集大通网络爬虫与可视化平台

项目文档(Version 1.4)

2019年5月22日

最后修改：2019年6月15日

目 录

**[目 录 2](#_Toc16002)**

**[第1章 绪论 1](#_Toc16264)**

[1.1 系统研究背景 1](#_Toc8627)

[1.2 主要开发内容 1](#_Toc11242)

[1.3.1 网络爬虫系统 1](#_Toc6606)

[1.3.2 识别系统 1](#_Toc3705)

[1.3.2 数据可视化平台 2](#_Toc2467)

**[第2章 网络爬虫的设计与实现 3](#_Toc4118)**

[2.1 网络爬虫系统简介 3](#_Toc19095)

[2.2 项目结构 3](#_Toc20596)

[2.3 项目部署 6](#_Toc497)

[2.3.1 IDEA导入 6](#_Toc15025)

[2.3.2 Eclipse导入 8](#_Toc9595)

[2.3.3 项目本地化修改 8](#_Toc28655)

[2.3.4 项目打包jar 8](#_Toc21066)

[2.3.5 部署jar定时脚本 11](#_Toc32644)

[2.4 项目添加新模块 19](#_Toc28329)

[2.4.1 集大通数据分析 20](#_Toc8619)

[2.4.2 新模块的设计 22](#_Toc6270)

[2.4.3 新模块的实现 23](#_Toc4334)

[2.5 本章小结 24](#_Toc11161)

**[第3章 数据可视化平台的设计与实现 25](#_Toc16277)**

[3.1 可视化平台简介 25](#_Toc18390)

[3.2 项目结构 26](#_Toc20963)

[3.2.1 服务端代码结构 27](#_Toc15611)

[3.2.1 客户端代码结构 29](#_Toc19179)

[3.3 项目部署 29](#_Toc12969)

[3.3.1 项目导入 29](#_Toc2357)

[3.3.2 项目本地化修改 29](#_Toc21741)

[3.3.2 项目打包war 30](#_Toc22943)

[3.3.2 部署war包 30](#_Toc21034)

[3.4 系统实现 30](#_Toc23058)

[3.4.1 首页模块 31](#_Toc7536)

[3.4.2 搜索页面模块 32](#_Toc29440)

[3.4.3 邮件模块 32](#_Toc16516)

[3.4.4 前后端通讯设计与实现 33](#_Toc9679)

[3.5 本章小结 38](#_Toc8989)

**[第4章 识别系统的设计与实现 39](#_Toc8087)**

[4.1 识别系统简介 39](#_Toc6441)

[4.1 签名嵌入项目 39](#_Toc15075)

**[第5章 数据库的设计与部署 40](#_Toc22548)**

[5.1 数据库环境简介 40](#_Toc12367)

[5.1 数据库设计 40](#_Toc13158)

[5.2 数据库部署 42](#_Toc30007)

**[第6章 关键功能代码及说明 43](#_Toc17206)**

[6.1 调用百度AI接口 43](#_Toc218)

[6.2 集大通爬虫 44](#_Toc27452)

[6.3 使用Echats图表 46](#_Toc13213)

[6.3 服务端向客户端发送ws消息 47](#_Toc12180)

[6.4 客户端向服务端发送ws消息 50](#_Toc14671)

[6.5 Quartz任务定时器 52](#_Toc27755)

**[第7章 系统运行环境和软件安装 54](#_Toc18077)**

[7.1 系统运行环境 54](#_Toc28710)

[7.2 IntelliJ IDEA安装 54](#_Toc21272)

[7.3 Java安装 55](#_Toc13263)

[7.3 Tomcat安装 55](#_Toc15480)

[7.4 MySQL安装 56](#_Toc3009)

**[常见问题及解决方法 57](#_Toc25867)**

**[文档日志 59](#_Toc15087)**

# 第1章 绪论

## 1.1 系统研究背景

集大通是集美大学校内的微博平台，师生可以在该平台上发布生活动态或完成在线充值缴费的功能。但随着人数使用的增多，并且由于用户关注度集中在首页微博，如果充斥大量负面情绪，将极大影响师生的体验，降低集大通的微博质量，减少用户使用频次。同时，部分微博存在风险性和欺诈性，对于风险意识不足的大学生，极易上当受骗。目前，集大通的微博屏蔽依靠管理员手动删除，效率不高。因此，为管理员开发一套监测系统，把微博类型识别出来，同时分析用户舆情，可以极大提高微博舆情的处置效率。

## 1.2 主要开发内容

本文的阐述对象是“集大通网络爬虫与可视化平台”，将围绕网络爬虫的开发、舆情识别系统的设计和数据可视化平台而展开。由爬虫获取原始数据，并调用舆情识别API接口将分类后的结构化数据保存至数据库，最终由数据展示平台实时监测集大通数据，本文主要工作如下：

### 1.3.1 网络爬虫系统

集大通微博数据的获取。编写相应APP爬虫，获取集大通用户、微博、回复、图片等数据，并将数据结构化存入数据库。

### 1.3.2 识别系统

目前(2019.5.15)该平台舆情识别接口调用百度AI情绪识别接口(<http://ai.baidu.com/tech/nlp_apply/sentiment_classify>)。由于百度AI接口的适用对象多为外卖、美食等团购平台，所以在识别集大通微博情绪时有较大错误率。

该平台后续开发计划：利用现有集大通微博和回复数据，标注数据和划分训练集和测试集，构建情绪识别器，实现符合微博应用场景的情绪识别接口。

### 1.3.2 数据可视化平台

使用SSM框架作为网站开发框架，amazeui和echarts作为前端组件。websocket实现客户端和服务端的实时通讯。

# 第2章 网络爬虫的设计与实现

## 2.1 网络爬虫系统简介

图2-1是集大通的网络爬虫系统的流程图，展示整个系统的工作流程。程序根据集大通首页微博URL地址为入口，获取每条微博、回复和用户数据，下载数据获得的JSON格式数据，然后用JSON解析器解析，提取符合爬取要求的URL地址放入待爬取URL地址池中。微博和回复数据将调用百度API接口获得对应的情绪分类。JSON解析器解析将自动构建对应对象的实体，通过数据库驱动程序存入数据库中。

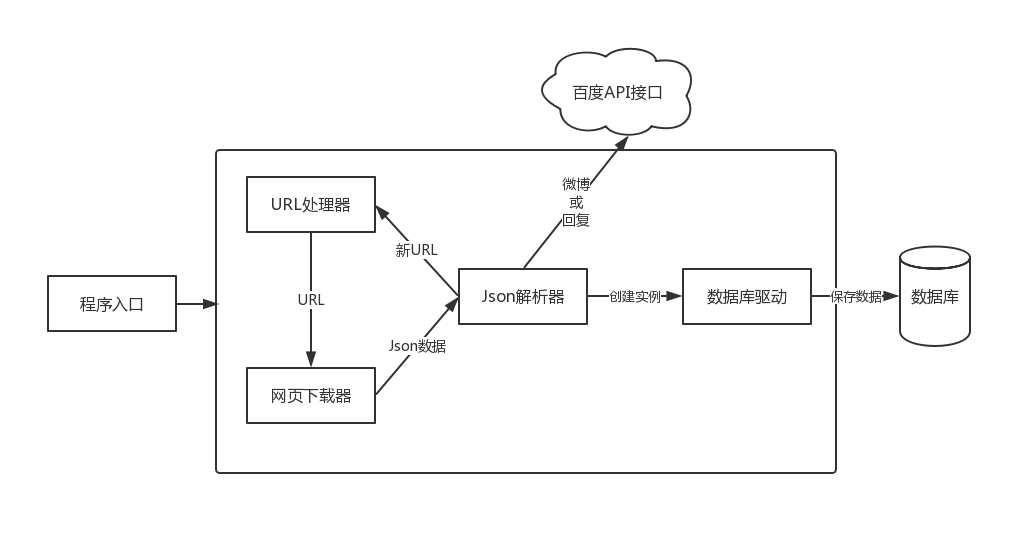


图2-1 网络爬虫工作流程图

## 2.2 项目结构

在源代码文件夹下的jdtFetch文件是网络爬虫系统的源代码。项目结构图见图2-2。

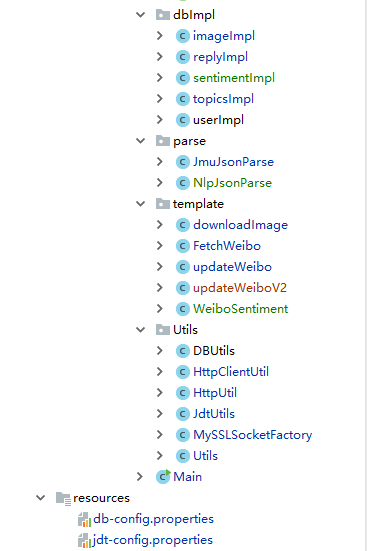
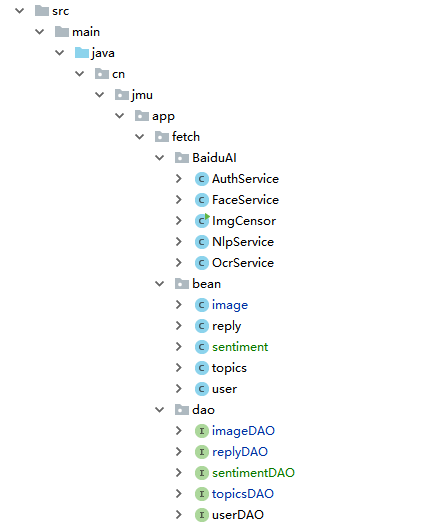


图2-2 jdtFetch项目结构图

本项目目录下包含以下文件：

1. .idea 本项目开发工具是IntelliJ IDEA，这个文件夹是开发工具自动生成的文件，可以忽视。
2. out 项目打包成jar时生成的输出目录，用于存放编译后的jar文件
3. src 项目源码
4. target 这个目录只要用于存放项目构建的输出文件，比如class文件以及打包后的包文件等。
5. jdtFetch.iml文件 IntelliJ IDEA自动生成的文件
6. pom.xml文件 本项目用Maven构建，这个文件是Maven的配置文件

以下是src/main文件夹详细内容：

src.main目录下的resources文件为系统配置文件

* jdt-config.properties 为集大通爬虫参数，包含登陆账号密码，集大通请求包中的token
* db-config.properties 为数据库配置

src.main.java目录下为爬虫代码，cn.jmu.app.featch为项目名，其包结构如下

* BaiduAI：调用百度AI的接口包
* AuthService.java 对应多种AI接口的token
* FaceService.java 处理人脸识别
* ImgCensor.java 识别图片是否为广告等
* NlpService.java 处理文字内容的情感倾向等
* OcrService.java 处理图片的文字识别等
* bean：DAO设计模式下的数据库实体类
* image.java 与第1章数据库设计一致，下同
* reply.java
* sentiment.java
* topics.java
* user.java
* dao：DAO接口
* imageDAO.java DAO抽象类，对应自定义的数据库增删改查，下同
* replyDAO.java
* sentimentDAO.java
* topicsDAO.java
* userDAO.java
* dbImpl：DAO实现类
* imageImpl.java 为DAO的实现类，下同
* replyImpl.java
* sentimentImpl.java
* topicsImpl.java
* userImpl.java
* parse：解析Json数据的方法
* JmuJsonParse.java 解析所有集大通的Json数据，便于存入数据库
* NlpJsonParse.java 解析百度AI返回的Json数据
* template：根据不同需求，爬虫爬取的模式
* downloadImage.java 下载image数据库保存的图片url
* FetchWeibo.java 根据集大通按条件查找好友的接口，爬取全集大通的微博
* updateWeibo.java 读取首页1000条微博，用于更新数据
* WeiboSentiment.java 分析每条微博的情感倾向
* updateWeiboV2.java 读取首页1000条微博，但只更新48小时的内容
* Utils：工具集
* DBUtils.java DAO模式的数据库方法
* HttpClientUtil.java 爬虫的http请求方法
* HttpUtil.java 用java自带的包，构造的http请求方法
* JdtUtils.java 保存集大通的接口url地址
* MySSLSocketFactory.java 解决https请求失败的问题
* Utils.java Base64编码解码等操作
* Main.java：主类

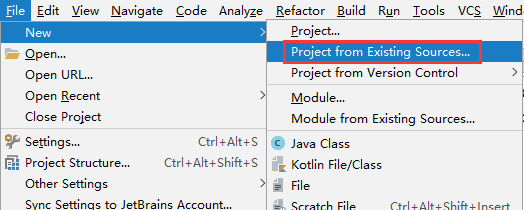
## 2.3 项目部署

项目部署包含项目运行环境的部署，代码本地化运行，jar定时任务的部署等。

### 2.3.1 IDEA导入

本项目使用IDEA开发环境，因此将详细介绍IDEA导入该项目的过程。

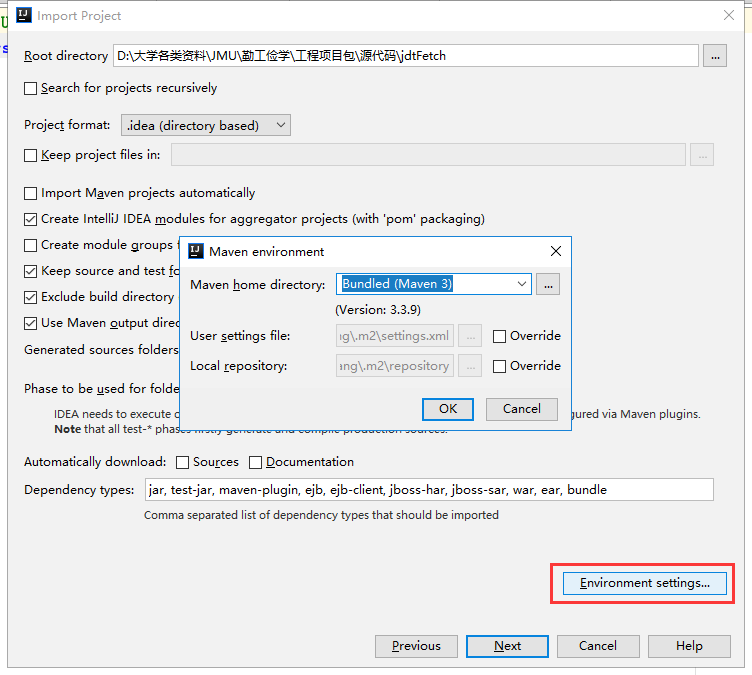
1.首先选择File->New->Project from Existing Sources自己将要导入的Web项目。



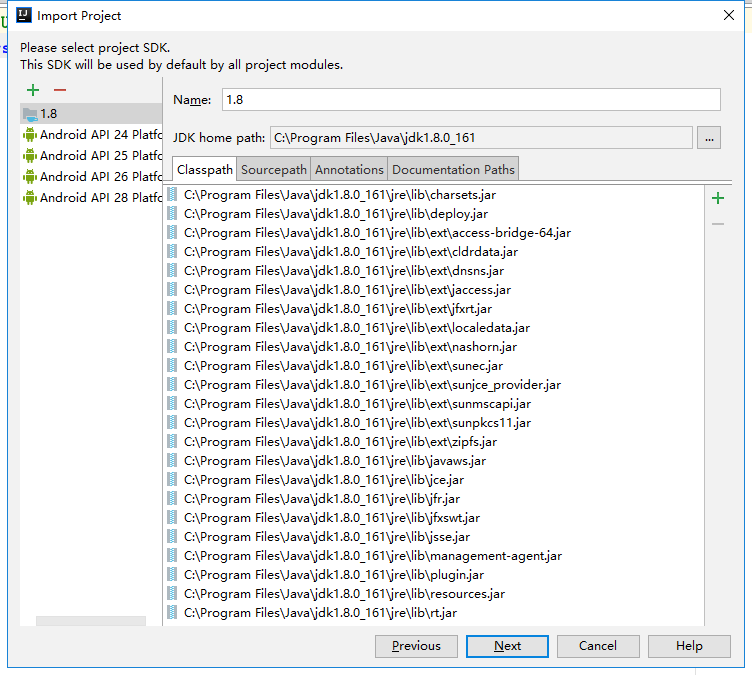
2.选择将项目以Maven的形式导入，然后选择Next

|  |  |
| --- | --- |
|  |  |

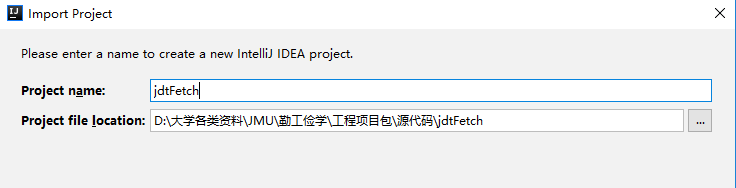
3.在这一步有一些配置选项，一般来说使用默认就可以，本项目使用Maven3构建。



