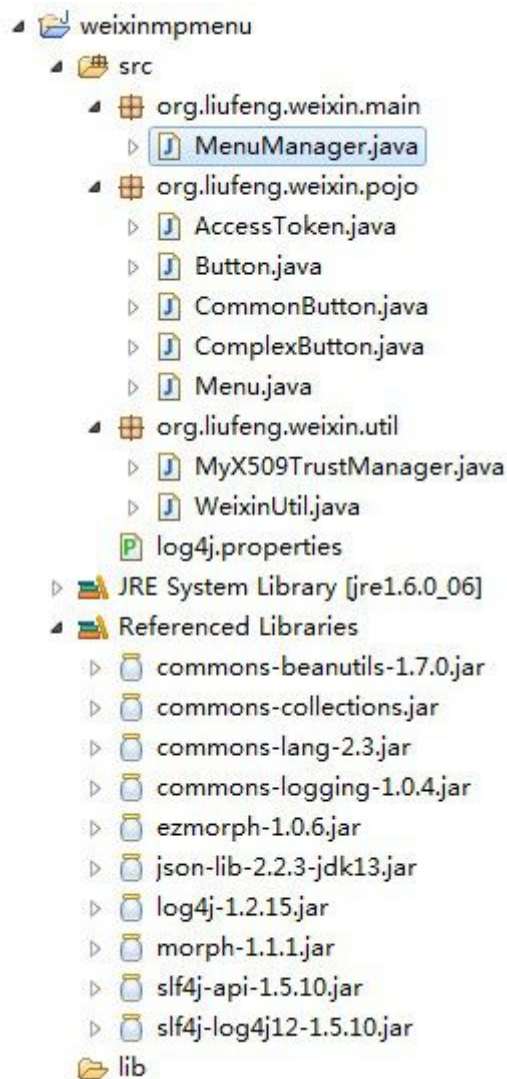
```

108.         ComplexButton mainBtn1 = new ComplexButton();
109.         mainBtn1.setName("生活助手");
110.         mainBtn1.setSub_button(new CommonButton[] { btn11, btn12, btn13, bt
            n14 });
111.
112.         ComplexButton mainBtn2 = new ComplexButton();
113.         mainBtn2.setName("休闲驿站");
114.         mainBtn2.setSub_button(new CommonButton[] { btn21, btn22, btn23, bt
            n24, btn25 });
115.
116.         ComplexButton mainBtn3 = new ComplexButton();
117.         mainBtn3.setName("更多体验");
118.         mainBtn3.setSub_button(new CommonButton[] { btn31, btn32, btn33 });
119.
120.         /**
121.          * 这是公众号 xiaoqrobot 目前的菜单结构，每个一级菜单都有二级菜单项<br>
122.          *
123.          * 在某个一级菜单下没有二级菜单的情况，menu 该如何定义呢？<br>
124.          * 比如，第三个一级菜单项不是“更多体验”，而直接是“幽默笑话”，那么 menu 应该
            这样定义：<br>
125.          * menu.setButton(new Button[] { mainBtn1, mainBtn2, btn33 });
126.          */
127.         Menu menu = new Menu();
128.         menu.setButton(new Button[] { mainBtn1, mainBtn2, mainBtn3 });
129.
130.         return menu;
131.     }
132. }

```

注意：在运行以上代码时，需要将 appId 和 appSecret 换成你自己公众号的。

整个工程的结构



为了保证文章的完整独立性和可读性，我是新建了一个 **Java Project**（**Java web 工程**也可以，没有太大关系），没有在前几篇文章所讲到的 **weixinCourse** 工程中添加代码。如果需要，读者可以自己实现将菜单创建的代码移到自己已有的工程中去。

图中所有 **Java** 文件的源代码都在文章中贴出并进行了说明，图中使用到的 **jar** 也是 **Java** 开发中通用的 **jar** 包，很容易在网上下载到。

工程中引入的 **jar** 包主要分为两类：

- 1) 第一类是 **json** 开发工具包，用于 **Java** 对象和 **Json** 字符串之间的转换；**json** 开发工具包一共有 3 个 **jar**：**ezmorph-1.0.6.jar**，**json-lib-2.2.3-jdk13.jar** 和 **morph-1.1.1.jar**。
- 2) 第二类是 **slf4j** 日志工具包，用于记录系统运行所产生的日志，日志可以输出到控制台或文件中。

整个工程中，唯一没有讲到的是 **src** 下的 **log4j.properties** 的配置，也把它贴出来，方便大家参考，这样才是一个完整的工程源码。**log4j.properties** 文件的内容如下：

[java] view plaincopy

```
1. log4j.rootLogger=info,console,file
2.
3. log4j.appender.console=org.apache.log4j.ConsoleAppender
4. log4j.appender.console.layout=org.apache.log4j.PatternLayout
5. log4j.appender.console.layout.ConversionPattern=[%-5p] %m%n
6.
7. log4j.appender.file=org.apache.log4j.DailyRollingFileAppender
8. log4j.appender.file.DatePattern='-'yyyy-MM-dd
9. log4j.appender.file.File=./logs/weixinmpmenu.log
10. log4j.appender.file.Append=true
11. log4j.appender.file.layout=org.apache.log4j.PatternLayout
12. log4j.appender.file.layout.ConversionPattern=[%-5p] %d %37c %3x - %m%n
```

如何响应菜单点击事件

自定义菜单的创建工作已经完成，那么该如何接收和响应菜单的点击事件呢，也就是说在公众帐号后台处理程序中，如何识别用户点击的是哪个菜单，以及做出响应。这部分内容其实在教程的第 5 篇 [各种消息的接收与响应](#) 中已经讲解清楚了。

来看一下第一篇教程 `weixinCourse` 项目中的 `CoreService` 类要怎么改写，才能接收响应菜单点击事件，该类修改后的完整代码如下：

[java] view plaincopy

```
1. package org.liufeng.course.service;
2.
3. import java.util.Date;
4. import java.util.Map;
5.
6. import javax.servlet.http.HttpServletRequest;
7.
8. import org.liufeng.course.message.resp.TextMessage;
9. import org.liufeng.course.util.MessageUtil;
10.
11. /**
12.  * 核心服务类
13.  *
14.  * @author liufeng
15.  * @date 2013-05-20
16.  */
17. public class CoreService {
18.     /**
19.      * 处理微信发来的请求
20.      *
21.      * @param request
```

```
22.     * @return
23.     */
24.     public static String processRequest(HttpServletRequest request) {
25.         String respMessage = null;
26.         try {
27.             // 默认返回的文本消息内容
28.             String respContent = "请求处理异常，请稍候尝试！";
29.
30.             // xml 请求解析
31.             Map<String, String> requestMap = MessageUtil.parseXml(request);
32.
33.             // 发送方帐号 (open_id)
34.             String fromUserName = requestMap.get("FromUserName");
35.             // 公众帐号
36.             String toUserName = requestMap.get("ToUserName");
37.             // 消息类型
38.             String msgType = requestMap.get("MsgType");
39.
40.             // 回复文本消息
41.             TextMessage textMessage = new TextMessage();
42.             textMessage.setToUserName(fromUserName);
43.             textMessage.setFromUserName(toUserName);
44.             textMessage.setCreateTime(new Date().getTime());
45.             textMessage.setMsgType(MessageUtil.RESP_MESSAGE_TYPE_TEXT);
46.             textMessage.setFuncFlag(0);
47.
48.             // 文本消息
49.             if (msgType.equals(MessageUtil.REQ_MESSAGE_TYPE_TEXT)) {
50.                 respContent = "您发送的是文本消息！";
51.             }
52.             // 图片消息
53.             else if (msgType.equals(MessageUtil.REQ_MESSAGE_TYPE_IMAGE)) {
54.                 respContent = "您发送的是图片消息！";
55.             }
56.             // 地理位置消息
57.             else if (msgType.equals(MessageUtil.REQ_MESSAGE_TYPE_LOCATION))
58.             {
59.                 respContent = "您发送的是地理位置消息！";
60.             }
61.             // 链接消息
62.             else if (msgType.equals(MessageUtil.REQ_MESSAGE_TYPE_LINK)) {
63.                 respContent = "您发送的是链接消息！";
64.             }
65.         }
66.     }
67. }
```

```

64.         // 音频消息
65.         else if (msgType.equals(MessageUtil.REQ_MESSAGE_TYPE_VOICE)) {
66.             respContent = "您发送的是音频消息！";
67.         }
68.         // 事件推送
69.         else if (msgType.equals(MessageUtil.REQ_MESSAGE_TYPE_EVENT)) {
70.             // 事件类型
71.             String eventType = requestMap.get("Event");
72.             // 订阅
73.             if (eventType.equals(MessageUtil.EVENT_TYPE_SUBSCRIBE)) {
74.                 respContent = "感谢您的关注！";
75.             }
76.             // 取消订阅
77.             else if (eventType.equals(MessageUtil.EVENT_TYPE_UNSUBSCRIBE)) {
78.                 // TODO 取消订阅后用户再收不到公众号发送的消息，因此不需要回复消息
79.             }
80.             // 自定义菜单点击事件
81.             else if (eventType.equals(MessageUtil.EVENT_TYPE_CLICK)) {
82.                 // 事件 KEY 值，与创建自定义菜单时指定的 KEY 值对应
83.                 String eventKey = requestMap.get("EventKey");
84.
85.                 if (eventKey.equals("11")) {
86.                     respContent = "天气预报菜单项被点击！";
87.                 } else if (eventKey.equals("12")) {
88.                     respContent = "公交查询菜单项被点击！";
89.                 } else if (eventKey.equals("13")) {
90.                     respContent = "周边搜索菜单项被点击！";
91.                 } else if (eventKey.equals("14")) {
92.                     respContent = "历史上的今天菜单项被点击！";
93.                 } else if (eventKey.equals("21")) {
94.                     respContent = "歌曲点播菜单项被点击！";
95.                 } else if (eventKey.equals("22")) {
96.                     respContent = "经典游戏菜单项被点击！";
97.                 } else if (eventKey.equals("23")) {
98.                     respContent = "美女电台菜单项被点击！";
99.                 } else if (eventKey.equals("24")) {
100.                     respContent = "人脸识别菜单项被点击！";
101.                 } else if (eventKey.equals("25")) {
102.                     respContent = "聊天唠嗑菜单项被点击！";
103.                 } else if (eventKey.equals("31")) {
104.                     respContent = "Q 友圈菜单项被点击！";
105.                 } else if (eventKey.equals("32")) {

```



```

106.                respContent = "电影排行榜菜单项被点击! ";
107.            } else if (eventKey.equals("33")) {
108.                respContent = "幽默笑话菜单项被点击! ";
109.            }
110.        }
111.    }
112.
113.        textMessage.setContent(respContent);
114.        respMessage = MessageUtil.textMessageToXml(textMessage);
115.    } catch (Exception e) {
116.        e.printStackTrace();
117.    }
118.
119.    return respMessage;
120. }
121. }

```

代码说明：

- 1) 第 69 行、第 81 行这两行代码说明了如何判断菜单的点击事件。当消息类型 `MsgType=event`，并且 `Event=CLICK` 时，就表示是自定义菜单点击事件；
- 2) 第 83 行是判断具体点击的是哪个菜单项，根据菜单的 `key` 值来判断；
- 3) 第 85~109 行表示当用户点击某个菜单项后，具体返回什么消息，我只是做个简单示例，统一返回文本消息，读者可以根据实际需要来灵活处理。

总结

到这里关于自定义菜单的创建、菜单事件的判断和处理响应就全部介绍完了。我只希望看过文章的人不要只是拷贝代码，如果是这样，我完全不用花这么多的时间来写这篇文章，直接把工程放在下载区多简单。另外，网上是有很多工具，让你填入 `appid`, `appsecret` 和菜单结构，提交就能创建菜单，请慎用！因为 `appid` 和 `appsecret` 一旦告诉别人，你的公众号的菜单控制权就在别人手上了，总会有别有用心的人出来搞点事的。

微信公众帐号开发教程第 15 篇-自定义菜单的 view 类型（访问网页）

引言及内容概要

距离写上一篇文章《自定义菜单的创建及菜单事件响应》整整过了两个月的时间，那时公众平台还没有开放 `view` 类型的菜单。在不久前，微信公众平台悄悄开放了 `view` 类型的菜单，却没有在首页发布任何通知，貌似微信团队很喜欢这么干。一个偶然的的机会，我留意到 API 文档的自定义菜单接口发生了变化，增加了对菜单 `view` 类型的说明：

view（访问网页）：

用户点击 view 类型按钮后，会直接跳转到开发者指定的 url 中。

于是我在第一时间更新了小q机器人（微信号：xiaoqrobot）的菜单，在一级菜单“更多”下增加了二级菜单“使用帮助”，点击该菜单项会直接跳转到网页，如下图所示。



最近也有不少网友问起这种类型的菜单是如何创建的，[本篇文章](#)就为大家介绍下 **view** 类型的自定义菜单该如何创建。

自定义菜单的两种类型（click 和 view）

公众平台 API 文档中给出了自定义菜单的 json 结构示例，我从中截取两个菜单项的 json 代码，一个是 click 类型，另一个是 view 类型，如下所示。

[html] view plaincopy

```
1.  {
2.    "type":"click",
3.    "name":"今日歌曲",
4.    "key":"V1001_TODAY_MUSIC"
5.  },
6.  {
7.    "type":"view",
8.    "name":"歌手简介",
9.    "url":"http://www.qq.com/"
```

```
10. }
```

从上面可以看出，两种类型的菜单除了 **type** 值不同之外，属性也有差别。**click** 类型的菜单有 **key** 属性，而 **view** 类型的菜单没有 **key** 属性，与之对应的是 **url** 属性。通过上一篇文章的学习我们知道，**key** 值是用于判断用户点击了哪个 **click** 类型的菜单项。而 **view** 类型的菜单没有 **key** 属性，目前无法在公众账号后台判断是否有用户点击了 **view** 类型的菜单项，也就没办法知道哪个用户点击了 **view** 类型的菜单项。

建立 **view** 类型的菜单对象

View 类型的菜单有 3 个属性：**type**、**name** 和 **url**。在[上一篇文章](#)中，我们创建了菜单项的基类 **Button**，**Button** 类只有一个属性 **name**。**View** 类型的菜单对象也需要继承 **Button** 类，代码如下：

```
[java] view plaincopy
```

```
1. package org.liufeng.weixin.pojo;
2.
3. /**
4.  * view 类型的菜单
5.  *
6.  * @author liuyq
7.  * @date 2013-04-10
8.  */
9. public class ViewButton extends Button {
10.     private String type;
11.     private String url;
12.
13.     public String getType() {
14.         return type;
15.     }
16.
17.     public void setType(String type) {
18.         this.type = type;
19.     }
20.
21.     public String getUrl() {
22.         return url;
23.     }
24.
25.     public void setUrl(String url) {
26.         this.url = url;
27.     }
28. }
```

