# 微信后端获取数据技术文档

## 主要目标

根据数据库中保存的公众号，通过 gsdata 接口抓取它们每天发布的文章并保存到数据库中，并每天获取它们的阅读量与点赞量，每篇文章跟踪七天。

## 获取方式

微信有一个获取阅读数和点赞数的接口（http://mp.weixin.qq.com/mp/getappmsgext），通过向这个接口发送 http post 请求，就可以获取。post 请求需要许多参数，包括以下：

__biz: 公众号唯一 id

mid：推送消息编号

sn：暂不详，不重要

idx：文章在消息中的编号

f：为固定值 json

is_need_ad：为固定值 0

is_only_read：为固定值 1

key：根据微信帐号随机生成

uin：微信帐号唯一标识

发送 post 请求的时候，需要填上以上参数。其中，__biz、mid、sn、idx 可以从微信文章的 url 中获得，

所以当想要知道一片文章的阅读数点赞数时，首先要从这篇文

章的 url 中解析出这四个参数，然后 f、is_need_ad、is_only_read 是固定值，要解决的只有 key 和 uin 两个参数。关于 key 和 uin 的获得，我们是通过代理服务器的方式来获取的。当你用客户端访问微信公众号里的文章时，就会产生 key，这个 key 在两个小时内有效。有一台 windows 电脑上，运行了微信客户端，通过点击脚本，每隔一段时间访问几篇文章，而这台电脑连接了我们的代理服务器，所有的请求都从代理服务器走。代理服务器会记录所有请求的信息到日志中，所以 key 和 uin 都在日志中。我们就可以通过解析日志来获取 key 和 uin。此处有一个注意事项：微信的接口访问不能太频繁，否则会被暂时封号，获取不了数据，现在的请求时间是 4s.

## 业 务 逻 辑

1. 我们跟踪了一批公众号，这些公众号存储在数据库中，第一步，从数据库中找出这些公众号
2. 根据公众号名称，调用一个第三方平台的接口，就可以获取每个公众号某一天发出的文章列表，然后存入数据库的文章表中（表名：wsa_article）
3. 我们现在对一篇文章要跟踪七天。所以从数据库的文章表中 select 出最近七天的文章进行处理，里面包含着文章 url，然后根据代理服务器日志里的 key 和 uin，就可以获取数据（注意获取间隔），获取的数据也要写到数据库的一张表中。（表名：wsa_article_stats）

4. 以上所说的数据库都是实验室里服务器上的数据库。还需要和远程阿里云服务器上的数据库做一个同步操作，暂不细谈。

## 代 码 解 释

### 一. WechatSupporter

该类为主类，按照抓取公众号文章，获得文章的阅读与点赞数，同步本地服务器数据库数据到阿里云数据库，多线程抓取HTML源码并保存到本地的业务逻辑来完成微信后端的数据获取。

```
public class WechatSupporter {
    static Logger logger = Logger.getLogger(WechatSupporter.class.getName());

    public static void main(String[] args) {
        DBConnectionPool.create();

        crawlArticle crawler = new crawlArticle();
        DataProvider dataProvider = new DataProvider();
        Counter counter = new Counter();
        syncMan syncman = new syncMan();

        logger.info("Start to crawl article...");
        crawler.crawl();

        logger.info("Start to get tasks..");
        try {
            dataProvider.getTasks();
        } catch (SQLException e) {
            logger.error(e.getClass() + " " + Arrays.toString(e.getStackTrace()));
        }

        logger.info("Start to process tasks...");
        try {
            dataProvider.doTasks();

            dataProvider.redoTasks();

        } catch (SQLException e) {

            logger.error(e.getClass() + " " + Arrays.toString(e.getStackTrace()));

        }


        logger.info("Start to count...");

        try {

            counter.doCount();

        } catch (SQLException e) {

            logger.error(e.getClass() + " " + Arrays.toString(e.getStackTrace()));

        }




        logger.info("Start to synchronize...");

        try {

            syncman.doSync();

        } catch (SQLException e) {

            logger.error(e.getClass() + "" + Arrays.toString(e.getStackTrace()));

        }


        logger.info("Start to get HTML...");

        ArrayList<Article> articles;

        articles = HTMLHelper.getArticles();


        int threadNums = 4;

        int block = articles.size() / threadNums;


        for (int i = 0; i < threadNums; i += 1) {

            int begin = block * i;

            int end = ((i == threadNums - 1) ? articles.size() : block * (i + 1));

            new Thread(new HTMLGetter(new ArrayList<>(articles.subList(begin, end)))).start();

        }

    }

}
```