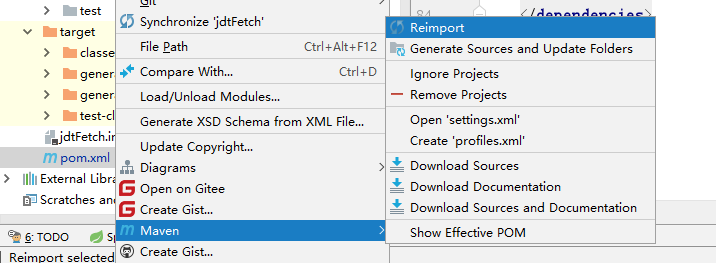
4.Next以后，在这一步还有JDK的配置，直接选择安装过的JDK就行，本项目使用JDK1.8。



5.填写项目名称jdfFetch。



6.右键pom.xml-->Maven-->Reimport，导入第三方包。



至此，在IDEA开发环境中，项目可以正常运行。

### 2.3.2 Eclipse导入

参见博文：<https://blog.csdn.net/w12345_ww/article/details/52094756>

### 2.3.3 项目本地化修改

项目要在本地能正常运行，需要配置项目运行参数和部署数据库。

项目参数均在resources目录下，修改流程如下：

1. 修改db-config.properties文件，配置本地的数据库账号和密码
2. 修改jdt-config.properties，配置你的集大通账号和密码

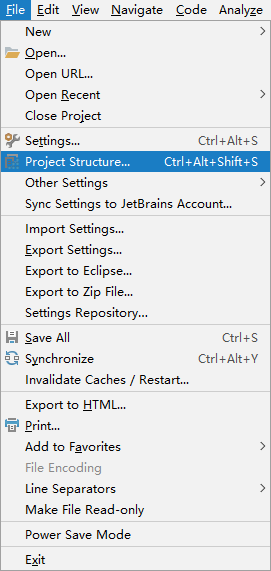
部署数据库见【第5章的5.3数据库部署】

### 2.3.4 项目打包jar

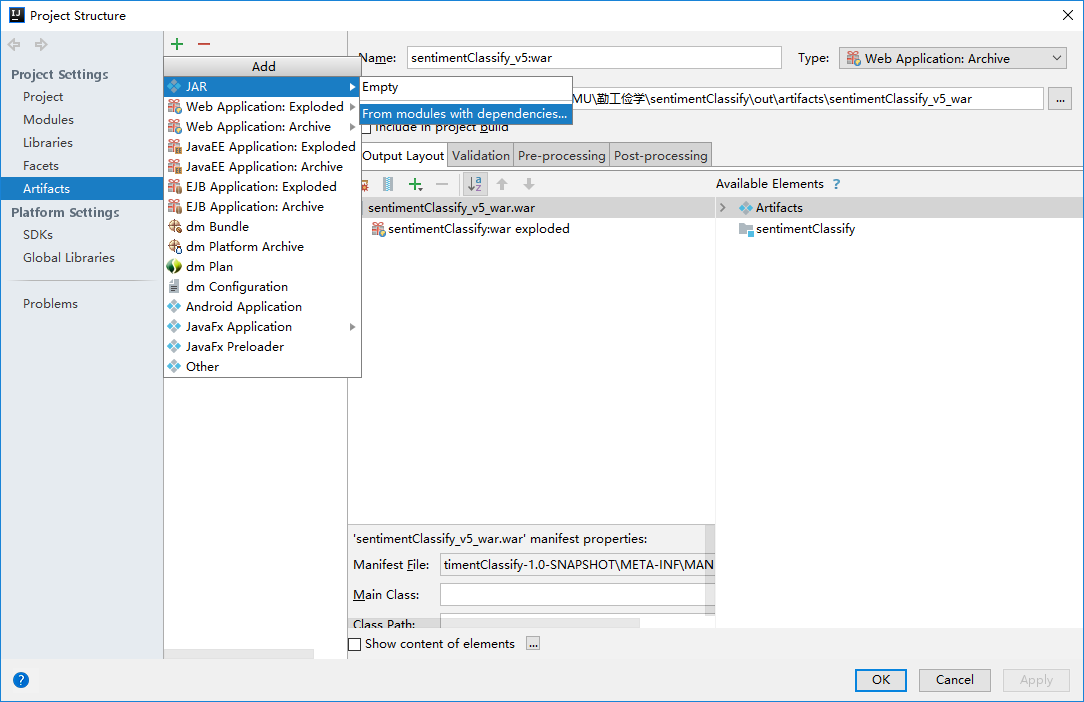
jar包是Java项目程序可执行文件，本项目打包成jar包目的是实现定时爬取的功能，这个功能由任务计划配置和爬出脚本组成。因此需要配合计划任务配置，创建一个爬出jar脚本。

演示过程使用IDEA开发环境，创建jar脚本步骤如下：

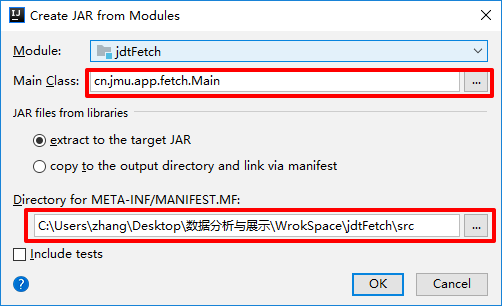
1. 选中Java项目工程名称，在菜单中选择 File->project structure... (快捷键Ctrl+Alt+Shift+S)。



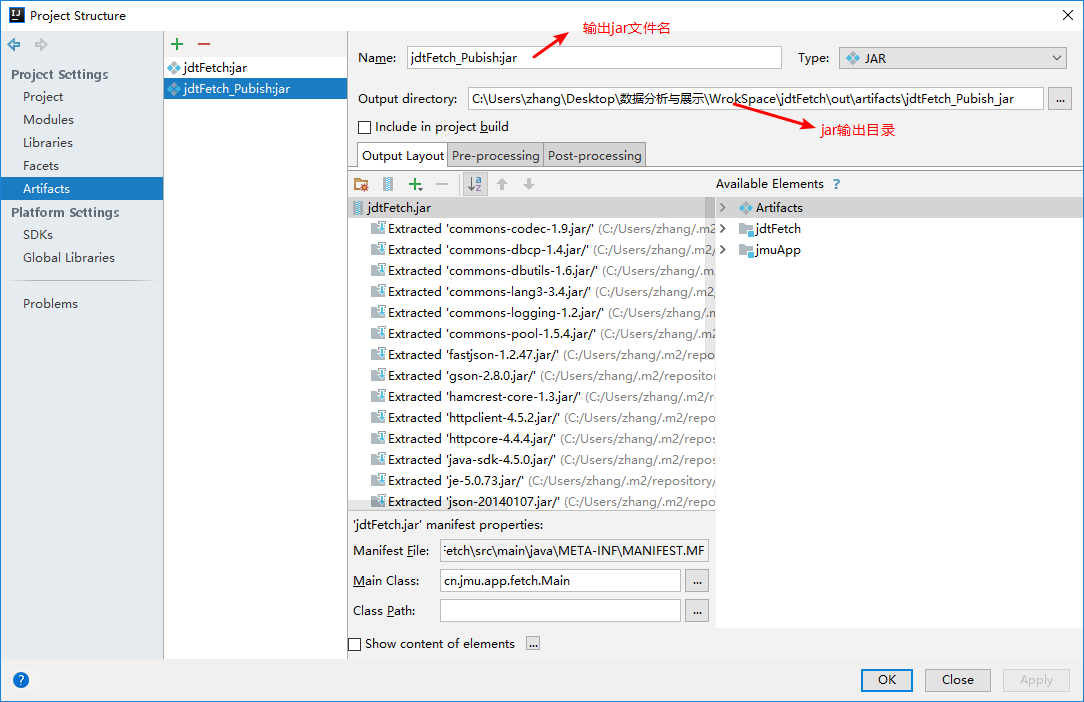
1. 在弹出的窗口中左侧选中"Artifacts"，点击"+"选择jar，然后选择"from modules with dependencies"。



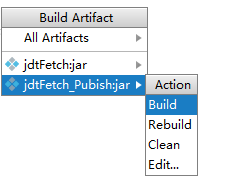
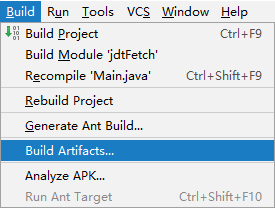
1. 在配置窗口中配置“Main Class”。配置“Directory for META-INF/MAINFEST.MF”，此项配置的缺省值是：D:\xxx\jdtFetch\src\main\java，需要改成：D:\xxx\jdtFetch\src。如果不这样修改，打成的jar包里没有包含META-INF/MAINFEST.MF文件，这个应该是个IDEA的BUG（参考：http://stackoverflow.com/questions/15724091/how-to-run-a-jar-file-created-using-intellij-12）

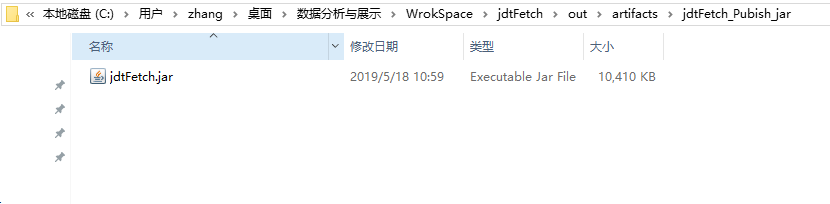


1. 修改jar文件名，文件名随意，只要不和以前的jar重名。



1. 完成后，点击OK，Apply等按钮，回到IDEA的主菜单，选择“Build - Build Artifacts”下的“Build”或者“Rebuild”即可生成最终的可运行的jar，在D:\xxx\jdtFetch\out\artifacts\jdtFetch\_Publish\_jar下面找到生成的目标jar，可以看到META-INF/MAINFEST.MF文件被正确包含，内容也正确。





1. 在windows系统下，cmd窗口运行jdtFetch.jar,在数据库开启的情况下，可以正确运行

java -jar jdtFetch.jar

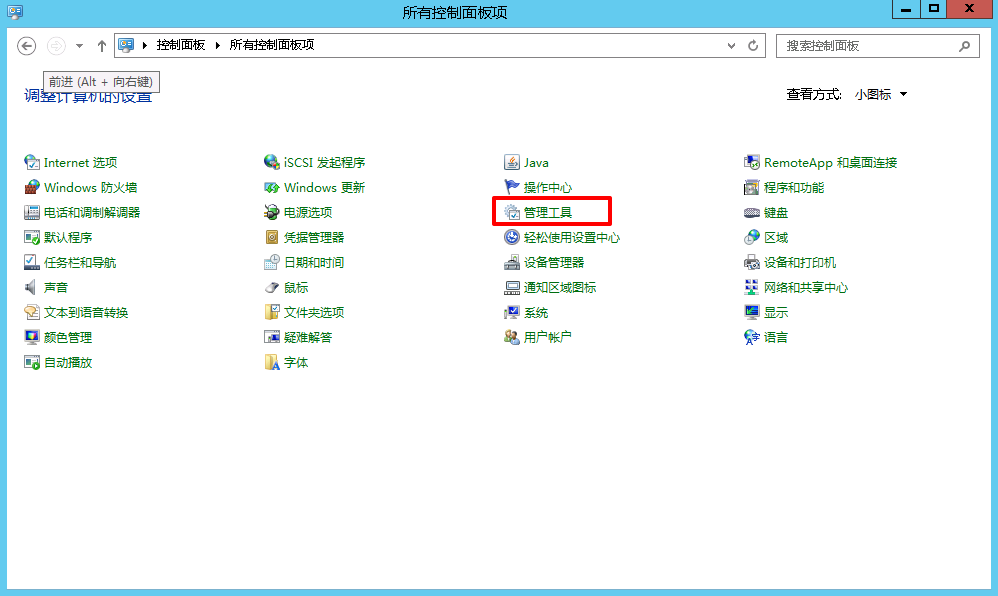
### 2.3.5 部署jar定时脚本

部署方式根据操作系统有所区别，本节将以Windows2008和Ubuntu演示部署定时脚本。

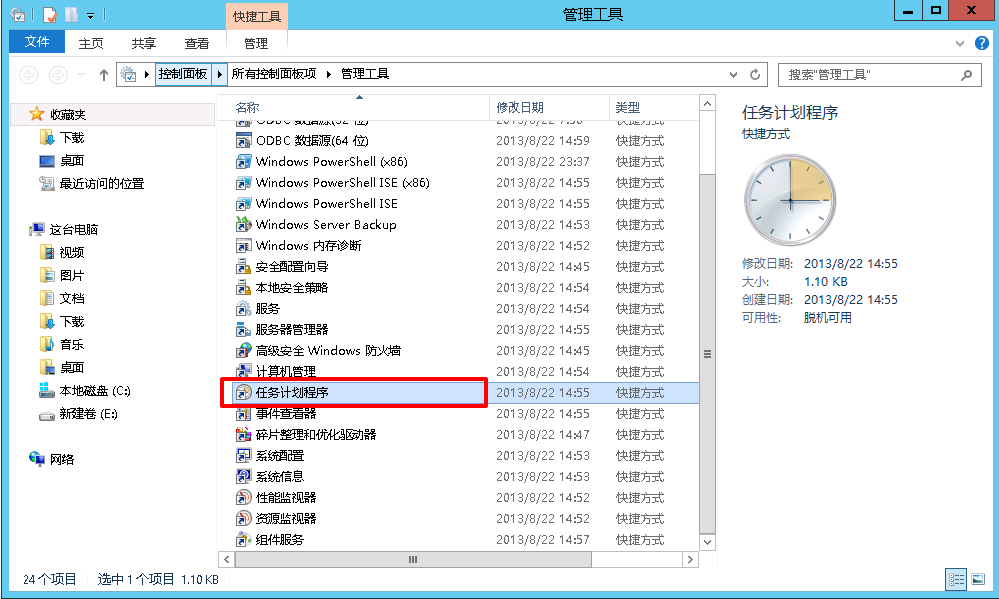
部署jar定时脚本需求是每隔1分钟，爬取一次集大通，保存数据至数据库。

**Windows2008**部署方式如下：

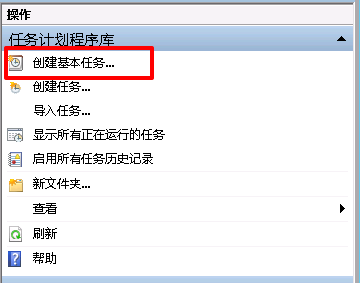
1. 打开“控制面板”，找到“管理工具”模块



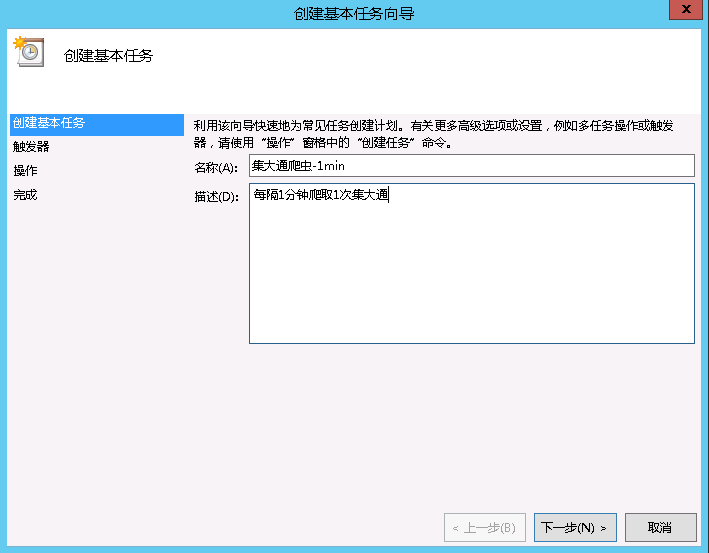
1. 找到“任务计划程序”