创建带 view 类型的菜单示例

我们对[前一篇文章](#)中给出的菜单创建代码进行调整，增加 view 类型的菜单项，完整的菜单创建代码如下：

[java] view plaincopy

```
1. package org.liufeng.weixin.main;
2.
3. import org.liufeng.weixin.pojo.AccessToken;
4. import org.liufeng.weixin.pojo.Button;
5. import org.liufeng.weixin.pojo.CommonButton;
6. import org.liufeng.weixin.pojo.ComplexButton;
7. import org.liufeng.weixin.pojo.Menu;
8. import org.liufeng.weixin.pojo.ViewButton;
9. import org.liufeng.weixin.util.WeixinUtil;
10. import org.slf4j.Logger;
11. import org.slf4j.LoggerFactory;
12.
13. /**
14.  * 菜单管理器类
15.  *
16.  * @author liufeng
17.  * @date 2013-08-08
18.  */
19. public class MenuManager {
20.     private static Logger log = LoggerFactory.getLogger(MenuManager.class);
21.
22.     public static void main(String[] args) {
23.         // 第三方用户唯一凭证
24.         String appId = "000000000000000000";
25.         // 第三方用户唯一凭证密钥
26.         String appSecret = "00000000000000000000000000000000";
27.
28.         // 调用接口获取 access_token
29.         AccessToken at = WeixinUtil.getAccessToken(appId, appSecret);
30.
31.         if (null != at) {
32.             // 调用接口创建菜单
33.             int result = WeixinUtil.createMenu(getMenu(), at.getToken());
34.
35.             // 判断菜单创建结果
36.             if (0 == result)
37.                 log.info("菜单创建成功！");
```

```
38.         else
39.             log.info("菜单创建失败，错误码: " + result);
40.     }
41. }
42.
43. /**
44.  * 组装菜单数据
45.  *
46.  * @return
47.  */
48. private static Menu getMenu() {
49.     CommonButton btn11 = new CommonButton();
50.     btn11.setName("天气预报");
51.     btn11.setType("click");
52.     btn11.setKey("11");
53.
54.     CommonButton btn12 = new CommonButton();
55.     btn12.setName("公交查询");
56.     btn12.setType("click");
57.     btn12.setKey("12");
58.
59.     CommonButton btn13 = new CommonButton();
60.     btn13.setName("周边搜索");
61.     btn13.setType("click");
62.     btn13.setKey("13");
63.
64.     CommonButton btn14 = new CommonButton();
65.     btn14.setName("历史上的今天");
66.     btn14.setType("click");
67.     btn14.setKey("14");
68.
69.     CommonButton btn15 = new CommonButton();
70.     btn15.setName("电影排行榜");
71.     btn15.setType("click");
72.     btn15.setKey("32");
73.
74.     CommonButton btn21 = new CommonButton();
75.     btn21.setName("歌曲点播");
76.     btn21.setType("click");
77.     btn21.setKey("21");
78.
79.     CommonButton btn22 = new CommonButton();
80.     btn22.setName("经典游戏");
81.     btn22.setType("click");
```

```
82.         btn22.setKey("22");
83.
84.         CommonButton btn23 = new CommonButton();
85.         btn23.setName("美女电台");
86.         btn23.setType("click");
87.         btn23.setKey("23");
88.
89.         CommonButton btn24 = new CommonButton();
90.         btn24.setName("人脸识别");
91.         btn24.setType("click");
92.         btn24.setKey("24");
93.
94.         CommonButton btn25 = new CommonButton();
95.         btn25.setName("聊天唠嗑");
96.         btn25.setType("click");
97.         btn25.setKey("25");
98.
99.         CommonButton btn31 = new CommonButton();
100.        btn31.setName("Q 友圈");
101.        btn31.setType("click");
102.        btn31.setKey("31");
103.
104.        CommonButton btn33 = new CommonButton();
105.        btn33.setName("幽默笑话");
106.        btn33.setType("click");
107.        btn33.setKey("33");
108.
109.        CommonButton btn34 = new CommonButton();
110.        btn34.setName("用户反馈");
111.        btn34.setType("click");
112.        btn34.setKey("34");
113.
114.        CommonButton btn35 = new CommonButton();
115.        btn35.setName("关于我们");
116.        btn35.setType("click");
117.        btn35.setKey("35");
118.
119.        ViewButton btn32 = new ViewButton();
120.        btn32.setName("使用帮助");
121.        btn32.setType("view");
122.        btn32.setUrl("http://liufeng.gotoip2.com/xiaoqrobot/help.jsp");
123.
124.        ComplexButton mainBtn1 = new ComplexButton();
125.        mainBtn1.setName("生活助手");
```

```

126.         mainBtn1.setSub_button(new Button[] { btn11, btn12, btn13, btn14, b
            tn15 });
127.
128.         ComplexButton mainBtn2 = new ComplexButton();
129.         mainBtn2.setName("休闲驿站");
130.         mainBtn2.setSub_button(new Button[] { btn21, btn22, btn23, btn24, b
            tn25 });
131.
132.         ComplexButton mainBtn3 = new ComplexButton();
133.         mainBtn3.setName("更多");
134.         mainBtn3.setSub_button(new Button[] { btn31, btn33, btn34, btn35, b
            tn32 });
135.
136.         /**
137.          * 这是公众号 xiaoqrobot 目前的菜单结构，每个一级菜单都有二级菜单项<br>
138.          *
139.          * 在某个一级菜单下没有二级菜单的情况，menu 该如何定义呢？<br>
140.          * 比如，第三个一级菜单项不是“更多体验”，而直接是“幽默笑话”，那么 menu 应该
            这样定义：<br>
141.          * menu.setButton(new Button[] { mainBtn1, mainBtn2, btn33 });
142.          */
143.         Menu menu = new Menu();
144.         menu.setButton(new Button[] { mainBtn1, mainBtn2, mainBtn3 });
145.
146.         return menu;
147.     }
148. }

```

119~122 行代码就是用于创建 **view** 类型菜单项的。上面的菜单结构也是小 q 机器人（微信号：xiaoqrobot）目前正在使用的，读者可以对照着理解。

补充：居然还有些网友问我上面的自定义菜单创建代码在哪里调用，问我为什么不把调用的代码也公开？搞的我都有点不好意思了。上面的菜单创建代码是写在 **main** 方法中，直接在开发工具中运行一下就可以了，这应该是最基础的 **Java** 知识了。另外，菜单只要创建一次，会一直存在的，不需要每次启动应用程序都去创建菜单。当然，你也可以把菜单的创建代码集成到项目中，并非一定要放在 **main** 方法中。程序是死的，人是活的，解决方法往往有很多种，怎么方便实用就怎么来！

微信公众帐号开发教程第 16 篇-应用实例之历史上的今天

[内容概要](#)

本篇文章主要讲解如何在微信公众帐号上实现“历史上的今天”功能。这个例子本身并不复杂，但希望通过对它的学习，读者能够对正则表达式有一个新的认识，能够学会运用现有的网络资源丰富自己的公众账号。

何谓历史上的今天

回顾历史的长河，历史是生活的一面镜子；以史为鉴，可以知兴衰；历史上的每一天，都是喜忧参半；可以了解历史的这一天发生的事件，借古可以鉴今，历史是不能忘记的。[查看历史上每天发生的重大事情，增长知识，开拓眼界，提高人文素养。](#)

寻找接口（数据源）

要实现查询“历史上的今天”，首先我们要找到相关数据源。笔者经过搜索发现，网络上几乎没有现成的“历史上的今天”API 可以使用，所以我们只能通过爬取、解析网页源代码的方式得到我们需要的数据。笔者发现网站 <http://www.rjiben.com/> 上包含“历史上的今天”功能，就用它做数据源了。

开发步骤

为了便于读者理解，我们需要清楚该应用实例的开发步骤，主要如下：

- 1) 发起 HTTP GET 请求，获取网页源代码。
- 2) 运用正则表达式从网页源代码中抽取我们需要的数据。
- 3) 对抽取得到的数据进行加工（使内容呈现更加美观）。
- 4) 将以上三步进行封装，供外部调用。
- 5) 在公众账号后台调用封装好的“历史上的今天”查询方法。

代码实现

笔者将上述步骤 1)、2)、3) 中的代码实现封装成了 `TodayInHistoryService` 类，并对外提供了 `getTodayInHistory()` 方法来获取“历史上的今天”。实现代码如下：

[java] view plaincopy

```
1. import java.io.BufferedReader;
```



```
2. import java.io.InputStream;
3. import java.io.InputStreamReader;
4. import java.net.HttpURLConnection;
5. import java.net.URL;
6. import java.text.DateFormat;
7. import java.text.SimpleDateFormat;
8. import java.util.Calendar;
9. import java.util.regex.Matcher;
10. import java.util.regex.Pattern;
11.
12. /**
13.  * 历史上的今天查询服务
14.  *
15.  * @author liufeng
16.  * @date 2013-10-16
17.  *
18.  */
19. public class TodayInHistoryService {
20.
21.     /**
22.      * 发起 http get 请求获取网页源代码
23.      *
24.      * @param requestUrl
25.      * @return
26.      */
27.     private static String httpRequest(String requestUrl) {
28.         StringBuffer buffer = null;
29.
30.         try {
31.             // 建立连接
32.             URL url = new URL(requestUrl);
33.             HttpURLConnection httpUrlConn = (HttpURLConnection) url.openConnection();
34.             httpUrlConn.setDoInput(true);
35.             httpUrlConn.setRequestMethod("GET");
36.
37.             // 获取输入流
38.             InputStream inputStream = httpUrlConn.getInputStream();
39.             InputStreamReader inputStreamReader = new InputStreamReader(inputStream, "utf-8");
40.             BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
41.
42.             // 读取返回结果
```

```

43.         buffer = new StringBuffer();
44.         String str = null;
45.         while ((str = bufferedReader.readLine()) != null) {
46.             buffer.append(str);
47.         }
48.
49.         // 释放资源
50.         bufferedReader.close();
51.         inputStreamReader.close();
52.         inputStream.close();
53.         httpUrlConn.disconnect();
54.     } catch (Exception e) {
55.         e.printStackTrace();
56.     }
57.     return buffer.toString();
58. }
59.
60. /**
61.  * 从 html 中抽取历史上的今天信息
62.  *
63.  * @param html
64.  * @return
65.  */
66. private static String extract(String html) {
67.     StringBuffer buffer = null;
68.     // 日期标签: 区分是昨天还是今天
69.     String dateTag = getMonthDay(0);
70.
71.     Pattern p = Pattern.compile("(.*)(<div class=\"listren\">(.?*)(</div>)(.*)");
72.     Matcher m = p.matcher(html);
73.     if (m.matches()) {
74.         buffer = new StringBuffer();
75.         if (m.group(3).contains(getMonthDay(-1)))
76.             dateTag = getMonthDay(-1);
77.
78.         // 拼装标题
79.         buffer.append("== ").append("历史上的
").append(dateTag).append(" ==").append("\n\n");
80.
81.         // 抽取需要的数据
82.         for (String info : m.group(3).split(" ")) {
83.             info = info.replace(dateTag, "").replace(" (图)
", "").replaceAll("</?[^>]+>", "").trim();

```

```
84.          // 在每行末尾追加 2 个换行符
85.          if (!"".equals(info)) {
86.              buffer.append(info).append("\n\n");
87.          }
88.      }
89.  }
90.  // 将 buffer 最后两个换行符移除并返回
91.  return (null == buffer) ? null : buffer.substring(0, buffer.lastIndexOf("\n\n"));
92.  }
93.
94.  /**
95.   * 获取前/后 n 天日期(M 月 d 日)
96.   *
97.   * @return
98.   */
99.  private static String getMonthDay(int diff) {
100.      DateFormat df = new SimpleDateFormat("M 月 d 日");
101.      Calendar c = Calendar.getInstance();
102.      c.add(Calendar.DAY_OF_YEAR, diff);
103.      return df.format(c.getTime());
104.  }
105.
106.  /**
107.   * 封装历史上的今天查询方法，供外部调用
108.   *
109.   * @return
110.   */
111.  public static String getTodayInHistoryInfo() {
112.      // 获取网页源代码
113.      String html = httpRequest("http://www.rijiben.com/");
114.      // 从网页中抽取信息
115.      String result = extract(html);
116.
117.      return result;
118.  }
119.
120.  /**
121.   * 通过 main 在本地测试
122.   *
123.   * @param args
124.   */
125.  public static void main(String[] args) {
126.      String info = getTodayInHistoryInfo();
```

```
127.         System.out.println(info);
128.     }
129. }
```