## 二. CrawlArticle

该类主要通过gsdata的接口访问获取需要爬取的公众号以及文章。

```
public class crawlArticle {
```

```
private static Logger logger = Logger.getLogger(crawlArticle.class.getName());


private final static String appId = "";

private final static String appKey = "";

private final static String apiUrl = "http://open.gsdata.cn/api/wx/opensearchapi/content_list";


private List<String> official_accounts; //保存获取的微信公众号

private int article_cnt;      //获得的文章数量

private int duplicated;       //重复的文章数量

private Map<String,Integer> hashMap;      //判断是否存在重复文章的hashmap


/**

 * @discription  从数据库获得要爬取的微信公众号

 * @throws Exception

 */

private void getAccounts() throws Exception {

    Connection conn = SourceDBPool.getConnection();


    ResultSet rs = conn.createStatement().executeQuery("select official_account from wsa_official_account");

    while (rs.next()) {

        official_accounts.add(rs.getString(1));

    }


    logger.info(official_accounts.size() + " official_accounts.");

    conn.close();

}


/**

 * @discription  通过gsdata提供的接口和jar包获得指定公众号的文章，并保存到数据库

 * @param account   要获取文章的公众号

 * @throws Exception

 */

private void getArticles(String account) throws Exception {

    DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");

    Date yesterday = new Date(System.currentTimeMillis() - (24 * 3600 * 1000));


    ApiSdk apiSdk = ApiSdk.getApiSdk(appId,appKey);

    Map<String, Object> map = new HashMap<>();


    map.put("wx_name",account);

    map.put("postdate",formatter.format(yesterday));


    String jsonReturned = apiSdk.callInterFace(apiUrl, map);

    JSONObject jsonObject = new JSONObject(jsonReturned);
```

```
JSONArray jsonArray = (JSONArray) jsonObject.getJSONObject("returnData").get("items");

article_cnt += jsonArray.length();

Connection conn = SourceDBPool.getConnection();
String sql = "insert into wsa_article(official_account, publish_time, title, summary, url, "
        + "add_time, ranking, source_url, author, copyright) values(?,?,?,?,?,?,?,?,?,?)";
for (int i = 0; i < jsonArray.length(); ++i) {
    String title = (String) ((JSONObject) jsonArray.get(i)).get("title");
    String toHash = account + title;
    if (hashMap.containsKey(toHash)) {
        ++duplicated;
        --article_cnt;
        continue;
    } else {
        hashMap.put(toHash,1);
    }

    String url = (String) ((JSONObject) jsonArray.get(i)).get("url");
    String publish_time = (String) ((JSONObject) jsonArray.get(i)).get("posttime");
    String summary = (String) ((JSONObject) jsonArray.get(i)).get("content");
    String add_time = (String) ((JSONObject) jsonArray.get(i)).get("add_time");
    String author = (String) ((JSONObject) jsonArray.get(i)).get("author");
    String source_url = (String) ((JSONObject) jsonArray.get(i)).get("sourceurl");
    String copyright = (String) ((JSONObject) jsonArray.get(i)).get("copyright");
    String ranking = (String) ((JSONObject) jsonArray.get(i)).get("top");

    PreparedStatement pst = conn.prepareStatement(sql);
    pst.setString(1, account);
    pst.setString(2, publish_time);
    pst.setString(3, title);
    pst.setString(4, summary);
    pst.setString(5, url);
    pst.setString(6, add_time);
    pst.setString(7, ranking);
    pst.setString(8, source_url);
    pst.setString(9, author);
    pst.setString(10, copyright);

    pst.addBatch();
    pst.executeBatch();
    pst.close();

    logger.info(account + ' ' + publish_time + ' ' + title + ' ' + summary + ' ' +
                    url + ' ' + add_time + ' ' + ranking + ' ' + source_url + ' ' +
                    author + ' ' + copyright);
        }
        conn.close();
    }

    /**
     * @discription  开始抓取操作
     */
    void crawl(){
        official_accounts = new ArrayList<>();
        hashMap = new HashMap<>();

        try {
            getAccounts();
        } catch (Exception e) {
            logger.error(e.getClass() + " " + Arrays.toString(e.getStackTrace()));
        }

        int repeat =2;
        while ((repeat--)>0) {
            for (String accout : official_accounts) {
                try {
                    getArticles(accout);
                } catch (Exception e) {
                    logger.error(e.getClass() + " " + Arrays.toString(e.getStackTrace()));
                }
            }
        }

        for (Entry<String, Integer> entry : hashMap.entrySet()) {
            logger.info("key= " + entry.getKey());
        }

        logger.info("Duplicates : " + duplicated);
        logger.info("There are " + article_cnt + " new articles.");
    }

    public static void main(String[] args) {
        crawlArticle ca = new crawlArticle();
        long start = System.currentTimeMillis();
        ca.crawl();
        System.out.println("Time : " + (System.currentTimeMillis() - start));
    }
```

```
}
```

