第3章

微信公众平台应用开发入门

本章主要介绍微信公众平台应用开发时如何实现对 access_token 的获取、如何实现网络检测、如何实现对 IP 地址的获取和如何验证消息是否来自微信服务器。


视频讲解

3.1 获取 access_token

3.1.1 说明

获取 access_token 的接口 URL 为 https://api.weixin.qq.com/cgi-bin/token?grant_type=client_credential&appid=APPID&secret=APPSECRET。其中，必需的参数 grant_type 取值为 client_credential。代表用户的唯一凭证参数 appid 也是必需的参数，要用具体值替换占位的 APPID。代表用户密钥的 secret(即 appsecret)也是必需的参数，也要用具体值替换占位的 APPSECRET。在接口 URL 中，“?”后面的字符串中，以“&”为分隔符，分成若干个等式。每个等式中前面小写字符串(如 appid)代表参数，而每个等式中后面的小写字符串代表参数值(如 client_credential)，等式中后面的全部大写字符串(如 APPID)起占位作用，调用接口时要用实际的参数值来代替它。后面章节的接口也遵守此约定。

3.1.2 创建项目并修改文件 pom.xml

按照 2.2 节的方法创建项目 wxgzptkfbook，修改文件 pom.xml，文件 pom.xml 修改后的代码如例 3-1 所示。修改文件 pom.xml 主要是增加项目依赖的代码。

【例 3-1】 修改后的文件 pom.xml 代码示例。

```
<?xml version = "1.0" encoding = "UTF - 8"?>
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/
xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.5.2</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>edu.bookcode</groupId>
    <artifactId>wxgzptkfbook</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>wxgzptkfbook</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>11</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <!--XML处理类中的添加-->
        <dependency>
            <groupId>net.sf.json-lib</groupId>
            <artifactId>json-lib</artifactId>
            <version>0.9</version>
            <!--高版本需要JDK13或者JDK15-->
        </dependency>
        <dependency>
            <groupId>com.thoughtworks.xstream</groupId>
            <artifactId>xstream</artifactId>
            <version>1.4.14</version>
```

```
</dependency>
<dependency>
    <groupId>org.dom4j</groupId>
    <artifactId>dom4j</artifactId>
    <version>2.1.1</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>druid-spring-boot-starter</artifactId>
    <version>1.1.14</version>
</dependency>
<dependency>
    <groupId>cn.hutool</groupId>
    <artifactId>hutool-all</artifactId>
    <version>4.5.11</version>
</dependency>
 <!-- SpringBoot 集成 thymeleaf 模板 -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.58</version>
</dependency>
<dependency>
    <groupId>com.squareup.okhttp3</groupId>
    <artifactId>okhttp</artifactId>
    <version>3.13.1</version>
</dependency>
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>2.9.0</version>
</dependency>
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.3.2</version>
</dependency>
<dependency>
    <groupId>com.google.code.gson</groupId>
```

```
            <artifactId>gson</artifactId>
            <version>2.8.5</version>
        </dependency>
        </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                    <fork>true</fork>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

3.1.3　创建类 TemptTOKEN

在包 edu.bookcode 中创建 service 子包，并在包 edu.bookcode.service 中创建类 TemptTOKEN，代码如例 3-2 所示。

【例 3-2】 类 TemptTOKEN 的代码示例。

```
package edu.bookcode.service;
import lombok.Data;
import lombok.NoArgsConstructor;
@NoArgsConstructor //代表无参构造方法
@Data
//@Data 等于@Setter、@Getter、@ToString、@EqualsAndHashCode 等方法
public class TemptTOKEN {
    private String accessToken;
    private final String expiresIn = "7200";
    private Long createdTime = 0L;
    public TemptTOKEN(String accessToken) {
        this.accessToken = accessToken;
        this.createdTime = System.currentTimeMillis();
    }
    public boolean isExpired(){
        Long currentTime = System.currentTimeMillis();
        Long realTime = this.getCreatedTime();
        if ((currentTime - realTime) < 7200 * 1000) {
            return false;
        } else {
```

