目錄

Wx-tools是基于微信公众平台API的轻量级框架。 基于Wx-tools你可以开速开发一个订阅号/服务号的web应用后台。

特性：

- 统一、简单的API，可以快速上手。
- 链式赋值编程，更加容易理解和使用。
- 对于微信服务器发来的消息，提供匹配器（Matcher），拦截器（interceptor），处理器（Handler）接口，定制实现，具有可扩展性。

源码：

- wx-tools-2.1.3-code.zip

文档:

- 开发文档 （文档写得比较详细，可以看看快速入门~）

实例教程

- CSDN实例教程

Jar下载：

- wx-tools-2.1.3.jar

讨论：

- BUG反馈及建议：https://github.com/antgan/wx-tools/issues
- 微信开发交流QQ群：570937047

博主有话说：

- 大四快毕业了，最近才把工作和实习定了下来。闲暇有空，结合了在几个项目的实践中，把遇到的问题和设计重新整理修复了一遍。受益匪浅。但是小弟能力不才，如果有BUG或者其他建议，希望能提出来~让wx-tools更加好用。喜欢的话欢迎star哦~无限感激！

1)wx-tools概述

Wx-tools是基于微信公众平台API的轻量级框架。 基于Wx-tools你可以开速开发一个订阅号/服务号的web应用后台。

wx-tools框架使用起来非常简单，关键词有**6**个。

- WxConfig 基本配置库
- WxService 微信公众平台统一API Service接口
- WxMessageRouter 消息路由器
- WxMessageMatcher (interface) 消息匹配器
- WxMessageInterceptor (interface) 消息拦截器
- WxMessageHandler (interface) 消息处理器

先有个大概了解，后续会详细讲到如何使用。

该框架还有个特点就是链式赋值，用过jQuery的人都知道这样的写法

```
$("#id").hide().attr().html()....
```

而wx-tools对于微信服务器发过来的消息是这样处理的。

```
//初始化统一API调用入口
WxService wxService = new WxService();
//来自微信服务器的消息
WxXmlMessage msg = XStreamTransformer.fromXml(WxXmlMessage.class
, request.getInputStream());
//实例化消息路由器，作用是将消息路由去匹配器，拦截器，处理器。
WxMessageRouter router = new WxMessageRouter(wxService);
//添加路由规则，只处理满足规则的消息，可以自定义匹配器，拦截器，处理。
//每条路由规则必须以next()或者end()结束。否则不生效。这个后续会讲到。
//这里意思是，只接收TEXT类型的消息，交给DemoMatcher匹配器、DemoIntercep
tor拦截器、DemoHandler处理器处理。
router.rule().msgType(WxConsts.XML_MSG_TEXT).matcher(new DemoMat
cher()).interceptor(new DemoInterceptor()).handler(new DemoHandl
er()).end();
//开始路由
router.route(wx);
```

是不是很简单？ 可定制的匹配器、拦截器、处理器，让你更加灵活的处理复杂的消息和业务。

下一篇会说明当前版本实现了哪些功能。

1.1)功能实现列表

> 更多功能 微信公众平台API官方文档 可以查看官方文档。

- 基本接口
  - 验证服务器地址的有效性
  - 获取access token
  - 获取jsapi_ticket
  - 获取微信服务器IP地址
- 菜单栏
  - 自定义菜单创建接口
  - 自定义菜单查询接口
  - 自定义菜单栏删除接口
  - 自定义菜单事件推送
  - 个性化菜单接口
  - 获取自定义菜单配置接口
- 消息管理
  - 微信服务器发来的消息通过消息路由器管理。
  - 群发接口
  - 模板消息
- OAuth2.0 认证
  - 网页授权获取用户基本信息
- 素材管理
  - 新增临时素材
  - 获取临时素材
  - 新增永久素材
  - 获取永久素材
  - 删除永久素材
  - 修改永久图文素材
  - 获取素材总数
  - 获取素材列表
- 用户管理
  - 用户分组管理

    - 设置用户备注名
    - 获取用户基本信息(UnionID机制)
    - 获取用户列表
    - 获取用户地理位置
- 账号管理
    - 生成带参数的二维码
    - 长链接转短链接接口
    - 微信认证事件推送
- JSSDK
    - JSSDK大部分支持

wx-tools暂时不支持小店的功能。

JS-SDK的签名相关接口是有的，所以支持部分JSSDK的api。详情参考微信官方文档。

不过，没实现也不怕，wx-tools内置了有关httpClient的方法可供开发者自行实现接口。

后续会讲到~

1.2)总体构成

wx-tools的功能组件分为WxMessageRouter，WxMessageMatcher，WxMessageInterceptor，WxMessageHandler四大组件。

这四大组件构成了对微信服务器发送过来的消息进行拓展性的处理。

其中，以下三个是接口，开发者可实现并构建自己的匹配器，拦截器，处理器。

- WxMessageMatcher
- WxMessageInterceptor
- WxMessageHandler

此外，还有几个类需要注意一下

- WxConsts 类
  - 封装所有微信公众平台API的常量类型，包括接口请求路径，事件等。
- WxConfig 类
  - 基本配置库。里面包含了AppId，AppSecret等信息。wx-tools已经提供了个基于内存管理的配置库。暂不支持自行拓展，如有需要持久化到数据库，需要自己实现。注意：配置库对于整个程序是单例的。
- WxService 类
  - 微信统一的API Service入口，继承IService接口，所有接口都从这里调用。
- WxErrorException 类
  - 微信异常
- WxErrorExceptionHandler 接口
  - 开发者可自行实现该接口，处理微信异常。

1.3)框架依赖

wx-tools依赖于以下几个jar包

- http
  - org.apache.httpcomponents -> httpclient
  - org.apache.httpcomponents -> httpmime
- JSON处理
  - org.codehaus.jackson -> jackson-mapper-asl
- XML处理
  - com.thoughtworks.xstream -> xstream
- IO
  - commons-io