## 三．DataProvider

从数据库中获取文章，从微信接口获取数据，再将获取的数据存入数据库

```
public class DataProvider {
    private static Logger logger = Logger.getLogger(DataProvider.class.getName());


    private ArrayList<ArrayList<String>> tasks;        //要获取数据的文章
    private ArrayList<ArrayList<String>> redos;        //要重新获取数据的文章
    private Key key;


    /**
     * @Description  构造函数
     */
    public DataProvider() {
        tasks = new ArrayList<>();
        redos = new ArrayList<>();
        key = new LogExtracter().getKey();
    }


    /**
     * @Description  通过LogExtracter，更换key值
     */
    private void changeKey() {
        key = new LogExtracter().getKey();
        logger.info("key changed: " + key);
    }


    /**
     * @Description  访问微信接口获得数据
     * @param key key值
     * @param url  文章地址
     * @return  返回获得的数据
     * @throws IOException
     */
    private String getappmsgext(Key key, String url) throws IOException {


        Map<String, String> params = NetUtil.parseQuery(url);
        String api = "http://mp.weixin.qq.com/mp/getappmsgext";
        return Request.Post(api)
                .addHeader("User-Agent", "Mozilla/5.0 (iPhone; CPU iPhone OS 9_2_1 like Mac OS X) AppleWebKit/601.1.46 " +
                        "(KHTML, like Gecko)       Mobile/13D15 MicroMessenger/6.3.13 NetType/WIFI Language/zh_CN")
                .addHeader("Accept", "*/*")
                .addHeader("Accept-Charset", "utf-8, iso-8859-1, utf-16, *;q=0.7")
                .addHeader("Accept-Language", "zh-CN")
                .addHeader("Connection", "keep-alive")
                .addHeader("X-Requested-With", "XMLHttpRequest")
                .bodyForm(Form.form()
                        .add("__biz", params.get("__biz"))
                        .add("mid", params.get("mid"))
                        .add("sn", params.get("sn"))
                        .add("idx", params.get("idx"))
                        .add("f", "json")
                        .add("is_need_ad", "0")
                        .add("key", key.getKey())
                        .add("uin", key.getUin())
                        .add("is_only_read", "1").build())
                .execute().returnContent().asString();
    }


    /**
     * @Description  检索发布七天内的文章并保存到tasks
     * @throws SQLException
     */
    @SuppressWarnings({"SqlDialectInspection", "SqlNoDataSourceInspection"})
    void getTasks() throws SQLException {
        Connection conn = SourceDBPool.getConnection();


        ResultSet rs = conn.createStatement().executeQuery("select * from wsa_article where to_days(now()) - " +
                "to_days(publish_time) <= 7");


        //noinspection Duplicates
        while (rs.next()) {
            ArrayList<String> row = new ArrayList<>();
            for (int i = 1; i <= 11; i += 1) {
                row.add(rs.getString(i));
            }
            tasks.add(row);
        }


        conn.close();
    }
```

```
/**
 * @Description  解析微信接口返回的数据并将有效数据保存到数据库中
 * @throws SQLException
 */
@SuppressWarnings({"SqlDialectInspection", "SqlNoDataSourceInspection"})
void doTasks() throws SQLException {
    Connection conn = SourceDBPool.getConnection();
    logger.info("tasks size: " + tasks.size());
    for (ArrayList<String> row : tasks) {
        try {
            Thread.sleep(4000);
            String id = row.get(0);
            String url = row.get(5);

            String response = getappmsgext(this.key, url);

            logger.info(response);

            JSONObject jsonObject = new JSONObject(response);
            JSONObject data = (JSONObject) jsonObject.get("appmsgstat");
            Integer read_num = (Integer) data.get("read_num");
            Integer like_num = (Integer) data.get("like_num");

            String sql = "insert into wsa_article_stats(article_id, read_count, like_count, add_time) values(?,?,?,now())";
            PreparedStatement pst = conn.prepareStatement(sql);
            pst.setString(1, id);
            pst.setString(2, read_num.toString());
            pst.setString(3, like_num.toString());

            logger.info(pst);

            pst.addBatch();
            pst.executeBatch();
            pst.close();
        } catch (Exception e) {
            logger.error(e.getClass() + " " + Arrays.toString(e.getStackTrace()));
            if (e instanceof JSONException) {
                changeKey();
            }
            redos.add(row);
        }
    }
}
```

```
    /**
     * @Description 将获取数据失败的文章重新获取一次
     * @throws SQLException
     */
    void redoTasks() throws SQLException {
        while (redos.size() > 0) {
            logger.info("redo begins, redo nums: " + redos.size());
            tasks = new ArrayList<>(redos);
            redos.clear();

            doTasks();
        }
    }

    public static void main(String[] args) throws Exception {
//          String article_url =
"http://mp.weixin.qq.com/s?__biz=MjM5MTM1NzQ3Mg%3D%3D&mid=2652379422&idx=2&sn=b1bb6f4a7d66cb8abb8c014e5f03eb2f&scene=4#wechat_redirect";
//          byte[] bytes = article_url.getBytes();
//          System.out.println(new DataProvider().getappmsgext(new LogExtracter().getKey(), new String(bytes, "utf-8")));
        DataProvider dp = new DataProvider();
    }

}
```