```
                return true;
            }
        }
}
```

3.1.4 创建类 URLtoTokenUtil

在包 edu.bookcode.service 中创建类 URLtoTokenUtil，代码如例 3-3 所示。

【例 3-3】 类 URLtoTokenUtil 的代码示例。

```
package edu.bookcode.service;
import java.io.InputStream;
import java.net.URL;
import java.net.URLConnection;
public class URLtoTokenUtil {
public static String getTemptURLToken(String strURL) {
      try {
            URL urlObj = new URL(strURL);
            URLConnection urlConnection = urlObj.openConnection();
            InputStream inputStream = urlConnection.getInputStream();
            byte[] bytes = new byte[1024];
            int len;
            StringBuilder stringBuilder = new StringBuilder();
            while ((len = inputStream.read(bytes) )!= -1){
                  stringBuilder.append(new String(bytes, 0, len));
            }
            String s = stringBuilder.toString().replace("}",",");
            s = s + "\"createdTime\":" + System.currentTimeMillis() + "}";
            return s;
      } catch (Exception e) {
            e.printStackTrace();
      }
      return null;
}
}
```

3.1.5 创建类 TemptTokenUtil

在包 edu.bookcode.service 中创建类 TemptTokenUtil，代码如例 3-4 所示。

【例 3-4】 类 TemptTokenUtil 的代码示例。

```
package edu.bookcode.service;
import net.sf.json.JSONObject;
public class TemptTokenUtil {
    public String getTokenInfo(){
     String strAccess = "https://api.weixin.qq.com/cgi-bin/token?grant_type=client_
credential&appid=AppID&secret=AppSECRET";
        String strAppID = "wxd2f278459c83a8e2";              //需要修改成读者自己的 AppID
```

```
        //需要修改成读者自己的 appsecret
        String strAppSECRET = "b62a858ebe3ab2238a4eaaf423369cef";
        String strURL = strAccess.replace("AppID",strAppID).replace("AppSECRET",strAppSECRET);
        String cont = URLtoTokenUtil.getTemptURLToken(strURL);
        JSONObject jsonObject = JSONObject.fromObject(cont);
        String strAccessToken = jsonObject.getString("access_token");
        long time = jsonObject.getLong("createdTime");
        //目前有 2h 的限制,2h 之内可以继续使用,超过 2h 需要重新生成
        if(System.currentTimeMillis() - time >= 7200 * 1000) {
            cont = URLtoTokenUtil.getTemptURLToken(strURL);
        }
        return strAccessToken;
    }
}
```

3.1.6　创建类 TemptTOKENController

在包 edu.bookcode 中创建 controller 子包,并在包 edu.bookcode.controller 中创建类 TemptTOKENController,代码如例 3-5 所示。

【例 3-5】 类 TemptTOKENController 的代码示例。