代码解读：

- 1) 27-58 行代码是 `httpRequest()` 方法，用于发起 `http get` 请求，获取指定 `url` 的网页源代码。
- 2) 66-92 行代码是 `extract()` 方法，运用正则表达式从网页源代码中抽取“历史上的今天”数据。
- 3) 111-118 行代码是 `getTodayInHistory()` 方法，封装给外部调用查询“历史上的今天”。
- 4) 125-128 行代码是 `main` 方法，用于在本地的开发工具中测试。
- 5) 75-76 行代码的作用是判断获取到的“历史上的今天”数据是当天的还是前一天的（因为不能保证 `www.rjiben.com` 上的数据一定在凌晨零点准时更新，所以为了保证数据的准确性必须做此判断）。
- 6) 第 71 行代码是本文的重点，笔者编写的正则表达式规则是“`(.*)(<div class=\"listtren\">)(.*?)(</div>)(.*)`”。正则表达式规则需要根据网页源代码进行编写的，特别是包含“历史上的今天”数据的那部分 `HTML` 标签，所以我们先来查看网页源代码。通过 `httpRequest("http://www.rjiben.com/")` 方法获取到的网页源代码，与我们通过浏览器访问 `http://www.rjiben.com/` 页面再点击右键选择“查看网页源代码”所得到的结果完全一致。我们通过浏览器查看 `http://www.rjiben.com/` 的网页源代码，然后找到“历史上的今天”数据所在位置，如下图所示：

```

<div class="listren">
<ul>

<li><a href="/news6836/" title="0690年10月16日 武则天登上皇位">0690年10月16日 武则天登上皇位</a>&nbsp;&nbsp;&nbsp;</li>
<li><a href="/news6837/" title="1854年10月16日 唯美主义运动的倡导者王尔德诞辰">1854年10月16日 唯美主义运动的倡
<li><a href="/news6838/" title="1854年10月16日 德国社会主义活动家考茨基诞生">1854年10月16日 德国社会主义活动家
<li><a href="/news6839/" title="1908年10月16日 阿尔巴尼亚领导人恩维尔·霍查诞辰">1908年10月16日 阿尔巴尼亚领导
<li><a href="/news6840/" title="1913年10月16日 中国“两弹一星”元勋钱三强诞辰">1913年10月16日 中国“两弹一星”
<li><a href="/news6841/" title="1922年10月16日 开滦煤矿工人罢工失败">1922年10月16日 开滦煤矿工人罢工失败</a>&nbsp;&nbsp;&nbsp;
<li><a href="/news6842/" title="1927年10月16日 德国诺贝尔文学奖得主格拉斯诞生">1927年10月16日 德国诺贝尔文学奖
<li><a href="/news6843/" title="1933年10月16日 抗日同盟军失败">1933年10月16日 抗日同盟军失败</a>&nbsp;&nbsp;&nbsp;</li>
<li><a href="/news6844/" title="1950年10月16日 人民解放军进军西藏">1950年10月16日 人民解放军进军西藏</a>&nbsp;&nbsp;&nbsp;
<li><a href="/news6845/" title="1954年10月16日 俞平伯《关于红楼梦研究问题的信》发表">1954年10月16日 俞平伯《关
<li><a href="/news6846/" title="1959年10月16日 美军将领、国务卿马歇尔去世">1959年10月16日 美军将领、国务卿马歇
<li><a href="/news6847/" title="1964年10月16日 勃列日涅夫取代赫鲁晓夫 成为苏共中央第一书记">1964年10月16日 勃
<li><a href="/news6848/" title="1964年10月16日 我国第一颗原子弹爆炸成功">1964年10月16日 我国第一颗原子弹爆炸成
<li><a href="/news6849/" title="1973年10月16日 震撼世界的石油危机爆发">1973年10月16日 震撼世界的石油危机爆发</
<li><a href="/news6850/" title="1978年10月16日 约翰·保罗二世当选新教皇">1978年10月16日 约翰·保罗二世当选新教
<li><a href="/news6851/" title="1979年10月16日 哈克将军宣布巴基斯坦推迟大选解散政党">1979年10月16日 哈克将军宣

```

从上面的源代码截图中可以看到，我们需要的数据被包含在`<div class="listren">`标签内，

这样就不难理解为什么正则表达式要这样写：

```
(.*)<div class=\"listren\">(.?*)</div>(.*)
```

我们使用括号()将正则表达式规则分成了 5 组，下面是这些分组的说明：

第 1 组：(.)表示网页源代码中`<div class="listren">`标签之前还有任意多个字符。

第 2 组：(`<div class=\"listren\">`)中的反斜杠表示转义，所以该规则就是用于匹配`<div class="listren">`。

第 3 组：(.?)表示在标签`<div class="listren">`和`</div>`之间的所有内容，这才是我们真正需要的数据所在。

第 4 组：(`</div>`)就是用于匹配`<div class="listren">`的结束标签。

第 5 组：(.)表示在`</div>`标签之后还有任意多的字符。

掌握了正则表达式规则的含义，就不难理解为什么在 `extract()` 方法中全都是在使用 `m.group(3)`，因为 `m.group(3)` 就表示匹配到数据的第 3 个分组。`m.group(3)` 的内容如下：

[html] view plaincopy

```
1. <ul>                                <li><a href="/news6836/" title="0690年10月16日 武则天登上皇位">0690年10月16日 武则天登上皇位</a>    (图)
</li>                                <li><a href="/news6837/" title="1854年10月16日 唯美主义运动的倡导者王尔德诞辰">1854年10月16日 唯美主义运动的倡导者王尔德诞辰
</a> </li>                            <li><a href="/news6838/" title="1854年10月16日 德国社会主义活动家考茨基诞生">1854年10月16日 德国社会主义活动家考茨基诞生
</a> </li>                            <li><a href="/news6839/" title="1908年10月16日 阿尔巴尼亚领导人恩维尔·霍查诞辰">1908年10月16日 阿尔巴尼亚领导人恩维尔·霍查诞辰</a>    (图)</li>
                                <li><a href="/news6840/" title="1913年10月16日 中国“两弹一星”元勋钱三强诞辰">1913年10月16日 中国“两弹一星”元勋钱三强诞辰</a>    (图)
</li>                                <li><a href="/news6841/" title="1922年10月16日 开滦煤矿工人罢工失败">1922年10月16日 开滦煤矿工人罢工失败</a>    (图)
</li>                                <li><a href="/news6842/" title="1927年10月16日 德国诺贝尔文学奖得主格拉斯诞生">1927年10月16日 德国诺贝尔文学奖得主格拉斯诞生
</a>    (图) </li>                            <li><a href="/news6843/" title="1933年10月16日 抗日同盟军失败">1933年10月16日 抗日同盟军失败</a>    (图)
</li>                                <li><a href="/news6844/" title="1950年10月16日 人民解放军进军西藏">1950年10月16日 人民解放军进军西藏</a>    (图)
</li>                                <li><a href="/news6845/" title="1954年10月16日 俞平伯《关于红楼梦研究问题的信》发表">1954年10月16日 俞平伯《关于红楼梦研究问题的信》发表</a>    (图) </li>
                                <li><a href="/news6846/" title="1959年10月16日 美军将领、国务卿马歇尔去世">1959年10月16日 美军将领、国务卿马歇尔去世</a>    (图) </li>
                                <li><a href="/news6847/" title="1964年10月16日 勃列日涅夫取代赫鲁晓夫 成为苏共中央第一书记">1964年10月16日 勃列日涅夫取代赫鲁晓夫 成为苏共中央第一书记
</a> </li>                            <li><a href="/news6848/" title="1964年10月16日 我国第一颗原子弹爆炸成功">1964年10月16日 我国第一颗原子弹爆炸成功</a>    (图)
</li>                                <li><a href="/news6849/" title="1973年10月16日 震撼世界的石油危机爆发">1973年10月16日 震撼世界的石油危机爆发</a>    (图)
</li>                                <li><a href="/news6850/" title="1978年10月16日 约翰·保罗二世当选新教皇">1978年10月16日 约翰·保罗二世当选新教皇
</a> </li>                            <li><a href="/news6851/" title="1979年10月16日 哈克将军宣布巴基斯坦推迟大选解散政党">1979年10月16日 哈克将军宣布巴基斯坦推迟大选解散政党
</a> </li>                            <li><a href="/news6852/" title="1984年10月16日 图图主教荣获“诺贝尔和平奖”">1984年10月16日 图图主教荣获“诺贝尔和平
奖”</a> </li>                                <li><a href="/news6853/" title="1988年10
```

```

月 16 日 北京正负电子对撞机对撞成功">1988 年 10 月 16 日 北京正负电子对撞机对撞成功
</a>    (图) </li>                                <li><a href="/news6854/" title="1991 年
10 月 16 日 美国小镇枪杀案 22 人丧生">1991 年 10 月 16 日 美国小镇枪杀案 22 人丧生
</a>    </li>                                <li><a href="/news6855/" title="1991 年 10 月
16 日 莫扎特死因有新说">1991 年 10 月 16 日 莫扎特死因有新说
</a>    </li>                                <li><a href="/news6856/" title="1991 年 10 月
16 日 钱学森获“国家杰出贡献科学家”殊荣">1991 年 10 月 16 日 钱学森获“国家杰出贡献科学
家”殊荣</a>    (图)
</li>                                <li><a href="/news6857/" title="1994 年 10 月 16 日 德
国总理科尔四连任">1994 年 10 月 16 日 德国总理科尔四连任
</a>    </li>                                <li><a href="/news6858/" title="1994 年 10 月
16 日 第十二届广岛亚运会闭幕">1994 年 10 月 16 日 第十二届广岛亚运会闭幕
</a>    </li>                                <li><a href="/news6859/" title="1994 年 10 月
16 日 修秦陵制秦俑工匠墓葬被发现">1994 年 10 月 16 日 修秦陵制秦俑工匠墓葬被发现
</a>    </li>                                <li><a href="/news6860/" title="1995 年 10 月
16 日 美国百万黑人男子大游行">1995 年 10 月 16 日 美国百万黑人男子大游行</a>    (图)
</li>                                </ul>

```

可以看到，通过正则表达式抽取得到的 `m.group(3)` 中仍然有大量的 `html` 标签、空格、换行、无关字符等。我们要想办法把它们全部过滤掉，第 83 行代码的作用正是如此。

组装文本消息

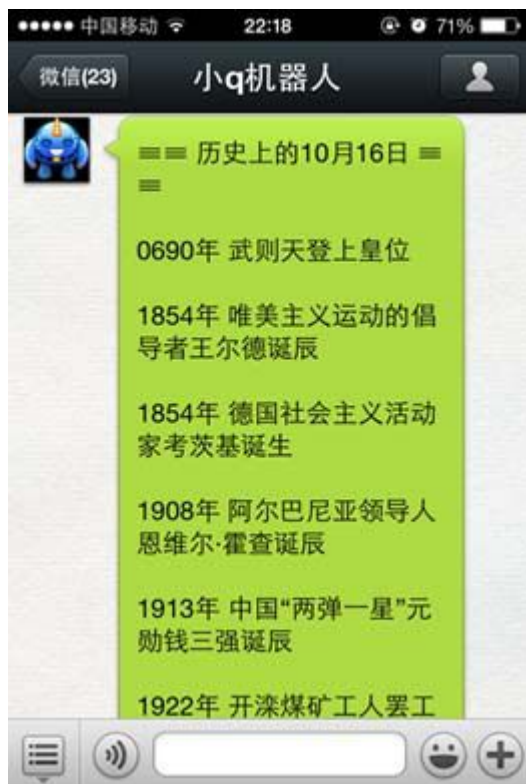
[java] view plaincopy

```

1.  // 组装文本消息（历史上的今天）
2.  TextMessage textMessage = new TextMessage();
3.  textMessage.setToUserName(fromUserName);
4.  textMessage.setFromUserName(toUserName);
5.  textMessage.setCreateTime(new Date().getTime());
6.  textMessage.setMsgType(WeixinUtil.RESP_MESSAGE_TYPE_TEXT);
7.  textMessage.setFuncFlag(0);
8.  textMessage.setContent(TodayInHistoryService.getTodayInHistoryInfo());

```

对于公众帐号的消息回复在 [本系列教程的第 5 篇](#) 已经讲的很详细了，所以在这里笔者只是简单的组装了文本消息。最后，我们来看一下在微信公众帐号上的演示效果：



说明：与其说这是一篇关于公众帐号应用开发的教程，倒不如说这是一篇关于网页数据爬取的教程。本文旨在为读者开辟思路，介绍一种数据获取方式。当然，这种做法也是有弊端的，当网页改版源代码结构发生变化时，就需要重新改写数据抽取代码。没有做不到，只有想不到！

微信公众帐号开发教程第 17 篇-应用实例之智能翻译

内容概要

本篇文章为大家演示如何在微信公众帐号上实现“智能翻译”，本例中翻译功能是通过调用“[百度翻译 API](#)”实现的。智能翻译是指用户任意输入想要翻译的内容（单词或句子），系统能自动识别用户采用的语言，并将其翻译为其他语言，目前支持的翻译方向：中->英、英->中和日->中。下面我们来看看智能翻译最终做出来的效果：



我们通过输入关键词“翻译”或者点击菜单“翻译”能够看到该功能的使用帮助，然后输入“翻译+内容”就能对内容进行翻译了。

百度翻译 API 介绍

点击查看[百度翻译 API 使用说明](#)，其实这份文档已经说的很详细了，笔者只是将我们调用该接口时最关心的内容摘取出来，主要如下：

1) 通过发送 HTTP GET 请求调用百度翻译 API。

2) 百度翻译 API 请求地址：

`http://openapi.baidu.com/public/2.0/bmt/translate`

3) 调用 API 需要传递 from、to、client_id 和 q 四个参数，描述如下：

key	value	描述
from	源语言语种: 语言代码或 auto	仅支持特定的语言组合，下面会单独进行说明
to	目标语言语种: 语言代码或 auto	仅支持特定的语言组合，下面会单独进行说明
client_id	开发者在百度连接平台上注册得到的授权 API key	请阅读 如何获取 api key
q	待翻译内容	该字段必须为 UTF-8 编码，并且以 GET 方式调用 API 时，需要进行 urlencode 编码。

在调用接口前，我们要先获取到 api key。获取方式比较简单，根据提示一步步操作就可以，笔者就不再赘述了。

- 4) 对于智能翻译, 参数 **from** 和 **to** 的传都是 **auto**。
- 4) 参数 **q** 的编码方式为 **UTF-8**, 传递之前要进行 **urlencode** 编码。
- 5) 接口返回结果示例如下:

```
{ "from": "en", "to": "zh", "trans_result": [ { "src": "today", "dst": "\u4eca\u5929" } ] }
```

返回结果里的中文是 **unicode** 编码, 需要通过 **json_decode** 进行转换, 转换后的示例如下:

```
{
  "from": "en",
  "to": "zh",
  "trans_result": [
    {
      "src": "today",
      "dst": "今天"
    },
    {
      "src": "tomorrow",
      "dst": "明天"
    }
  ]
}
```

JSON 处理工具包 Gson 介绍

Gson 是 Google 提供的用于在 Java 对象和 JSON 数据之间进行转换的 Java 类库。通过使用 Gson 类库, 我们可以将 JSON 字符串转成 Java 对象, 反之亦然。下载地址:

<https://code.google.com/p/google-gson/downloads/list>, Gson 的使用比较简单, 直接调用它的方法 **toJson()**或**fromJson()**就能完成相应的转换, 但需要注意的是: 在使用 Gson 将 json 字符串转换成 Java 对象之前, 需要先创建好与目标 Java 对象。读者可以在维基百科上学习它的使用示例 <http://zh.wikipedia.org/wiki/Gson>。

代码实现

1) 创建与百度翻译 API 返回的 JSON 相对应的 Java 类

[java] view plaincopy

```
1. import java.util.List;
2.
3. /**
4.  * 调用百度翻译 api 查询结果
5.  *
```

```

6.  * @author liufeng
7.  * @date 2013-10-21
8.  */
9.  public class TranslateResult {
10.     // 实际采用的源语言
11.     private String from;
12.     // 实际采用的目标语言
13.     private String to;
14.     // 结果体
15.     private List<ResultPair> trans_result;
16.
17.     public String getFrom() {
18.         return from;
19.     }
20.
21.     public void setFrom(String from) {
22.         this.from = from;
23.     }
24.
25.     public String getTo() {
26.         return to;
27.     }
28.
29.     public void setTo(String to) {
30.         this.to = to;
31.     }
32.
33.     public List<ResultPair> getTrans_result() {
34.         return trans_result;
35.     }
36.
37.     public void setTrans_result(List<ResultPair> trans_result) {
38.         this.trans_result = trans_result;
39.     }
40. }

