## [微信公众平台的Java开发详解（工程代码+解析）](http://blog.csdn.net/pamchen/article/details/38718947)

说明：  
本次的教程主要是对**微信公众平台开发者模式**的讲解，网络上很多类似文章，但很多都让初学微信开发的人一头雾水，所以总结自己的微信开发经验，将微信开发的整个过程系统的列出，并对主要代码进行讲解分析，让初学者尽快上手。

在阅读本文之前，应对微信公众平台的官方开发文档有所了解，知道接收和发送的都是xml格式的数据。另外，在做内容回复时用到了***[图灵机器人的api接口](http://www.tuling123.com/" \t "_blank)***，这是一个**自然语言解析的开放平台**，可以帮我们解决整个微信开发过程中最困难的问题，此处不多讲，下面会有其详细的调用方式。

**1.1 在登录微信官方平台之后，开启开发者模式，此时需要我们填写url和token，所谓url就是我们自己服务器的接口，用WechatServlet.java来实现，相关解释已经在注释中说明，代码如下：**

**[java]** [view plaincopy](http://blog.csdn.net/pamchen/article/details/38718947)

1. **package** demo.servlet;
3. **import** java.io.BufferedReader;
4. **import** java.io.IOException;
5. **import** java.io.InputStream;
6. **import** java.io.InputStreamReader;
7. **import** java.io.OutputStream;
9. **import** javax.servlet.ServletException;
10. **import** javax.servlet.http.HttpServlet;
11. **import** javax.servlet.http.HttpServletRequest;
12. **import** javax.servlet.http.HttpServletResponse;
14. **import** demo.process.WechatProcess;
15. /\*\*
16. \* 微信服务端收发消息接口
17. \*
18. \* @author pamchen-1
19. \*
20. \*/
21. **public** **class** WechatServlet **extends** HttpServlet {
23. /\*\*
24. \* The doGet method of the servlet. <br>
25. \*
26. \* This method is called when a form has its tag value method equals to get.
27. \*
28. \* @param request
29. \*            the request send by the client to the server
30. \* @param response
31. \*            the response send by the server to the client
32. \* @throws ServletException
33. \*             if an error occurred
34. \* @throws IOException
35. \*             if an error occurred
36. \*/
37. **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)
38. **throws** ServletException, IOException {
39. request.setCharacterEncoding("UTF-8");
40. response.setCharacterEncoding("UTF-8");
42. /\*\* 读取接收到的xml消息 \*/
43. StringBuffer sb = **new** StringBuffer();
44. InputStream is = request.getInputStream();
45. InputStreamReader isr = **new** InputStreamReader(is, "UTF-8");
46. BufferedReader br = **new** BufferedReader(isr);
47. String s = "";
48. **while** ((s = br.readLine()) != **null**) {
49. sb.append(s);
50. }
51. String xml = sb.toString(); //次即为接收到微信端发送过来的xml数据
53. String result = "";
54. /\*\* 判断是否是微信接入激活验证，只有首次接入验证时才会收到echostr参数，此时需要把它直接返回 \*/
55. String echostr = request.getParameter("echostr");
56. **if** (echostr != **null** && echostr.length() > 1) {
57. result = echostr;
58. } **else** {
59. //正常的微信处理流程
60. result = **new** WechatProcess().processWechatMag(xml);
61. }
63. **try** {
64. OutputStream os = response.getOutputStream();
65. os.write(result.getBytes("UTF-8"));
66. os.flush();
67. os.close();
68. } **catch** (Exception e) {
69. e.printStackTrace();
70. }
71. }
73. /\*\*
74. \* The doPost method of the servlet. <br>
75. \*
76. \* This method is called when a form has its tag value method equals to
77. \* post.
78. \*
79. \* @param request
80. \*            the request send by the client to the server
81. \* @param response
82. \*            the response send by the server to the client
83. \* @throws ServletException
84. \*             if an error occurred
85. \* @throws IOException
86. \*             if an error occurred
87. \*/
88. **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)
89. **throws** ServletException, IOException {
90. doGet(request, response);
91. }
93. }

**1.2 相应的web.xml配置信息如下，在生成WechatServlet.java的同时，可自动生成web.xml中的配置。前面所提到的url处可以填写例如：**

***http;//服务器地址/项目名/wechat.do***

**[html]** [view plaincopy](http://blog.csdn.net/pamchen/article/details/38718947)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<web-app** version="2.5"
3. xmlns="http://java.sun.com/xml/ns/javaee"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
6. http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"**>**
7. **<servlet>**
8. **<description>**This is the description of my J2EE component**</description>**
9. **<display-name>**This is the display name of my J2EE component**</display-name>**
10. **<servlet-name>**WechatServlet**</servlet-name>**
11. **<servlet-class>**demo.servlet.WechatServlet**</servlet-class>**
12. **</servlet>**
14. **<servlet-mapping>**
15. **<servlet-name>**WechatServlet**</servlet-name>**
16. **<url-pattern>**/wechat.do**</url-pattern>**
17. **</servlet-mapping>**
18. **<welcome-file-list>**
19. **<welcome-file>**index.jsp**</welcome-file>**
20. **</welcome-file-list>**
21. **</web-app>**