```
package edu.bookcode.controller;
import edu.bookcode.service.TemptTokenUtil;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class TemptTOKENController {
    //下面一行是运行本类时的相对地址
    @RequestMapping("/")
    //为了测试方便,在运行其他类时,必须注释掉上一行代码,即修改相对地址
    //并可以去掉下一行代码的注释,修改本类的相对地址
    //@RequestMapping("/testAccessToken")
    //具体的操作方法可参考配套视频中的演示说明
    void  getAccessTemptTOKEN()  {
            System.out.println("临时 token 对象信息:" + new TemptTokenUtil().getTokenInfo());
    }
}
```

3.1.7　运行程序

启动内网穿透工具 Ngrok(或服务器,为了简化开发本书采用 Ngrok)后,在 IDEA 中运行项目入口类 WxgzptkfbookApplication。

运行工具 Postman(可参考相关资源下载、安装 Postman),在 URL 地址栏中输入 http://localhost: 8080/,在方法中选择 POST 方法,单击 Send 按钮,结果显示 200 OK 即表示程序运行正常,如图 3-1 所示。此时在控制台中相关的输出结果如图 3-2 所示。

在手机微信公众号中输入任意文本(输出结果与输入的文本内容无关),如“你好”,如

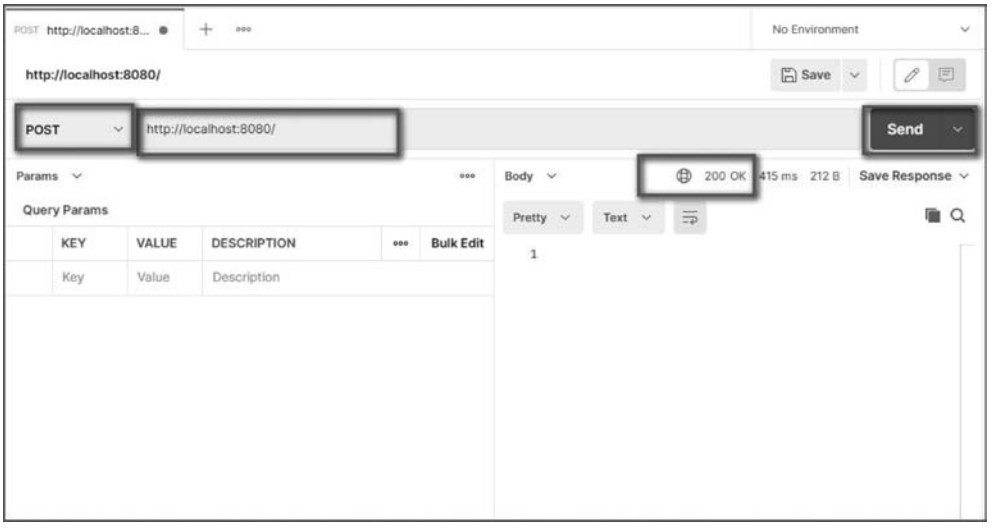

图 3-1　在 Postman 的 URL 中输入 http://localhost：8080/并选择 POST 方法后单击 Send 按钮的结果

```
临时token对象信息:51_npk_oWc7pA3qaj8TmZcdMUKSRuyTt34LyUGLEl5SQV
 -l1TX8NoFLZQOAmj_5ScMzWB6onrAfcIUPQwMkv1wrnF7J7kbADbM72P9OsC39CXPO__umnJ4pESo4eLwNtynJxavHsTkEuLtUed
 ZjNQPiAIAEHG
```

图 3-2　对工具 Postman 进行操作后在控制台中输出的临时 access_token 结果

图 3-3 所示。此时在控制台中相关的输出结果如图 3-4 所示。对照图 3-2 和图 3-4 可以发现，两次操作产生的 access_token 具体信息有差异。读者自己运行时的结果和本书的结果（见图 3-2 或图 3-4）也有差异，而且每隔 2h 的操作结果也会有差异，只要确保能正常输出 access_token 信息即可。后面章节程序中读者的运行结果和本书的结果也可能会有差异，只要能正确输出即可。

图 3-3　在手机微信公众号中输入文本“你好”

```
临时token对象信息:45_XmG8X3fgUrzj5gCyr0zV6joY7_7oFrFAu67-h8F1iC4vkl92flWpM7RCrjBgVYxCu3B_QJ3
-eYIE6ueEINCg4-zoJJqQhSlDQARu2iEiEjoIck007he1MZZwuUhZeAybftahJlXnCOci8b2tHFGcACAMKL
```

图 3-4　在手机微信公众号中输入文本“你好”后在控制台中输出的临时 access_token 结果

3.1.8　运行程序或调试接口的方法说明

在此基础上，对比 1.3.3 节中使用微信公众平台接口调试工具对此接口的调试内容，可以发现手机微信公众号、工具 Postman、微信公众平台接口调试工具三种运行（或调试）方法的差异。在这三种方法中，手机微信公众号运行程序效果最好，但是来回在手机、IDEA 开发环境中操作略显复杂，特别是在学习微信公众平台应用开发的初期（出错率相对偏多时）较为麻烦。利用微信公众平台接口调试工具进行调试相对复杂。因此，本书在可以不必在手机微信公众号中测试时，优先选用工具 Postman 运行程序，其次使用手机微信公众号运行程序。

3.2　网络检测


视频讲解

3.2.1　说明

网络检测 API 可以帮助排查回调连接失败问题，该接口的 URL 为 https://api.weixin.qq.com/cgi-bin/callback/check?access_token=ACCESS_TOKEN，公众号的参数 access_token 是必需的，开发时需要用生成的 access_token 值去替换占位参数 ACCESS_TOKEN。后面章节应用开发中参数 access_token 的含义、用法相同。接口 POST 请求的输入参数为 JSON 格式，如例 3-6 所示。

【例 3-6】 JSON 格式的输入参数示例。