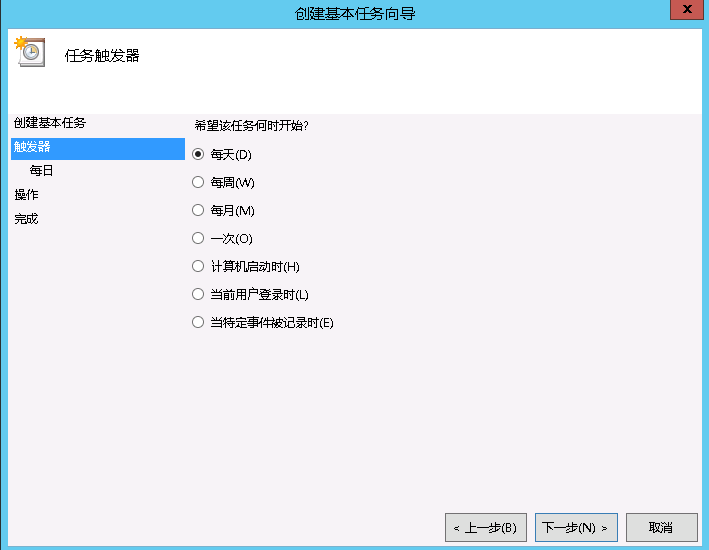
1. 点击“创建基本任务”



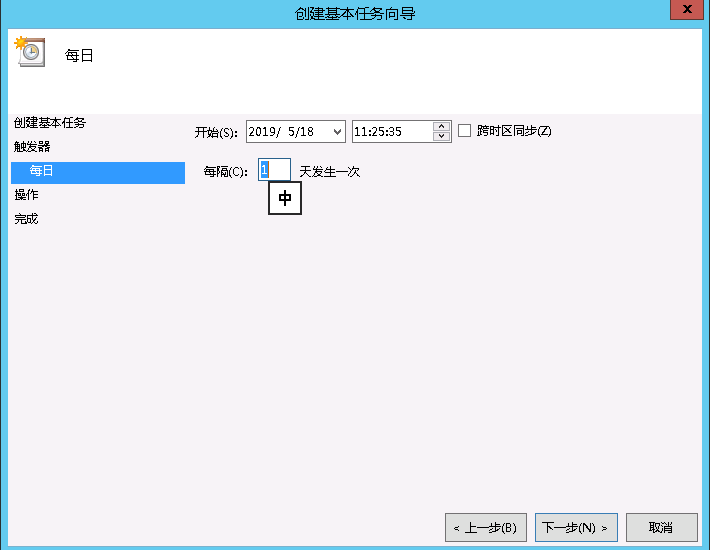
1. 填写名称和描述，点击下一步



1. 先选择每天，点击下一步。在创建任务成功后，修改为1分钟

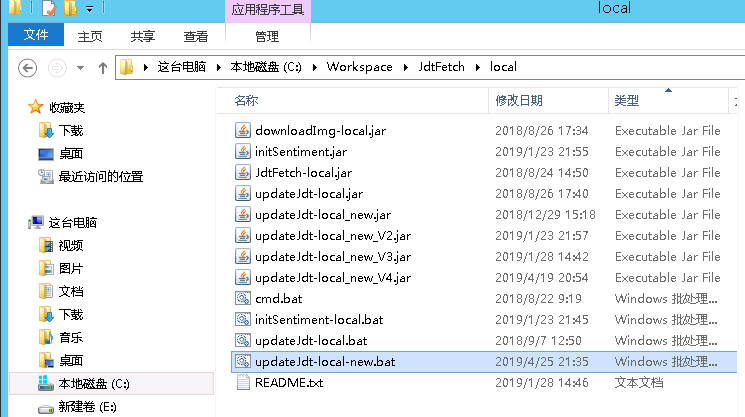


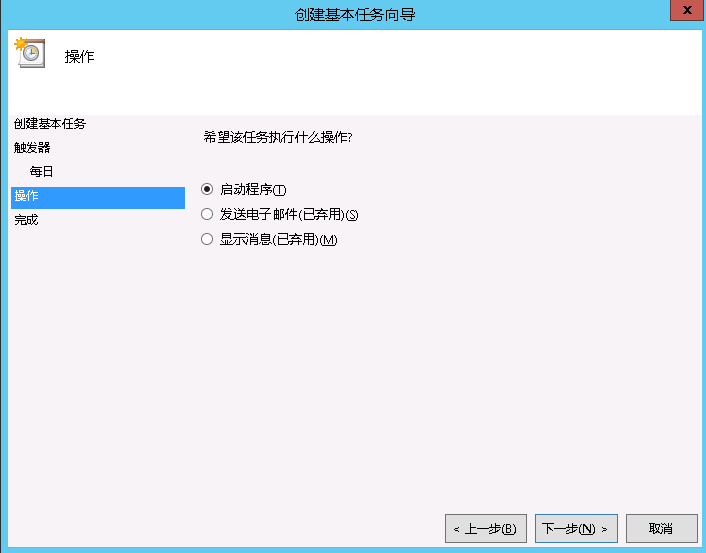
1. 每隔1天，点击下一步。

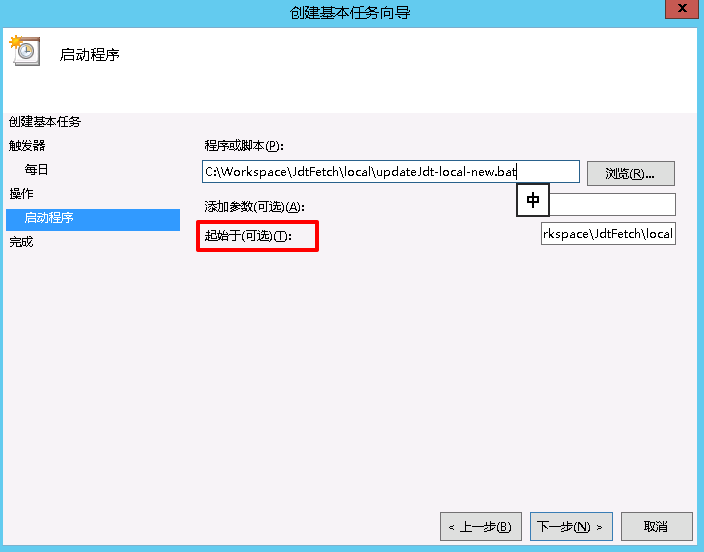


1. 配置操作，点击下一步进入详情配置。这里建议编写一个bat脚本，用于执行jar文件。bat脚本内容如下，注意jar文件需要在bat脚本目录下。在操作详情配置页面，选择创建的脚本，起始于填写脚本的文件目录。

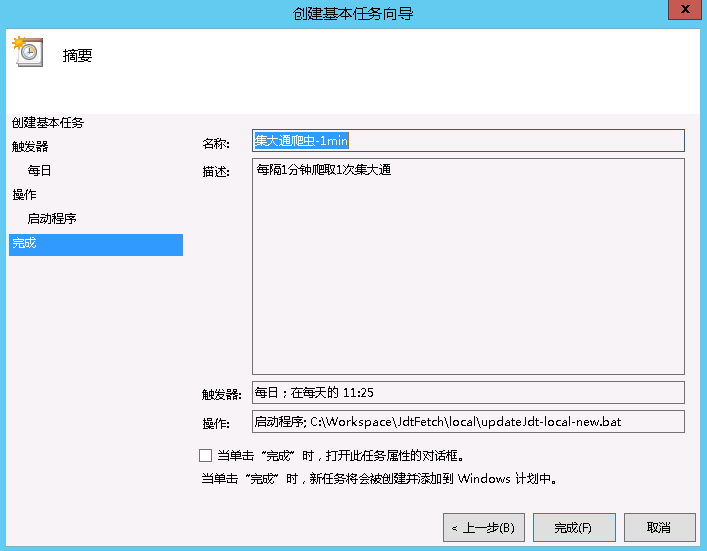
java -jar jdtFetch.jar



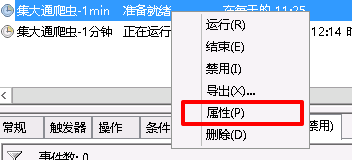




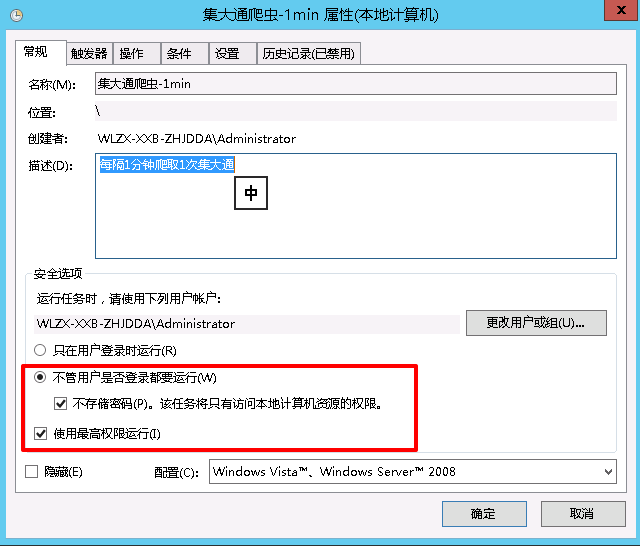
1. 点击完成



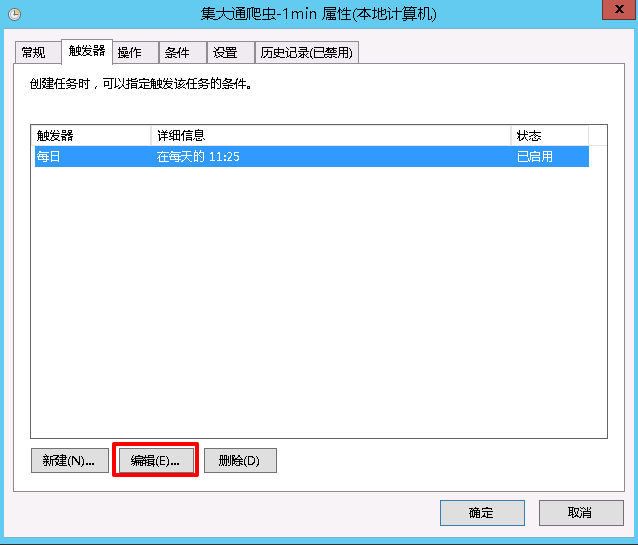
1. 右键刚刚创建的任务，点击属性

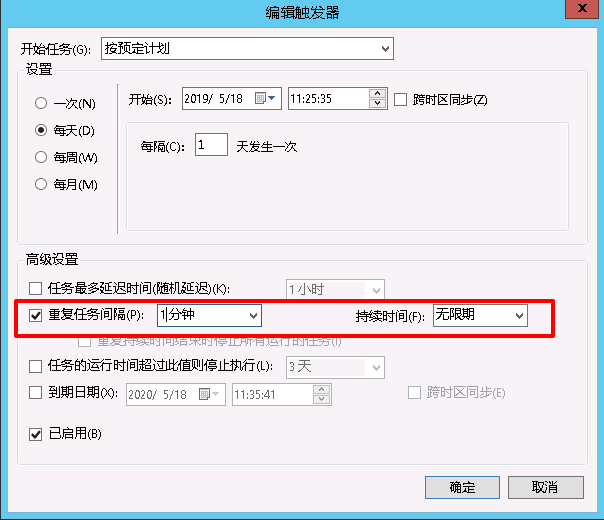


1. 属性的常规页面，需要如下配置，目的是保证退出远程服务器时也可以正常运行脚本

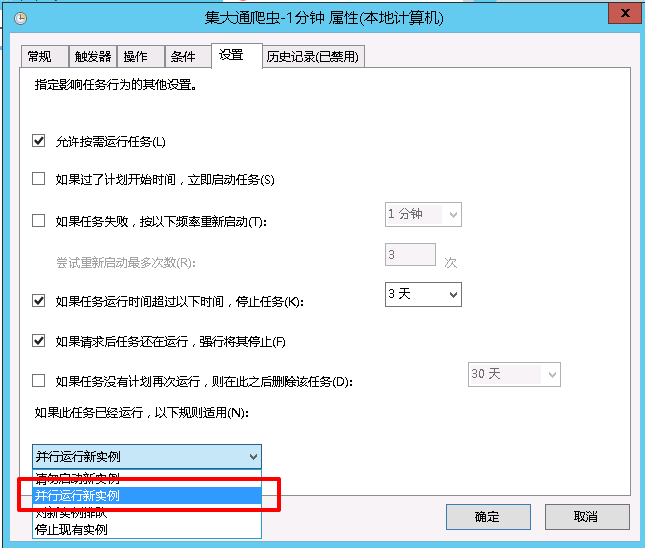


1. 进入触发器页面，点击编辑，编辑内容如下图





1. 进入设置页面，编辑内容如下图



**Ubuntu(Linux)**部署方式如下：

1. 安装Java环境和crontab任务程序
2. 使用crontab运行脚本，输入命令“crontab -e”，新增如下语句。

\*/1 \* \* \* \* /home/java/jdk1.8/bin/java -jar /home/chasen/jdtFetch.jar >> /home/chasen/jdtFetch.log

其中“\*/1 \* \* \* \*”表示每隔1分钟运行脚本，“/home/java/jdk1.8/bin/java”表示java的运行目录，“-jar /home/chasen/jdtFetch.jar”表示运行jar包，

“>> /home/chasen/jdtFetch.log”表示将jar包运行日志保存到本地。

1. 文件夹目录结构如下，runjdtFetch.sh在此次部署中没有作用，可以无视



## 2.4 项目添加新模块

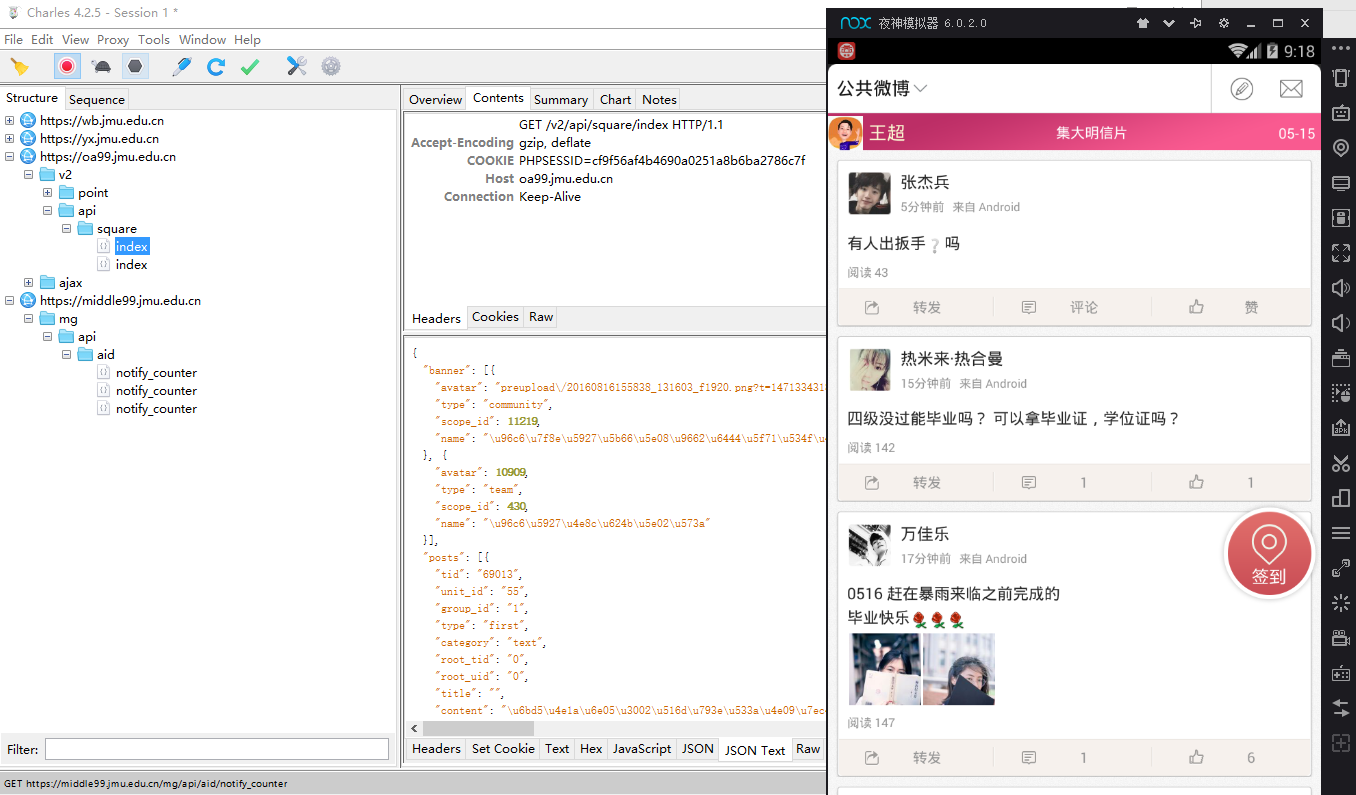
本项目目前具有集大通全微博遍历和首页微博更新。对于未来新的需求和功能，可以参见本小节的开发过程，本小节将假设需要开发“集大通抽奖信息”的爬虫，演示从数据抓取分析到项目代码更新的全过程。