**1.3 通过以上代码，我们已经实现了微信公众平台开发的框架，即开通开发者模式并成功接入、接收消息和发送消息这三个步骤。**

**下面就讲解其核心部分——解析接收到的xml数据，并以文本类消息为例，通过图灵机器人api接口实现智能回复。**  
  
  
2.1 首先看一下整体流程处理代码，包括：xml数据处理、调用图灵api、封装返回的xml数据。

**[java]** [view plaincopy](http://blog.csdn.net/pamchen/article/details/38718947)

1. **package** demo.process;
3. **import** java.util.Date;
5. **import** demo.entity.ReceiveXmlEntity;
7. /\*\*
8. \* 微信xml消息处理流程逻辑类
9. \* @author pamchen-1
10. \*
11. \*/
12. **public** **class** WechatProcess {
13. /\*\*
14. \* 解析处理xml、获取智能回复结果（通过图灵机器人api接口）
15. \* @param xml 接收到的微信数据
16. \* @return  最终的解析结果（xml格式数据）
17. \*/
18. **public** String processWechatMag(String xml){
19. /\*\* 解析xml数据 \*/
20. ReceiveXmlEntity xmlEntity = **new** ReceiveXmlProcess().getMsgEntity(xml);
22. /\*\* 以文本消息为例，调用图灵机器人api接口，获取回复内容 \*/
23. String result = "";
24. **if**("text".endsWith(xmlEntity.getMsgType())){
25. result = **new** TulingApiProcess().getTulingResult(xmlEntity.getContent());
26. }
28. /\*\* 此时，如果用户输入的是“你好”，在经过上面的过程之后，result为“你也好”类似的内容
29. \*  因为最终回复给微信的也是xml格式的数据，所有需要将其封装为文本类型返回消息
30. \* \*/
31. result = **new** FormatXmlProcess().formatXmlAnswer(xmlEntity.getFromUserName(), xmlEntity.getToUserName(), result);
33. **return** result;
34. }
35. }