```
{
    "action": "all",
    "check_operator": "DEFAULT"
}
```

接口 POST 请求的两个参数都是必需的。action 执行的检测动作，允许的值包括 dns（做域名解析）、ping（做 ping 检测）、all（dns 和 ping 都做）。check_operator 指定平台从某个运营商进行检测，允许的值为 CHINANET（电信出口）、UNICOM（联通出口）、CAP（腾讯自建出口）、DEFAULT（根据 IP 来选择运营商）。

3.2.2　创建类 CommonUtil

继续在 3.1 节的基础上进行开发，在包 edu.bookcode.service 中创建类 CommonUtil，代码如例 3-7 所示。

【例 3-7】 类 CommonUtil 的代码示例。

```
package edu.bookcode.service;
import net.sf.json.JSONObject;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.URL;
import java.net.URLConnection;
public class CommonUtil {
 public static JSONObject httpsRequest(String requestUrl, String requestMethod, String outputStr) {
      JSONObject jsonObject = null;
      URL url;
      try {
            url = new URL(requestUrl);
            URLConnection conn = url.openConnection();
            conn.setDoOutput(true);
            conn.setDoInput(true);
            conn.setUseCaches(false);
            if (null != outputStr) {
                  OutputStream outputStream = conn.getOutputStream();
                  outputStream.write(outputStr.getBytes("UTF-8"));
                  outputStream.close();
            }
            InputStream inputStream = conn.getInputStream();
            InputStreamReader inputStreamReader = new InputStreamReader(inputStream, "utf-8");
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
            String str ;
            StringBuffer buffer = new StringBuffer();
            while ((str = bufferedReader.readLine()) != null) {
                  buffer.append(str);
            }
            bufferedReader.close();
            inputStreamReader.close();
            inputStream.close();
            jsonObject = JSONObject.fromObject(buffer.toString());
      } catch (Exception e) {
            e.printStackTrace();
      }
      return jsonObject;
 }
}
```

3.2.3　创建类 TestNetController

在包 edu.bookcode.controller 中创建类 TestNetController，代码如例 3-8 所示。

【例 3-8】 类 TestNetController 的代码示例。

