**NoSQL作业报告**

1. 经济情况分析
2. 问题描述

根据作业要求及所提供数据需得出以下三个问题的结果：

* 经济是否处于通货膨胀？
* 经济是否过热？
* 失业是否在加剧？

提供的数据：

|  |  |
| --- | --- |
| 文件名 | 包含字段 |
| CPI.csv | 指标代码,指标全称,数据发布时间,数据日期,数据值,单位,地区,数据来源,数据更新时间 |
| GDP.csv | 指标代码,指标全称,数据发布时间,数据日期,数据值,单位,地区,来源,数据更新时间 |
| PMI.csv | 指标代码,指标全称,数据发布时间,数据日期,数据值,单位,地区,数据来源,数据更新时间 |
| 美国GDP.csv | 指标代码,指标全称,数据发布时间,数据日期,数据值,单位,地区,数据来源,数据更新时间 |
| 美国就业与失业.csv | 指标代码,指标全称,数据发布时间,数据日期,数据值,单位,地区,数据来源,数据更新时间 |

任务描述：

* 利用所学到的NoSQL技术，加载并进行数据处理，包括：

 经济是否处于通膨？

 当CPI>3%时，我们称为INFLATION，就是通货膨胀；

 CPI>5%的增幅时，称为SERIOUS INFLATION，就是严重通货膨胀。

 经济是否过热？

 CPI/PPI上涨（过高），一般是经济偏热（过热）的表面特征

 PMI与上月进行比较，大于50%，表示经济上升，反之则趋向下降。

 失业是否在加剧？

 统计每个季度的农业和非农业失业率以及其变化

* 要求：

 在MongoDB上实现

实现说明：

* 编写将数据载入Mongodb的JAVA程序，实现命令行执行导入程序。例如：java –jar NoSQL.jar –a ImportDataToMongodb –p /home/tianwen/bigdata/，-a参数表示指定行为，ImportDataToMongodb表示将数据导入mongodb行为，-p表示指定数据目录，后面跟目录地址。

编写分析经济形势的命令行程序，实现按日期查询数据的功能。例如：java –jar NoSQL.jar –a EconomicSituation –d 2014-08-31，表示显示2014-08-31的经济情况。-d表示指定日期参数，后面跟要查询的日期。

* 算法
* 第一个问题可根据CPI数据直接得出指定日期是否处于通货膨胀，即CPI>3时通货膨胀，CPI>5时严重通货膨胀；
* 第二个问题由于数据中没有PPI数据，无法得出经济是否处于过热还是过高，只能通过已有PMI数据得出经济趋于上升或下降，即PMI大于50%表示趋于上升，反之趋于下降；
* 第三个问题由于数据只有美国2014年失业数据，所以只能基于美国失业数据计算。公式为失业变化率等于当月失业率减去上月失业率除以上月失业率。大于0表示失业率上升，小于零表示失业率下降，等于0表示不变。因为没有可参考的失业率变化快慢指标，所以本作业中将假设失业率变化率的绝对值大于2%为变化较快，否则表示变化缓慢。另外失业率是否偏高参考了网上资料，当失业率大于5%表示偏高，大于4小于5表示略高，小于等于4表示正常。

1. 数据加载代码

ImportDataToMongodb.java

public class ImportDataToMongodb {

/\*\*

\* 导入指定文件数据到mongodb表中

\*/

public static void ImportCPI(DBCollection coll,String dataFile) {

try {

FileInputStream in = new FileInputStream(dataFile);

byte[] b = new byte[3];

in.read(b);

String code = "GBK";

if (b[0] == -17 && b[1] == -69 && b[2] == -65){

code = "UTF-8";

}

InputStreamReader inputStreamReader = new InputStreamReader(in,code);

BufferedReader reader = new BufferedReader(inputStreamReader);

String firstLine = reader.readLine(); //第一行信息，为英文标题，可以作为数据项的键

String keys[] = firstLine.split(",");

reader.readLine();//第二行信息，为标题信息，不用,如果需要，注释掉

String line = null;

while((line = reader.readLine()) != null) {

String item[] = line.split(",");

//System.out.println(line);

BasicDBObject doc = null;

for(int i=0;i<item.length;i++) {

if(i == 0){

doc = new BasicDBObject("indicID", item[i]);

} else {

doc.append(keys[i], String.valueOf(item[i]));

}

}

coll.insert(doc);

}

in.close();

reader.close();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void importFilesToMongo(String host, int port, String dbName, String dirPath)

{

// TODO Auto-generated method stub

try{

@SuppressWarnings("deprecation")

Mongo mongo = new Mongo(host, port);

@SuppressWarnings("deprecation")

DB db = mongo.getDB(dbName);

// 遍历所有集合的名字

Set<String> colls = db.getCollectionNames();

for (String s : colls) {

// 先删除所有Collection(类似于关系数据库中的"表")

if (!s.equals("system.indexes")) {

db.getCollection(s).drop();

}

}

File dir = new File(dirPath);

String[] fileList = dir.list();

for (int i = 0; i < fileList.length; i++) {

String filePath = dirPath +"/" + fileList[i];

File file = new File(filePath);

String fileName = file.getName();