建议用**Maven**构建项目。因为依赖的**jar**包中也会依赖其他包。

如果出现ClassNotFound等异常欢迎留言。

maven pom.xml

```
    <!-- 补全依赖 -->
    <!-- HttpClient -->
    <dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpclient</artifactId>
        <version>4.3.6</version>
    </dependency>
    <!-- http://mvnrepository.com/artifact/org.apache.httpcompon
ents/httpmime -->
    <dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpmime</artifactId>
        <version>4.3.6</version>
    </dependency>

    <!-- JSON -->
    <dependency>
        <groupId>org.codehaus.jackson</groupId>
        <artifactId>jackson-mapper-asl</artifactId>
        <version>1.9.13</version>
    </dependency>
    <!-- XML -->
    <!-- http://mvnrepository.com/artifact/com.thoughtworks.xstr
eam/xstream -->
    <dependency>
        <groupId>com.thoughtworks.xstream</groupId>
        <artifactId>xstream</artifactId>
        <version>1.4.7</version>
    </dependency>
    <!-- IO -->
    <!-- http://mvnrepository.com/artifact/commons-io/commons-io
 -->
    <dependency>
        <groupId>commons-io</groupId>
        <artifactId>commons-io</artifactId>
        <version>2.4</version>
    </dependency>
```

注意：不要与项目中的**jar**包重复哦，如果已经有了，就不用添加了~

2)快速开始

wx-tools只有一个jar包： **wx-tools-{version}.jar**

GitHub最新下载地址：wx-tools-2.0.0.jar

大概步骤：

1. 创建Web工程
2. 导入wx-tools-{version}.jar包
3. 编写wx.properties配置文件
4. 导入依赖包(可查看上一篇#框架依赖)
5. 接入微信公众平台，验证服务器地址的有效性
6. 实现自己的业务逻辑

简单吗？接下来写一个简单的Demo来讲解如何使用wx-tools。

Demo代码下载

2.1)创建Web工程

使用maven创建，或者在eclipse创建web项目。

maven创建项目指令

```
mvn archetype:generate -DgroupId=wxtools.demo -DartifactId=demo
-DarchetypeArtifactId=maven-archetype-webapp -DarchetypeCatalog=
local
```

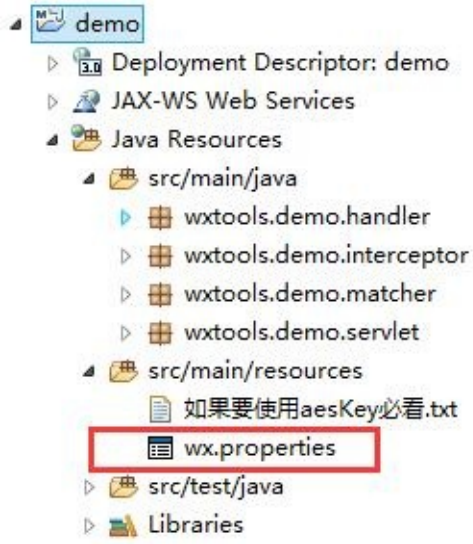
- 注意：此指令创建的web工程版本是2.3的，比较低。可以修改web.xml，变成3.0

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" x
mlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="htt
p://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee
/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>demo</display-name>
</web-app>
```

创建完毕后，导入**wx-tools-{version}.jar**

- 注意，导入后记得修改jar包的编码。window下默认读GBK，而框架本身是UTF-8。如果需要看源代码的务必手动修改编码。
- 修改方式：在eclipse的工程下，对着jar包右键 --> Properties --> Encoding --> UTF-8

建立基本的**package**，如图。



搭好项目基本框架后，在**src/main/resources**或者**src/main/java**下新建**wx.properties**，如上图

```
#配置如下
wx.appId=wxb1bff1627d37417b
wx.appSecret=dd037d9b9b4eea00fba14167a9f3c75d
wx.token=antgan
wx.aesKey=f82PVzQsKG5d8en3DwnZ8VVEoGInkmsWz3X3HsreEqe
wx.mchId=
```

不要填错了哦！注意大小写。

2.2)添加依赖

如果使用maven构建项目，可以直接添加如下坐标下载依赖jar。

maven pom.xml

```
    <!-- 补全依赖 -->
    <!-- 1：日志； java日志：slf4j，log4j，logback，common-logging
        slf4j接口/规范
        log4j,logback,common-logging，实现
        此处使用slf4j+logback -->
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>1.7.6</version>
    </dependency>

    <!-- 实现slf4j日志 -->
    <dependency>
        <groupId>ch.qos.logback</groupId>
        <artifactId>logback-classic</artifactId>
        <version>1.1.1</version>
    </dependency>

    <!-- HttpClient -->
    <dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpclient</artifactId>
        <version>4.3.6</version>
    </dependency>
    <!-- http://mvnrepository.com/artifact/org.apache.httpcompon
ents/httpmime -->
    <dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpmime</artifactId>
        <version>4.3.6</version>
    </dependency>
```