## 四．DBConnectionPool

数据库连接池，为整个程序链接释放数据库提高效率。

```
public class DBConnectionPool
{
    public static ConnectionPool SourceDBPool ; //本地服务器数据库连接池
    public static ConnectionPool DestDBPool ;   //阿里云服务器数据库连接池
    private static Logger logger = Logger.getLogger(DBConnectionPool.class.getName());

    /**
     *@Description 创建数据库链接池
     */
    static void create()  {
        if (SourceDBPool != null) {
            SourceDBPool.release();
        }
        if (DestDBPool != null) {
            DestDBPool.release();
        }
        try {
            Class c = Class.forName("com.mysql.jdbc.Driver");
            Driver driver = (Driver) c.newInstance();
            DriverManager.registerDriver(driver);
            SourceDBPool = new ConnectionPool("sourceDBPool", 3, 30, 50, 90,
                    "jdbc:mysql://10.10.65.172:3306/wsa?useUnicode=true&characterEncoding=utf-8", "wechat", "sklcc");
            DestDBPool = new ConnectionPool("sourceDBPool", 3, 30, 50, 90,
                "jdbc:mysql://:3306/wsa?useUnicode=true&characterEncoding=utf-8", "wsa", "sklcc");
        } catch (Exception e) {
            logger.info("Failed to connect DataBase!" + " " + Arrays.toString(e.getStackTrace()));

        }

    }

    /**
     *@Description 释放数据库链接池
     */
    static void release() {
        SourceDBPool.release();
        DestDBPool.release();
    }


    public static void main(String[] args) {

    }
}
```