**2.2 解析接收到的xml数据，此处有两个类，ReceiveXmlEntity.java和ReceiveXmlProcess.java，通过反射的机制动态调用实体类中的set方法，可以避免很多重复的判断，提高代码效率，代码如下：**

**[java]** [view plaincopy](http://blog.csdn.net/pamchen/article/details/38718947)

1. **package** demo.entity;
2. /\*\*
3. \* 接收到的微信xml实体类
4. \* @author pamchen-1
5. \*
6. \*/
7. **public** **class** ReceiveXmlEntity {
8. **private** String ToUserName="";
9. **private** String FromUserName="";
10. **private** String CreateTime="";
11. **private** String MsgType="";
12. **private** String MsgId="";
13. **private** String Event="";
14. **private** String EventKey="";
15. **private** String Ticket="";
16. **private** String Latitude="";
17. **private** String Longitude="";
18. **private** String Precision="";
19. **private** String PicUrl="";
20. **private** String MediaId="";
21. **private** String Title="";
22. **private** String Description="";
23. **private** String Url="";
24. **private** String Location\_X="";
25. **private** String Location\_Y="";
26. **private** String Scale="";
27. **private** String Label="";
28. **private** String Content="";
29. **private** String Format="";
30. **private** String Recognition="";
32. **public** String getRecognition() {
33. **return** Recognition;
34. }
35. **public** **void** setRecognition(String recognition) {
36. Recognition = recognition;
37. }
38. **public** String getFormat() {
39. **return** Format;
40. }
41. **public** **void** setFormat(String format) {
42. Format = format;
43. }
44. **public** String getContent() {
45. **return** Content;
46. }
47. **public** **void** setContent(String content) {
48. Content = content;
49. }
50. **public** String getLocation\_X() {
51. **return** Location\_X;
52. }
53. **public** **void** setLocation\_X(String locationX) {
54. Location\_X = locationX;
55. }
56. **public** String getLocation\_Y() {
57. **return** Location\_Y;
58. }
59. **public** **void** setLocation\_Y(String locationY) {
60. Location\_Y = locationY;
61. }
62. **public** String getScale() {
63. **return** Scale;
64. }
65. **public** **void** setScale(String scale) {
66. Scale = scale;
67. }
68. **public** String getLabel() {
69. **return** Label;
70. }
71. **public** **void** setLabel(String label) {
72. Label = label;
73. }
74. **public** String getTitle() {
75. **return** Title;
76. }
77. **public** **void** setTitle(String title) {
78. Title = title;
79. }
80. **public** String getDescription() {
81. **return** Description;
82. }
83. **public** **void** setDescription(String description) {
84. Description = description;
85. }
86. **public** String getUrl() {
87. **return** Url;
88. }
89. **public** **void** setUrl(String url) {
90. Url = url;
91. }
92. **public** String getPicUrl() {
93. **return** PicUrl;
94. }
95. **public** **void** setPicUrl(String picUrl) {
96. PicUrl = picUrl;
97. }
98. **public** String getMediaId() {
99. **return** MediaId;
100. }
101. **public** **void** setMediaId(String mediaId) {
102. MediaId = mediaId;
103. }
104. **public** String getEventKey() {
105. **return** EventKey;
106. }
107. **public** **void** setEventKey(String eventKey) {
108. EventKey = eventKey;
109. }
110. **public** String getTicket() {
111. **return** Ticket;
112. }
113. **public** **void** setTicket(String ticket) {
114. Ticket = ticket;
115. }
116. **public** String getLatitude() {
117. **return** Latitude;
118. }
119. **public** **void** setLatitude(String latitude) {
120. Latitude = latitude;
121. }
122. **public** String getLongitude() {
123. **return** Longitude;
124. }
125. **public** **void** setLongitude(String longitude) {
126. Longitude = longitude;
127. }
128. **public** String getPrecision() {
129. **return** Precision;
130. }
131. **public** **void** setPrecision(String precision) {
132. Precision = precision;
133. }
134. **public** String getEvent() {
135. **return** Event;
136. }
137. **public** **void** setEvent(String event) {
138. Event = event;
139. }
140. **public** String getMsgId() {
141. **return** MsgId;
142. }
143. **public** **void** setMsgId(String msgId) {
144. MsgId = msgId;
145. }
146. **public** String getToUserName() {
147. **return** ToUserName;
148. }
149. **public** **void** setToUserName(String toUserName) {
150. ToUserName = toUserName;
151. }
152. **public** String getFromUserName() {
153. **return** FromUserName;
154. }
155. **public** **void** setFromUserName(String fromUserName) {
156. FromUserName = fromUserName;
157. }
158. **public** String getCreateTime() {
159. **return** CreateTime;
160. }
161. **public** **void** setCreateTime(String createTime) {
162. CreateTime = createTime;
163. }
164. **public** String getMsgType() {
165. **return** MsgType;
166. }
167. **public** **void** setMsgType(String msgType) {
168. MsgType = msgType;
169. }
170. }

**[java]** [view plaincopy](http://blog.csdn.net/pamchen/article/details/38718947)

1. **package** demo.process;
3. **import** java.lang.reflect.Field;
4. **import** java.lang.reflect.Method;
5. **import** java.util.Iterator;
6. **import** org.dom4j.Document;
7. **import** org.dom4j.DocumentHelper;
8. **import** org.dom4j.Element;
10. **import** demo.entity.ReceiveXmlEntity;
11. /\*\*
12. \* 解析接收到的微信xml，返回消息对象
13. \* @author pamchen-1
14. \*
15. \*/
16. **public** **class** ReceiveXmlProcess {
17. /\*\*
18. \* 解析微信xml消息
19. \* @param strXml
20. \* @return
21. \*/
22. **public** ReceiveXmlEntity getMsgEntity(String strXml){
23. ReceiveXmlEntity msg = **null**;
24. **try** {
25. **if** (strXml.length() <= 0 || strXml == **null**)
26. **return** **null**;
28. // 将字符串转化为XML文档对象
29. Document document = DocumentHelper.parseText(strXml);
30. // 获得文档的根节点
31. Element root = document.getRootElement();
32. // 遍历根节点下所有子节点
33. Iterator<?> iter = root.elementIterator();
35. // 遍历所有结点
36. msg = **new** ReceiveXmlEntity();
37. //利用反射机制，调用set方法
38. //获取该实体的元类型
39. Class<?> c = Class.forName("demo.entity.ReceiveXmlEntity");
40. msg = (ReceiveXmlEntity)c.newInstance();//创建这个实体的对象
42. **while**(iter.hasNext()){
43. Element ele = (Element)iter.next();
44. //获取set方法中的参数字段（实体类的属性）
45. Field field = c.getDeclaredField(ele.getName());
46. //获取set方法，field.getType())获取它的参数数据类型
47. Method method = c.getDeclaredMethod("set"+ele.getName(), field.getType());
48. //调用set方法
49. method.invoke(msg, ele.getText());
50. }
51. } **catch** (Exception e) {
52. // TODO: handle exception
53. System.out.println("xml 格式异常: "+ strXml);
54. e.printStackTrace();
55. }
56. **return** msg;
57. }
58. }