```
    <!-- JSON -->
    <dependency>
        <groupId>org.codehaus.jackson</groupId>
        <artifactId>jackson-mapper-asl</artifactId>
        <version>1.9.13</version>
    </dependency>
    <!-- XML -->
    <!-- http://mvnrepository.com/artifact/com.thoughtworks.xstr
eam/xstream -->
    <dependency>
        <groupId>com.thoughtworks.xstream</groupId>
        <artifactId>xstream</artifactId>
        <version>1.4.7</version>
    </dependency>
    <!-- IO -->
    <!-- http://mvnrepository.com/artifact/commons-io/commons-io
 -->
    <dependency>
        <groupId>commons-io</groupId>
        <artifactId>commons-io</artifactId>
        <version>2.4</version>
    </dependency>
```

如果不是maven构建的项目，可以自行下载相应的jar包，放在WEB-INF/lib文件夹下。

如果与项目中的jar包冲突或重复，只留一个就可以了~

2.3)验证服务器地址的有效性

这时候，wx-tools下的所有api都可以调用了。

我们验证一下服务器的有效性。官方文档

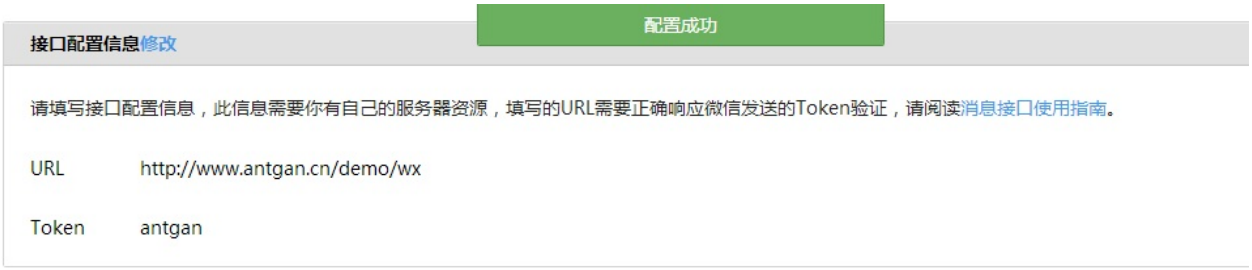
创建servlet，名为DemoServlet.java 当然你也可以使用主流的SpringMVC框架，一样的用法。【建议使用SpringMVC，Serlvet太繁琐了】

```
/**
 * <pre>
 * Demo Servlet
 *
 * 注意：WxConfig请调用getInstance()
 * 因为对于全局是唯一的。采用单例模式。
 * </pre>
 *
 * @author antgan
 * @date 2016/12/15
 *
 */
@WebServlet("/wx")
public class DemoServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    //实例化 统一业务API入口
    private IService iService = new WxService();

    protected void doGet(HttpServletRequest request, HttpServlet
Response response)
            throws ServletException, IOException {
        // 验证服务器的有效性
        PrintWriter out = response.getWriter();
        String signature = request.getParameter("signature");
        String timestamp = request.getParameter("timestamp");
        String nonce = request.getParameter("nonce");
        String echostr = request.getParameter("echostr");
        if (iService.checkSignature(signature, timestamp, nonce,
 echostr)) {
            out.print(echostr);
        }
    }

    protected void doPost(HttpServletRequest request, HttpServle
tResponse response)
            throws ServletException, IOException {
        //暂时省略，后面会讲到。
    }
}
```

然后去微信公众平台后台或者测试号后台填写资料验证即可。



2.4)接收微信服务器发来的消息

当你验证服务器有消息成功后，微信服务器就会把你的公众号任何事件和消息，以post请求推送到你验证的那个url地址上。

所以，把DemoServlet.java的doPost方法完善一下，并创建一个简单的WxMessageHandler。

- 创建类**DemoHandler.java** 实现 **WxMessageHandler**接口

```
/**
 * 示例：DemoHandler
 * 目的：返回用户 “恭喜你猜对了”
 * @author antgan
 * @date 2016/12/15
 *
 */
public class DemoHandler implements WxMessageHandler{

    @Override
    public WxXmlOutMessage handle(WxXmlMessage wxMessage, Map<St
ring, Object> context, IService iService)
            throws WxErrorException {
        //必须以build()作为结尾，否则不生效。
        WxXmlOutMessage xmlOutMsg = WxXmlOutMessage.TEXT().conte
nt("恭喜你猜对了").toUser(wxMessage.getFromUserName()).fromUser(wx
Message.getToUserName()).build();
        return xmlOutMsg;
    }

}
```

- 完善doPost方法，这里示例是判断了消息的加密模式，如果是明文模式，可以不用加解密消息。

```
protected void doPost(HttpServletRequest request, HttpServletRes
```

```
ponse response) throws ServletException, IOException {
       request.setCharacterEncoding("UTF-8");
        response.setCharacterEncoding("UTF-8");
        // 返回消息给微信服务器
        PrintWriter out = response.getWriter();
        // 获取encrypt_type 消息加解密方式标识
        String encrypt_type = request.getParameter("encrypt_type
");
        // 创建一个路由器
        WxMessageRouter router = new WxMessageRouter(iService);
        try {
            // 判断消息加解密方式，如果是加密模式。encrypt_type==aes
            if (encrypt_type != null && "aes".equals(encrypt_typ
e)) {
//                String signature = request.getParameter("signa
ture");
                String timestamp = request.getParameter("timesta
mp");
                String nonce = request.getParameter("nonce");
                String msg_signature = request.getParameter("msg
_signature");

                // 微信服务器推送过来的加密消息是XML格式。使用WxXmlMess
age中的decryptMsg()解密得到明文。
                WxXmlMessage wx = WxXmlMessage.decryptMsg(reques
t.getInputStream(), WxConfig.getInstance(), timestamp,
                        nonce, msg_signature);
                System.out.println("消息：\n " + wx.toString());
                // 添加规则；这里的规则是指所有消息都交给交给DemoHandle
r处理
                // 注意！！每一个规则，必须由end()或者next()结束。不然
不会生效。
                // end()是指消息进入该规则后不再进入其他规则。 而next()
是指消息进入了一个规则后，如果满足其他规则也能进入，处理。
                router.rule().handler(new DemoHandler()).end();
                // 把消息传递给路由器进行处理，得到最后一个handler处理的
结果
                WxXmlOutMessage xmlOutMsg = router.route(wx);
                if (xmlOutMsg != null) {
                    // 将要返回的消息加密，返回
```