## 五．HTMLGetter

通过相关jar包获取微信文章的HTML源码，并保存到本地服务器，同时判断下文章是否删除，是否修改。

```
/**
 * @Description 多线程类获取HTML代码源码
 */
@SuppressWarnings({"SqlDialectInspection", "SqlNoDataSourceInspection"})
public class HTMLGetter implements Runnable {

    private ArrayList<Article> errorTasks;  //保存获取HTML源代码错误的文章

    public HTMLGetter(ArrayList<Article> articles) {
        this.errorTasks = articles;
    }


    /**
     * @Description  开始线程
     */
    public void run() {


        while (errorTasks.size() > 0) {
            HTMLHelper.logger.info("HTMLGetter thread, task size: " + errorTasks.size());
            ArrayList<Article> temp = new ArrayList<>(errorTasks);
            errorTasks.clear();
            for (Article article : temp) {
                try {
                    String oldContent = HTMLHelper.exists(article.id);
                    String newContent = HTMLHelper.getContent(article.url);
                    if (oldContent == null) {
                        HTMLHelper.insertContent(article.id, newContent);
                    } else {
                        HTMLHelper.setDeleteFlag(article.id, newContent, oldContent);
                    }
                } catch (Exception e) {
                    errorTasks.add(article);
                    HTMLHelper.logger.error("error article: " + article);
                    HTMLHelper.logger.error(e.getClass() + " " + Arrays.toString(e.getStackTrace()));
                }
            }
        }
    }


    public static void main(String[] args) throws IOException, SQLException {
        String id = "145917";
        String url = "http://mp.weixin.qq.com/s?__biz=MjM5MTM1NzQ3Mg==&mid=2652382266&idx=3&sn=897e6020c0f43850c4256aa0b8eb08eb&scene=4#wechat_redirect";
        String oldContent = HTMLHelper.exists(id);
        String newContent = HTMLHelper.getContent(url);
        HTMLHelper.setDeleteFlag(id, newContent, oldContent);
    }
}


/**
 * @Description HTMLGetter的工具类
 */
@SuppressWarnings({"SqlDialectInspection", "SqlNoDataSourceInspection"})
```

```
class HTMLHelper {

    static Logger logger = Logger.getLogger(HTMLHelper.class.getName());


    /**
     * @Description  从数据库获得最近七天的文章
     * @return  获得的文章
     */
    static ArrayList<Article> getArticles() {
        ArrayList<Article> articles = new ArrayList<>();


        String sql = "select id, url from wsa_article where to_days(now()) - to_days(publish_time) <= 7";
        ResultSet rs;
        try (Connection connection = SourceDBPool.getConnection()) {
            rs = connection.createStatement().executeQuery(sql);
            while (rs.next()) {
                articles.add(new Article(rs.getString(1), rs.getString(2)));
            }
        } catch (SQLException e) {
            logger.error(e.getClass() + " " + Arrays.toString(e.getStackTrace()));
        }


        return articles;
    }


    /**
     * @Description  从文章地址获得HTML源码
     * @param url  文章地址
     * @return HTML源代码
     * @throws IOException
     */
    static String getContent(String url) throws IOException {
        Document doc = Jsoup.connect(url).get();
        doc.select("script").remove();
        return new String(Base64.getEncoder().encode(doc.toString().getBytes()));
    }


    /**
     * @Description  判断html是否存在
     * @param id  文章id
     * @return  如果存在，返回html代码，如果不存在，返回null
     * @throws SQLException
     */
    static String exists(String id) throws SQLException {
        String res = null;
        Connection connection = SourceDBPool.getConnection();


        String sql = "select content from wsa_article_content where article_id = " + id;

        ResultSet rs = connection.createStatement().executeQuery(sql);

        if (rs.next()) {

                res = rs.getString(1);

        }


        connection.close();

        return res;

}




/**

  * @Description  如果html代码不存在，将html源码保存到数据库

  * @param id  文章id

  * @param content  文章内容

  * @throws SQLException

  */

static void insertContent(String id, String content) throws SQLException {


        Connection connection = SourceDBPool.getConnection();


        String sql = "insert into wsa_article_content(article_id, content) values(?,?)";

        PreparedStatement pst = connection.prepareStatement(sql);

        pst.setString(1, id);

        pst.setString(2, content);


        logger.info("Content inserted, article_id: " + id);


        pst.addBatch();

        pst.executeBatch();

        pst.close();


        connection.close();

}


/**

  * @Description  通过html长度，判断文章内容是否改变，若改变在数据库中设置flag

  * @param id  文章id

  * @param newContent    新的HTML代码

  * @param oldContent    数据库中保存的旧的html代码

  * @throws SQLException
```

```
     */
    static void setDeleteFlag(String id, String newContent, String oldContent) throws SQLException {
        Boolean modified = false;

        if (newContent.length() == oldContent.length() && newContent.equals(oldContent)) {
            return;
        }

        if (newContent.length() > 20000) {
            modified = true;
        }

//          Connection connection = SourceDBPool.getConnection();
        Connection sourceConn = SourceDBPool.getConnection();
        Connection destConn = DestDBPool.getConnection();
        String sql;

        if (modified) {
            sql = "update wsa_article set deleted = 2, delete_time = now() where id = ?";
            PreparedStatement pst = sourceConn.prepareStatement(sql);
            pst.setString(1, id);
            pst.executeUpdate();
            pst.close();

            pst = destConn.prepareStatement(sql);
            pst.setString(1, id);
            pst.executeUpdate();
            pst.close();

        } else {
            String deleted;
            sql = "select deleted from wsa_article where id = " + id;
            ResultSet rs = sourceConn.createStatement().executeQuery(sql);
            rs.next();
            deleted = rs.getString(1);
            if (deleted.equals("1")) {
                logger.warn("Content had been deleted already!");
                sourceConn.close();
                destConn.close();
                return;
            }

            sql = "update wsa_article set deleted = 1, delete_time = now() where id = ?";
            PreparedStatement pst = sourceConn.prepareStatement(sql);
```

```
                pst.setString(1, id);
                pst.executeUpdate();
                pst.close();

                pst = destConn.prepareStatement(sql);
                pst.setString(1, id);
                pst.executeUpdate();
                pst.close();

                logger.info("Content deleted, article_id: " + id);
            }
            sourceConn.close();
            destConn.close();
        }

        /**
         * @Description  将HTML代码保存到本地文件中
         * @param id
         * @throws SQLException
         * @throws FileNotFoundException
         */
        @SuppressWarnings("unused")
        static void saveToFile(String id) throws SQLException, FileNotFoundException {
            byte[] html = Base64.getDecoder().decode(exists(id).getBytes());

            PrintWriter pw = new PrintWriter(id+".html");
            pw.write(new String(html));
            pw.flush();
            pw.close();
        }
}

/**
 * @Description  文章的定义结构体
 */
@SuppressWarnings("ALL")
class Article {
        String id;    //文章在数据中的id
        String url; //文章地址
        public Article(String i, String str) {
            id = i;
            url = str;
        }
```

```
    public String toString() {
        return "id="+id+" url="+url;
    }
}
```