```
package edu.bookcode.controller;
import edu.bookcode.service.CommonUtil;
import edu.bookcode.service.TemptTokenUtil;
import net.sf.json.JSONObject;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class TestNetController {
    //下面一行是运行本类时的相对地址
    @RequestMapping("/")
    //为了测试方便,在运行其他类时,必须注释掉上一行代码,即修改相对地址
    //并可以去掉下一行代码的注释,修改本类的相对地址
    //@RequestMapping("/testNet")
    void  testNet() {
        String strAPI = "https://api.weixin.qq.com/cgi-bin/callback/check?access_token=
ACCESS_TOKEN";
        String requestUrl = strAPI.replace("ACCESS_TOKEN",new TemptTokenUtil().getTokenInfo());
        String data = "{" +
                      "    \"action\": \"all\", " +
                      "    \"check_operator\": \"DEFAULT\"\n" +
                      "}" ;
        JSONObject jsonObject = CommonUtil.httpsRequest(requestUrl, "POST",data);
        System.out.println(jsonObject);
    }
}
```

3.2.4　运行程序

启动内网穿透工具后，按照例 3-5 中注释给出的提示修改类 TemptTOKENController 的相对地址，由原来的@RequestMapping("/")改成@RequestMapping("/testAccessToken")，这样修改可以避免程序运行时相对地址的冲突，因为此时新增加的 TestNetController 的相对地址也是@RequestMapping("/")。后面章节中修改程序的相对地址的方法、原因也是如此。这种方法主要是为了开发时调试、运行程序的方便，并降低程序出错的可能性。开发完成后正式运行程序时，可以通过相对地址的不同来实现程序的整合、集成(可参考 15 章案例的整合方法)。

在 IDEA 中运行项目入口类 WxgzptkfbookApplication。

运行工具 Postman，在 URL 地址栏中输入 http://localhost：8080/，在方法中选择 POST 方法，单击 Send 按钮，结果显示 200 OK 表示程序运行成功。为了节省篇幅，后面章

节将这些操作和结果简单表述为“在工具 Postman 的 URL 中输入 http://localhost:8080/，选择 POST 方法成功运行程序”。此时在控制台中相关的输出结果如图 3-5 所示。

{"ping":[{"package_loss":"0%","ip":"134.175.220.239","from_operator":"CAP","time":"28.666ms"}],
"dns":[{"real_operator":"CAP","ip":"134.175.220.239"}]}

图 3-5 网络检测时在控制台中输出的结果


视频讲解

3.3 获取 IP 地址

3.3.1 说明

公众号有时需要获知微信服务器 IP 地址列表以便进行相关限制。此时可以调用接口 https://api.weixin.qq.com/cgi-bin/get_api_domain_ip?access_token=ACCESS_TOKEN。

与之相对应，微信服务器调用公众号服务器所使用的出口 IP（即回调 IP）的接口 URL 为 https://api.weixin.qq.com/cgi-bin/getcallbackip?access_token=ACCESS_TOKEN。

3.3.2 创建类 WXServerInfoController

继续在 3.2 节的基础上进行开发。在包 edu.bookcode.controller 中创建类 WXServerInfoController，代码如例 3-9 所示。

【例 3-9】 类 WXServerInfoController 的代码示例。

```
package edu.bookcode.controller;
import edu.bookcode.service.TemptTokenUtil;
import net.sf.json.JSONObject;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import edu.bookcode.service.URLtoTokenUtil;
@RestController
public class WXServerInfoController {
    //下面一行是运行本类时的相对地址
    @RequestMapping("/")
    //为了测试方便,在运行其他类时,必须注释掉上一行代码,即修改相对地址
    //并可以去掉下一行代码的注释,修改本类的相对地址
    //@RequestMapping("/testIP")
    void  getIpAddress() {
String getIPAPI = "https://api.weixin.qq.com/cgi-bin/get_api_domain_ip?access_token=
ACCESS_TOKEN";
String getCallbackIP = "https://api.weixin.qq.com/cgi-bin/getcallbackip?access_token=
ACCESS_TOKEN";
        String strToken = new TemptTokenUtil().getTokenInfo();
        String strIPURL = getIPAPI.replace("ACCESS_TOKEN",strToken );
        String strCallbackURL = getCallbackIP.replace("ACCESS_TOKEN",strToken );
```

```
        String ipResult = URLtoTokenUtil.getTemptURLToken(strIPURL);
        String callbackIP = URLtoTokenUtil.getTemptURLToken(strCallbackURL);
        JSONObject jsonObject = JSONObject.fromObject(ipResult);
        JSONObject callbackJSON = JSONObject.fromObject(callbackIP);
        System.out.println("IP:" + jsonObject);
        System.out.println("Callback IP:" + callbackJSON);
    }
}
```

3.3.3　运行程序

启动内网穿透工具后，按照例 3-8 中注释给出的提示修改 TestNetController 的相对地址，并再次在 IDEA 中运行项目入口类 WxgzptkfbookApplication。

在工具 Postman 的 URL 中输入 http://localhost：8080/，选择 POST 方法成功运行程序后（详细的操作方法可参考 3.2.4 节，后面章节相同），控制台中的输出结果如图 3-6 所示。