```
                    out.print(WxXmlOutMessage.encryptMsg(WxConfi
g.getInstance(), xmlOutMsg.toXml(), timestamp, nonce));// 返回给
用户。
                }
            //如果是明文模式，执行以下语句
            } else {
                // 微信服务器推送过来的是XML格式。
                WxXmlMessage wx = XStreamTransformer.fromXml(WxX
mlMessage.class, request.getInputStream());
                System.out.println("消息：\n " + wx.toString());
                // 添加规则；这里的规则是指所有消息都交给DemoHandler处
理
                // 注意！！每一个规则，必须由end()或者next()结束。不然
不会生效。
                // end()是指消息进入该规则后不再进入其他规则。 而next()
是指消息进入了一个规则后，如果满足其他规则也能进入，处理。
                router.rule().handler(new DemoHandler()).end();
                // 把消息传递给路由器进行处理
                WxXmlOutMessage xmlOutMsg = router.route(wx);
                if (xmlOutMsg != null)
                    out.print(xmlOutMsg.toXml());// 因为是明文，所
以不用加密，直接返回给用户。
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            out.close();
        }
```

这样子写的效果就是：公众号后台接收所有类型的消息，都交给DemoHandler处理器处理，最后返回了一句“恭喜你猜对了”给用户。

是不是很简单？

看到这里你已经入门了。

接下来加入匹配器Matcher和Interceptor拦截器来处理更加复杂的业务。

2.5)WxMessageMatcher接口实现

WxMessageMatcher匹配器接口，可以自定义更加复杂的匹配逻辑，如格式验证。

- 假设业务场景：
  - 有一个活动，让用户猜一个英文单词。这里只是举个例子。答案是Matcher。答对的用户收到：恭喜你答对了。
- 创建**DemoMatcher.java** 实现 **WxMessageMatcher**接口

```
/**
* Demo 简单的匹配器，可以用于更加复杂的消息验证操作
* @author antgan
*
*/
public class DemoMatcher implements WxMessageMatcher{
  //答案是Matcher，如果匹配Matcher返回true；反之，false。
  public boolean match(WxXmlMessage message) {
      if(message.getContent().equals("Matcher")){
        return true;
      }
      return false;
  }
}
```

- 修改DemoSerlvet里的doPost方法中的路由器规则，添加DemoMatcher

```
router.rule().matcher(new DemoMatcher()).handler(new DemoMessage
Handler()).end();
```

接下来就交给wx-tools帮你匹配啦~

2.6)WxMessageInterceptor接口实现

WxMessageInterceptor拦截器接口，可以处理更加复杂的验证。例如身份验证。

- 业务场景：在上一篇的猜单词活动中，增加一个规则。
  - 只有用户关注公众号时长大于3天才能参与活动。
- 创建DemoInterceptor.java 实现 WxMessageInterceptor接口

```
/**
 * Demo 拦截器，可以通过WxService做更加复杂的拦截，例如身份验证，权限验证等操作。
 * @author antgan
 *
 */
public class DemoInterceptor implements WxMessageInterceptor{
    public boolean intercept(WxXmlMessage wxMessage, Map<String, Object> context, IService wxService)
            throws WxErrorException {
        //可以使用wxService的微信API方法
        //可以在Handler和Interceptor传递消息，使用context上下文
        //可以实现自己的业务逻辑

        //这里就不编写验证关注三天以上的用户了
        if(/*用户关注时长大于3天*/){
            return true;
        }
        return false;
    }
}
```

- 修改
- DemoSerlvet里的doPost方法中的路由器规则，添加DemoInterceptor

```
router.rule().matcher(new DemoMatcher()).interceptor(new DemoInt
erceptor()).handler(new DemoMessageHandler()).end();
```

搞定。接下来交给wx-tools去做吧~ 简单吧！

2.7)关于WxMessageRouter的详解

WxMessageRouter消息路由器，到底是个什么东东呢？接下来详细讲解一下。

提到这个路由器，就要说说另一个东西：WxMessageRouterRule。简称规则(Rule)。

定义规则，用于对来自微信服务器的消息进行过滤和筛选，只针对有效消息进行处理，提高服务器处理效率。

通过链式配置路由规则(Rule)，根据规则把来自微信的消息交给handler处理。

说明：

1. 配置路由规则时尽量按照从细到粗的原则，否则可能消息可能会被提前处理
2. 默认情况下消息只会被处理一次，除非使用 {WxMessageRouterRule的next()方法}
3. 规则的结束必须用{WxMessageRouterRule的end()方法}或者{WxMessageRouterRule的next()方法}，否则不会生效。

使用方法：

```
//初始化一个路由器，把wxService传入。
WxMessageRouter router = new WxMessageRouter(wxService);
//新建路由规则，通过rule()方法创建新的规则，然后链式填写过滤条件。MSG_TYPE等参数填入WxConst中的常量，这里不作展示，可以查看WxConst代码或官方文档，有注释。
router.rule().msgType("MSG_TYPE").event("EVENT").eventKey("EVENT_KEY").content("CONTENT").matcher(matcher).interceptor(interceptor, ...).handler(handler, ...).end().rule().msgType("MSG_TYPE")...//另外一个匹配规则.end();

// 将WxXmlMessage交给消息路由器，处理后得到结果。
WxXmlOutMessage xmlOutMsg = router.route(wxXmlMessage);
```

关于路由规则条件

1. 对于一条消息(WxXMLMessage)允许多个规则(Rule)去进行过滤和处理。用**next()**方法去连接两个规则。但是最后必须是以**end()**方法结束。
2. 每条规则可以允许多个拦截器(Interceptor)，多个处理器(Handler)处理。返回最后一个**Handler**处理的结果。
3. 路由规则还提供正则表达式过滤，对于简单的过滤需求，如只接受数字消息。不想繁琐的建立Matcher匹配器。可以如下写法。