```

注意：这里的类名可以任意取，但是成员变量的名字应于翻译 API 返回的 JSON 字符串中的属性名保持一致，否则将 JSON 转换成 TranslateResult 对象时会报错。

TranslateResult 类中的 trans_result 属性是一个 ResultPair 集合，该类的代码如下：

[java] view plaincopy

```

1.  /**
2.   * 结果对
3.   *

```

```

4.  * @author liufeng
5.  * @date 2013-10-21
6.  */
7.  public class ResultPair {
8.      // 原文
9.      private String src;
10.     // 译文
11.     private String dst;
12.
13.     public String getSrc() {
14.         return src;
15.     }
16.
17.     public void setSrc(String src) {
18.         this.src = src;
19.     }
20.
21.     public String getDst() {
22.         return dst;
23.     }
24.
25.     public void setDst(String dst) {
26.         this.dst = dst;
27.     }
28. }

```

说明：这两个类的封装是 Gson 类库所要求的，如果读者不是用 Gson 解析 json 字符串，而是用 JSON-lib，就没有必要封装这两个类。

2) 接口调用

[java] view plaincopy

```

1.  import java.io.BufferedReader;
2.  import java.io.InputStream;
3.  import java.io.InputStreamReader;
4.  import java.io.UnsupportedEncodingException;
5.  import java.net.HttpURLConnection;
6.  import java.net.URL;
7.  import com.google.gson.Gson;
8.
9.  /**
10.   *
11.   * @author liufeng
12.   * @date 2013-10-21

```

```
13. */
14. public class BaiduTranslateService {
15.     /**
16.      * 发起 http 请求获取返回结果
17.      *
18.      * @param requestUrl 请求地址
19.      * @return
20.      */
21.     public static String httpRequest(String requestUrl) {
22.         StringBuffer buffer = new StringBuffer();
23.         try {
24.             URL url = new URL(requestUrl);
25.             HttpURLConnection httpUrlConn = (HttpURLConnection) url.openConnection();
26.
27.             httpUrlConn.setDoOutput(false);
28.             httpUrlConn.setDoInput(true);
29.             httpUrlConn.setUseCaches(false);
30.
31.             httpUrlConn.setRequestMethod("GET");
32.             httpUrlConn.connect();
33.
34.             // 将返回的输入流转换成字符串
35.             InputStream inputStream = httpUrlConn.getInputStream();
36.             InputStreamReader inputStreamReader = new InputStreamReader(inputStream, "utf-8");
37.             BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
38.
39.             String str = null;
40.             while ((str = bufferedReader.readLine()) != null) {
41.                 buffer.append(str);
42.             }
43.             bufferedReader.close();
44.             inputStreamReader.close();
45.             // 释放资源
46.             inputStream.close();
47.             inputStream = null;
48.             httpUrlConn.disconnect();
49.
50.         } catch (Exception e) {
51.         }
52.         return buffer.toString();
53.     }
```

```
54.
55.     /**
56.      * utf 编码
57.      *
58.      * @param source
59.      * @return
60.      */
61.     public static String urlEncodeUTF8(String source) {
62.         String result = source;
63.         try {
64.             result = java.net.URLEncoder.encode(source, "utf-8");
65.         } catch (UnsupportedEncodingException e) {
66.             e.printStackTrace();
67.         }
68.         return result;
69.     }
70.
71.     /**
72.      * 翻译（中->英 英->中 日->中）
73.      *
74.      * @param source
75.      * @return
76.      */
77.     public static String translate(String source) {
78.         String dst = null;
79.
80.         // 组装查询地址
81.         String requestUrl = "http://openapi.baidu.com/public/2.0/bmt/translate?client_id=AAAAAAAAAAAAAAAAAAAA&q={keyWord}&from=auto&to=auto";
82.         // 对参数 q 的值进行 urlEncode utf-8 编码
83.         requestUrl = requestUrl.replace("{keyWord}", urlEncodeUTF8(source));
84.
85.         // 查询并解析结果
86.         try {
87.             // 查询并获取返回结果
88.             String json = httpRequest(requestUrl);
89.             // 通过 Gson 工具将 json 转换成 TranslateResult 对象
90.             TranslateResult translateResult = new Gson().fromJson(json, TranslateResult.class);
91.             // 取出 translateResult 中的译文
92.             dst = translateResult.getTrans_result().get(0).getDst();
93.         } catch (Exception e) {
94.             e.printStackTrace();
```

```

95.         }
96.
97.         if (null == dst)
98.             dst = "翻译系统异常，请稍候尝试! ";
99.         return dst;
100.    }
101.
102.    public static void main(String[] args) {
103.        // 翻译结果: The network really powerful
104.        System.out.println(translate("网络真强大"));
105.    }
106. }

```

代码解读：

- 1) 第 21-53 行封装了一个 http 请求方法 `httpRequest()`，相信读过之前教程的读者已经很熟悉了。
- 2) 第 61-69 行封装了一个 `urlEncodeUTF8()` 方法，用于对 url 中的参数进行 UTF-8 编码。
- 3) 第 81 行代码中的 `client_id` 需要替换成自己申请的 api key。
- 4) 第 83 行代码是对 url 中的中文进行编码。以后凡是遇到通过 url 传递中文参数的情况，一定要显示地对中文进行编码，否则很可能出现程序在本机能正常运行，但部署到服务器上却有问题，因为本机与服务器的默认编码方式可能不一样。
- 5) 第 88 行代码就是调用百度翻译 API。
- 6) 第 90 行代码是使用 Gson 工具将 json 字符串转换成 `TranslateResult` 对象，是不是发现 Gson 的使用真的很简单？另外，前面提到过调用百度翻译 API 返回的 json 里如果有中文是用 unicode 表示的，形如“`\u4eca\u5929`”，那为什么这里没有做任何处理？因为 Gson 的内部实现已经帮我们搞定了。

公众账号后台调用

在公众账号后台，需要对接收到的文本消息进行判断，如果是“翻译”两个字开头的，就认为是在使用智能翻译功能，然后将“翻译”两个字之后的内容作为翻译对象，调用 API 进行翻译；如果输入的只有“翻译”两个字，就提示智能翻译功能的使用指南。关键代码如下：

[java] view plaincopy

```

1.  // 文本消息
2.  if (WeixinUtil.REQ_MESSAGE_TYPE_TEXT.equals(msgType)) {
3.      String content = requestMap.get("Content").trim();
4.      if (content.startsWith("翻译")) {
5.          String keyWord = content.replaceAll("^翻译", "").trim();
6.          if ("".equals(keyWord)) {
7.              textMessage.setContent(getTranslateUsage());
8.          } else {

```

```

9.         textMessage.setContent(BaiduTranslateService.translate(keyWord));
10.     }
11.     out.print(WeixinUtil.textMessageToXml(textMessage));
12. }
13. }

```

第 7 行 `getTranslateUsage()` 方法得到的就是智能翻译功能的使用指南，代码如下：

[java] view plaincopy

```

1.  /**
2.   * Q 译通使用指南
3.   *
4.   * @return
5.   */
6.  public static String getTranslateUsage() {
7.      StringBuffer buffer = new StringBuffer();
8.      buffer.append(XiaoqUtil.emoji(0xe148)).append("Q 译通使用指南")
9.      buffer.append("Q 译通为用户提供专业的多语言翻译服务，目前支持以下翻译方向：")
10.     buffer.append("    中 -> 英").append("\n");
11.     buffer.append("    英 -> 中").append("\n");
12.     buffer.append("    日 -> 中").append("\n\n");
13.     buffer.append("使用示例: ").append("\n");
14.     buffer.append("    翻译我是中国人").append("\n");
15.     buffer.append("    翻译 dream").append("\n");
16.     buffer.append("    翻译さようなら").append("\n\n");
17.     buffer.append("回复“?”显示主菜单");
18.     return buffer.toString();
19. }

```

说明：希望通过本例的学习，除了掌握百度翻译 API 的调用之外，读者还能够掌握 json 字符串的解析方法，这样就能够自己学会调用更多互联网上开放的接口。

微信公众帐号开发教程第 18 篇-应用实例之音乐搜索

引言及内容概要

微信公众平台支持向用户回复音乐消息，用户收到音乐消息后，点击即可播放音乐。通过音乐消息，公众账号可以实现音乐搜索（歌曲点播）功能，即用户输入想听的音乐名称，公众

账号返回对应的音乐（歌曲）。读者可以关注 xiaoqrobot 体验该功能，操作指南及使用如下所示。



考虑到歌曲名称有重复的情况，用户还可以同时指定歌曲名称、演唱者搜索歌曲。下面就为读者详细介绍歌曲点播功能的实现过程。

音乐消息说明

在[微信公众平台开发者文档](#)中提到，向用户回复音乐消息需要构造如下格式的 XML 数据。

```
[html] view plaincopy
1.  <xml>
2.    <ToUserName><![CDATA[toUser]]></ToUserName>
3.    <FromUserName><![CDATA[fromUser]]></FromUserName>
4.    <CreateTime>12345678</CreateTime>
5.    <MsgType><![CDATA[music]]></MsgType>
6.    <Music>
7.      <Title><![CDATA[TITLE]]></Title>
8.      <Description><![CDATA[DESCRIPTION]]></Description>
9.      <MusicUrl><![CDATA[MUSIC_Url]]></MusicUrl>
10.     <HQMusicUrl><![CDATA[HQ_MUSIC_Url]]></HQMusicUrl>
11.     <ThumbMediaId><![CDATA[media_id]]></ThumbMediaId>
12.   </Music>
13. </xml>
```

上面 XML 中，需要注意的是<Music>节点中的参数，说明如下：

- 1) 参数 Title: 标题，本例中可以设置为歌曲名称；
- 2) 参数 Description: 描述，本例中可以设置为歌曲的演唱者；
- 3) 参数 MusicUrl: 普通品质的音乐链接；
- 4) 参数 HQMusicUrl: 高品质的音乐链接，在 WIFI 环境下会优先使用该链接播放音乐；
- 5) 参数 ThumbMediaId: 缩略图的媒体 ID，通过上传多媒体文件获得；它指向的是一张图片，最终会作为音乐消息左侧绿色方形区域的背景图片。

上述 5 个参数中，最为重要的是 MusicUrl 和 HQMusicUrl，这也是本文的重点，如何根据歌曲名称获得歌曲的链接。如果读者只能得到歌曲的一个链接，可以将 MusicUrl 和 HQMusicUrl 设置成一样的。至于 ThumbMediaId 参数，必须是通过微信认证的服务号才能得到，普通的服务号与订阅号可以忽略该参数，也就是说，在回复给微信服务器的 XML 中可以不包含 ThumbMediaId 参数。

百度音乐搜索 API 介绍

上面提到，给用户回复音乐消息最关键在于如何根据歌曲名称获得歌曲的链接，我们必须找一个现成的音乐搜索 API，除非读者自己有音乐服务器，或者只向用户回复固定的几首音乐。百度有一个非公开的音乐搜索 API，之所以说非公开，是因为笔者没有在百度官网的任何地方看到有关该 API 的介绍，但这并不影响读者对本例的学习，我们仍然可以调用它。百度音乐搜索 API 的请求地址如下：

[html] view plaincopy

1. `http://box.zhangmen.baidu.com/x?op=12&count=1&title=TITLE`
`AUTHOR`
2. `$$`

`http://box.zhangmen.baidu.com` 为百度音乐盒的首页地址，上面的链接中不用管参数 `op` 和 `count`，重点关注 `TITLE` 和 `AUTHOR`，`TITLE` 表示歌曲名称，`AUTHOR` 表示演唱者，`AUTHOR` 可以为空，参数 `TITLE` 和 `AUTHOR` 需要进行 URL 编码（UTF-8 或 GB2312 均可）。例如，要搜索歌曲零点乐队的“相信自己”，可以像下面这样：

[html] view plaincopy

1. // GB2312 编码的音乐搜索链接
2. `http://box.zhangmen.baidu.com/x?op=12&count=1&title=%CF%E0%D0%C5%D7%D4%BC%BA`
`$$`
3. // UTF-8 编码的音乐搜索链接
4. `http://box.zhangmen.baidu.com/x?op=12&count=1&title=%E7%9B%B8%E4%BF%A1%E8%87`
`%AA%E5%B7%B1$$`

通过浏览器访问上面的地址，返回的是如下格式的 XML 数据：

[html] view plaincopy

```
1.  <result>
2.    <count>1</count>
3.    <url>
4.      <encode>
5.        <![CDATA[http://zhangmenshiting.baidu.com/data2/music/44277542/Z
WZla2xra2pfn6NndK6ap5WXcJVob5puZ2trbWprmnBjZ2xolpeZa2drZmWZmZmd12hjZWhvnWlpYm
RtZmltcGplZFqin5t1YWBobW5qcGxia2NmZ2twbzE$]]>
6.      </encode>
7.      <decode>
8.        <![CDATA[44277542.mp3?xcod=a39c6698955c82594aab36931dcbe60139f
180191368931&mid=0.59949419022597]]>
9.      </decode>
10.     <type>8</type>
11.     <lrcid>64644</lrcid>
12.     <flag>1</flag>
13.   </url>
14.   <durl>
15.     <encode>
16.       <![CDATA[http://zhangmenshiting2.baidu.com/data2/music/44277530/
ZWZla2xramhfn6NndK6ap5WXcJVob5puZ2trbWprmnBjZ2xolpeZa2drZmWZmZmd12hjaGhvnZ5q1
GRpbpedamJla1qin5t1YWBobW5qcGxia2NmZ2twbzE$]]>
17.     </encode>
18.     <decode>
19.       <![CDATA[44277530.mp3?xcod=a39c6698955c82594aab36931dcbe60439f
f9b159af2138&mid=0.59949419022597]]>
20.     </decode>
21.     <type>8</type>
22.     <lrcid>64644</lrcid>
23.     <flag>1</flag>
24.   </durl>
25.   <p2p>
26.     <hash>022bc0fbf66cd19bea96db49634419dc2600393f</hash>
27.     <url>
28.       <![CDATA[ ]>
29.     </url>
30.     <type>mp3</type>
31.     <size>5236902</size>
32.     <bitrate>192</bitrate>
33.   </p2p>
34. </result>
```

返回结果中的主要参数说明如下：

1) <count> 表示搜索到的音乐数；

2) <url>中包含了普通品质的音乐链接，<durl>中包含了高品质音乐的链接；

3) <encode>中包含了加密后的音乐链接，实际上只是对音乐名称进行了加密，<decode>中包含了解密后的音乐名称。因此，要获取音乐的链接就需要重点分析<encode>和<decode>中的内容，下面会专门为读者进行介绍。