### 2.4.1 集大通数据分析

由于集大通是一个移动端应用，因此需要使用安卓模拟器和抓包软件分析数据交互过程，推荐使用夜神安卓模拟器和Charles网络流量分析器。这两款应用常被用于移动端App网络爬虫的前期分析。

夜神模拟器和Charles需要进行配置，配置过程参见：<https://it.baiked.com/dev/4031.html>。配置完成后，去集大通官网(<http://99.jmu.edu.cn/>)下载集大通apk文件，拖动apk文件至夜神模拟器，即可安装集大通。

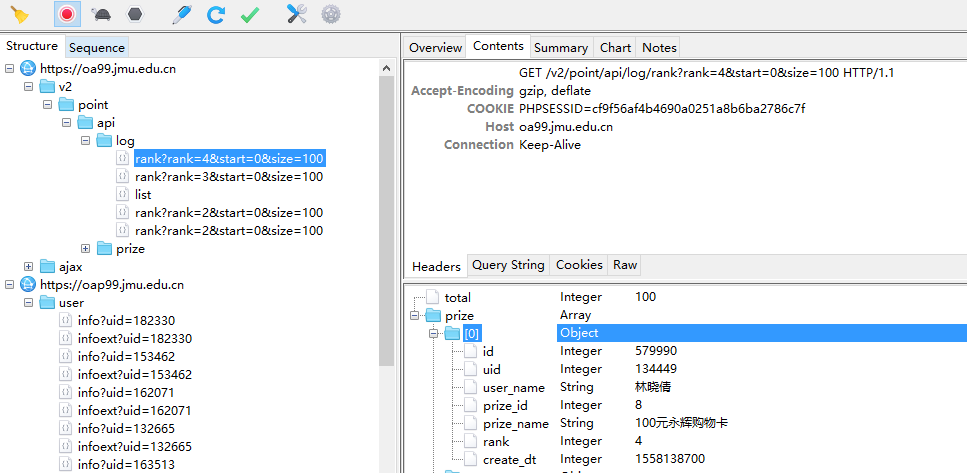
打开夜神模拟器中的集大通和Charles，如果配置成功，可以看到Charles获取到集大通的数据，如下图



安卓应用多为JSON数据进行交互通信，因此本项目有编写对应的JSON解析器。点击微博首页顶部抽奖信息栏，配合Charles观察抽奖信息的数据URL地址。

|  |  |
| --- | --- |
|  |  |

进入抽奖列表页面，点击更多。可以观察到Charles捕获到最近100条抽奖信息，如下图。



数据URL地址为：

https://oa99.jmu.edu.cn/v2/point/api/log/rank?rank=4&start=0&size=100

可以看到请求方式为GET，参数分别为rank、start、size，其中请求头必须包含COOKIE，即登陆凭证。返回JSON数据如下。

{

"total": 100,

"prize": [{

"id": 579990,

"uid": 134449,

"user\_name": "\u6797\u6653\u5029",

"prize\_id": 8,

"prize\_name": "100\u5143\u6c38\u8f89\u8d2d\u7269\u5361",

"rank": 4,

"create\_dt": 1558138700

}, {

"id": 578505,

"uid": 153955,

"user\_name": "\u5218\u4f73\u5cfb",

"prize\_id": 44,

"prize\_name": "\u5145\u7535\u5b9d",

"rank": 4,

"create\_dt": 1557702271

}

}]

}

### 2.4.2 新模块的设计

有了2.4.1节数据包的分析，首先要设计数据库结构。本节给出参考数据库表PrizeInfo。

|  |  |  |
| --- | --- | --- |
| 字段 | 类型（长度） | 备注 |
| pid | varchar(7) | 抽奖记录ID |
| uid | varchar(7) | 用户ID |
| uName | varchar(50) | 用户名称 |
| prizeId | varchar(7) | 奖品ID |
| prizeName | varchar(20) | 奖品名称 |
| rank | varchar(2) | 奖品类别（一等奖、二等奖...） |
| dataTime | varchar(10) | 时间戳 |

然后在**bean**目录下创建对应实体prizeInfo类，由于本次需求只涉及数据插入，因此在**dao**目录下创建prizeInfoDAO接口类，并在**dbImpl**目录下的prizeInfoImpl类编写addPrizeInfo(prizeInfo pi)实现方法。至此，数据存储这层的代码设计完成。

接下来，在**Utils**目录下的JdtUtils类添加抽奖信息的URL。在**template**目录下创建对应爬虫FetchPrize类，涉及JSON解析的代码要写在**parse**目录下的JmuJsonParse类下。

### 2.4.3 新模块的实现

本节将着重阐述FetchPrize类和JmuJsonParse类的实现代码，其余代码都可以机械地完成。

1. FetchPrize类是具体实现抽奖信息获取的爬虫类，负责调用数据层和解析层的代码，也是在Main主类中调用的类方法。代码如下：

**public class** FetchPrize {  
 *// 抽奖信息写入数据库* **public static void** doInitPrize() **throws** IOException, SQLException {  
 HttpClientUtil hcu = **new** HttpClientUtil();  
 prizeInfoImpl pimpl = **new** prizeInfoImpl();  
 JmuJsonParse jjp = **new** JmuJsonParse();  
  
 **for**(**int** i=1;i<=6;i++) {  
 *// 抽奖URL* String url = JdtUtils.*getPrizeUrl*(i);  
 *// Json数据* String prizeRes = hcu.doJmuGet(url, JdtUtils.*COOKIE*);  
 *// 解析数据* List<prizeInfo> prizeList = jjp.prizeParse(prizeRes);  
 **for** (prizeInfo pi : prizeList) {  
 System.***out***.println(pi.getpId());  
 pimpl.addPrizeInfo(pi);  
 }  
 }  
 }  
}

其中，调用HttpClientUtil的doJmuGet通用Get请求方法，实现JSON数据的获取，调用JmuJsonParse完成JSON数据解析至prizeInfo实例化的对象。最后调用prizeInfoImpl类完成数据存储。

1. JmuJsonParse类实现JSON解析，具体代码如下：

*// 集大通抽奖Json数据解析***public** List<prizeInfo> prizeParse(String result){  
 List<prizeInfo> prizeList = **new** ArrayList<prizeInfo>();  
 prizeInfo bean;  
 JSONObject obj = JSON.*parseObject*(result);  
 **if**(obj.getJSONArray(**"prize"**) != **null**) {  
 JSONArray prizeArray = obj.getJSONArray(**"prize"**);  
 **for** (**int** i=0; i<prizeArray.size(); i++){  
 JSONObject perObj = prizeArray.getJSONObject(i);  
 bean = **new** prizeInfo();  
 String pId = perObj.getString(**"id"**);  
 String uId = perObj.getString(**"uid"**);  
 String uName = perObj.getString(**"user\_name"**);  
 String prizeId = perObj.getString(**"prize\_id"**);  
 String prizeName = perObj.getString(**"prize\_name"**);  
 String rank = perObj.getString(**"rank"**);  
 String dateTime = perObj.getString(**"create\_dt"**);  
  
 bean.setpId(pId);  
 bean.setuName(uName);  
 bean.setuId(uId);  
 bean.setPrizeId(prizeId);  
 bean.setPrizeName(prizeName);  
 bean.setRank(rank);  
 bean.setDateTime(dateTime);  
  
 prizeList.add(bean);  
 }  
 }  
 **return** prizeList;  
}

## 2.5 本章小结

本章围绕集大通网络爬虫的设计与实现，重点介绍了项目部署过程和对于新模块开发的流程。

# 第3章 数据可视化平台的设计与实现

## 3.1 可视化平台简介

本项目可视化平台可以实时监测高危用户，查看用户历史消息，舆情舆情，每周或上周报表等功能。

为使可视化平台的处理逻辑更清晰，提高平台的可维护性和代码的复用性，当需求发生变化时，可以方便增加业务和模块，设计如图3-1所示的分层架构。平台分为显示层、控制层、业务层和数据层四层，各层各司其职，具有较强的内聚性，降低了层与层之间的耦合程度。

平台每个功能层的详情如下：

1. 显示层：数据交互层，系统展示实时数据，并处理后台定时socket消息。
2. 控制层：URLs路由功能，将用户请求跳转至对应页面。
3. 业务层：该层负责系统整体逻辑，包含服务端定时发送数据、处理客户端消息、异常处理等。
4. 数据层：为业务层提供数据或数据存储。本系统采用MySQL作为数据库。

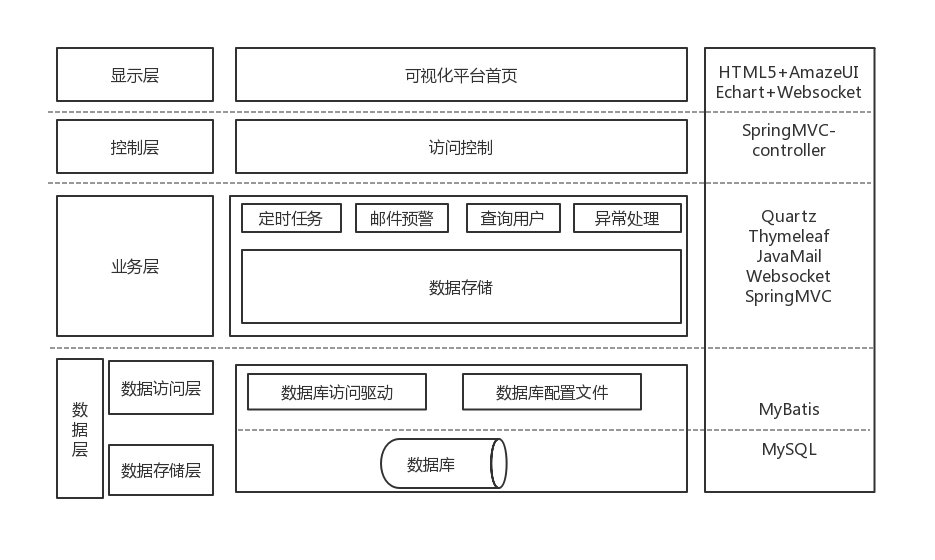


图3-1 可视化平台架构

## 3.2 项目结构

在源代码文件夹下的sentimentClassify文件是数据可视化平台的源代码。项目结构图见图3-2，图3-3。图3-2展示服务端结构，图3-3展示客户端结构。

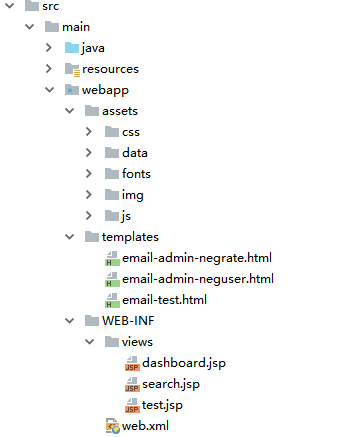
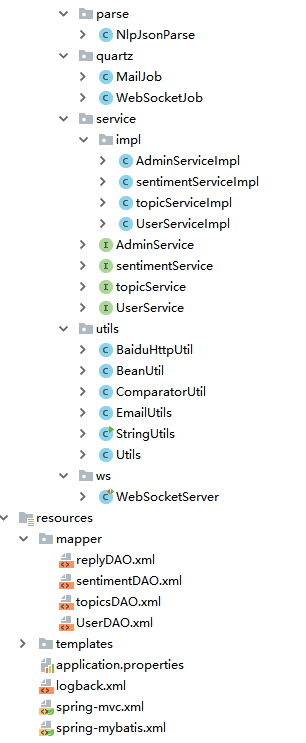
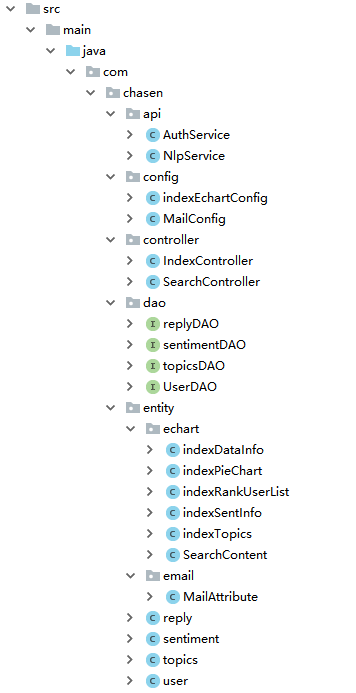


图3-2 sentimentClassify项目结构图

### 3.2.1 服务端代码结构

服务端代码主要包括src.main.java和src.main.resources目录下文件：

src.main目录下的**resources**文件为系统配置文件

* logback.xml 控制台日志输出方式配置
* application.properties 应用配置文件
* spring-mvc.xml Spring配置文件
* spring-mybatis.xml Mybatis配置文件
* template：邮件html模板
  + email-admin-negrate.html 负面预警html模板
  + email-admin-neguser.html 高危用户html模板
  + email-test.html 测试模板
* mapper：数据库对象映射文件夹
  + replyDAO.xml replyDAO的映射
  + sentimentDAO.xml sentimentDAO的映射
  + topicsDAO.xml topicsDAO的映射

src.main.**java**目录下为服务端业务层和控制层的代码，com.chasen为项目名，其结构如下

* api：调用百度AI的接口包
  + AuthService.java 对应多种AI接口的token
  + NlpService.java 自然语言接口，处理文本情绪识别等
* config：内存型配置
  + indexEchartConfig.java 前端图表模块的配置
  + MailConfig.java 邮件时间间隔配置（已弃用）
* controller：控制层代码，负责处理URL地址
  + IndexController.java 负责首页的地址跳转
  + SearchController.java 负责处理搜索页面Ajax请求
* dao：数据库DAO
  + replyDAO.java reply表DAO
  + sentimentDAO.java sentiment表DAO
  + topicsDAO topics表DAO
  + UserDAO user表DAO
* entity：数据库表的实体
  + reply.java 对应reply表
  + sentiment.java 对应sentiment表
  + topics.java 对应topics表
  + user.java 对应user表
  + echart：显示层数据对象
    - indexDataInfo.java 首页监测详情信息，如微博总数，回复总数
    - indexEchartConfig.java echart图标的配置信息，诸如配置高危用户显示时间段
    - indexPieChart.java 首页2小时舆情饼图
    - indexRankUserList.java 首页高危用户对象
    - indexSentInfo.java 统计一段时间内正负情绪的数量
    - indexTopics .java 首页最近微博的对象
    - SearchContent.java 搜索页面显示用户历史记录的对象
  + email：显示层数据对象
    - MailAttribute.java 邮件模板中的对象实体
* parse：解析JSON格式的数据
  + NlpJsonParse.java 处理百度AI返回的JSON数据
* quartz：定时任务
  + WebSocketJob.java 配合spring-mvc.xml的Quartz任务,每隔1分钟向客户端发送一次数据
  + MailJob.java 邮件定时任务，监测负面率和高危用户
* service：DAO接口的逻辑整合
  + sentimentService.java 处理情绪识别的逻辑功能接口
  + topicService.java 处理微博的逻辑功能接口
  + UserService.java 处理用户表的接口
  + AdminService.java 处理管理员操作的接口，如邮件任务
  + impl：接口实现
    - sentimentServiceImpl.java sentimentService接口的具体实现
    - topicServiceImpl.java topicService接口的具体实现
    - AdminServiceImpl.java AdminService接口的实现
    - UserServiceImpl.java UserService接口的实现
* utils：工具包
  + BaiduHttpUtil.java 百度AI提供的http请求方法
  + BeanUtil.java 解决Spring无法注入对象的问题
  + ComparatorUtil.java 处理对象大小比较，这里只写了首页的最近微博根据时间排序
  + Utils.java 工具类
* ws：实现WebSocket
  + WebSocketServer.java 服务端WebScoket的具体实现代码