```
//正则表达式：^[0-9]*$只接受数字消息，其他消息过滤。
router.rule().rContent("^[0-9]*$").handler(new DemoHandler()
).end();
```

去除多余消息，高效处理争对性消息，眞是好用又简单。

3)关于开发与API的拓展

1. 面对wx-tools，无从下手怎么办？

在开发的过程中，对于一个陌生的框架，肯定会无从下手。 wx-tools提供源代码查看。

其实wx-tools架构很简单，无需想复杂了。

全部微信公众号的API接口都在**IService**接口里，默认实现类为**WxService**。

知道这点，然后明白有**Matcher**，**Interceptor**，**Handler**，路由**Router**，规则**Rule**就可以了。非常简单易用。

2. 关于API拓展

参考3.1)开发与API拓展。

3.1) 开发时如何查看API接口

1.如何查看已经实现的API接口？

wx-tools开放源代码查看，需要开发什么功能。可以参考【1.1中功能实现列表】或直接按Ctrl直接查看**IService**接口。

如：现在我要实现**OAuth2.0**用户认证的功能。我该如何使用这个框架呢？

- 按Ctrl查看IService接口源代码，搜索关键字oauth，找到如下方法
  - **oauth2buildAuthorizationUrl** 获取OAuth2.0认证URL
  - **oauth2ToGetAccessToken** 通过code获取AccessToken
  - **oauth2ToGetRefreshAccessToken** 强制刷新AccessToken
  - **oauth2ToGetUserInfo** 获取用户信息
  - **oauth2CheckAccessToken** 检查AccessToken是否有效

查看源代码，有详细的注解帮助你快速理解各个接口或方法，直接调用使用即可。

2. 如果有些接口未实现怎么办？例如微信小店等接口。

wx-tools的API拓展，小弟能力有限，暂时想不到更好的设计模式去设计WxService。 所以拓展只能自己新建的类，继承WxService，然后添加新的API方法。

- 注意：新建类的构造方法要加上**super()**，因为在WxService中httpClient还未初始化，在WxService的无参构造方法中才初始化。源代码如下：

```
/**
 * 构造方法，初始化httpClient
 */
public WxService() {
    httpClient = HttpClients.createDefault();
}
```

新建的类，假设它叫NewService.java

```
/**
 * NewService的构造方法
 */
public NewService() {
    super();//用于初始化httpClient
}
```

3.2)博主有话说

如果你看到了这里，非常感谢你支持我，给我star给我力量。

大学的尾巴悄悄来临，我也顺利找到了实习和工作。闲暇期间，想起wx-tools的问题还没解决。（舍友写项目用了我的wx-tools出现了线程爆炸，难以使用，文档不全等现象。但忙于找工作就放着wx-tools不管了）

终于有时间了，静下心来。看看别人优秀的源代码。心想着我的wx-tools就应该是简单易用，轻量级可拓展。于是，wx-tools-2.0.0.jar又诞生了。

相比之前的wx-tools。这个版本比较大的变化如下：

- 更简化的配置(wx.properties文件来配置)。
- 更加简单简约的使用方法（链式编程！酷炫！）
- 去掉了线程池。（就是这玩意导致舍友项目爆炸了）
- 去掉了鸡肋的重复消息检查器（**Checker**）。（当初为什么要加检查重复消息呢？黑人问号？？？）
- **jssdk**的支持。（不支持微信小店）

哎！其实对比了几个微信公众平台开发框架，wx-tools走的路线完全不同。不知道大家喜不喜欢呢~ 喜欢希望你给我star咯~wx-tools Github戳这里！！给星星！爱你么么哒！

我也创了个QQ群（570937047），欢迎大家加入，讨论JAVA开发或者对酒当歌。

最后！欢迎大家提BUG或者pull request！小弟能力不足！就这样了！

4)API的用法及写法（使用示例）

在本章，针对每一个API接口的调用将有详细的代码展示。

接口列表：

- createMenu - 创建菜单栏
- deleteMenu - 删除菜单栏
- getMenu - 获取菜单栏
- getMenuCurInfo - 获取当前菜单栏配置信息
- createUserGroup - 创建用户分组
- queryAllUserGroup - 查询所有用户分组
- queryGroupIdByOpenId - 查询某用户在哪个分组
- updateUserGroupName - 更新用户分组名字
- movingUserToNewGroup - 移动某用户到某分组
- batchMovingUserToNewGroup - 批量移动用户到某分组
- deleteUserGroup - 删除用户分组
- updateUserRemark - 更新用户备注
- getUserInfoByOpenId - 获取用户信息基本信息
- batchGetUserInfo - 批量查询用户信息
- batchGetUserOpenId - 批量查询关注者openid
- oauth2buildAuthorizationUrl - Oauth2.0 认证
- oauth2ToGetAccessToken- Oauth2.0 认证
- oauth2ToGetUserInfo- Oauth2.0 认证
- uploadTempMedia - 上传临时资源
- downloadTempMedia - 下载临时资源
- uploadMedia - 上传永久资源
- downloadMedia - 下载永久资源
- downloadNewsMedia - 下载图文资源
- downloadVideoMedia - 下载视频资源
- deleteMediaMaterial - 删除永久资源
- imageDomainChange - 上传图片变成腾讯域名下的图片
- getMaterialCount - 获取永久资源个数
- batchGetMeterial - 批量获取永久资源

- sendAllByGroup - 群发-通过组
- sendAllByOpenid - 群发-通过openid
- sendAllPreview - 群发预览
- sendAllDelete - 删除群发
- sendAllGetStatus - 查询群发状态
- templateSetIndustry - 设置模板行业
- templateGetIndustry - 查询模板行业
- templateGetId - 获取模板ID
- templateGetList - 获取模板列表
- templateDelete - 删除模板
- templateSend - 发送模板
- addNewsMedia - 新增图文
- updateNewsInfo - 修改图文
- createJsapiConfig - 获取jssdk config
- unifiedOrder【未单元测试】 - 统一下单
- createQrCode - 生成二维码
- downloadQrCode - 下载二维码
- getShortUrl - 获取短链接
- getCallbackIp - 获取微信服务器ip

4.1)菜单栏相关API

> 包括如下接口：
>
> - createMenu - 创建菜单栏
> - deleteMenu - 删除菜单栏
> - getMenu - 获取菜单栏
> - getMenuCurInfo - 获取当前菜单栏配置信息

1. createMenu 创建菜单栏