**2.3 调用图灵机器人api接口，获取智能回复内容：**

**[java]** [view plaincopy](http://blog.csdn.net/pamchen/article/details/38718947)

1. **package** demo.process;
3. **import** java.io.IOException;
4. **import** java.io.UnsupportedEncodingException;
5. **import** java.net.URLEncoder;
7. **import** org.apache.http.HttpResponse;
8. **import** org.apache.http.client.ClientProtocolException;
9. **import** org.apache.http.client.methods.HttpGet;
10. **import** org.apache.http.impl.client.HttpClients;
11. **import** org.apache.http.util.EntityUtils;
12. **import** org.json.JSONException;
13. **import** org.json.JSONObject;
15. /\*\*
16. \* 调用图灵机器人api接口，获取智能回复内容
17. \* @author pamchen-1
18. \*
19. \*/
20. **public** **class** TulingApiProcess {
21. /\*\*
22. \* 调用图灵机器人api接口，获取智能回复内容，解析获取自己所需结果
23. \* @param content
24. \* @return
25. \*/
26. **public** String getTulingResult(String content){
27. /\*\* 此处为图灵api接口，参数key需要自己去注册申请，先以11111111代替 \*/
28. String apiUrl = "http://www.tuling123.com/openapi/api?key=11111111&info=";
29. String param = "";
30. **try** {
31. param = apiUrl+URLEncoder.encode(content,"utf-8");
32. } **catch** (UnsupportedEncodingException e1) {
33. // TODO Auto-generated catch block
34. e1.printStackTrace();
35. } //将参数转为url编码
37. /\*\* 发送httpget请求 \*/
38. HttpGet request = **new** HttpGet(param);
39. String result = "";
40. **try** {
41. HttpResponse response = HttpClients.createDefault().execute(request);
42. **if**(response.getStatusLine().getStatusCode()==200){
43. result = EntityUtils.toString(response.getEntity());
44. }
45. } **catch** (ClientProtocolException e) {
46. e.printStackTrace();
47. } **catch** (IOException e) {
48. e.printStackTrace();
49. }
50. /\*\* 请求失败处理 \*/
51. **if**(**null**==result){
52. **return** "对不起，你说的话真是太高深了……";
53. }
55. **try** {
56. JSONObject json = **new** JSONObject(result);
57. //以code=100000为例，参考图灵机器人api文档
58. **if**(100000==json.getInt("code")){
59. result = json.getString("text");
60. }
61. } **catch** (JSONException e) {
62. // TODO Auto-generated catch block
63. e.printStackTrace();
64. }
65. **return** result;
66. }
67. }

**2.4 将结果封装为微信规定的xml格式，并返回给1.1中创建的servlet接口。**

**[java]** [view plaincopy](http://blog.csdn.net/pamchen/article/details/38718947)

1. **package** demo.process;
3. **import** java.util.Date;
4. /\*\*
5. \* 封装最终的xml格式结果
6. \* @author pamchen-1
7. \*
8. \*/
9. **public** **class** FormatXmlProcess {
10. /\*\*
11. \* 封装文字类的返回消息
12. \* @param to
13. \* @param from
14. \* @param content
15. \* @return
16. \*/
17. **public** String formatXmlAnswer(String to, String from, String content) {
18. StringBuffer sb = **new** StringBuffer();
19. Date date = **new** Date();
20. sb.append("<xml><ToUserName><![CDATA[");
21. sb.append(to);
22. sb.append("]]></ToUserName><FromUserName><![CDATA[");
23. sb.append(from);
24. sb.append("]]></FromUserName><CreateTime>");
25. sb.append(date.getTime());
26. sb.append("</CreateTime><MsgType><![CDATA[text]]></MsgType><Content><![CDATA[");
27. sb.append(content);
28. sb.append("]]></Content><FuncFlag>0</FuncFlag></xml>");
29. **return** sb.toString();
30. }
31. }

总结，以上便是微信公众平台开发的全部流程，整体来看并不复杂，要非常感谢**图灵机器人提供的api接口**，帮我们解决了智能回复这一高难度问题。其他类型的消息处理与示例中类似，有兴趣的开发者可以联系我进行交流学习，希望本文对大家有所帮助。