### 3.2.1 客户端代码结构

服务端代码主要包含在src.main.**webapp**目录下：

* asset：网页静态资源
* templates:邮件html模板
* WEB-INF：web配置和前端Html代码
  + views：html代码
    - dashboard.jsp 首页页面
    - search.jsp 搜索页面
    - test.jsp 测试页面
  + web.xml：配置文件

## 3.3 项目部署

项目部署包含项目导入，项目本地化修改，项目打包和部署。

### 3.3.1 项目导入

本可视化WEB项目同样使用Maven构建，根据不同的开发环境，参见【2.3.1 IDEA导入】和【2.3.2 Eclipse导入】

### 3.3.2 项目本地化修改

项目要在本地能正常运行，需要配置项目的数据库、邮件发送方和接收方。

项目参数均在resources目录下，只需要修改application.properties中对应的参数。其中邮箱授权码非邮箱密码，需要手动开通，参加以下网易和QQ邮箱授权码开通方式。

QQ邮箱授权码：

<https://service.mail.qq.com/cgi-bin/help?subtype=1&&id=28&&no=1001256>

网易邮箱授权码：

<http://help.163.com/14/0923/22/A6S1FMJD00754KNP.html>

如果使用网易企业邮箱，如201972000001@jmu.edu.cn，邮件的服务器与普通邮箱不同，需要通过网易官网查询对应的服务器地址和端口，查询地址如下。配置文件默认使用符合集美大学邮箱的配置。

网易企业邮箱配置查询：

<https://qiye.163.com/help/client-profile.html>

### 3.3.2 项目打包war

war包(Web application Archive)，与jar基本相同，但它通常表示这是一个Java的Web应用程序的包，tomcat这种Servlet容器会认出war包并自动部署。

用IDEA打包war包比较简单，参见博文：

<https://blog.csdn.net/qq_1290259791/article/details/78619353>

用Eclipse打包war包也很方便，参见博文：

<https://blog.csdn.net/gaohuanjie/article/details/52354367>

### 3.3.2 部署war包

将打包好的war包放入Tomcat的webapp目录下即可。然后需要去项目名，否则websocket会报错。Tomcat版本必须大于8.5。

去项目名步骤如下：

在tomcat\conf目录下找到server.xml，在<Host></Host>配置里面添加一行配置

<Context docBase="D:\apache-tomcat-8.0.36\webapps\sentimentClassify" path="/" reloadable="true" />

docBase是项目的物理路径（项目不必非要放在tomcat下面，可以放在磁盘的任意位置，只要此处物理路径配置正确，tomcat能找到项目位置就可以），path是虚拟路径。将path设置成/便可不需要项目名称便可访问项目。

<Host></Host>里面可以设置多个项目，每个项目一行Context设置。

## 3.4 系统实现

本系统展示首页如图3-3。每一个图标对应一个模块，系统可以根据具体需求改变对应模块或新增模块，因此本系统代码具有较高的复用性。

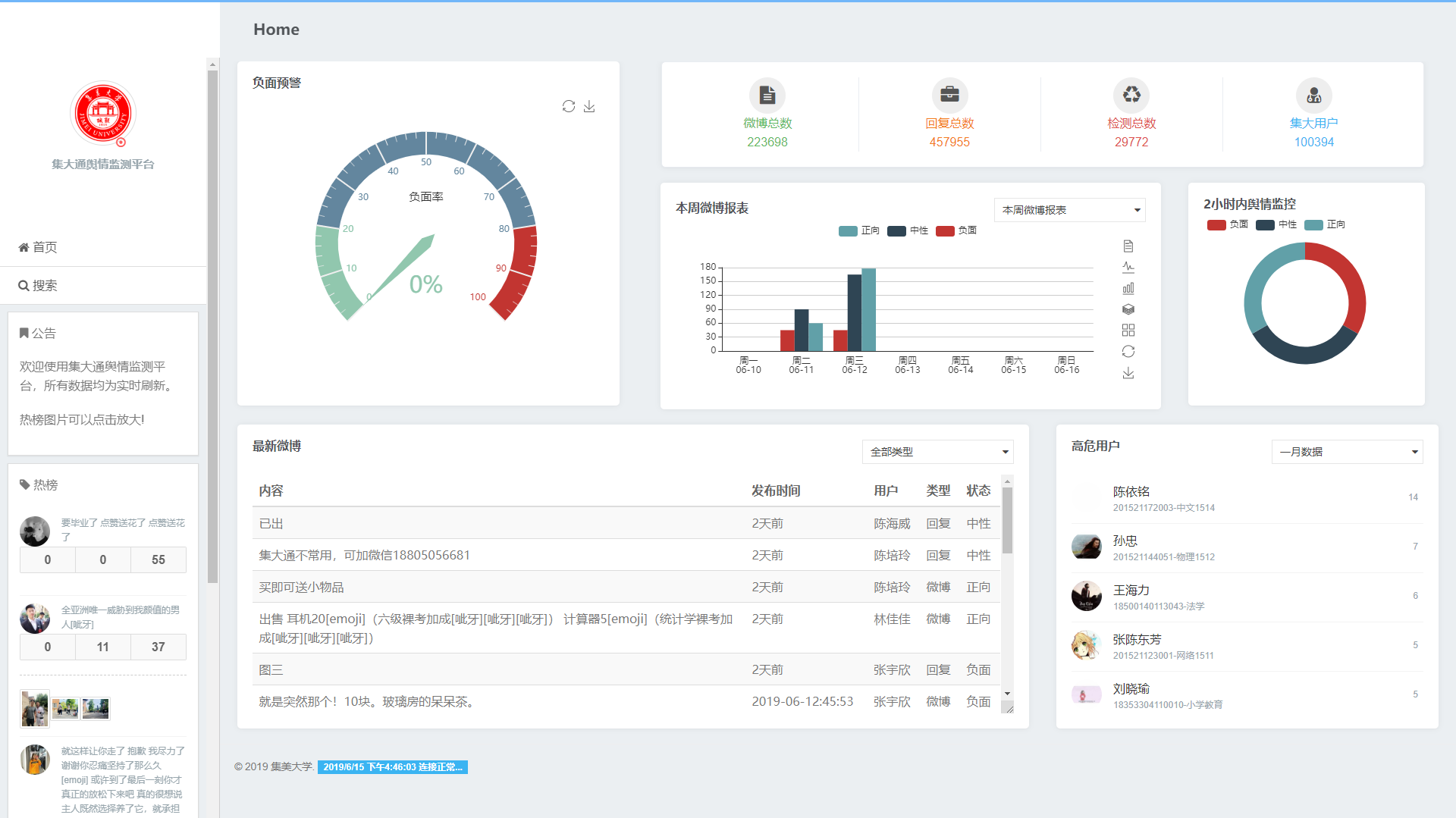


图3-3 监测平台首页

### 3.4.1 首页模块

首页与服务端通讯方式采用websocket，首页的模块可以向服务端发送消息获取数据，服务端会根据Quartz定时任务发送消息。首页模块详情介绍见下表。

|  |  |  |
| --- | --- | --- |
| 功能 | 功能描述 | 备注 |
| 热榜 | 显示最热门的5条微博，每条微博显示用户头像，微博内容，微博图片和微博转发数、回复数和点赞数；  计算的时间范围是两天（48小时）内的微博； | 热榜权重=0.2\*回复数+0.4\*转发数+0.4\*点赞数+0.001\*浏览数 |
| 负面预警 | 自动计算显示1小时的负面率，可下载保存报表图片 |  |
| 数据统计 | 显示微博数、回复数、检查数、集大用户数 | 为保证及时响应，数据采用不可靠查询，即数据不准确 |
| 舆情类型监控（小时） | 显示2小时舆情监控类型分布：正面、中性、负面 |  |
| 舆情类型监控（周） | 显示本周，上周舆情监控类型分布：正面、中性、负面，显示为报表，允许报表样式切换，可下载保存报表图片 |  |
| 最新微博 | 显示近的20条微博与20条回复内容：内容、时间、用户、类型、状态（可筛选显示微博与回复的类型，全部，负面） |  |
| 高危用户 | 显示最前的5条负面微博或回复数量用户，可点击查看高危用户的历史记录（可筛选显示高危用户的周期，一周，二周，三周，一月，完整 | 高危用户显示的数字是在设定的时间范围内，发表的负面微博或回复数量 |
| 连接状态 | 显示连接状态与获取数据的连接时间 |  |

### 3.4.2 搜索页面模块

搜索页面包含用户查询和用户记录两个模块，如图3-4所示。因为根据姓名查询可能存在多个用户。点击查看按钮即可显示该用户的历史发言记录，记录包含微博和回复。模块详情介绍如下表。

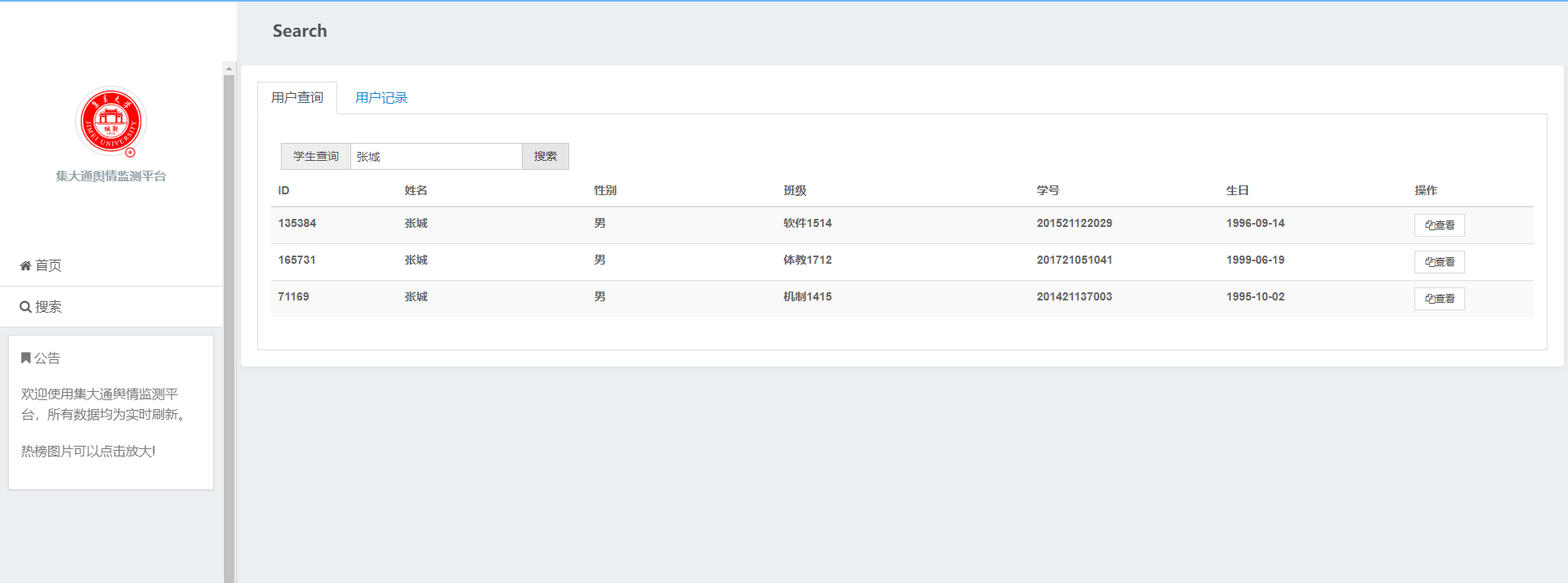


图3-4 监测平台搜索页面

|  |  |  |
| --- | --- | --- |
| 功能 | 功能描述 | 备注 |
| 用户查询 | 输入用户名，可以显示集大通用户信息。  点击查询按钮，将跳转到用户记录模块，显示该用户的历史记录。 | ID表示集大通ID，非用户学号。  获取数据方式采用Ajax请求。 |
| 用户记录 | 必须通过用户查询模块后，才能触发用户记录。 | 转发数、回复数和点赞数对于回复类型都显示0。  获取数据方式采用Ajax请求。 |

### 3.4.3 邮件模块

邮件模块同样采用Quartz定时器，每个一段时间查看负面率和高危用户。邮件模块采用html模板和邮件对象结合的方式发生，效果如图3-5。邮件模块包含监测负面率和高危用户，每成功发生一封邮件，将设定一段冷却时间，以防频繁发送邮件。模块详情介绍如下表。

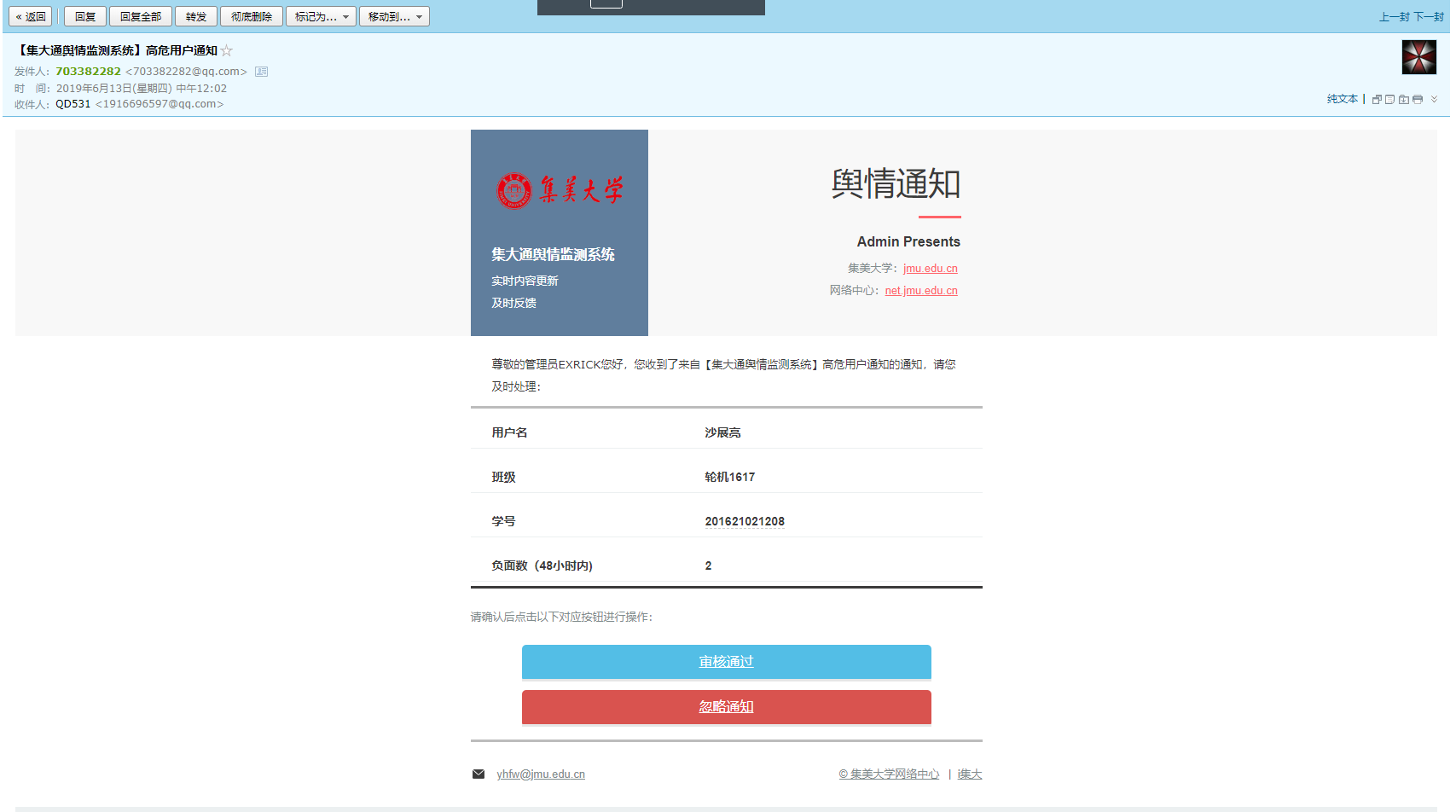


图3-5 高危用户邮件示例

|  |  |  |
| --- | --- | --- |
| 功能 | 功能描述 | 备注 |
| 负面预警邮件 | 当前负面率大于55%时，发送邮件 | 每隔1分钟监测一次。  成功发送后，将冷却30分钟后再次监测。 |
| 高危用户邮件 | 用户在48小时内，负面数量大于5条时，发送邮件。 | 每隔1小时监测一次。  成功发送后，将冷却24小时再次监测。 |