4) <type>表示音乐文件的类型，如 rm、wma、mp3 等；

5) <lrcid>是歌词的 ID，<url>中的歌词 ID 为 64644，那么如何得到歌词呢？本例并不关心歌词，只是附带提一下。歌词的地址如下：

[html] view plaincopy

1. <http://box.zhangmen.baidu.com/bdlrc/646/64644.lrc>

其中，<http://box.zhangmen.baidu.com/bdlrc/>是固定值；646 为歌词所在目录名，计算方法为歌词 ID（64644）除以 100，取整数部分；64644.lrc 是歌词文件名。

下面来看如何从<encode>和<decode>中得到音乐链接。为了便于说明，笔者将上面搜索结果中的<url>和<durl>部分抽取出来，并进行了标注，如下图所示。

```
▼<url>
  ▼<encode>
    ▼<![CDATA[
      http://zhangmenshiting.baidu.com/data2/music/44277542/ZWZ1a2xra2pfn6NndK6ap5WXcJVob5puZ2t
    ]]>
  </encode>
  ▼<decode>
    ▼<![CDATA[
      44277542.mp3?xcode=a39c6698955c82594aab36931dcbef60139f180191368931&mid=0.59949419022597
    ]]>
  </decode>
  <type>8</type>
  <lrcid>64644</lrcid>
  <flag>1</flag>
</url>
▼<durl>
  ▼<encode>
    ▼<![CDATA[
      http://zhangmenshiting2.baidu.com/data2/music/44277530/ZWZ1a2xramhfn6NndK6ap5WXcJVob5puZ2t
    ]]>
  </encode>
  ▼<decode>
    ▼<![CDATA[
      44277530.mp3?xcode=a39c6698955c82594aab36931dcbef60439ff9b159af2138&mid=0.59949419022597
    ]]>
  </decode>
  <type>8</type>
  <lrcid>64644</lrcid>
  <flag>1</flag>
</durl>
```

上图中，1 和 2 拼接起来是普通品质音乐的链接，3 和 4 拼接起来是高品质音乐的链接。也就是说，普通品质和高品质音乐链接如下：

[html] view plaincopy

1. // 普通品质音乐链接
2. <http://zhangmenshiting.baidu.com/data2/music/44277542/44277542.mp3?xcodes=a39c6698955c82594aab36931dcbef60139f180191368931>
3. // 高品质音乐链接
4. <http://zhangmenshiting2.baidu.com/data2/music/44277530/44277530.mp3?xcodes=a39c6698955c82594aab36931dcbef60439ff9b159af2138>

参数 `xcodes` 可以理解为随机验证码，每次搜索得到的值都不一样，如果不带该参数会报未授权异常“401 Authorization Required”。需要注意的是，`xcodes` 是有时间限制的，超过限制再访问链接会报异常：{"Error":{"code":"2","Message":"object not exists","LogId":"3456414897"}}。在 `xcodes` 有效的前提下，通过浏览器访问上面的音乐链接，会提示下载音乐。

编程调用百度音乐搜索 API

知道如何从 API 返回结果中得到音乐链接后，就可以编写程序来实现了。笔者将发送 HTTP 请求、URL 编码、解析 XML 等操作全部封装在 `BaiduMusicService` 类中，该类的代码如下：

[java] view plaincopy

```
1. import java.io.InputStream;
2. import java.io.UnsupportedEncodingException;
3. import java.net.HttpURLConnection;
4. import java.net.URL;
5. import java.util.List;
6.
7. import org.dom4j.Document;
8. import org.dom4j.Element;
9. import org.dom4j.io.SAXReader;
10.
11. import org.liufeng.course.message.resp.Music;
12.
13. /**
14.  * 百度音乐搜索 API 操作类
15.  *
16.  * @author liufeng
17.  * @date 2013-12-09
18.  */
19. public class BaiduMusicService {
20.     /**
21.      * 根据名称和作者搜索音乐
22.      *
23.      * @param musicTitle 音乐名称
24.      * @param musicAuthor 音乐作者
25.      * @return Music
26.      */
```

```

27.     public static Music searchMusic(String musicTitle, String musicAuthor) {
28.         // 百度音乐搜索地址
29.         String requestUrl = "http://box.zhangmen.baidu.com/x?op=12&count=1&title={TITLE}
                                     AUTHOR
30. $$";
31.         // 对音乐名称、作者进 URL 编码
32.         requestUrl = requestUrl.replace("{TITLE}", urlEncodeUTF8(musicTitle))
;
33.         requestUrl = requestUrl.replace("{AUTHOR}", urlEncodeUTF8(musicAuthor));
34.         // 处理名称、作者中间的空格
35.         requestUrl = requestUrl.replaceAll("\\\\+", "%20");
36.
37.         // 查询并获取返回结果
38.         InputStream inputStream = httpRequest(requestUrl);
39.         // 从返回结果中解析出 Music
40.         Music music = parseMusic(inputStream);
41.
42.         // 如果 music 不为 null, 设置标题和描述
43.         if (null != music) {
44.             music.setTitle(musicTitle);
45.             // 如果作者不为"", 将描述设置为作者
46.             if (!"".equals(musicAuthor))
47.                 music.setDescription(musicAuthor);
48.             else
49.                 music.setDescription("来自百度音乐");
50.         }
51.         return music;
52.     }
53.
54.     /**
55.      * UTF-8 编码
56.      *
57.      * @param source
58.      * @return
59.      */
60.     private static String urlEncodeUTF8(String source) {
61.         String result = source;
62.         try {
63.             result = java.net.URLEncoder.encode(source, "UTF-8");
64.         } catch (UnsupportedEncodingException e) {

```

```
65.         e.printStackTrace();
66.     }
67.     return result;
68. }
69.
70. /**
71.  * 发送 http 请求取得返回的输入流
72.  *
73.  * @param requestUrl 请求地址
74.  * @return InputStream
75.  */
76. private static InputStream httpRequest(String requestUrl) {
77.     InputStream inputStream = null;
78.     try {
79.         URL url = new URL(requestUrl);
80.         HttpURLConnection httpUrlConn = (HttpURLConnection) url.openConnection();
81.         httpUrlConn.setDoInput(true);
82.         httpUrlConn.setRequestMethod("GET");
83.         httpUrlConn.connect();
84.         // 获得返回的输入流
85.         inputStream = httpUrlConn.getInputStream();
86.     } catch (Exception e) {
87.         e.printStackTrace();
88.     }
89.     return inputStream;
90. }
91.
92. /**
93.  * 解析音乐参数
94.  *
95.  * @param inputStream 百度音乐搜索 API 返回的输入流
96.  * @return Music
97.  */
98. @SuppressWarnings("unchecked")
99. private static Music parseMusic(InputStream inputStream) {
100.     Music music = null;
101.     try {
102.         // 使用 dom4j 解析 xml 字符串
103.         SAXReader reader = new SAXReader();
104.         Document document = reader.read(inputStream);
105.         // 得到 xml 根元素
106.         Element root = document.getRootElement();
107.         // count 表示搜索到的歌曲数
```

```
108.         String count = root.element("count").getText();
109.         // 当搜索到的歌曲数大于 0 时
110.         if (!"0".equals(count)) {
111.             // 普通品质
112.             List<Element> urlList = root.elements("url");
113.             // 高品质
114.             List<Element> durlList = root.elements("durl");
115.
116.             // 普通品质的 encode、decode
117.             String urlEncode = urlList.get(0).element("encode").getText
                ();
118.             String urlDecode = urlList.get(0).element("decode").getText
                ();
119.             // 普通品质音乐的 URL
120.             String url = urlEncode.substring(0, urlEncode.lastIndexOf("/")
                + 1) + urlDecode;
121.             if (-1 != urlDecode.lastIndexOf("&"))
122.                 url = urlEncode.substring(0, urlEncode.lastIndexOf("/")
                + 1) + urlDecode.substring(0, urlDecode.lastIndexOf("&"));
123.
124.             // 默认情况下, 高音质音乐的 URL 等于 普通品质音乐的 URL
125.             String durl = url;
126.
127.             // 判断高品质节点是否存在
128.             Element durlElement = durlList.get(0).element("encode");
129.             if (null != durlElement) {
130.                 // 高品质的 encode、decode
131.                 String durlEncode = durlList.get(0).element("encode").g
                    etText();
132.                 String durlDecode = durlList.get(0).element("decode").g
                    etText();
133.                 // 高品质音乐的 URL
134.                 durl = durlEncode.substring(0, durlEncode.lastIndexOf("/")
                    + 1) + durlDecode;
135.                 if (-1 != durlDecode.lastIndexOf("&"))
136.                     durl = durlEncode.substring(0, durlEncode.lastIndex
                        Of("/") + 1) + durlDecode.substring(0, durlDecode.lastIndex
                            Of("&"));
137.             }
138.             music = new Music();
139.             // 设置普通品质音乐链接
140.             music.setMusicUrl(url);
141.             // 设置高品质音乐链接
142.             music.setHQMusicUrl(durl);
143.         }
```



```

144.         } catch (Exception e) {
145.             e.printStackTrace();
146.         }
147.         return music;
148.     }
149.
150.     // 测试方法
151.     public static void main(String[] args) {
152.         Music music = searchMusic("相信自己", "零点乐队");
153.         System.out.println("音乐名称: " + music.getTitle());
154.         System.out.println("音乐描述: " + music.getDescription());
155.         System.out.println("普通品质链接: " + music.getMusicUrl());
156.         System.out.println("高品质链接: " + music.getHQMusicUrl());
157.     }
158. }

```

下面对代码进行简单的说明：

- 1) 代码中的 **Music** 类是对音乐消息的封装（不包括 **ThumbMediaId** 参数），读者可以在[本系列教程的第 4 篇](#)中找到该类的定义；
- 2) 运行上述代码需要引入 **dom4j** 的 JAR 包，笔者使用的是 **dom4j-1.6.1.jar**；
- 3) **searchMusic()**方法是提供给外部调用的，在 **CoreService** 类中会调用该方法获得音乐消息需要的 **Music** 相关的 4 个参数（**Title**、**Description**、**MusicUrl** 和 **HQMusicUrl**）；
- 4) **parseMusic()**方法用于解析 XML，读者可以结合代码中的注释和之前对 XML 的分析进行理解，这里就不再赘述了。
- 5) 116 行、127 行中的 **get(0)**表示返回多条音乐结果时默认取第一条。

公众账号后台的实现

在公众账号后台的 **CoreService** 类中，需要对用户发送的文本消息进行判断，如果是“歌曲”两个字开头，就认为用户是在使用“歌曲点播”功能，此时需要对“歌曲”两个字之后的内容进行判断，如果包含“@”符号，就表示需要按演唱者搜索，否则不指定演唱者。**CoreService** 类的完整代码如下：

[java] view plaincopy

```

1. package org.liufeng.course.service;
2.
3. import java.util.Date;
4. import java.util.Map;
5. import javax.servlet.http.HttpServletRequest;
6. import org.liufeng.course.message.resp.Music;
7. import org.liufeng.course.message.resp.MusicMessage;
8. import org.liufeng.course.message.resp.TextMessage;

```

```
9. import org.liufeng.course.util.MessageUtil;
10.
11. /**
12.  * 核心服务类
13.  *
14.  * @author liufeng
15.  * @date 2013-12-10
16.  */
17. public class CoreService {
18.     /**
19.      * 处理微信发来的请求
20.      *
21.      * @param request
22.      * @return
23.      */
24.     public static String processRequest(HttpServletRequest request) {
25.         // 返回给微信服务器的 xml 消息
26.         String respXml = null;
27.         // 文本消息内容
28.         String respContent = null;
29.         try {
30.             // xml 请求解析
31.             Map<String, String> requestMap = MessageUtil.parseXml(request);
32.
33.             // 发送方帐号 (open_id)
34.             String fromUserName = requestMap.get("FromUserName");
35.             // 公众帐号
36.             String toUserName = requestMap.get("ToUserName");
37.             // 消息类型
38.             String msgType = requestMap.get("MsgType");
39.
40.             // 回复文本消息
41.             TextMessage textMessage = new TextMessage();
42.             textMessage.setToUserName(fromUserName);
43.             textMessage.setFromUserName(toUserName);
44.             textMessage.setCreateTime(new Date().getTime());
45.             textMessage.setMsgType(MessageUtil.RESP_MESSAGE_TYPE_TEXT);
46.
47.             // 文本消息
48.             if (MessageUtil.REQ_MESSAGE_TYPE_TEXT.equals(msgType)) {
49.                 // 文本消息内容
50.                 String content = requestMap.get("Content").trim();
51.                 // 如果以“歌曲”2 个字开头
52.                 if (content.startsWith("歌曲")) {
```

```

52.                // 将歌曲 2 个字及歌曲后面的+、空格、-等特殊符号去掉
53.                String keyWord = content.replaceAll("^歌曲
    [\\+ ~!@#%^- _=]?", "");
54.                // 如果歌曲名称为空
55.                if ("".equals(keyWord)) {
56.                    respContent = getUsage();
57.                } else {
58.                    String[] kwArr = keyWord.split("@");
59.                    // 歌曲名称
60.                    String musicTitle = kwArr[0];
61.                    // 演唱者默认为空
62.                    String musicAuthor = "";
63.                    if (2 == kwArr.length)
64.                        musicAuthor = kwArr[1];
65.
66.                    // 搜索音乐
67.                    Music music = BaiduMusicService.searchMusic(musicTit
    le, musicAuthor);
68.                    // 未搜索到音乐
69.                    if (null == music) {
70.                        respContent = "对不起，没有找到你想听的歌曲
    <" + musicTitle + ">。";
71.                    } else {
72.                        // 音乐消息
73.                        MusicMessage musicMessage = new MusicMessage();
74.
75.                        musicMessage.setToUserName(fromUserName);
76.                        musicMessage.setFromUserName(toUserName);
77.                        musicMessage.setCreateTime(new Date().getTime());
78.                        musicMessage.setMsgType(MessageUtil.RESP_MESSAGE
    _TYPE_MUSIC);
79.                        musicMessage.setMusic(music);
80.                        respXml = MessageUtil.musicMessageToXml(musicMes
    sage);
81.                    }
82.                }
83.            }
84.            // 未搜索到音乐时返回使用指南
85.            if (null == respXml) {
86.                if (null == respContent)
87.                    respContent = getUsage();
88.                textMessage.setContent(respContent);