```
        WxMenu menu = new WxMenu();
        List<WxMenuButton> btnList = new ArrayList<>();

        //设置CLICK类型的按钮1
        WxMenuButton btn1 = new WxMenuButton();
        btn1.setType(WxConsts.BUTTON_CLICK);
        btn1.setKey("btn1_key");
        btn1.setName("CLICK按钮1");

        //设置VIEW类型的按钮2
        WxMenuButton btn2 = new WxMenuButton();
        btn2.setType(WxConsts.BUTTON_VIEW);
        btn2.setUrl("http://www.baidu.com");
        btn2.setName("VIEW按钮2");

        //设置含有子按钮的按钮3
        List<WxMenuButton> subList = new ArrayList<>();
        //子按钮
        WxMenuButton btn3_1 = new WxMenuButton();
        btnbtn3_1 .setType(WxConsts.BUTTON_VIEW);
        btnbtn3_1 .setUrl("http://www.baidu.com");
        btnbtn3_1 .setName("子按钮3_1");
        WxMenuButton btn3_2 = new WxMenuButton();
        btn3_2.setType(WxConsts.BUTTON_VIEW);
        btn3_2.setUrl("http://www.baidu.com");
```

```
        btn3_2.setName("子按钮3_2");
        subList.add(btn3_1);
        subList.add(btn3_2);
        //把子按钮列表设置进按钮3
        WxMenuButton btn3 = new WxMenuButton();
        btn3.setName("子按钮3");
        btn3.setSub_button(subList);

        //将三个按钮设置进btnList
        btnList.add(btn1);
        btnList.add(btn2);
        btnList.add(btn3);
        //设置进菜单类
        menu.setButton(btnList);
        //调用API即可
        try {
            //参数1--menu   ，参数2--是否是个性化定制。如果是个性化菜单栏
，需要设置MenuRule
            iService.createMenu(menu, false);
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        //个性化菜单栏
        WxMenuRule rule = new WxMenuRule();
        rule.setProvince("广东");
        menu.setMatchrule(rule);

        try {
            //参数1--menu   ，参数2--是否是个性化定制。如果是个性化菜单栏
，需要设置MenuRule
            iService.createMenu(menu, true);
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

2. deleteMenu 删除菜单栏

```
try {
    iService.deleteMenu();
} catch (WxErrorException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

//如果是删除个性化菜单栏，需要传入MenuID
try {
    iService.deleteMenu("MenuID");
} catch (WxErrorException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

---

3. getMenu 获取菜单栏

```
try {
    WxMenuResult result = iService.getMenu();
    System.out.println(result.toString());
} catch (WxErrorException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

---

4. getMenuCurInfo 获取现有菜单栏配置

```
        try {
            WxCurMenuInfoResult result = iService.getMenuCurInfo
();
            System.out.println(result.toString());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

4.2)用户相关API

> 包括如下接口：
>
> - createUserGroup - 创建用户分组
> - queryAllUserGroup - 查询所有用户分组
> - queryGroupIdByOpenId - 查询某用户在哪个分组
> - updateUserGroupName - 更新用户分组名字
> - movingUserToNewGroup - 移动某用户到某分组
> - batchMovingUserToNewGroup - 批量移动用户到某分组
> - deleteUserGroup - 删除用户分组
> - updateUserRemark - 更新用户备注
> - getUserInfoByOpenId - 获取用户信息基本信息
> - batchGetUserInfo - 批量查询用户信息
> - batchGetUserOpenId - 批量查询关注者openid
> - oauth2buildAuthorizationUrl - Oauth2.0 认证
> - oauth2ToGetAccessToken- Oauth2.0 认证
> - oauth2ToGetUserInfo- Oauth2.0 认证

1. createUserGroup 创建用户分组

```
        try {
            WxUserGroupResult result = iService.createUserGroup(
"组名");
            System.out.println(result.getGroup().getId());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

2. queryAllUserGroup 查询用户分组

```
        try {
            WxUserGroupResult result = iService.queryAllUserGrou
p();
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

3. queryGroupIdByOpenId 查询用户在哪个分组

```
        try {
            int groupId = iService.queryGroupIdByOpenId("openid"
);
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

4. updateUserGroupName 修改分组名

```
        try {
            //组ID，新组名
            iService.updateUserGroupName(1, "new group name");
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

5. movingUserToNewGroup 移动用户在某组

```
        try {
            WxError result = iService.movingUserToNewGroup("open
id", 1);
            System.out.println(result.getErrcode());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

6. batchMovingUserToNewGroup 批量移动用户在某组

```
        List<String> openidList = new ArrayList<>();
        openidList.add("openid1");
        openidList.add("openid2");

        try {
            iService.batchMovingUserToNewGroup(openidList, 2);
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

7. deleteUserGroup 删除用户分组

```
        try {
            iService.deleteUserGroup(2);
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

8. updateUserRemark 修改用户备注

```
        try {
            iService.updateUserRemark("openid", "备注名");
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

9. getUserInfoByOpenId 获取用户基本信息

```
        try {
            WxUser user = iService.getUserInfoByOpenId(new WxUse
rGet("openid", WxConsts.LANG_CHINA));
            System.out.println(user.toString());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

10. batchGetUserInfo 批量获取用户信息

```
        List<WxUserGet> list = new ArrayList<>();
        WxUserGet userGet1 = new WxUserGet("openid", WxConsts.LA
NG_CHINA);
        WxUserGet userGet2 = new WxUserGet("openid", WxConsts.LA
NG_CHINA);
        list.add(userGet1);
        list.add(userGet2);
        try {
            WxUserList userList = iService.batchGetUserInfo(list
);
            System.out.println(userList.toString());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

11. batchGetUserOpenId 批量获取关注者openid