```
IP:{"ip_list":["101.226.212.27","112.60.0.226","112.60.0.235","116.128.163.147","117.184.242.111",
 "121.51.130.115","121.51.166.37","121.51.90.217","180.97.7.108","182.254.88.157","183.3.234.152",
 "183.57.48.62","203.205.239.82","203.205.239.94","36.152.5.109","58.246.220.31","58.251.80.204","58
 .251.82.216","108.108.10.128"]}
Callback IP:{"ip_list":["118.126.124.0/24","119.29.180.0/24","119.29.9.0/24","162.62.80.0/24","162.62
 .81.0/24","42.192.0.0/24","42.192.6.0/24","42.192.7.0/24","81.69.18.0/24","81.69.19.0/24","81.69.229
 .0/24","81.71.140.0/24","81.71.19.0/24","42.192.6.0/24","42.192.0.0/24","175.24.211.0/24","81.69.229
 .0/24","81.69.101.0/24","81.69.103.0/24","101.226.103.0/24"]}
```

图 3-6　获取 IP 地址时控制台中的输出结果

3.4　验证消息来自微信服务器


视频讲解

3.4.1　说明

在公众号管理后台设置 URL 等信息后，微信服务器将发送请求到填写的公众号服务器 URL 上，请求携带参数包括微信加密签名（signature）、时间戳（timestamp）、随机数（nonce）和随机字符串（echostr）。

signature 结合了在公众号管理后台填写的 Token（或 token）参数（此 Token 和临时 access_token 不同）。若确认请求来自微信服务器，原样返回参数 echostr 内容，则接入生效，否则接入失败。

验证消息来自微信服务器（或称后台）的流程包括：

（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）获得加密后的字符串与 signature 对比。

该流程如图 3-7 所示。

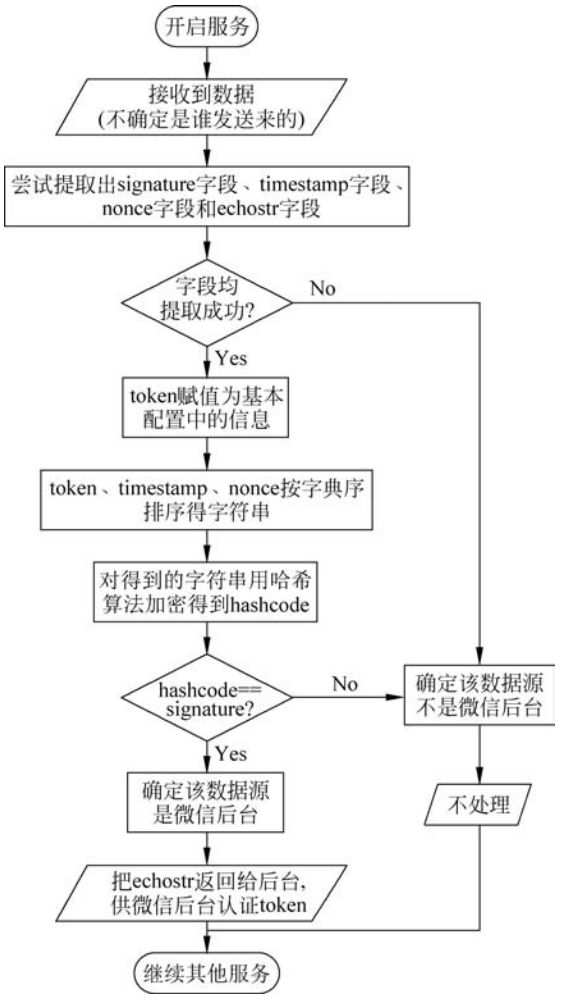

图 3-7　验证消息来自微信服务器(即微信后台)的流程

3.4.2　创建类 CheckUtil

继续在 3.3 节的基础上进行开发。在包 edu.bookcode 中创建 util 子包，并在包 edu.bookcode.util 中创建类 CheckUtil，代码如例 3-10 所示。

【例 3-10】 类 CheckUtil 的代码示例。