```

```

89.             respXml = MessageUtil.textMessageToXml(textMessage);
90.         }
91.     } catch (Exception e) {
92.         e.printStackTrace();
93.     }
94.     return respXml;
95. }
96.
97. /**
98.  * 歌曲点播使用指南
99.  *
100.  * @return
101.  */
102. public static String getUsage() {
103.     StringBuffer buffer = new StringBuffer();
104.     buffer.append("歌曲点播操作指南").append("\n\n");
105.     buffer.append("回复: 歌曲+歌名").append("\n");
106.     buffer.append("例如: 歌曲存在").append("\n");
107.     buffer.append("或者: 歌曲存在@汪峰").append("\n\n");
108.     buffer.append("回复“?”显示主菜单");
109.     return buffer.toString();
110. }
111.}

```

上述代码的逻辑比较简单，用户发送“歌曲+名称”或者“歌曲+名称@演唱者”就能搜索歌曲，搜索不到时会提示用户，如果发送其他内容回复歌曲点播功能的用法。

微信公众平台开发教程第 19 篇-应用实例之人脸检测

在笔者的公众账号小 q 机器人（微信号：xiaoqrobot）中有一个非常好玩的功能“人脸检测”，它能够检测出用户发送的图片中有多少张人脸，并且还能分析出每张脸所对应的人种、性别和年龄。几乎每天都有一些用户在使用“人脸检测”，该功能的趣味性和娱乐性在于能够让用户知道自己的长相与真实年龄是否相符，是否男（女）性化🤔。本文将为读者介绍人脸检测应用的完整实现过程。

人脸检测属于人脸识别的范畴，它是一个复杂的具有挑战性的模式匹配问题，国内外许多组织、科研机构都在专门研究该问题。国内的 Face++ 团队专注于研发人脸检测、识别、分析和重建技术，并且向广大开发者开放了人脸识别 API，本文介绍的“人脸检测”应用正是基于 Face++ API 进行开发。

Face++简介

Face++是北京旷视科技有限公司旗下的人脸识别云服务平台，Face++平台通过提供云端 API、离线 SDK、以及面向用户的自主研发产品等形式，将人脸识别技术广泛应用到互联网及移动应用场景中。Face++为广大开发者提供了简单易用的 API，开发者可以轻松搭建属于自己的云端身份认证、用户兴趣挖掘、移动体感交互、社交娱乐分享等多种类型的应用。

Face++提供的技术服务包括人脸检测、人脸分析和人脸识别，主要说明如下：

1) 人脸检测：可以从图片中快速、准确的定位面部的关键区域位置，包括眉毛、眼睛、鼻子、嘴巴等。

2)人脸分析:可以从图片或实时视频流中分析出人脸的性别(准确度达 96%)、年龄、种族等多种属性。

3) 人脸识别：可以快速判定两张照片是否为同一个人，或者快速判定视频中的人像是否为某一位特定的人。

Face++的中文网址为 <http://cn.faceplusplus.com/>，要使用 Face++ API，需要注册成为 Face++ 开发者，也就是要注册一个 Face++ 账号。注册完成后，先创建应用，创建应用时需要填写“应用名称”、“应用描述”、“API 服务器”、“应用类型”和“应用平台”，读者可以根据实际情况填写。应用创建完成后，可以看到应用的详细信息，如下图所示。

应用名称	小q机器人
API KEY	af035040403000000000000000000000
API SECRET	0p0d0w0000000000000000000000000000 重置API_SECRET ?
API URL	apicn.faceplusplus.com ?
创建时间	2013-04-16 23:51:06
应用描述	一款集天气、游戏、聊天等众多生活娱乐功能于一体的应用
应用状态	上线版
API服务器	阿里云(中国)
应用类型	娱乐
应用平台	其他
离线SDK	IOS 离线检测器下载 ANDROID 离线检测器下载

上图中，最重要的是 API KEY 和 API SECRET，在调用 Face++提供的 API 时，需要传入这两个参数。

人脸检测 API 介绍

在 Face++ 网站的“API 文档”中，能够看到 Face++ 提供的所有 API，我们要使用的人脸检测接口是 **detect** 分类下的“/detection/detect”，它能够检测出给定图片(Image)中的所有人脸(Face)的位置和相应的面部属性，目前面部属性包括性别(gender)、年龄(age)、种族(race)、微笑程度(smiling)、眼镜(glass)和姿势(pose)。

读者可以在 <http://cn.faceplusplus.com/uc/doc/home?id=69> 中了解到人脸检测接口的详细信息，该接口的请求地址如下：

[html] view plaincopy

1. http://apicn.faceplusplus.com/v2/detection/detect?url=URL&api_secret=API_SECRET&api_key=API_KEY

调用上述接口，必须要传入参数 **api_key**、**api_secret** 和待检测的图片。其中，待检测的图片可以是 **URL**，也可以是 **POST** 方式提交的二进制数据。在微信公众平台后台，接收用户发送的图片，得到的是图片的访问路径（PicUrl），因此，在本例中，直接使用待检测图片的 **URL** 是最方便的。调用人脸检测接口返回的是 **JSON** 格式数据如下：

[html] view plaincopy

```
1. {
2.   "face": [
3.     {
4.       "attribute": {
5.         "age": {
6.           "range": 5,
7.           "value": 23
8.         },
9.         "gender": {
10.          "confidence": 99.9999,
11.          "value": "Female"
12.        },
13.        "glass": {
14.          "confidence": 99.945,
15.          "value": "None"
16.        },
17.        "pose": {
18.          "pitch_angle": {
19.            "value": 17
20.          },
21.          "roll_angle": {
22.            "value": 0.735735
23.          },
24.          "yaw_angle": {
```

```
25.             "value": -2
26.         }
27.     },
28.     "race": {
29.         "confidence": 99.6121,
30.         "value": "Asian"
31.     },
32.     "smiling": {
33.         "value": 4.86501
34.     }
35. },
36. "face_id": "17233b4b1b51ac91e391e5afe130eb78",
37. "position": {
38.     "center": {
39.         "x": 49.4,
40.         "y": 37.6
41.     },
42.     "eye_left": {
43.         "x": 43.3692,
44.         "y": 30.8192
45.     },
46.     "eye_right": {
47.         "x": 56.5606,
48.         "y": 30.9886
49.     },
50.     "height": 26.8,
51.     "mouth_left": {
52.         "x": 46.1326,
53.         "y": 44.9468
54.     },
55.     "mouth_right": {
56.         "x": 54.2592,
57.         "y": 44.6282
58.     },
59.     "nose": {
60.         "x": 49.9404,
61.         "y": 38.8484
62.     },
63.     "width": 26.8
64. },
65. "tag": ""
66. }
67. ],
68. "img_height": 500,
```

```
69.     "img_id": "22fd9efc64c87e00224c33dd8718eec7",
70.     "img_width": 500,
71.     "session_id": "38047ad0f0b34c7e8c6efb6ba39ed355",
72.     "url": "http://cn.faceplusplus.com/wp-content/themes/faceplusplus.zh/assets/img/demo/1.jpg?v=4"
73. }
```

这里只对本文将要实现的“人脸检测”功能中主要用到的参数进行说明，参数说明如下：

- 1) **face** 是一个数组，当一张图片中包含多张人脸时，所有识别出的人脸信息都在 **face** 数组中。
- 2) **age** 中的 **value** 表示估计年龄，**range** 表示误差范围。例如，上述结果中 **value=23**，**range=5**，表示人的真实年龄在 18 岁至 28 岁左右。
- 3) **gender** 中的 **value** 表示性别，男性为 **Male**，女性为 **Female**；**gender** 中的 **confidence** 表示检测结果的可信度。
- 4) **race** 中的 **value** 表示人种，黄色人种为 **Asian**，白色人种为 **White**，黑色人种为 **Black**；**race** 中的 **confidence** 表示检测结果的可信度。
- 5) **center** 表示人脸框中心点坐标，可以将 **x** 用于计算人脸的左右顺序，即 **x** 坐标的值越小，人脸的位置越靠近图片的左侧。

人脸检测 API 的使用方法

为了方便开发者调用人脸识别 API，Face++ 团队提供了基于 Objective-C、Java（Android）、Matlab、Ruby、C# 等多种语言的 **开发工具包**，读者可以在 Face++ 网站的“工具下载”版块下载相关的 SDK。在本例中，笔者并不打算使用官方提供的 SDK 进行开发，主要原因如下：1) 人脸检测 API 的调用比较简单，自己写代码实现也并不复杂；2) 如果使用 SDK 进行开发，笔者还要花费大量篇幅介绍 SDK 的使用，这些并不是本文的重点；3) 自己写代码实现比较灵活。当图片中有多张人脸时，人脸检测接口返回的数据是无序的，开发者可以按照实际使用需求进行排序，例如，将图片中的人脸按照从左至右的顺序进行排序。

编程调用人脸检测 API

首先，要对人脸检测接口返回的结构进行封装，建立与之对应的 Java 对象。由于人脸检测接口返回的参数较多，笔者只是将本例中需要用到的参数抽取出来，封装成 **Face** 对象，对应的代码如下：

[java] view plaincopy

```
1. package org.liufeng.course.pojo;
2.
3. /**
4.  * Face Model
5.  */
```



```
6.  * @author liufeng
7.  * @date 2013-12-18
8.  */
9.  public class Face implements Comparable<Face> {
10.     // 被检测出的每一张人脸都在 Face++系统中的标识符
11.     private String faceId;
12.     // 年龄估计值
13.     private int ageValue;
14.     // 年龄估计值的正负区间
15.     private int ageRange;
16.     // 性别: Male/Female
17.     private String genderValue;
18.     // 性别分析的可信度
19.     private double genderConfidence;
20.     // 人种: Asian/White/Black
21.     private String raceValue;
22.     // 人种分析的可信度
23.     private double raceConfidence;
24.     // 微笑程度
25.     private double smilingValue;
26.     // 人脸框的中心点坐标
27.     private double centerX;
28.     private double centerY;
29.
30.     public String getFaceId() {
31.         return faceId;
32.     }
33.
34.     public void setFaceId(String faceId) {
35.         this.faceId = faceId;
36.     }
37.
38.     public int getAgeValue() {
39.         return ageValue;
40.     }
41.
42.     public void setAgeValue(int ageValue) {
43.         this.ageValue = ageValue;
44.     }
45.
46.     public int getAgeRange() {
47.         return ageRange;
48.     }
49.
```

```
50.     public void setAgeRange(int ageRange) {
51.         this.ageRange = ageRange;
52.     }
53.
54.     public String getGenderValue() {
55.         return genderValue;
56.     }
57.
58.     public void setGenderValue(String genderValue) {
59.         this.genderValue = genderValue;
60.     }
61.
62.     public double getGenderConfidence() {
63.         return genderConfidence;
64.     }
65.
66.     public void setGenderConfidence(double genderConfidence) {
67.         this.genderConfidence = genderConfidence;
68.     }
69.
70.     public String getRaceValue() {
71.         return raceValue;
72.     }
73.
74.     public void setRaceValue(String raceValue) {
75.         this.raceValue = raceValue;
76.     }
77.
78.     public double getRaceConfidence() {
79.         return raceConfidence;
80.     }
81.
82.     public void setRaceConfidence(double raceConfidence) {
83.         this.raceConfidence = raceConfidence;
84.     }
85.
86.     public double getSmilingValue() {
87.         return smilingValue;
88.     }
89.
90.     public void setSmilingValue(double smilingValue) {
91.         this.smilingValue = smilingValue;
92.     }
93.
```

```

94.     public double getCenterX() {
95.         return centerX;
96.     }
97.
98.     public void setCenterX(double centerX) {
99.         this.centerX = centerX;
100.    }
101.
102.     public double getCenterY() {
103.         return centerY;
104.     }
105.
106.     public void setCenterY(double centerY) {
107.         this.centerY = centerY;
108.    }
109.
110.    // 根据人脸中心点坐标从左至右排序
111.    @Override
112.     public int compareTo(Face face) {
113.         int result = 0;
114.         if (this.getCenterX() > face.getCenterX())
115.             result = 1;
116.         else
117.             result = -1;
118.         return result;
119.     }
120. }

```

与普通 Java 类不同的是, **Face** 类实现了 **Comparable** 接口, 并实现了该接口的 **compareTo()** 方法, 这正是 **Java** 中对象排序的关键所在。112-119 行代码是通过比较每个 **Face** 的脸部中心点的横坐标来决定对象的排序方式, 这样能够实现检测出的多个 **Face** 按从左至右的先后顺序进行排序。

接下来, 是人脸检测 API 的调用及相关处理逻辑, 笔者将这些实现全部封装在 **FaceService** 类中, 该类的完整实现如下:

[java] view plaincopy

```

1.  package org.liufeng.course.service;
2.
3.  import java.io.BufferedReader;
4.  import java.io.InputStream;
5.  import java.io.InputStreamReader;
6.  import java.net.HttpURLConnection;
7.  import java.net.URL;
8.  import java.util.ArrayList;