```
        try {
            //第一个openid之后拉取
            WxUserListResult result = iService.batchGetUserOpenI
d("next openid");
            System.out.println(result.getNext_openid());
            System.out.println(result.getData());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

12. oauth2buildAuthorizationUrl Oauth2.0 认证获取用户信息--第一步：构造URL获取Code

```
        try {
            String oauthUrl = iService.oauth2buildAuthorizationU
rl("回调URL",WxConsts.OAUTH2_SCOPE_USER_INFO, "自定义携带参数");

        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

13. oauth2ToGetAccessToken Oauth2.0 认证获取用户信息--第二步：拿code换token和openid

```
        try {
            WxOAuth2AccessTokenResult result = iService.oauth2To
GetAccessToken("code");
            System.out.println(result.getAccess_token());
            System.out.println(result.getOpenid());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

14. oauth2ToGetUserInfo Oauth2.0 认证获取用户信息--第三步：拿token换用户信息

```
        try {
            WxUser user = iService.oauth2ToGetUserInfo("token",
new WxUserGet("openid", WxConsts.LANG_CHINA));
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

4.3)资源上传相关API

> 包括如下接口：
>
> - uploadTempMedia - 上传临时资源
> - downloadTempMedia - 下载临时资源
> - uploadMedia - 上传永久资源
> - downloadMedia - 下载永久资源
> - downloadNewsMedia - 下载图文资源
> - downloadVideoMedia - 下载视频资源
> - deleteMediaMaterial - 删除永久资源
> - imageDomainChange - 上传图片变成腾讯域名下的图片
> - getMaterialCount - 获取永久资源个数
> - batchGetMeterial - 批量获取永久资源

1. uploadTempMedia 上传临时文件到微信服务器

```
        //可以上传file或者InputSteam，拿到MediaID
        try {
            WxMediaUploadResult result = iService.uploadTempMedi
a(WxConsts.MEDIA_IMAGE,new File("E://test.jpg"));
            System.out.println(result.getMedia_id());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

2. downloadTempMedia 下载临时文件，存在**E://temp**文件夹

```
        try {
            File file = iService.downloadTempMedia("media_id",ne
w File("E://temp"));
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

3. uploadMedia 上传永久文件到微信服务器。可以传**File**或者输入流

```
        try {
            //这里注意，如果是上传非视频格式的素材，第三个参数(WxVideoIn
troduction)为null即可
            WxMediaUploadResult result1 = iService.uploadMedia(W
xConsts.MEDIA_VOICE, new File("E://test.m4a"), null);

            //如果是上传视频Video，可以添加描述
            WxVideoIntroduction intro = new WxVideoIntroduction(
);
            intro.setTitle("视频1");
            intro.setIntroduction("描述1");
            WxMediaUploadResult result2 = iService.uploadMedia(W
xConsts.MEDIA_VIDEO, new File("E://test.mp4"), intro);

        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

4. downloadMedia 下载永久文件，与临时文件一样用法（注意：图文和视频需要使用另外的方法）

```
        try {
            File file = iService.downloadMedia("media_id",new Fi
le("E://temp"));
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

5. downloadNewsMedia 下载图文素材

```
        try {
            //图文结果
            WxNewsMediaResult result = iService.downloadNewsMedi
a("media_id");
            System.out.println(result.toString());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

6. downloadVideoMedia 下载视频素材

```
        try {
            //视频结果，取出URL即可下载
            WxVideoMediaResult result = iService.downloadVideoMe
dia("media_id", new File("E://temp"));
            System.out.println(result.toString());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

7. deleteMediaMaterial 删除素材资源

```
        try {
            WxError result = iService.deleteMediaMaterial("media
_id");
            System.out.println(result.getErrcode());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

8. imageDomainChange 上传图片变成腾讯域名下的图片

```
        try {
            WxMediaUploadResult result = iService.imageDomainCha
nge(new File("E://test.jpg"));
            System.out.println(result.getUrl());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

9. getMaterialCount 获取永久素材数量接口

```
        try {
            WxMaterialCountResult result = iService.getMaterialC
ount();
            System.out.println(result.getImage_count());
            System.out.println(result.getNews_count());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

10. batchGetMeterial 批量获取永久素材资源信息

```
        try {
            WxBatchGetMaterialResult result = iService.batchGetM
eterial(WxConsts.MEDIA_IMAGE, 0, 5);
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

4.4)消息相关API

> 包括如下接口：
>
> - sendAllByGroup - 群发-通过组
> - sendAllByOpenid - 群发-通过openid
> - sendAllPreview - 群发预览
> - sendAllDelete - 删除群发
> - sendAllGetStatus - 查询群发状态
> - templateSetIndustry - 设置模板行业
> - templateGetIndustry - 查询模板行业
> - templateGetId - 获取模板ID
> - templateGetList - 获取模板列表
> - templateDelete - 删除模板
> - templateSend - 发送模板

1. sendAllByGroup 通过用户组来群发

```
        WxGroupSender sender = new WxGroupSender();
        //设置哪些组需要接受群发
        sender.setFilter(new SenderFilter(true, 1));
        //群发文本内容
        sender.setText(new Text("文本内容"));
        //群发图片，以此类推
        sender.setImage(new Media("media_id"));
        try {
            SenderResult result = iService.sendAllByGroup(new Wx
GroupSender());
            System.out.println(result.toString());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