### 3.4.4 前后端通讯设计与实现

本节将详细阐述“高危用户”的前后端代码实现原理，高危用户指在一段时间内，负面微博数量最多的5位用户，同时管理员可以点击用户查看近期用户发言历史。高危用户交互流程见图3-4。

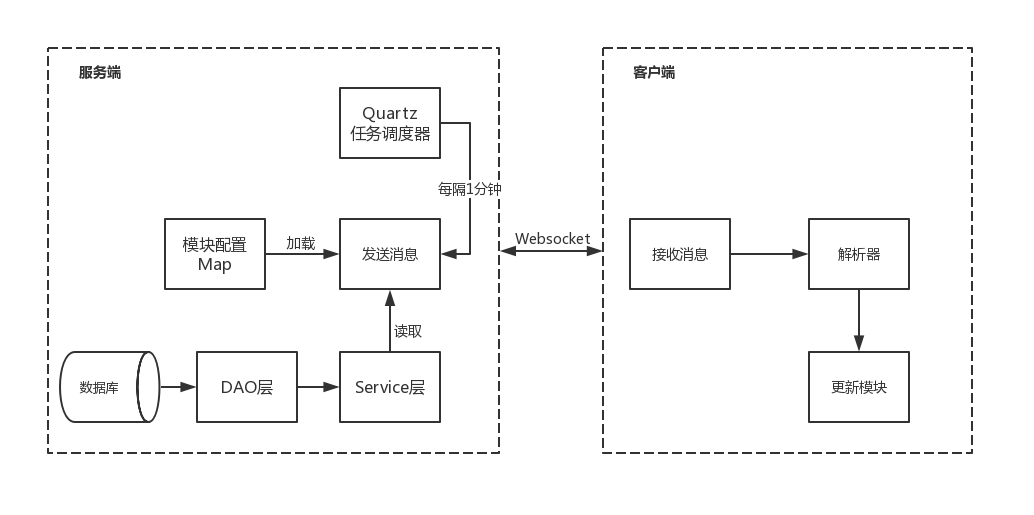


图3-4 高危用户模块工作流程图

客户端成功与服务端建立起Websocket连接时，服务端Websocket的OnOpen方法将根据客户端session发送消息，同时Quartz任务调度器监测到服务端有新的连接session，就会每隔1分钟发送各类模块消息包。客户端收到服务端websocket消息包，将调用解析器来分析出各类消息的处理方法，最后更新显示层的高危用户模块。主要关键代码和类如下：

1. Quartz任务调度器，用于实现实时数据展示的功能，涉及WebSocketJob类，代码和配置如下：

**public void** task(){  
 *//获取WebSocketServer对象的映射。* CopyOnWriteArraySet<WebSocketServer> wsArray = WebSocketServer.*getWebSocketSet*();  
 ApplicationContext act = BeanUtil.*getApplicationContext*();  
 WebSocketServer wss = act.getBean(WebSocketServer.**class**);  
 **if** (wsArray.size() != 0){  
 **for** (WebSocketServer item : wsArray) {  
 **try** {  
 *//向客户端推送消息* wss.sendViewMessage(item.getSession());  
 }**catch** (Exception e){  
 e.printStackTrace();  
 }  
 }  
 }**else** {  
 System.***out***.println(**"WebSocket未连接"**);  
 }  
}

如果websocket的session个数不为0，将向客户端发送消息，否则将打印“WebScoket未连接”。Quartz配置文件在spring-mvc.xml下：

*<!-- 要执行的任务类 -->*<**bean id="WebSocketJob" class="com.chasen.quartz.WebSocketJob"**></**bean**>  
*<!-- 将需要执行的定时任务注入job中 -->*<**bean id="jobDetail1" class="org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean"**>  
 <**property name="targetObject" ref="WebSocketJob"**/>  
 *<!-- 任务类中需要执行的方法 -->* <**property name="targetMethod" value="task"**></**property**>  
 *<!-- 上一次未执行完成的，要等待有再执行。 -->* <**property name="concurrent" value="false"** />  
</**bean**>  
*<!-- 基本的定时器，会绑定具体的任务。 -->*<**bean id="trigger" class="org.springframework.scheduling.quartz.SimpleTriggerFactoryBean"**>  
 <**property name="jobDetail" ref="jobDetail1"**/>  
 <**property name="startDelay" value="3000"**/>  
 <**property name="repeatInterval" value="60000"**/>  
</**bean**>

其中startDelay表示任务调度器的时延，本文设定3000毫秒，repeatInterval表示发送间隔，本文设置1分钟。

1. 数据库，DAO层和Service层为系统框架中的数据层，部分关键代码如下：

**package** com.chasen.service;  
  
**public interface** adclassifyService {

// 高危用户  
 String rankUserNegative(**int** year, **int** month, **int** day, **int** hour);  
}

上述代码为Service层的接口，输入参数为时间范围，接口方法的具体实现方法代码如下：

**public** String rankUserNegative(**int** year,**int** month,**int** day,**int** hour){  
 String currentStamp = **""**;  
 String recentStamp = **""**;  
 **try**{  
 Date currentDate = Utils.*getMyTime*(0,0,0,0);  
 currentStamp = Utils.*dateToStamp*(**sdf**.format(currentDate));  
 Date recentDate = Utils.*getMyTime*(-year,-month,-day,-hour);  
 recentStamp = Utils.*dateToStamp*(**sdf**.format(recentDate));  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 List<indexRankUserList> indexRankUserList = **adclassifyDAO**.rankUserNegative(currentStamp,recentStamp);  
 JSONObject message = **new** JSONObject();  
 message.put(**"type"**,**"indexRankUserList"**);  
 JSONArray rankUserArray = JSONArray.*parseArray*(JSON.*toJSONString*(indexRankUserList));  
 message.put(**"content"**,rankUserArray);  
 **return** JSON.*toJSONString*(message);  
}

代码调用DAO层，输入当前当前时间戳和查询过去时间点的时间戳，将相应实体封装成JSON格式，便于前端数据读取和分析。

1. 消息发送是服务端向客户端发送消息的方法，如下代码为该方法。

**public void** sendMessage(String message,Session session) **throws** IOException{  
 session.getBasicRemote().sendText(message);}

1. 接收消息是客户端JS编写的代码，是同属Websocket前后端通信的方法，代码如下：

**$**(**function** (){  
 **if** (!***window***.***WebSocket***) {  
 ***window***.***WebSocket*** = ***window***.MozWebSocket;  
 }  
 **if** (***window***.***WebSocket***) {  
 ***socket*** = **new *WebSocket***(**'ws://'** + ***window***.location.host + **"/chart"**);  
 *// 收到消息* ***socket***.onmessage = **function** (event) {  
 *analysisMessage*(event.data);  
 };  
 } **else** {  
 *alert*(**"你的浏览器不支持 WebSocket！"**);  
 };  
});

如上述代码所示，成功接收到消息后会调用analysisMessage方法，来解析消息。

1. 解析器是用于选择消息处置的更新模块方法，代码如下：

**function** *analysisMessage*(message){  
 message = ***JSON***.parse(message);  
 **if**(message.**type** == **"indexRankUserList"**){  
 *set\_indexRankUserList\_table*(message.**content**);  
 }  
 **else** {  
 ***console***.info(message.**type**);  
 }  
}

1. 客户端可以向服务端发送消息，在高危用户模块，用户选择时间范围就是一种消息发送的过程。监听首页点击事件的代码在js目录下的indexFunction.js

**function** *indexRankUserNeg\_selectOnchange*() {  
 **var** myselect=***document***.getElementById(**"rankUserNeg\_select"**);  
 *// 下拉框选择的编号* **var** index=myselect.selectedIndex;  
 **switch** (index)  
 {  
 **case** 0:  
 *// 1周  
 webSocketSend*(**"rankUserNeg"**,**"oneWeek"**);  
 **break**;  
 **case** 1:  
 *// 两周  
 webSocketSend*(**"rankUserNeg"**,**"twoWeek"**);  
 **break**;  
 **case** 2:  
 *// 三周  
 webSocketSend*(**"rankUserNeg"**,**"threeWeek"**);  
 **break**;  
 **case** 3:  
 *// 一个月  
 webSocketSend*(**"rankUserNeg"**,**"oneMonth"**);  
 **break**;  
 **default**:  
 **break**;  
 }  
}

当用户点击下拉框，修改时间范围，就会触发上述JS代码，代码将调用webSocketSend()方法发送消息给服务端。

1. 发送代码写在websocket.js，同样构造JSON格式

**function** *webSocketSend*(viewId,type) {  
 **var** myDate = **new *Date***();*//获取系统当前时间* **var** date = myDate.toLocaleString();  
 ***console***.info(***socket***);  
 ***socket***.send(***JSON***.stringify({  
 **message** : {  
 **viewId**: viewId,  
 **type** : type,  
 **time** : date  
 }  
 }));  
 ***console***.info(**"msg send yet"**);  
}

1. 客户端发送成功后，服务端的OnMessage方法将被触发，根据消息的type来调用不同的处理方法，同时修改模块配置Map，即indexChartConfig对象。

@OnMessage  
**public void** onMessage(String message, Session session) **throws** IOException, InterruptedException {  
 Date currentTime = **new** Date();  
 SimpleDateFormat formatter = **new** SimpleDateFormat(**"MM-dd HH:mm:ss"**);  
 String msg = String.*format*(**"%s:来自客户[%d]端的消息: %s"**, formatter.format(currentTime), Thread.*currentThread*().getId(), message);  
 System.***out***.println(msg);  
 JSONObject msgJson = JSON.*parseObject*(message);  
 String viewId = msgJson.getJSONObject(**"message"**).getString(**"viewId"**);  
 String type = msgJson.getJSONObject(**"message"**).getString(**"type"**);  
 *indexChartConfig*.put(viewId,type);  
 procClientMessage(viewId,session);  
}

## 3.5 本章小结

本章主要阐述数据可视化平台的设计与实现，简要介绍项目部署的过程。在系统实现部分，以高危用户模块为例，展示服务端和客户端处理消息的方法，并分析各项方法的代码和配置文件。

# 第4章 识别系统的设计与实现

## 4.1 识别系统简介

识别系统是网络爬虫的一个API接口，目前系统调用的百度AI接口。使用百度AI接口，需要注册百度账号使用，然后在AI控制台使用相应模块，百度AI会返回一串签名，这是程序发送请求时必要token。

详情内容，参见百度AI文档：[http://ai.baidu.com/docs#/Begin/top](http://ai.baidu.com/docs" \l "/Begin/top)

## 4.1 签名嵌入项目

调用百度AI接口的项目主要是网络爬虫，在BaiduAI目录下的AuthService.java是负责集成每个模块的API KEY。每个API KEY都有使用年限，如果API KEY过期，需要重新再百度AI控制台新建模块。

因此，如果项目在识别情绪时出错，可能原因就在百度AI的接口过期了，需要重新新建模块。

以下是AuthService.java下负责自然语言处理的API KEY使用方法。

**public static** String getNlpAuth() {  
 *// 官网获取的 API Key 更新为你注册的  
 // String clientId = "百度云应用的AK";* String clientId = **"IhO1Qz0p4DaSRnER7W8PCytn"**;  
 *// 官网获取的 Secret Key 更新为你注册的  
 // String clientSecret = "百度云应用的SK";* String clientSecret = **"aESSx5W6SxKg444TFAGYTgps97d7Gwbz"**;  
 **return** *getAuth*(clientId, clientSecret);  
}

# 第5章 数据库的设计与部署

## 5.1 数据库环境简介

本项目数据库使用MySQL，版本需要大于5.5。本章将阐述数据库的设计和数据库部署方法。

## 5.1 数据库设计

数据库包含topics，reply,image,user,sentiment五张表。

topics（微博数据）

|  |  |  |
| --- | --- | --- |
| 字段 | 类型（长度） | 备注 |
| tid | varchar(7) | 微博ID |
| uid | varchar(6) | 用户ID |
| unit\_id | varchar(2) | 默认55；？ |
| group\_id | varchar(2) | 默认55；？ |
| content | varchar(200) | 微博外层内容（即最先显示的内容） |
| location | varcahr(2) | 默认为空；？ |
| type | varchar(7) | first：原创; forward：转发 |
| category | varchar(8) | text：content比较少的  longtext：content内容较多 |
| extra\_id | varchar(60) | 图片id字符串 |
| root\_tid | varchar(2) | 0：原创; 非0：转发tid |
| post\_time | varchar(10) | 时间戳 |
| from\_string | varchar(7) | Android; iPhone |
| weibo\_tid | varchar(2) | 默认为0；？ |
| is\_shield | varchar(2) | 默认为0；？ |
| article | varchar(2000) | longtext的完整内容 |
| replys | int | 回复数 |
| forwards | int | 转发数 |
| praises | int | 点赞数 |
| glances | int | 阅读数 |
| praised | varchar(2) | 默认为0；？ |
| favorited | varchar(2) | 默认为0；？ |

user（用户数据）

|  |  |  |
| --- | --- | --- |
| 字段 | 类型（长度） | 备注 |
| uid | varchar(7) | 用户ID |
| username | varchar(50) | 用户姓名 |
| gender | varchar(2) | 性别 |
| nativecode | varchar(6) | 地区编号 |
| classname | varchar(10) | 班级 |
| type | varchar(2) | 学生默认为2；？ |
| birthday | varchar(11) | 格式：2000-07-19 |
| workid | varchar(14) | 学号 |
| joindate | varchar(10) | 加入时间戳 |

image（微博图片）

|  |  |  |
| --- | --- | --- |
| 字段 | 类型（长度） | 备注 |
| imgid | varchar(6) | 图片ID |
| tid | varchar(7) | 微博ID |
| imge\_url | varchar(75) | 图片保存的URL |

reply（微博回复）

|  |  |  |
| --- | --- | --- |
| 字段 | 类型（长度） | 备注 |
| rid | varchar(7) | 回复ID |
| content | varchar(200) | 回复内容 |
| uid | varchar(6) | 用户ID |
| to\_tid | varchar(7) | 所属微博ID |
| to\_rid | varchar(6) | 0：给微博回复；非0：给留言用户回复 |
| post\_time | varchar(10) | 回复时间戳 |