```

```
9. import java.util.Collections;
10. import java.util.List;
11. import org.liufeng.course.pojo.Face;
12. import net.sf.json.JSONArray;
13. import net.sf.json.JSONObject;
14.
15. /**
16.  * 人脸检测服务
17.  *
18.  * @author liufeng
19.  * @date 2013-12-18
20.  */
21. public class FaceService {
22.     /**
23.      * 发送 http 请求
24.      *
25.      * @param requestUrl 请求地址
26.      * @return String
27.      */
28.     private static String httpRequest(String requestUrl) {
29.         StringBuffer buffer = new StringBuffer();
30.         try {
31.             URL url = new URL(requestUrl);
32.             HttpURLConnection httpUrlConn = (HttpURLConnection) url.openConnection();
33.             httpUrlConn.setDoInput(true);
34.             httpUrlConn.setRequestMethod("GET");
35.             httpUrlConn.connect();
36.             // 将返回的输入流转换成字符串
37.             InputStream inputStream = httpUrlConn.getInputStream();
38.             InputStreamReader inputStreamReader = new InputStreamReader(inputStream, "utf-8");
39.             BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
40.
41.             String str = null;
42.             while ((str = bufferedReader.readLine()) != null) {
43.                 buffer.append(str);
44.             }
45.             bufferedReader.close();
46.             inputStreamReader.close();
47.             // 释放资源
48.             inputStream.close();
49.             inputStream = null;
```

```
50.         httpUrlConn.disconnect();
51.
52.     } catch (Exception e) {
53.         e.printStackTrace();
54.     }
55.     return buffer.toString();
56. }
57.
58. /**
59.  * 调用 Face++ API 实现人脸检测
60.  *
61.  * @param picUrl 待检测图片的访问地址
62.  * @return List<Face> 人脸列表
63.  */
64. private static List<Face> faceDetect(String picUrl) {
65.     List<Face> faceList = new ArrayList<Face>();
66.     try {
67.         // 拼接 Face++人脸检测的请求地址
68.         String queryUrl = "http://apicn.faceplusplus.com/v2/detection/de
        tect?url=URL&api_secret=API_SECRET&api_key=API_KEY";
69.         // 对 URL 进行编码
70.         queryUrl = queryUrl.replace("URL", java.net.URLEncoder.encode(pi
        cUrl, "UTF-8"));
71.         queryUrl = queryUrl.replace("API_KEY", "替换成自己的 API Key");
72.         queryUrl = queryUrl.replace("API_SECRET", "替换成自己的
        API Secret");
73.         // 调用人脸检测接口
74.         String json = httpRequest(queryUrl);
75.         // 解析返回 json 中的 Face 列表
76.         JSONArray jsonArray = JSONObject.fromObject(json).getJSONArray("
        face");
77.         // 遍历检测到的人脸
78.         for (int i = 0; i < jsonArray.size(); i++) {
79.             // face
80.             JSONObject faceObject = (JSONObject) jsonArray.get(i);
81.             // attribute
82.             JSONObject attrObject = faceObject.getJSONObject("attribute")
            ;
83.             // position
84.             JSONObject posObject = faceObject.getJSONObject("position");
85.
86.             Face face = new Face();
            face.setFaceId(faceObject.getString("face_id"));
```

```

87.                face.setAgeValue(attrObject.getJSONObject("age").getInt("value"));
88.                face.setAgeRange(attrObject.getJSONObject("age").getInt("range"));
89.                face.setGenderValue(genderConvert(attrObject.getJSONObject("gender").getString("value")));
90.                face.setGenderConfidence(attrObject.getJSONObject("gender").getDouble("confidence"));
91.                face.setRaceValue(raceConvert(attrObject.getJSONObject("race").getString("value")));
92.                face.setRaceConfidence(attrObject.getJSONObject("race").getDouble("confidence"));
93.                face.setSmilingValue(attrObject.getJSONObject("smiling").getDouble("value"));
94.                face.setCenterX(posObject.getJSONObject("center").getDouble("x"));
95.                face.setCenterY(posObject.getJSONObject("center").getDouble("y"));
96.                faceList.add(face);
97.            }
98.            // 将检测出的 Face 按从左至右的顺序排序
99.            Collections.sort(faceList);
100.        } catch (Exception e) {
101.            faceList = null;
102.            e.printStackTrace();
103.        }
104.        return faceList;
105.    }
106.
107.    /**
108.     * 性别转换（英文->中文）
109.     *
110.     * @param gender
111.     * @return
112.     */
113.    private static String genderConvert(String gender) {
114.        String result = "男性";
115.        if ("Male".equals(gender))
116.            result = "男性";
117.        else if ("Female".equals(gender))
118.            result = "女性";
119.
120.        return result;
121.    }

```

```
122.
123.     /**
124.      * 人种转换（英文->中文）
125.      *
126.      * @param race
127.      * @return
128.      */
129.     private static String raceConvert(String race) {
130.         String result = "黄色";
131.         if ("Asian".equals(race))
132.             result = "黄色";
133.         else if ("White".equals(race))
134.             result = "白色";
135.         else if ("Black".equals(race))
136.             result = "黑色";
137.         return result;
138.     }
139.
140.     /**
141.      * 根据人脸识别结果组装消息
142.      *
143.      * @param faceList 人脸列表
144.      * @return
145.      */
146.     private static String makeMessage(List<Face> faceList) {
147.         StringBuffer buffer = new StringBuffer();
148.         // 检测到 1 张脸
149.         if (1 == faceList.size()) {
150.             buffer.append("共检测到 ").append(faceList.size()).append(" 张人
151. 脸").append("\n");
152.             for (Face face : faceList) {
153.                 buffer.append(face.getRaceValue()).append("人种,");
154.                 buffer.append(face.getGenderValue()).append(",");
155.                 buffer.append(face.getAgeValue()).append("岁左右
156. ").append("\n");
157.             }
158.         }
159.         // 检测到 2-10 张脸
160.         else if (faceList.size() > 1 && faceList.size() <= 10) {
161.             buffer.append("共检测到 ").append(faceList.size()).append(" 张人
162. 脸, 按脸部中心位置从左至右依次为: ").append("\n");
163.             for (Face face : faceList) {
164.                 buffer.append(face.getRaceValue()).append("人种,");
165.                 buffer.append(face.getGenderValue()).append(",");
```

```
163.         buffer.append(face.getAgeValue()).append("岁左右
    ").append("\n");
164.     }
165. }
166. // 检测到 10 张脸以上
167. else if (faceList.size() > 10) {
168.     buffer.append("共检测到 ").append(faceList.size()).append(" 张人
        脸").append("\n");
169.     // 统计各人种、性别的人数
170.     int asiaMale = 0;
171.     int asiaFemale = 0;
172.     int whiteMale = 0;
173.     int whiteFemale = 0;
174.     int blackMale = 0;
175.     int blackFemale = 0;
176.     for (Face face : faceList) {
177.         if ("黄色".equals(face.getRaceValue()))
178.             if ("男性".equals(face.getGenderValue()))
179.                 asiaMale++;
180.         else
181.             asiaFemale++;
182.         else if ("白色".equals(face.getRaceValue()))
183.             if ("男性".equals(face.getGenderValue()))
184.                 whiteMale++;
185.         else
186.             whiteFemale++;
187.         else if ("黑色".equals(face.getRaceValue()))
188.             if ("男性".equals(face.getGenderValue()))
189.                 blackMale++;
190.         else
191.             blackFemale++;
192.     }
193.     if (0 != asiaMale || 0 != asiaFemale)
194.         buffer.append("黄色人种: ").append(asiaMale).append("男
            ").append(asiaFemale).append("女").append("\n");
195.     if (0 != whiteMale || 0 != whiteFemale)
196.         buffer.append("白色人种: ").append(whiteMale).append("男
            ").append(whiteFemale).append("女").append("\n");
197.     if (0 != blackMale || 0 != blackFemale)
198.         buffer.append("黑色人种: ").append(blackMale).append("男
            ").append(blackFemale).append("女").append("\n");
199. }
200. // 移除末尾空格
```



```

201.         buffer = new StringBuffer(buffer.substring(0, buffer.lastIndexOf("\n")));
202.         return buffer.toString();
203.     }
204.
205.     /**
206.      * 提供给外部调用的人脸检测方法
207.      *
208.      * @param picUrl 待检测图片的访问地址
209.      * @return String
210.      */
211.     public static String detect(String picUrl) {
212.         // 默认回复信息
213.         String result = "未识别到人脸，请换一张清晰的照片再试！";
214.         List<Face> faceList = faceDetect(picUrl);
215.         if (null != faceList) {
216.             result = makeMessage(faceList);
217.         }
218.         return result;
219.     }
220.
221.     public static void main(String[] args) {
222.         String picUrl = "http://pic11.nipic.com/20101111/6153002_002722872554_2.jpg";
223.         System.out.println(detect(picUrl));
224.     }
225. }

```

上述代码虽然多，但条理很清晰，并不难理解，所以笔者只挑重点的进行讲解，主要说明如下：

- 1) 70 行：参数 url 表示图片的链接，由于链接中存在特殊字符，作为参数传递时必须进行 URL 编码。请读者记住：不管是什么应用，调用什么接口，凡是通过 GET 传递的参数中可能会包含特殊字符，都必须进行 URL 编码，除了中文以外，特殊字符还包括等号“=”、与“&”、空格“ ”等。
- 2) 76-97 行：使用 JSON-lib 解析人脸检测接口返回的 JSON 数据，并将解析结果存入 List 中。
- 3) 99 行：对集合中的对象进行排序，使用 Collections.sort()方法排序的前提是集合中的 Face 对象实现了 Comparable 接口。
- 4) 146-203 行：组装返回给用户的消息内容。考虑到公众平台的文本消息内容长度有限制，当一张图片中识别出的人脸过多，则只返回一些汇总信息给用户。
- 5) 211-219 行：detect()方法是 public 的，提供给其他类调用。笔者可以在本地的开发工具中运行上面的 main()方法，测试 detect()方法的输出。

公众账号后台的实现

在公众账号后台的 **CoreService** 类中，需要对用户发送的消息类型进行判断，如果是图片消息，则调用人脸检测方法进行分析，如果是其他消息，则返回人脸检测的使用指南。

CoreService 类的完整代码如下：

[java] view plaincopy

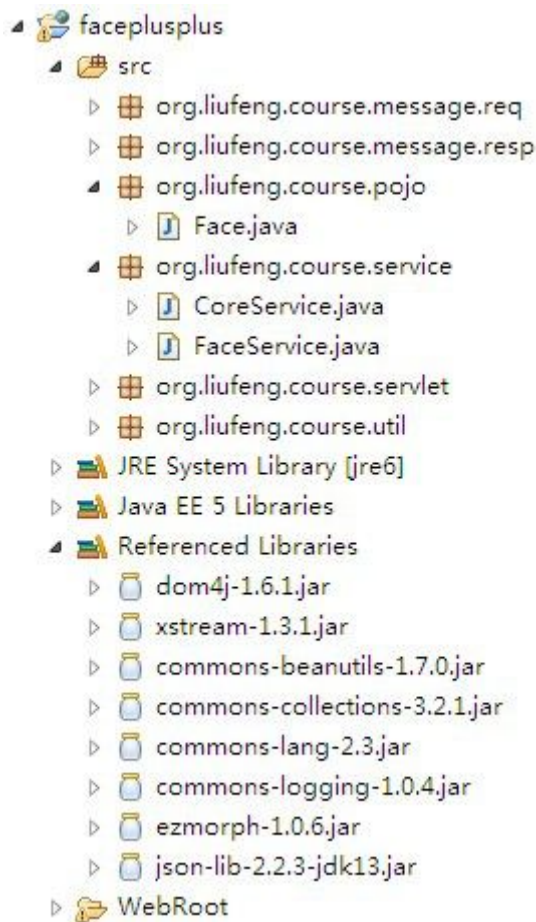
```
1. package org.liufeng.course.service;
2.
3. import java.util.Date;
4. import java.util.Map;
5. import javax.servlet.http.HttpServletRequest;
6. import org.liufeng.course.message.resp.TextMessage;
7. import org.liufeng.course.util.MessageUtil;
8.
9. /**
10.  * 核心服务类
11.  *
12.  * @author liufeng
13.  * @date 2013-12-19
14.  */
15. public class CoreService {
16.     /**
17.      * 处理微信发来的请求
18.      */
19.     public static String processRequest(HttpServletRequest request) {
20.         // 返回给微信服务器的 xml 消息
21.         String respXml = null;
22.         try {
23.             // xml 请求解析
24.             Map<String, String> requestMap = MessageUtil.parseXml(request);
25.
26.             // 发送方帐号 (open_id)
27.             String fromUserName = requestMap.get("FromUserName");
28.             // 公众帐号
29.             String toUserName = requestMap.get("ToUserName");
30.             // 消息类型
31.             String msgType = requestMap.get("MsgType");
32.
33.             // 回复文本消息
34.             TextMessage textMessage = new TextMessage();
35.             textMessage.setToUserName(fromUserName);
```

```

35.         textMessage.setFromUserName(toUserName);
36.         textMessage.setCreateTime(new Date().getTime());
37.         textMessage.setMsgType(MessageUtil.RESP_MESSAGE_TYPE_TEXT);
38.
39.         // 图片消息
40.         if (MessageUtil.REQ_MESSAGE_TYPE_IMAGE.equals(msgType)) {
41.             // 取得图片地址
42.             String picUrl = requestMap.get("PicUrl");
43.             // 人脸检测
44.             String detectResult = FaceService.detect(picUrl);
45.             textMessage.setContent(detectResult);
46.         }
47.         // 其它类型的消息
48.         else
49.             textMessage.setContent(getUsage());
50.
51.         respXml = MessageUtil.textMessageToXml(textMessage);
52.     } catch (Exception e) {
53.         e.printStackTrace();
54.     }
55.     return respXml;
56. }
57.
58. /**
59.  * 人脸检测帮助菜单
60.  */
61. public static String getUsage() {
62.     StringBuffer buffer = new StringBuffer();
63.     buffer.append("人脸检测使用指南").append("\n\n");
64.     buffer.append("发送一张清晰的照片，就能帮你分析出种族、年龄、性别等信息").append("\n");
65.     buffer.append("快来试试你是不是长得太着急");
66.     return buffer.toString();
67. }
68. }

```

到这里，人脸检测应用就全部开发完成了，整个项目的完整结构如下：



运行结果如下:



笔者用自己的相片测试了两次，测试结果分别是 26 岁、30 岁，这与笔者的实际年龄相差不大，可见，Face++ 的人脸检测准确度还是比较高的。为了增加人脸检测应用的趣味性和娱乐性，笔者忽略了年龄估计值的正负区间。读者可以充分发挥自己的想像力和创造力，使用 Face++ API 实现更多实用、有趣的功能。**应用开发不是简单的接口调用！**

微信公众平台开发教程第 20 篇-新手解惑 40 则

笔者在 CSDN 博客频道推出微信公众平台开发教程之后，接触了许多公众平台开发爱好者，也帮助他们解决了许多实际的问题，当然这其中有很多问题都是重复的，因此，笔者将这些问题及解答整理出来，以帮助更多初学者少走弯路。

1、订阅号与服务号的主要区别是什么？

订阅号每天能群发一条消息，没有自定义菜单及高级接口权限；服务号有自定义菜单及高级接口权限，但每月只能群发一条消息。

2、到底该申请订阅号还是服务号？

申请哪种类型的公众账号，主要取决于账号的用途。服务号主要面向企业和组织，旨在为用户提供服务；订阅号主要面向媒体和个人，旨在为用户提供信息和资讯。

3、订阅号是否支持编程开发？

不管是订阅号，还是服务号，在高级功能中都有编辑模式和开发模式，**订阅号也支持编程开发，同样也能与企业系统对接。**

4、为什么申请的公众账号没有高级功能？

公众账号注册后，要经过微信团队的审核，在审核未完成之前不显示高级功能。一般审核会在 15 个工作日内完成，如果一两周没审核通过均属正常现象，还请耐心等待。

5、现在订阅号能否申请自定义菜单？

不能。那**为什么有些订阅号有自定义菜单？**这是历史遗留问题。**2013 年 8 月 5 日**，随着微信 5.0 的发布，公众账号被划分为订阅号和服务号，所有的公众账号都被默认为订阅号，并且有一次转服务号的机会，许多在此之前申请到自定义菜单的账号并没有转为服务号，所以就存在一些订阅号有自定义菜单，例如：**36 氪、蓉城先锋、天府之光**等。