2. sendAllByOpenid 针对某群人的openid群发

```
        WxOpenidSender sender = new WxOpenidSender();
        List<String> openidList = new ArrayList<>();
        openidList.add("openid1");
        openidList.add("openid2");
        sender.setTouser(openidList);
        //群发文本内容
        sender.setText(new Text("文本内容"));
        //群发图片，以此类推
        sender.setImage(new Media("media_id"));
        try {
            SenderResult result = iService.sendAllByOpenid(sende
r);
            System.out.println(result.toString());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

3. sendAllPreview 群发预览

```
        PreviewSender sender = new PreviewSender();
        //设置openid或者微信号，优先级为wxname高
        sender.setTouser("openid");
        sender.setTowxname("微信号");
        //群发文本内容
        sender.setText(new Text("文本内容"));
        //群发图片，以此类推
        sender.setImage(new Media("media_id"));
        try {
            SenderResult result = iService.sendAllPreview(sender
);
            System.out.println(result.toString());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

4. sendAllDelete 删除群发

```
        try {
            //此参数在发送接口 返回
            SenderResult result = iService.sendAllDelete("msg_id
");
            System.out.println(result.toString());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

5. sendAllGetStatus 获取群发状态

```
        try {
            SenderResult result = iService.sendAllGetStatus("msg
_id");
            System.out.println(result.toString());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

6. templateSetIndustry 设置模板消息的行业

```
        //行业代码参考官方文档。
        try {
            iService.templateSetIndustry("1", "4");
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

7. templateGetIndustry获取模板消息的行业设置

```
        try {
            IndustryResult result = iService.templateGetIndustry
();
            System.out.println(result.getPrimary_industry());
            System.out.println(result.getSecondary_industry());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

8. templateGetId通过短ID获取模板ID

```
        //模板库中模板的编号，有“TM**”和“OPENTMTM**”等形式
        try {
            TemplateResult result = iService.templateGetId("temp
late_id_short");
            System.out.println(result.toString());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

9. templateGetList 获取模板列表

```
        try {
            TemplateListResult result = iService.templateGetList
();
            System.out.println(result.toString());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

10. templateDelete 删除模板

```
        try {
            iService.templateDelete("template_id");
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

11. templateSend 模板消息发送

```
        TemplateSender sender = new TemplateSender();
        sender.setTouser("openid");
        sender.setTemplate_id("templateId");
        sender.setData("Object：与模板内容对应的对象");
        sender.setUrl("url");
        try {
            TemplateSenderResult result = iService.templateSend(
sender);
            System.out.println(result.toString());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

4.5)图文相关API

> 包括如下接口：
>
> - addNewsMedia - 新增图文
> - updateNewsInfo - 修改图文

1. addNewsMedia 添加图文

```
        WxNewsInfo news1 = new WxNewsInfo();
        news1.setTitle("标题1");
        news1.setThumb_media_id("图片media_id");
        news1.setContent("xxx");
        //....设置图文内容
        WxNewsInfo news2 = new WxNewsInfo();
        news2.setTitle("标题1");
        news2.setThumb_media_id("图片media_id");
        news2.setContent("xxx");

        List<WxNewsInfo> newsList = new ArrayList<>();
        newsList.add(news1);
        newsList.add(news2);

        try {
            String mediaId = iService.addNewsMedia(newsList);
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

2. updateNewsInfo 修改图文内容

```
        WxNewsInfo news1 = new WxNewsInfo();
        news1.setTitle("标题1");
        news1.setThumb_media_id("图片media_id");
        news1.setContent("xxx");
        try {
            //参数1：media_id；参数2：图文位置下标。从0开始；参数3：新的
图文对象
            WxError result = iService.updateNewsInfo("media_id",
 0, news1);
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

4.6)JS SDK相关API

> 包括如下接口：
>
> - createJsapiConfig - 获取jssdk config

1. createJsapiConfig 获取JSSDK 中config的配置

```
        List<String> jsApiList = new ArrayList<>();
        //需要用到哪些JS SDK API 就设置哪些
        jsApiList.add("chooseImage");//拍照或从手机相册中选图接口
        jsApiList.add("onMenuShareQZone");//获取“分享到QQ空间”按钮点击状态及自定义分享内容接口

        try {
            //把config返回到前端进行js调用即可。
            WxJsapiConfig config = iService.createJsapiConfig("调用jssdk的完整url", jsApiList);
            System.out.println(config.toString());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

4.7)支付相关API

> 包括如下接口：
>
> - unifiedOrder【未单元测试】- 统一下单

1. unifiedOrder 统一下单接口【未测试】

```
        //下订单的商品价格相关信息
        PayOrderInfo order = new PayOrderInfo();
        order.setOrderId("订单ID");
        order.setOrderName("商品名");
        order.setDetail("商品详情");
        order.setTotalFee("100");//一块钱，单位为分

        try {
            InvokePay invokePay = iService.unifiedOrder(order, "
支付后回调页面URL", "待支付的用户openid");
            //只要把invokePay返回到前端调用JSSDK中的chooseWXPay方法即
可
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

4.8)另外一些API调用示例，如二维码生产，短链接等

> 包括如下接口：
>
> - createQrCode - 生成二维码
> - downloadQrCode - 下载二维码
> - getShortUrl - 获取短链接
> - getCallbackIp - 获取微信服务器ip

1. createQrCode 生成二维码

```
        WxQrcode code = new WxQrcode();
        code.setAction_name("actionName");
        code.setAction_info(new WxQrActionInfo(new WxScene("scen
e_id/str")));
        code.setExpire_seconds(720);
        try {
            QrCodeResult result = iService.createQrCode(code);
            System.out.println(result.getUrl());
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

2. downloadQrCode 下载二维码，需要用到**createQrCode**中的**ticket**

```
        try {
            File file = iService.downloadQrCode(new File("E://te
mp"), "ticket");
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

3. getShortUrl 长链接变短链接

```
        try {
            String shortUrl = iService.getShortUrl("long_url");
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

---

4. getCallbackIp获取微信服务器的ip段

```
        try {
            String [] ipList = iService.getCallbackIp();
        } catch (WxErrorException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```