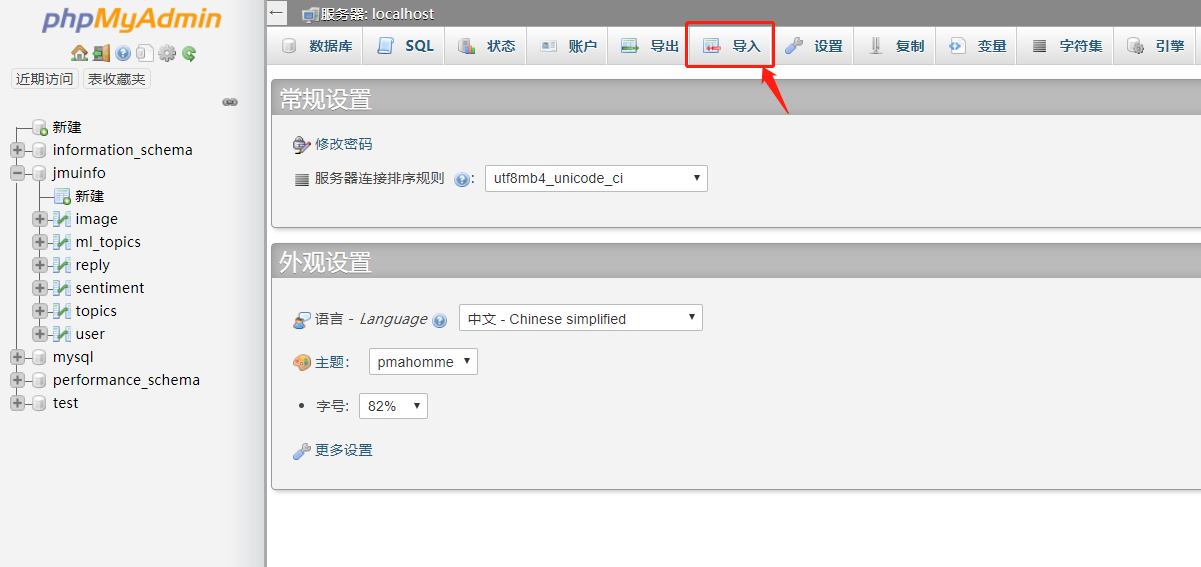
sentiment

|  |  |  |
| --- | --- | --- |
| 字段 | 类型（长度） | 备注 |
| sid | int | 情绪分析ID(自增长) |
| tid | varchar(7) | 微博ID |
| rid | varchar(7) | 微博下方回复ID（与tid相同表示根微博） |
| uid | varchar(6) | 用户ID |
| pos\_prob | float | 表示属于积极类别的概率 ，取值范围[0,1] |
| neg\_prob | float | 表示属于消极类别的概率 ，取值范围[0,1] |
| sentiment | varchar(2) | 表示情感极性分类结果，  0:负向，1:中性，2:正向 |
| post\_time | varchar(10) | 时间戳 |

## 5.2 数据库部署

#### **1.2.1 phpmyadmin**

1.点击进入“导入”页面



2.选择“工程项目包”→“源代码”→“jmuinfo.sql”

#### **1.2.2 mysql控制台导入**

使用mysql命令导入

mysql -u用户名 -p密码 < [要导入的数据库数据(jmuinfo.sql)]

# 第6章 关键功能代码及说明

## 6.1 调用百度AI接口

网络爬虫项目（项目名：jdtFetch）的情绪识别依靠百度AI。百度AI目前提供许多检测接口，并且大部分免费使用（详情见：ai.baidu.com）。网络爬虫项目已经编写人脸识别、图像内容识别、文字识别、情绪识别等接口请求的代码，但仅完成情绪识别JSON结果的解析。

以下示例展示情绪识别接口的调用，有如下步骤：

1. 调用接口，情绪识别接口代码位于cn.jmu.app.fetch.BaiduAI.NlpService

*/\*\*  
 \* 情感分析接口  
 \** ***@param text*** *待分析文本  
 \*/***public static** String getSentiment(String text){  
 *// 接口地址* String url = **"https://aip.baidubce.com/rpc/2.0/nlp/v1/sentiment\_classify"**;  
 *// 构造POST请求参数* JSONObject paramObj = **new** JSONObject();  
 paramObj.put(**"text"**,text);  
 **return** *getRes*(url,paramObj);  
}

1. 获得检测结果，调用情绪识别代码如下，topicsResult表示微博文本的情绪检测结果，第二段代码是JSON返回示例。

String topicsResult = NlpService.*getSentiment*(t.getContent());

{

"text":"苹果是一家伟大的公司",

"items":[

{

"sentiment":2, //表示情感极性分类结果

"confidence":0.40, //表示分类的置信度

"positive\_prob":0.73, //表示属于积极类别的概率

"negative\_prob":0.27 //表示属于消极类别的概率

}

]

}

1. 解析结果，情绪检测结果的解析代码位于cn.jmu.app.fetch.parse.NlpJsonParse

*/\*\*  
 \* com.alibaba.fastjson.\*  
 \* NLP数据JSON解析器  
 \*/***public class** NlpJsonParse {  
 **public** sentiment sentimentParse(String result){  
 sentiment sentiment = **new** sentiment();  
 JSONObject sentimentJson = JSON.*parseObject*(result);  
 JSONObject item = sentimentJson.getJSONArray(**"items"**).getJSONObject(0);  
  
 sentiment.setPos\_prob(item.getFloat(**"positive\_prob"**));  
 sentiment.setNeg\_prob(item.getFloat(**"negative\_prob"**));  
 sentiment.setSentiment(item.getString(**"sentiment"**));  
  
 **return** sentiment;  
 }  
}

## 6.2 集大通爬虫

集大通爬虫是用于获取集大通微博，回复，用户等数据的工具，爬虫代码写在jdtFetch项目的cn.jmu.app.fetch.template目录下，该目录下的每个类都对应一种功能的爬虫。集大通是一款移动端App，数据使用JSON格式封装，本项目采用阿里巴巴开源JSON解析工具fastJson，集大通的解析器在cn.jmu.app.fetch.parse.JmuJsonParse类下。

强烈建议使用安卓模拟器和流量分析器观察集大通数据交互的过程，具体过程参见【第2章 2.4项目添加新模块】

以下示例展示爬取集大通首页的过程，代码位于cn.jmu.app.fetch.template.updateWeibo类下。

爬虫爬取首页微博数据过程是根据每条微博为锚点，一次遍历该条微博下的回复，该微博发表用户数据和该微博的图片数据，将以上获取的数据存入对应的列表，然后循环读取下一条，直到上次更新的时间点结束。当所有微博更新完毕，将从数据库读取更新的微博和回复文本内容，用于情绪监测。

**do** {  
 **if**(flag == 1)  
 *// 第一页的微博URL区别于第二页的微博URL，因此要用到flag* url = JdtUtils.*getIndexWbUrl*();  
 **else** url = JdtUtils.*getIndexSeqWbUrl*(LastID);  
 System.***out***.println(url);  
 *// 获取首页微博数据* String wbRes = **""**;  
 **try**{  
 wbRes = hcu.doJmuWeiboGet(url, JdtUtils.*COOKIE*);  
 }**catch** (Exception e){  
 **continue**;  
 }  
 System.***out***.println(wbRes);  
  
 *// 微博为空，跳出* **if**(jjp.isWbNull(wbRes))  
 **break**;  
 *// 解析器分析* LastID = jjp.getWbLastID(wbRes);  
 List<topics> topicsList = jjp.weiboParse(wbRes);  
 *// 微博重复跳出* topics newestTopics = topicsList.get(0);  
 **int** topStamp = Integer.*parseInt*(newestTopics.getPost\_time());  
 **if**(topStamp < recentStamp)  
 **break**;  
 *//****TODO:微博写入数据库* for**(topics top : topicsList){  
 System.***out***.println(**"正在写入微博："**+top.getTid());  
 topImpl.insertTopics(top);  
 *// 防用户缺漏  
 getUserByWb*(top.getUid());  
 }  
 *// 读取微博回复* FetchWeibo.*readWbReply*(topicsList);  
 *// 解析微博图片* List<image> imgList = jjp.wbImgParse(wbRes);  
 *//****TODO:图片写入数据库* for**(image img : imgList){  
 System.***out***.println(**"正在写入图片："**+img.getImgid());  
 imgImpl.insetImage(img);  
 }  
 flag ++;  
 Thread.*currentThread*().*sleep*(100);  
}**while**(**true**);

## 6.3 使用Echats图表

ECharts，一个使用 JavaScript 实现的开源可视化库，可以流畅的运行在 PC 和移动设备上，兼容当前绝大部分浏览器，提供直观，交互丰富，可高度个性化定制的数据可视化图表。

数据可视化平台多个模块使用Echarts组件，如负面预警，周报表图表等。

官网教学文档：echarts.baidu.com/tutorial.html

官网图表库：echarts.baidu.com/examples

社区图表库：gallery.echartsjs.com/explore.html

数据可视化平台Echarts代码集中于src.main.webapp.assets.js.charts，可视化平台首页加载Echarts代码在该目录下的indexChart.js，以下展示负面预警模块代码，其中负面预警模块效果见图6-1，

原始模板见：<https://www.echartsjs.com/examples/editor.html?c=gauge>。



图6-1 负面预警模块效果图

1. 根据前端DOM的id，初始化echarts实例。其中**indexPie\_NegRateChart**是负面预警模块的前端id。

**$**(***document***).ready(**function**(){  
 ***indexPie\_NegRateChart*** = **echarts**.init(***document***.getElementById(**"indexPie\_NegRateChart"**));  
});

1. 根据rate大小，配置和显示图表。以下配置代码是提供给websocket消息处理的方法，客户端websocket消息处理位于src.main.webapp.assets.js.websocket.js下

*/\*\*  
 \* 负面预警模块\_加载echart配置  
 \*/***function** *SetIndexPie\_NegRateOptions*(rate)  
{  
 **option** = {  
 **tooltip** : {  
 **formatter**: **"{a} <br/>{b} : {c}%"** },  
 **toolbox**: {  
 **feature**: {  
 **restore**: {},  
 **saveAsImage**: {}  
 }  
 },  
 **series**: [  
 {  
 **name**: **'1小时内指标'**,  
 **type**: **'gauge'**,  
 **detail**: {**formatter**:**'{value}%'**},  
 **data**: [{**value**: rate, **name**: **'负面率'**}]  
 }  
 ]  
 };  
  
 ***indexPie\_NegRateChart***.setOption(**option**);  
}

## 6.3 服务端向客户端发送ws消息

服务端向客户端发送websocket消息分为两种情况，第一种是主动消息，即结合Quartz任务定时器，每隔1分钟向客户端发送固定的模块消息；第二种是消息反馈，即客户端向服务端请求数据，服务端根据请求数据类型，向客户端发送对应的消息。消息反馈见【6.4 客户端向服务端发送ws消息】。

所以服务端websocket代码都位于com.chasen.ws下。

1. 这里示例是主动消息。服务端代码位于com.chasen.ws.WebSocketServer

*/\*\*  
 \* 连接建立成功调用的方法  
 \*/*@OnOpen  
**public void** onOpen(Session session) **throws** InterruptedException, ParseException, IOException {  
 **this**.**session** = session;  
 *webSocketSet*.add(**this**); *//加入set中  
 addOnlineCount*(); *//在线数加1* System.***out***.println(**"有新连接加入！当前在线人数为"** + *getOnlineCount*());  
 sendViewMessage(session);  
}

1. 当客户端成功与服务端建立websocket连接，就会执行onOPen方法，向webSocketSet变量添加客户端session，然后调用sendViewMessage(session)方法执行第一次主动消息。其后发送都依靠Quartz定时任务器。

*/\*\*  
 \* 向客户端发送全部模块JSON消息  
 \** ***@param session*** *sessiohn对象  
 \*/***public void** sendViewMessage(Session session) {  
 **try** {  
 Thread.*sleep*(600);  
 *// 高危用户模块* sendMessage(**sentimentService**.rankUserNegative(*indexChartConfig*.get(**"rankUserNeg"**)),session);  
 *// 最近微博模块* sendMessage(**topicService**.getNewSentContent\_Json(*indexChartConfig*.get(**"SentContent"**), Integer.*parseInt*(*indexChartConfig*.get(**"SentContent\_num"**))),session);  
 System.***out***.println(**"消息同步成功！"**);  
 }**catch** (Exception e){  
 System.***out***.println(e);  
 }  
}

为了节省篇幅，这里只展示高危用户和最近微博发送过程。每一个模块消息都对应Service层的一个方法，由于模块可能存在参数变化，如将高危用户展示时间范围改为一周，因此定义了一个Map变量**indexChartConfig**，内部存储所有模块的参数。

1. 服务端代码至此结束，下来展示客户端接收到消息后的处理步骤。客户端websocket代码位于src.main.webapp.assets.js.websocket.js

**var *socket*** = **null**;  
**$**(**function** (){  
 **if** (!***window***.***WebSocket***) {  
 ***window***.***WebSocket*** = ***window***.MozWebSocket;  
 }  
 **if** (***window***.***WebSocket***) {  
 ***socket*** = **new *WebSocket***(**'ws://'** + ***window***.location.host + **"/chart"**);  
 *// 收到消息* ***socket***.onmessage = **function** (event) {  
 *analysisMessage*(event.data);  
 *checkTimeout*();  
 };  
 *//连接* ***socket***.onopen = **function** (event) {  
 ***console***.info(**"连接开启"**);  
 };  
 *//关闭* ***socket***.onclose = **function** (event) {  
 ***console***.info(**"连接被关闭"**);  
 ***socket*** = **null**;  
 };  
 *//异常* ***socket***.onerror = **function** (event) {  
 ***console***.info(**"连接异常"**);  
 ***socket*** = **null**;  
 };  
 } **else** {  
 *alert*(**"你的浏览器不支持 WebSocket！"**);  
 };  
});

上述代码在页面被加载时自动执行，websocket地址为定义为ws://location.host/chart，重点介绍收到消息的过程，即socket.onmessage = function (event)方法。

1. 假如收到高危用户模块消息，则将执行 set\_indexRankUserList\_table(message.content);方法。

*/\*\*  
 \* 消息类型分析  
 \*/***function** *analysisMessage*(message){  
 message = ***JSON***.parse(message);  
 *// 高危用户模块* **if**(message.**type** == **"indexRankUserList"**){  
 *set\_indexRankUserList\_table*(message.**content**);  
 }  
 *// 其他* **else** {  
 ***console***.info(message.**type**);  
 }  
}

至此，服务端向客户端发送消息和处理消息的过程结束，等待下次消息发送，以此实现实时数据展示。

## 6.4 客户端向服务端发送ws消息

客户端向服务端发送ws消息，服务端将完成消息反馈。本节示例是高危用户选择时间范围。

1. 以下是高危用户模块下拉框前端代码，点击事件将触发*indexRankUserNeg\_selectOnchange*()方法。

<**select data-am-selected="{btnSize: 'sm'}" id ="rankUserNeg\_select" onchange="***indexRankUserNeg\_selectOnchange*();**"**>  
 <**option value="a"**>一周数据</**option**>  
 <**option value="b"**>二周数据</**option**>  
 <**option value="c"**>三周数据</**option**>  
 <**option value="d" selected**>一月数据</**option**>  
 <**option value="e"** >完整数据</**option**>  
</**select**>

1. 触发事件方法将读取下拉框选择编号，构建消息方法webSocketSend（）。第一个参数是模块id，在这里高危用户模块名称是**rankUserNeg，**第二个参数是相应模块的请求类型。

*/\*\*  
 \* 高危用户模块\_下拉框触发事件  
 \*/***function** *indexRankUserNeg\_selectOnchange*() {  
 **var** myselect=***document***.getElementById(**"rankUserNeg\_select"**);  
 *// 下拉框选择的编号* **var** index=myselect.selectedIndex;  
 **switch** (index)  
 {  
 **case** 0:  
 *// 1周  
 webSocketSend*(**"rankUserNeg"**,**"oneWeek"**);  
 **break**;  
 **case** 1:  
 *// 两周  
 webSocketSend*(**"rankUserNeg"**,**"twoWeek"**);  
 **break**;  
 **case** 2:  
 *// 三周  
 webSocketSend*(**"rankUserNeg"**,**"threeWeek"**);  
 **break**;  
 **case** 3:  
 *// 一个月  
 webSocketSend*(**"rankUserNeg"**,**"oneMonth"**);  
 **break**;  
 **case** 4:  
 *// 完整数据  
 webSocketSend*(**"rankUserNeg"**,**"inAll"**);  
 **break**;  
 **default**:  
 **break**;  
 }  
}