补充：2013 年 12 月 24 日，公众平台针对订阅号做了重要更新。政府、传统媒体、明星等非企业性质的订阅号可以申请微信认证，通过微信认证的订阅号可获得自定义菜单接口权限。

6、现在申请的订阅号能否转服务号？

不能。只有 2013 年 8 月 5 日微信 5.0 发布以前申请的订阅号才有一次转服务号的机会，在此之后申请的订阅号不能转服务号。

那如果真的有转服务号的需求怎么解决？只能重新申请一个服务号。

7、目前一个身份证号能申请几个公众账号？

2 个。

8、使用一个公司的材料能申请多少个公众账号？

没有限制。

9、在注册公众账号时，提示“你注册的公众号名称存在侵权风险，请先完成微博验证”，这是什么意思？

公众平台对一些可能存在侵权的关键词进行了保护，例如：“微信”、“移动”、“搜狐”等，如果注册的公众账号名称中包含这类关键词，提交时就会提示存在侵权风险。

遇到这种情况时，要么避开这些关键词换个名称注册，要么就根据提示先完成微博验证再继续注册。

10、个人能否申请服务号？

不能，个人只能申请订阅号。服务号的运营主体必须为组织，例如：企业、政府、其他组织等。

11、公众账号的名称可以重复吗？

公众账号的名称可以重复，不用担心被他人抢注。

12、公众账号的名称可以修改吗？

公众账号名称一经设置无法修改，公众平台没有提供账号改名的功能，因此在注册账号时取名应谨慎。

13、微信认证与微博认证有什么区别？

微信认证是针对于服务号，微博认证是针对于订阅号。也就是说，订阅号只能申请微博认证，服务号只能申请微信认证。

14、是否需要粉丝数达到 500 才能申请微信认证？

只要是服务号都可以申请微信认证，与粉丝数无关。只有订阅号申请微博认证才要求粉丝数必须达到 500。

15、编辑模式与开发模式能够同时使用吗？

不能，这两种模式是互斥的，开启编辑模式就必须关闭开发模式，开启开发模式就必须关闭编辑模式。

16、现在用的是编辑模式，以后还可以选择使用开发模式吗？

可以，任何时候都可以根据需要切换到另外一种模式。

17、编辑模式切换到开发模式之后，在编辑模式中设置的内容还在吗？还有效吗？

在编辑模式中设置的内容，只要自己不手动删除，会永远存在的，但这些设置在开发模式下无效。

18、开发模式的菜单为什么突然消失了？

菜单不会无缘无故的消失，如果开发人员没有手动删除，那一定是有人开启过编辑模式引起的。请注意：开启编辑模式后，在开发模式下创建的菜单会被删除。

19、使用开发模式需要具备哪些条件？

1) 至少掌握一门编程语言；2) 具备公网服务器资源。

20、微信公众平台支持哪些编程语言？应该如何选择？

凡是支持动态 Web 开发的编程语言都能够用于微信公众平台开发，例如：Java、PHP、ASP.NET、Ruby、Python、Node.js 等。

开者人员应该选择自己最擅长的编程语言进行开发，如果都不擅长怎么办？如果都不擅长，建议选择 Java 或 PHP，原因在于网上关于微信公众平台开发的资料大都是基于 Java 和 PHP 的，开发起来要相对容易。

21、没有公网服务器资源怎么办？

1) 免费：可以考虑使用云环境，例如，BAE（Baidu App Engine，百度应用引擎）和 SAE（Sina App Engine，新浪应用引擎）。

2) 付费：可以考虑租用 VPS（Virtual Private Server，虚拟专用服务器）或阿里云的云服务器。

如果仅是为了学习微信公众平台开发，个人建议使用 BAE。

22、如果想使用 Java 进行微信公众平台开发至少需要掌握哪些内容？

至少需要掌握 Java 基础知识、JSP、Servlet、Javabean 和 JDBC（操作数据库）。

23、公司的项目大都是基于 SSH 框架进行开发，能使用 SSH 开发微信公众账号吗？

当然可以，前面说过，凡是支持动态 Web 开发的编程语言都能用于微信公众平台开发。其实，**Struts 本质上也是 Servlet**。

24、柳峰老师，可以给我一份微信公众平台项目的源码吗？

想要源码的朋友请您免开尊口，我认为这不是一种很好的学习方式和态度，而是一种浮躁的表现。博客中的教程已经讲的很详细了，并且贴出了所有代码（一行也不少），如果这样还不愿意花点时间去理解、消化和动动手，我也无能为力！

PS：曾经也有一些开发者、创业团队和公司提出要买小 q 机器人（xiaoqrobot）的源码，有的开价是 5 位数，但都被我拒绝了。相比之下，我更愿意把小 q 机器人的完整实现过程写成一篇篇技术文章免费分享出来，带动更多的开发者加入到微信公众平台开发阵营！

25、公众账号能够通过程序主动向关注用户发消息吗？

截止目前，公众平台还没有开放主动向用户发消息的接口。**为什么招行可以？**我前面说的是没有“开放”主动发消息的接口，并不代表没有该接口。如果贵公司也有招行的实力，我相信你也有办法申请到；如果没有这样的实力，那就不要费事了。

26、订阅号使用开发模式能够向用户回复图片、语音和视频消息吗？

可以，虽然订阅号没有多媒体文件上传接口权限，无法通过上传多媒体文件到微信服务器获取 **MediaId**，但仍可以变相得到 **MediaId**，同样可以实现回复多媒体消息。变相的实现方法是将用户发送给公众账号的多媒体消息的 **MediaId** 记录下来，给用户回复多媒体消息时可以使用。

27、订阅号使用开发模式能够向用户回复音乐消息吗？

可以。

28、音乐消息包含参数 **ThumbMediaId**，没有高级接口权限的公众账号无法获得

ThumbMediaId，怎么回复音乐消息？

ThumbMediaId 不是音乐消息的必须参数，给用户回复音乐消息时可以不传 **ThumbMediaId** 参数，类似下面这种示例格式也能正确回复音乐消息：


```
1. <xml>
2.   <ToUserName><![CDATA[toUser]]></ToUserName>
3.   <FromUserName><![CDATA[fromUser]]></FromUserName>
4.   <CreateTime>12345678</CreateTime>
5.   <MsgType><![CDATA[music]]></MsgType>
6.   <Music>
7.     <Title><![CDATA[TITLE]]></Title>
8.     <Description><![CDATA[DESCRIPTION]]></Description>
9.     <MusicUrl><![CDATA[MUSIC_Url]]></MusicUrl>
10.    <HQMusicUrl><![CDATA[HQ_MUSIC_Url]]></HQMusicUrl>
11.  </Music>
12. </xml>
```

29、订阅号与非微信认证的服务号能够向回复哪些类型的消息？

在开发模式下，订阅号与非微信认证的服务号只能向用户回复文本消息、音乐消息和图文消息。

30、为什么项目代码与柳峰老师教程中的一样，发消息给公众账号却没有任何响应？

这是我写微信公众平台开发教程以来，初学者给我反馈最多的问题。可以肯定的是，至今为止，我博客中贴出的所有代码全部都能正常运行，没有任何问题。遇到上面这种问题大都是由以下三种情况引起：

- 1) 在公众平台开发模式下，成为开发者却忘记开启开发模式，即开发模式的开关是关闭状态。
- 2) 通过上传 WAR 包的方式部署应用时，导出的 WAR 包中没有包含 JAR。建议初学者直接将项目需要的 JAR 拷贝到项目中，这样通过开发工具导出的 WAR 包就会包含 JAR。
- 3) 项目中引入的第三方 JAR 包与笔者教程中使用的 JAR 包版本不一致。

31、为什么自定义菜单创建成功了，在微信客户端的公众账号上却不显示？

由于微信客户端缓存的原因，自定义菜单创建成功后，需要 24 小时以后才能显示出来。开发者在测试时，可以尝试取消关注公众账号后再次关注，这样能立即看到最新的菜单效果。

PS：菜单更新、菜单删除也会有缓存。

32、如果要更新公众账号的自定义菜单，需要先将原有菜单删除吗？

不需要，直接执行菜单创建方法即可，每次创建菜单会自动覆盖以前的菜单。

33、什么是微网站？

微网站是新瓶装老酒，被一些搞营销的人给神化了，以至于很多开发者都在问什么是微网站，如何开发微网站。微网站本质上就是以微信浏览器为入口的手机网站（Web APP），能够

兼容 Android、iOS、WP 等操作系统。开发微网站用到的技术与开发普通网站一样，都是基于 HTML（HTML5）、CSS、Javascript 等，所以有普通网站开发经验的开发者，完全有能力开发微网站。

PS：初学者以后再看到什么以“微”开头的新名词，例如：微商城、微客服、微统计，直接把“微”字去掉或者把“微”当作是“基于微信的”就不难理解了。

34、什么是模拟登录？模拟登录微信公众平台能够干什么？

模拟登录指的是通过程序模拟用户在浏览器上的操作。例如，我们通过浏览器访问微信公众平台，先要登录，登录成功后能够查看用户信息、给用户回复消息、群发消息等，其实通过程序也能够实现这些操作。

PS：对于模拟登录，官方并没有明确表态是允许还是禁止，请谨慎使用，万一哪天被封号就不划算了，也没法向关注你公众账号的用户交待。

35、微信认证是如何收费的？

服务号申请微信认证需要支付 300 元/次的审核服务费用，无论最终的认证审核通过与否，都需要支付这笔费用。微信认证成功后，认证的有效期是一年，在有效期快结束时还要再次申请微信认证。

36、微信支付如何申请？

截止目前，微信公众平台仍未开放微信支付权限的申请。为什么广东联通、小米手机这些账号有微信支付权限？这些公司大都与微信有着合作关系，提前享受这些权限一点也不奇怪。

37、临时带参二维码有哪些应用场景？

通过微信扫描二维码登录微信网页版，就是临时带参二维码的典型应用场景。

38、微信公众平台开发一般如何调试？

微信公众平台提供的在线接口调试工具旨在帮助开发者检测调用公众平台接口时传入的参数是否正确，这款工具对开发者的帮助其实并不大。对于调试本地运行的公众账号后台程序，这里给读者推荐两种方法：

- 1) 使用“微信开发调试小工具”，该工具支持在本地调试，工具的用法及下载请访问：<http://www.cnblogs.com/linkbiz/>。
- 2) 使用花生壳动态域名解析软件，通过路由器端口映射，可以将自己的电脑变成一台外网服务器，这样本机运行的公众账号后台程序就能直接与微信服务器进行交互了。

39、为什么项目在本地运行正常，也能获取到数据，部署到服务器上之后公众账号没有任何响应？

遇到这类情况，请读者尝试从以下几个方面排查问题：

- 1) 检查项目在服务器上是否部署成功，可以尝试方法以前能够正常运行的功能模块，看能否正确响应，以便缩小问题范围。
- 2) 检查项目中通过 URL 传递参数时，如果传递特殊字符（例如：中文、+、&等），是否对特殊字符进行了编码。
- 3) 检查程序的处理是否超时，如果超过 5 秒，公众账号不响应。
- 4) 检查返回的文本消息、图文消息是否超过限制（文本消息长度 ≤ 2048 字节，图文消息条数 ≤ 10 条），若超过限制，公众账号不响应。
- 5) 公众账号不响应也有可能是微信公众平台自身故障导致。

40、为什么 URL 在浏览器能访问，放到微信上却不能访问？

请检查 URL 中是否包含特殊字符，例如：中文、+、&等，PC 上的浏览器通常都会对 URL 中包含的特殊字符自动编码，但有些浏览器不会。为了保证所有的浏览器都能正常访问 URL，请务必对 URL 中包含特殊字符显示编码，显示编码的意思是代码中能够明确看出编码方式是 UTF8、GB2312 或者其它。例如像下面这样：

[java] view plaincopy

1. // 采用操作系统默认的字符集进行编码，在不同的操作系统上表现不一致，不推荐
2. java.net.URLEncoder.encode(chinese);

[java] view plaincopy

1. // 显示编码，推荐用法
2. java.net.URLEncoder.encode(chinese, "UTF-8");

PS: 很多初学者都认为只有 URL 中包含中文时才需要编码，结果导致 OAuth2.0 授权接口、通过 ticket 换取二维码接口总是调用不成功。OAuth2.0 授权接口中的回调地址 redirect_uri 中包含大量特殊字符必须进行编码，通过 ticket 换取二维码接口中的 ticket 中可能包含+号也要进行编码。

送给初学者一条中肯的建议：不要总是怀疑微信公众平台的接口或者有经验的开发者分享的程序代码有问题，最先应该怀疑自己写的程序有问题，这样才有助于发现问题，从而解决问题。请相信：一套久经考验的平台、程序被初学者发现 BUG 的情况并不多见。

微信公众平台开发教程第 21 篇-“可信网址”白名单

防欺诈警告

不知道读者是否留意过这种情况：通过微信内置浏览器打开带有表单的页面，点击其中任何一个表单项都会在窗口顶部显示红色背景的防欺诈警告信息“防欺诈盗号，请勿支付或输入 qq 密码”，如下图所示。



防欺诈警告是腾讯微信团队基于安全考虑而设计的，但这种设计会严重影响用户体验。微信公众平台有一个“可信网址”白名单，它是由微信团队负责管理的。如果微信公众号使用的网址在“可信网址”白名单之列，用户填写表单时就不会弹出防欺诈警告。例如，使用“招商银行信用卡中心”、“中国南方航空”、“广东联通”等公众账号的表单页面就不会出现防欺诈警告，这样的用户体验会好很多。

“可信网址”的申请

如果读者需要将所拥有或合法管理的网址加入“可信网址”白名单，需要向微信团队提供相关材料（申请书、申请人主体材料、申请网址及权利证明相关材料和申请人保证）进行申请，这些材料的说明如下。

- 1) 申请书下载地址：<https://mp.weixin.qq.com/html/edition/res/urlrequest.doc>。
- 2) 申请人主体材料包括：申请人的姓名（名称）、联系方式、地址及营业执照（单位）、身份证（个人）等证明权利人主体资格的材料。
- 3) 申请网址及权利证明相关材料：申请成为可信网址的域名及域名登记备案资料，如 ICP 备案证明、相关授权证明等证明材料。
- 4) 申请人保证：申请人承诺在申请书中的陈述和提供的相关材料皆是真实、有效和合法的，并保证对此独立承担完全责任，并就可能因此造成的损害进行赔偿，包括因腾讯根

据申请人申请网址或相关网站内容而给用户或腾讯公司造成的任何损失,包括但不限于用户或腾讯因此而产生的财产损失及腾讯名誉、商誉损害等。

PS: 申请“可信网址”的具体细节读者可以拨打微信客服电话或发邮件至 weixinmp@qq.com 进行咨询。