## 六．LogExtracter

从代理服务器日志中分析出key和uin参数，主要通过正则表达式来捕获分组实现。

```
public class LogExtracter {
    private static String logPath = "/home/sklcc/rahul/wechat/hijack-proxy/log/proxy.log";
//      private static String logPath = "/home/rahul/proxy.log";
    private static String logPattern = "\\[([^\\]]+)\\]\\s*\\[([^\\]]+)\\]\\s*\\[([^\\]]+)\\]\\s*-\\s+(.*)";      //正则表达式

    private ArrayList<String> querys;     //保存要匹配的代理服务器日志句段
    private Pattern pattern;

    private Stack<Key> keystack;      //保存匹配到的key值

    /**
     * @Description  构造函数
     */
    LogExtracter() {
        querys = new ArrayList<>();
        keystack = new Stack<>();
        pattern = Pattern.compile(logPattern);
    }

    /**
     * @Description  读取代理服务器中的每行语句
     */
    private void getQueryString() {
        querys.clear();

        try {
            FileReader reader = new FileReader(logPath);
            BufferedReader bfReader = new BufferedReader(reader);

            String tempLine;
            while ((tempLine = bfReader.readLine()) != null) {
                Matcher m = pattern.matcher(tempLine);
                if (m.find()) {
                    String json = m.group(4);
```

```
                        if (json.charAt(0) != '{') continue;

                        JSONObject tempObject = new JSONObject(json);

                        String requestUrl = tempObject.getJSONObject("request").getString("url");

                        URL urlObject = new URL(requestUrl);

                        querys.add(urlObject.getQuery());

                    }

                }


                bfReader.close();

                reader.close();

            } catch (Exception e) {

                e.printStackTrace();

            }

        }


        /**

         * @Description    获得匹配到的有效参数

         * @param query  需要正则匹配的语句

         * @throws UnsupportedEncodingException

         */

        private void getQueryParams(String query) throws UnsupportedEncodingException {

            if (query == null) return;


            Map<String, List<String>> params = new HashMap<>();

            for (String param : query.split("&")) {

                String[] pair = param.split("=");

                String key = URLDecoder.decode(pair[0], "utf-8");

                String value = "";


                if (pair.length > 1) {

                    value = URLDecoder.decode(pair[1], "utf-8");

                }


                List<String> values = params.get(key);

                if (values == null) {

                    values = new ArrayList<>();

                    params.put(key, values);

                }

                values.add(value);

            }


            if (params.containsKey("key") && params.containsKey("uin")) {

                keystack.push(new Key(params.get("key"), params.get("uin")));

            }
```

```
    }

    /**
     * @Description  逐一将每行语句匹配正则表达式
     */
    private void getKeys() {
        for (String s : querys) {
            try {
                getQueryParams(s);
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            }
        }
    }

    /**
     * @Description  获得key值
     * @return  返回最新的key值
     */
    Key getKey() {
        this.getQueryString();
        this.getKeys();

        return keystack.peek();
    }

    public static void main(String[] args) throws Exception {
        LogExtracter le = new LogExtracter();
        System.out.println(le.getKey());
    }
}

/**
 * @Description key的定义，保存key和uin
 */
class Key {
    private String key;
    private String uin;

    @SuppressWarnings("unused")
    Key(String key, String uin) {
        this.key = key;
        this.uin = uin;
    }
    Key(List<String> keyList, List<String> uinList) {
        this.key = keyList.get(0);
        this.uin = uinList.get(0);
    }

    @SuppressWarnings("unused")
    String getKey() {
        return this.key;
    }

    @SuppressWarnings("unused")
    String getUin() {
        return this.uin;
    }

    public String toString() {
        return "key="+this.key+" uin="+this.uin;
    }
}
```

## 七．NetUtil

网络方面的工具类，暂时只有一个函数，从url中解析出参数。

```
public class NetUtil {


    /**
     * @Description  从url中解析出有效参数
     * @param url
     * @return  返回保存有效参数的map
     * @throws MalformedURLException
     * @throws UnsupportedEncodingException
     */
    static Map<String, String> parseQuery(String url) throws MalformedURLException, UnsupportedEncodingException {
        URL urlObject = new URL(url);
        Map<String, String> params = new HashMap<>();
        String querystring = urlObject.getQuery();

        for (String param : querystring.split("&")) {
            int split = param.indexOf("=");
```

```
                    int length = param.length();

                    String key = URLDecoder.decode(param.substring(0, split), "utf-8");

                    String value = "";


                    if (split+1 < length) {

                         value = URLDecoder.decode(param.substring(split+1, length), "utf-8");

                    }


                    if (!params.containsKey(key)) {

                         params.put(key, value);

                    }

               }

               return params;

          }


          public static void main(String[] args) throws MalformedURLException, UnsupportedEncodingException {

               NetUtil.parseQuery("http://www.test.com/?__biz=3214321==&mid=321423");

          }

     }
```