1. 调用webSocketSend（）方法，构造JSON数据类型，向服务端发送消息，代码位于src.main.webapp.assets.js.websocket.js下。

*/\*\*  
 \* 向服务端发送websocket消息  
 \*/***function** *webSocketSend*(viewId,type) {  
 **var** myDate = **new *Date***();*//获取系统当前时间* **var** date = myDate.toLocaleString();  
 ***console***.info(***socket***);  
 ***socket***.send(***JSON***.stringify({  
 **message** : {  
 **viewId**: viewId, *// 模块id* **type** : type, *// 模块请求类型* **time** : date *// 时间* }  
 }));  
 ***console***.info(**"msg send yet"**);  
}

1. 当服务端接收到消息，位于com.chasen.ws.WebSocketServer的onMessage（）方法将会被执行，代码如下。

@OnMessage  
**public void** onMessage(String message, Session session) **throws** IOException, InterruptedException {  
 Date currentTime = **new** Date();  
 SimpleDateFormat formatter = **new** SimpleDateFormat(**"MM-dd HH:mm:ss"**);  
 String msg = String.*format*(**"%s:来自客户[%d]端的消息: %s"**, formatter.format(currentTime), Thread.*currentThread*().getId(), message);  
 System.***out***.println(msg);  
 *// 解析JSON数据* JSONObject msgJson = JSON.*parseObject*(message);  
 String viewId = msgJson.getJSONObject(**"message"**).getString(**"viewId"**);  
 String type = msgJson.getJSONObject(**"message"**).getString(**"type"**);  
 *// 修改模块配置  
 indexChartConfig*.put(viewId,type);  
 *// 消息反馈* procClientMessage(viewId,session);  
}

1. 最后，根据模块id，发送指定类型的消息。例如读取高危用户模块指定类型是通过*indexChartConfig*.get(**"rankUserNeg"**)实现的。其后客户端接收消息并处理的过程在【6.3 服务端向客户端发送ws消息】阐述过了。

*/\*\*  
 \* 处理客户端消息  
 \** ***@param viewId*** *客户端发送消息的类型  
 \*/***private void** procClientMessage(String viewId,Session session) {  
 **try** {  
 *// 高危用户* **if**(viewId.equals(**"rankUserNeg"**)) sendMessage(**sentimentService**.rankUserNegative(*indexChartConfig*.get(**"rankUserNeg"**)),session);  
 }**catch** (Exception e){  
 System.***out***.println(e);  
 }  
}

至此，客户端向服务端发送展示完毕。

## 6.5 Quartz任务定时器

Quartz任务定时器可以实现定时处理任务，在本项目中是每个1分钟向客户端发送一次消息。任务代码位于com.chasen.quartz.WebSocketJob，配置代码位于src.main.resources.spring-mvc.xml

Java调用Quartz框架见：<https://www.jianshu.com/p/628c674de5c4>

以下是Quartz完整配置代码，其中**startDelay**表示延时任务多少毫秒，**repeatInterval**表示发送间隔毫秒数。修改这两个参数可以改变任务器发送的频率。

*<!-- 要执行的任务类 -->*<**bean id="WebSocketJob" class="com.chasen.quartz.WebSocketJob"**></**bean**>  
  
*<!-- 将需要执行的定时任务注入job中 -->*<**bean id="jobDetail1" class="org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean"**>  
 <**property name="targetObject" ref="WebSocketJob"**/>  
 *<!-- 任务类中需要执行的方法 -->* <**property name="targetMethod" value="task"**></**property**>  
 *<!-- 上一次未执行完成的，要等待有再执行。 -->* <**property name="concurrent" value="false"** />  
</**bean**>  
  
*<!-- 基本的定时器，会绑定具体的任务。 -->*<**bean id="trigger" class="org.springframework.scheduling.quartz.SimpleTriggerFactoryBean"**>  
 <**property name="jobDetail" ref="jobDetail1"**/>  
 <**property name="startDelay" value="3000"**/>  
 <**property name="repeatInterval" value="60000"**/>  
</**bean**>  
  
<**bean id="scheduler" class="org.springframework.scheduling.quartz.SchedulerFactoryBean"**>  
 <**property name="triggers"**>  
 <**list**>  
 <**ref bean="trigger"**/>  
 </**list**>  
 </**property**>  
</**bean**>

# 第7章 系统运行环境和软件安装

## 7.1 系统运行环境

操作系统：Windows10，Windows2008

开发环境：IntelliJ IDEA 2018.1.6 (Ultimate Edition)

Java版本：1.8.0\_161

Web服务器：Tomcat8.5（版本大于8.5的Tomcat才能使用Websocket）

数据库：MySQL5.7

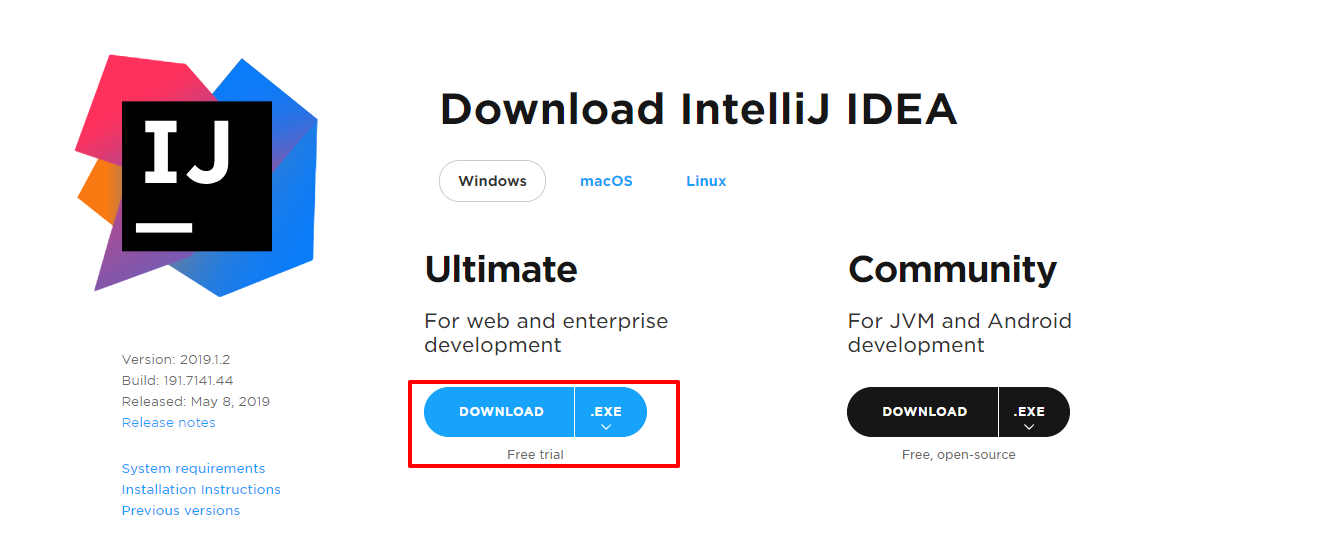
## 7.2 IntelliJ IDEA安装

本项目使用IntelliJ IDEA开发Java项目，相较于Eclipse用户友好度更高。因此建议使用IntelliJ IDEA作为首选Java开发工具。

IntelliJ IDEA分社区版（免费）和旗舰版（付费），旗舰版对于学生免费，可以使用集大邮箱注册账号，将自动认证为学生，即可免费使用旗舰版。社区版不支持Java Web开发，因此必须下载旗舰版。

官网下载地址：[https://www.jetbrains.com/idea/download/#section=windows](https://www.jetbrains.com/idea/download/" \l "section=windows)

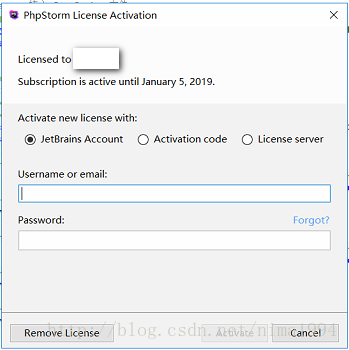
1. 点击下载旗舰版



1. 教育邮箱认证

教程参加：<https://blog.csdn.net/nima1994/article/details/78988595>

1. 安装下载好的IDEA，使用教育邮箱激活。



## 7.3 Java安装

目前Java已经更新至Java12，但还是推荐使用Java1.8。

1. Java1.8 JDK下载地址：

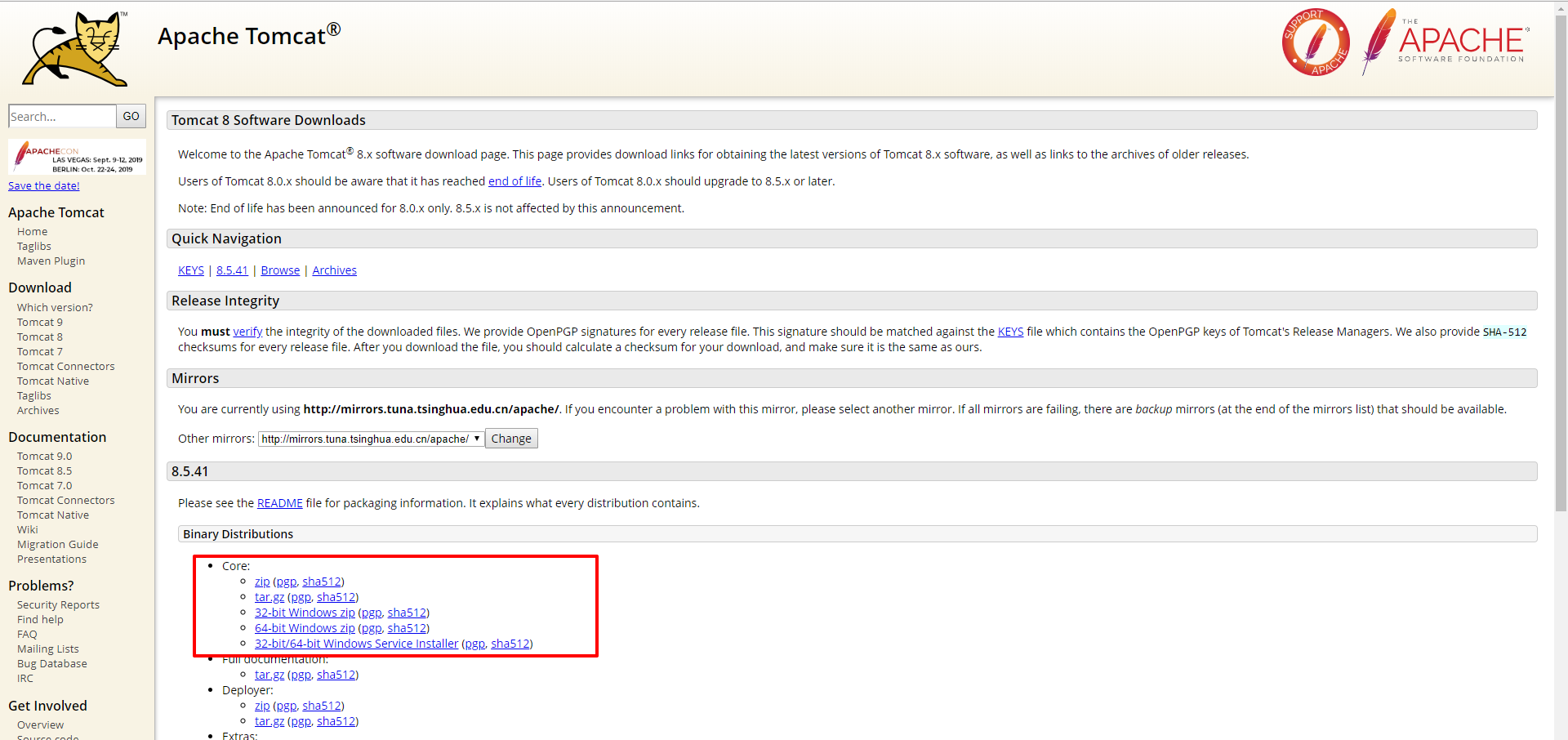
<https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

1. 安装过程参见：<https://blog.csdn.net/u014166319/article/details/71791287>

## 7.3 Tomcat安装

下载Tomcat8.5：<https://tomcat.apache.org/download-80.cgi>

根据操作系统下载对应的Tomcat



项目导入Tomcat以及配置修改见【第3章 3.3.2 部署war包】

## 7.4 MySQL安装

MySQL安装参见：<https://blog.csdn.net/hebbely/article/details/52370179>

建议配合phpmyadmin使用，这样便于数据库的管理。

# 常见问题及解决方法

* 爬虫问题

1. 问题：运行爬虫jar包时报错，错误提示获取连接失败

解决：1.查看jdtFetch数据库配置文件是否正确；2.检查数据库是否开启

Caused by: java.net.ConnectException: Connection refused: connect

at java.net.DualStackPlainSocketImpl.connect0(Native Method)

at java.net.DualStackPlainSocketImpl.socketConnect(Unknown Source)

at java.net.AbstractPlainSocketImpl.doConnect(Unknown Source)

at java.net.AbstractPlainSocketImpl.connectToAddress(Unknown Source)

at java.net.AbstractPlainSocketImpl.connect(Unknown Source)

at java.net.PlainSocketImpl.connect(Unknown Source)

at java.net.SocksSocketImpl.connect(Unknown Source)

at java.net.Socket.connect(Unknown Source)

at com.mysql.jdbc.StandardSocketFactory.connect(StandardSocketFactory.java:211)

at com.mysql.jdbc.MysqlIO.<init>(MysqlIO.java:298)

... 18 more

获取连接失败

Exception in thread "main" java.lang.NullPointerException

at cn.jmu.app.fetch.dbImpl.topicsImpl.findRecnetTopicsStamp(topicsImpl.java:64)

at cn.jmu.app.fetch.template.updateWeibo.readIndexTopics(updateWeibo.java:33)

at cn.jmu.app.fetch.Main.main(Main.java:14)

1. 问题：运行爬虫jar包时报错，看到百度AI返回如下JSON数据错误提示

解决：由于用户免费使用百度AI接口有请求量限制，因此不要使用同一个API KEY运行多个爬虫jar脚本。

{

"error\_code": 18,

"error\_msg": "Open api qps request limit reached"

}

1. 问题：运行爬虫jar包时报错，报错提示“找不到主类”或者“no main manifest attribute”“Could not find or load main class ”

解决：阅读【第2章 2.3.4部署打包jar】的步骤(3)，此类错误是在配置jar包中 出现了问题

* 数据可视化平台

1. 问题：数据可视化平台无法接收到服务端数据，如下图。

解决：1.检查Tomcat配置已将数据可视化平台war包设为根项目，参考【第3章 3.3.2 部署war包】server.xml配置。2.检查数据库是否连接正常；3.若以上2项都设置正确，可能是消息数据量过大导致堵塞，尝试多次刷新页面。



1. 问题：修改项目sentimentClassify代码，无法获取其他类的方法或属性。打印类方法均显示null值。

解决：这是由于Spring注入失败导致的，请使用com.chasen.utils.BeanUtil类的 方法来获取其他类。以下示例是获取WebSocketServer类。

*// 获得WebSocketServer类*ApplicationContext act = BeanUtil.*getApplicationContext*();  
WebSocketServer wss = act.getBean(WebSocketServer.**class**);

# 文档日志

* 2019年3月11日

完成第一版(v1.0)的编写工作，由于当时数据可视化平台处于开发阶段，故文档内容不够完整。

* 2019年5月19日

完成第一版的更新(v1.1)，撰写了整套系统的设计和实现过程。

* 2019年5月22日

完成v1.2版本，更新第6章，第7章和常见问题及解决方法的内容。

* 2019年6月8日

完成v1.3版本，更新项目代码，包括更新48小时内微博内容的转发、回复、点赞数量；数据可视化平台新增微博热榜（热榜权重计算方式：0.2\*回复数+0.4\*转发数+0.4\*点赞数+0.001\*浏览数），具体参见项目ComparatorUtil类；优化Quartz定时器的任务调度。

* 2019年6月15日

完成v1.4版本，更新邮件预警和查询系统，修改图片加载失败等Bug。在文档第3章【3.4 系统实现】小节中添加各类模块的详细介绍。