```
package edu.bookcode.util;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
public class CheckUtil {
//与微信公众号管理后台接口配置信息的 Token 保持一致,不是前面的 access_token
//修改成读者自己的 Token
private static final String TOKEN = "woodstoneweixingongzhonghao";
public static boolean checkSign(String signature, String timestamp, String nonce) {
```

```
        String[] paramArr = { TOKEN, timestamp, nonce };
        System.out.println("String of TOKEN :" + TOKEN);
        System.out.println("String of timestamp:" + timestamp);
        System.out.println("String of nonce:" + nonce);
        Arrays.sort(paramArr);
        String threeString = "";
        for (String each:paramArr) {
            threeString += each;
        }
        String mySHA1 = sha1(threeString);     //进行 sha1 加密
        return mySHA1.equalsIgnoreCase(signature);
    }
    //sha1 加密的实现方法
    //还可以参考微信公众平台官方文档给出的实现代码
    private static String sha1(String threeString) {
        StringBuilder stringBuilder = new StringBuilder();
        try {
            MessageDigest messageDigest = MessageDigest.getInstance("sha1");
            byte[] bytes = messageDigest.digest(threeString.getBytes());
            char[] chars = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'};
            for (byte b:bytes) {
                stringBuilder.append(chars[b >> 4&15]);
                stringBuilder.append(chars[b&15]);
            }
            System.out.println("compute signature: " + stringBuilder);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return stringBuilder.toString();
    }
}
```

3.4.3　创建类 VerifyWXServerController

在包 edu.bookcode.controller 中创建类 VerifyWXServerController，代码如例 3-11 所示。

【例 3-11】 类 VerifyWXServerController 的代码示例。

```
package edu.bookcode.controller;
import edu.bookcode.util.CheckUtil;
import org.springframework.web.bind.annotation.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@RestController
public class VerifyWXServerController {
    //下面一行是运行本类时的相对地址
    @RequestMapping("/")
    //为了测试方便,在运行其他类时,必须注释掉上一行代码,即修改相对地址
    //并可以去掉下一行代码的注释,修改本类的相对地址
```

```
    //@RequestMapping("/testVerifyInfo")
    public String verifyInfoFromWXServer(HttpServletRequest req,HttpServletResponse res) {
        String signature = req.getParameter("signature");
        String timestamp = req.getParameter("timestamp");
        String nonce = req.getParameter("nonce");
        String echostr = req.getParameter("echostr");
        System.out.println("signature:" + signature + ",timestamp:" + timestamp +
",nonce:" + nonce );
        if(CheckUtil.checkSign(signature,timestamp,nonce)){
            return echostr; //假如在配置接口时出现"配置失败"信息,可运行该类
        };
        return "ok";
    }
}
```

3.4.4 运行程序

启动内网穿透工具后，按照例 3-9 中注释给出的提示修改 WXServerInfoController 的相对地址，并再次在 IDEA 中运行项目入口类 WxgzptkfbookApplication。

在手机微信公众号中输入任何文本，如“你好”，在控制台中的输出结果如图 3-8 所示。图 3-8 中两处 signature 内容相同，说明通过校验（即消息来自微信服务器）。在公众号管理后台配置 URL 等信息时，若提示出现“配置失败”的错误，也可以通过运行 3.4 节程序来解决。

图 3-8 验证消息来自微信服务器时在手机微信公众号中输入文本后在控制台中输出的结果

习题 3

简答题

1. 简述验证消息是否来自微信服务器的方法。
2. 画出验证消息来自微信服务器的流程图。

实验题

1. 实现对 access_token 的获取。
2. 实现网络检测。
3. 实现对 IP 地址的获取。
4. 实现验证消息是否来自微信服务器。