## 八．syncMan

主要同步本地服务器与阿里云服务器之间的数据。

```
@SuppressWarnings({"SqlNoDataSourceInspection", "Duplicates"})

public class syncMan {

     static Logger logger = Logger.getLogger(syncMan.class.getName());

     @SuppressWarnings("SqlDialectInspection")

          void doSync() throws SQLException {

               Connection sourceConn = SourceDBPool.getConnection();

               Connection destConn = DestDBPool.getConnection();


               logger.info("Start wsa_article sync...");

               // do wsa_article sync

               ResultSet rs = sourceConn.createStatement().executeQuery("select * from wsa_article where to_days(now())" +

                         "- to_days(publish_time) = 1");


               ArrayList<ArrayList<String>> rows = new ArrayList<>();


               //noinspection Duplicates

               while (rs.next()) {

                    ArrayList<String> row = new ArrayList<>();

                    for (int i = 1; i <= 13; i += 1) {

                         row.add(rs.getString(i));

                    }
```

```
        rows.add(row);
    }

    for (ArrayList<String> row : rows) {
        String sql = "insert into wsa_article(id, official_account, publish_time, title, summary, url, "
                + "add_time, ranking, source_url, author, copyright, delete_time, deleted) values(?,?,?,?,?,?,?,?,?,?,?,?,?)";
        PreparedStatement pst = destConn.prepareStatement(sql);
        for (int i = 0; i < row.size(); i++) {
            pst.setString(i + 1, row.get(i));
        }
        logger.info("pst:" + pst);

        //noinspection Duplicates
        try {
            pst.addBatch();
            pst.executeBatch();
            pst.close();
        } catch (Exception e) {
            logger.error(e.getClass() + " " + e.getMessage());
        }
    }

    logger.info("Start wsa_article_stats sync...");
    // do wsa_article_stats sync
    rows.clear();
    rs = sourceConn.createStatement().executeQuery("select * from wsa_article_stats where to_days(now())" +
            "- to_days(add_time) = 0");
    while (rs.next()) {
        ArrayList<String> row = new ArrayList<>();
        for (int i = 1; i <= 7; i++) {
            row.add(rs.getString(i));
        }
        rows.add(row);
    }

    for (ArrayList<String> row : rows) {
        String sql = "insert into wsa_article_stats(id, article_id, read_count, like_count, add_time, delete_time, deleted) " +
                "values(?,?,?,?,?,?,?)";
        PreparedStatement pst = destConn.prepareStatement(sql);
        for (int i = 0; i < row.size(); i++) {
            pst.setString(i + 1, row.get(i));
        }
        logger.info("pst:" + pst);
        //noinspection Duplicates
        try {
            pst.addBatch();
            pst.executeBatch();
            pst.close();
        } catch (Exception e) {
            logger.error(e.getClass() + " " + e.getMessage());
        }
    }

    logger.info("Start wsa_group_stats sync...");
    //do wsa_group_stats sync
    rows.clear();
    rs = sourceConn.createStatement().executeQuery("SELECT * FROM wsa_group_stats");
    while (rs.next()) {
        ArrayList<String> row = new ArrayList<>();
        for (int i = 1; i <= 7; i++) {
            row.add(rs.getString(i));
        }
        rows.add(row);
    }

    for (ArrayList<String> row : rows) {
        String sql = "INSERT INTO wsa_group_stats(id,group_id, account_count, article_count, read_count, like_count, add_time) " +
                "VALUES(?,?,?,?,?,?,?)";
        PreparedStatement pst = destConn.prepareStatement(sql);
        for (int i = 0; i < row.size(); i++) {
            pst.setString(i + 1,row.get(i));
        }
        logger.info("pst:" + pst);

        //noinspection Duplicates
        try {
            pst.addBatch();
            pst.executeBatch();
            pst.close();
        } catch (Exception e) {
            logger.error(e.getClass() + " " + e.getMessage());
        }
    }



    sourceConn.close();
    destConn.close();
```

```
        }


        public static void main(String[] args) {
                DBConnectionPool.create();
                try {
                        new syncMan().doSync();
                } catch (Exception e) {
                        logger.error(e.getClass() + " " + e.getMessage());
                }
        }
}
```

## 九．Counter

主要每天计算公众号总数，文章总数，点赞量总数，阅读量总数，并保存到数据库中，所有文章均跟踪七天。

```
public class Counter {
        private static Logger logger =Logger.getLogger(Counter.class.getName());


        private ArrayList<String> group_accounts;     //保存每个分组的帐号
        private ArrayList<String> group_articles;     //保存每个分组的文章
        private ArrayList<Integer> groupAccountCount;     //保存每个分组的帐号数量
        private ArrayList<Integer> groupArticleCount;     //保存每个分组的文章数量
        private long allAccountCount ;                     //保存所有帐号的数量
        private long allArticleCount;                      //保存所有文章的数量


        /**
          * @Description  从数据库读取帐号并按照分组保存
          * @throws SQLException
          */
        private void getAccounts() throws SQLException {
                groupAccountCount = new ArrayList<>();
                allAccountCount = 0;
                int i = 0;


                Connection conn = SourceDBPool.getConnection();
                ResultSet rs = conn.createStatement().executeQuery("SELECT official_accounts FROM wsa_group");
                while (rs.next()) {
                        ArrayList<String> groupList = new ArrayList<>();
                        String accounts = rs.getString(1);
                        for (String account : accounts.split(","))
```

```
            {
                groupList.add('\'' + account + '\'');
            }
            group_accounts.add('(' + groupList.toString().substring(1,groupList.toString().length() - 1) + ')');    //transfer to string
            groupAccountCount.add(groupList.size());
            logger.info("Group" + (++i) +" has " + groupList.size() + " official_accounts.");
            allAccountCount += groupList.size();
        }
        logger.info("Total accounts: " + allAccountCount);
        conn.close();
    }


    /**
     * @Description  从数据库读取文章并按照分组保存
     * @throws SQLException
     */
    private void getArticles() throws SQLException {
        groupArticleCount = new ArrayList<>();
        allArticleCount = 0;
        int i = 0;

        Connection conn = SourceDBPool.getConnection();
        for (String group : group_accounts) {
            ArrayList<Integer> articleIDs = new ArrayList<>();
            String sql = "SELECT id FROM wsa_article WHERE official_account IN" + group + " AND deleted != 1";
            ResultSet rs = conn.createStatement().executeQuery(sql);
            while (rs.next())
            {
                articleIDs.add(Integer.valueOf(rs.getInt(1)));
            }
            group_articles.add('(' + articleIDs.toString().substring(1,articleIDs.toString().length() - 1) + ')');          //transfer to string
            groupArticleCount.add(articleIDs.size());
            logger.info("Group" + (++i) +" has " + articleIDs.size() + " articles.");
            allArticleCount += articleIDs.size();
        }
        logger.info("Total articles: " + allArticleCount);
        conn.close();
    }


    /**
     * @Description  计算公众号总数，文章总数，点赞量总数，阅读量总数
     * @throws SQLException
     */
    public void doCount() throws SQLException {
```

```
group_accounts = new ArrayList<>();

group_articles = new ArrayList<>();

getAccounts();

getArticles();


long[] read_count = new long[group_accounts.size()];

long[] like_count = new long[group_accounts.size()];

long readCounts = 0;

long likeCounts = 0;

int i = 0;


Connection connection = SourceDBPool.getConnection();


for (String group : group_articles) {


    //select the latest data

    String sql = "SELECT MAX(read_count),MAX(like_count) FROM wsa_article_stats WHERE article_id IN"

            + group + " GROUP BY article_id";

    ResultSet rs = connection.createStatement().executeQuery(sql);

    while (rs.next()) {

        read_count[i] += rs.getInt(1);

        like_count[i] += rs.getInt(2);

    }

    readCounts += read_count[i];

    likeCounts += like_count[i];

    ++i;

    logger.info("Group" + i + "    read_count:" + read_count[i-1] + " like_count:" + like_count[i-1]);

}

logger.info("Total read: " + readCounts + "    Total like: " + likeCounts);


//save to database

String sql = "INSERT INTO wsa_group_stats(group_id, account_count, article_count, read_count, like_count, add_time) " +

        "VALUES(?,?,?,?,?,now())";

for (i = 0; i < group_accounts.size(); ++i) {

    PreparedStatement psta = connection.prepareStatement(sql);

    psta.setInt(1, i+1);

    psta.setInt(2, groupAccountCount.get(i));

    psta.setLong(3, groupArticleCount.get(i));

    psta.setLong(4, read_count[i]);

    psta.setLong(5, like_count[i]);


    psta.addBatch();

    psta.executeBatch();

    logger.info("psta: " + psta);
```

```
                psta.close();
            }
            connection.close();
        }


        public static void main(String[] args) throws SQLException {
            DBConnectionPool.create();
            long start = System.currentTimeMillis();
            Counter ct = new Counter();
            ct.doCount();
            System.out.println("Time:" + (System.currentTimeMillis() - start));
        }
}
```