String temp[] = fileName.split("\\.");

if(temp.length != 2 || !temp[1].equals("csv"))

{

continue;

}

fileName = temp[0];

// 取得集合CPI(若：CPI不存在，mongodb将自动创建该集合)

DBCollection coll = db.getCollection(fileName);

// delete all

DBCursor dbCursor = coll.find();

for (DBObject dbObject : dbCursor) {

coll.remove(dbObject);

}

//String CPI = "/home/tianwen/Documents/bigdata/CPI.csv";

ImportCPI(coll, filePath);

}

mongo.close();

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* @param args

\*/

public static void main(String[] args) {

String host = Config.MONGODB\_HOST;

int port = Config.MONGODB\_PORT;

String dbName = Config.MONGODB\_DBNAME;

String dirPath = Config.DATA\_DIR;

importFilesToMongo(host, port, dbName, dirPath);

}

}

1. 数据统计代码

EconomicSituation.java

public class EconomicSituation {

public static HashMap<String, Float> getEconomicSituation(String host, int port, String dbName, String date) {

// TODO Auto-generated constructor stub

HashMap<String, Float> ret = new HashMap<String, Float>();

HashMap<Integer, String> monthLastDay = new HashMap<Integer, String>();

monthLastDay.put(1, "31");

monthLastDay.put(2, "28");

monthLastDay.put(3, "31");

monthLastDay.put(4, "30");

monthLastDay.put(5, "31");

monthLastDay.put(6, "30");

monthLastDay.put(7, "31");

monthLastDay.put(8, "31");

monthLastDay.put(9, "30");

monthLastDay.put(10, "31");

monthLastDay.put(11, "30");

monthLastDay.put(12, "31");

try{

@SuppressWarnings("deprecation")

Mongo mongo = new Mongo(host, port);

@SuppressWarnings("deprecation")

DB db = mongo.getDB(dbName);

String[] tmp = date.split("-");

int thisMonth = Integer.parseInt(tmp[1]);

date = tmp[0]+"-"+String.format("%02d", thisMonth)+"-"+monthLastDay.get(thisMonth);

DBCollection coll = db.getCollection("CPI");

BasicDBObject docFind = new BasicDBObject("periodDate", date).append("indicID", "M030000021");

DBObject findResult = coll.findOne(docFind);

Float cpi = Float.parseFloat((String) findResult.get("dataValue"));

ret.put("CPI", cpi);

coll = db.getCollection("PMI");

docFind = new BasicDBObject("periodDate", date).append("indicID", "M020000014");

findResult = coll.findOne(docFind);

Float pmi = Float.parseFloat((String) findResult.get("dataValue"));

ret.put("PMI", pmi);

coll = db.getCollection("美国就业与失业");

docFind = new BasicDBObject("periodDate", date).append("indicID", "G010000069");

findResult = coll.findOne(docFind);

Float 失业率 = Float.parseFloat((String) findResult.get("dataValue"));

ret.put("失业率", 失业率);

int lastMonth = thisMonth - 1;

if(lastMonth == 0){

ret.put("失业环比变化", Float.valueOf(0));

}else{

date = tmp[0]+"-"+String.format("%02d", lastMonth)+"-"+monthLastDay.get(lastMonth);

docFind = new BasicDBObject("periodDate", date).append("indicID", "G010000069");

findResult = coll.findOne(docFind);

Float 上月失业率 = Float.parseFloat((String) findResult.get("dataValue"));

Float 失业环比变化 = (失业率 - 上月失业率)/上月失业率\*100;

ret.put("失业环比变化", 失业环比变化);

}

mongo.close();

} catch (Exception e) {

e.printStackTrace();

}

return ret;

}

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

String host = Config.MONGODB\_HOST;

int port = Config.MONGODB\_PORT;

String dbName = Config.MONGODB\_DBNAME;

String date = "2014-03-31";

HashMap<String, Float> ret = getEconomicSituation(host, port, dbName, date);

Float cpi = ret.get("CPI");

System.out.print("中国经济CPI为"+cpi+",");

if(cpi > 5){

System.out.print("严重通胀");

} else if(cpi > 3) {

System.out.print("通胀");

} else {

System.out.print("不通胀");

}

Float pmi = ret.get("PMI");

System.out.print("，PMI为"+pmi+"，趋于");

if(pmi > 50){

System.out.print("上升");

} else {

System.out.print("下降");

}

System.out.println("。");

Float 失业率 = ret.get("失业率");

System.out.print("美国失业率是"+失业率+"%，");

if(失业率 > 5){

System.out.print("偏高");

} else if(失业率 > 4){

System.out.print("略高");

} else {

System.out.print("正常");

}

System.out.print("，环比");

Float 失业环比变化 = ret.get("失业环比变化");

if(失业环比变化 == 0){

System.out.println("无变化。");

}else{

DecimalFormat decimalFormat=new DecimalFormat(".00");//小数保留2位,不足用0补足.

if(失业环比变化>0){

System.out.print("上涨"+decimalFormat.format(失业环比变化)+"%，");

}else{

System.out.print("下降"+decimalFormat.format(Math.abs(失业环比变化))+"%，");

}

if(Math.abs(失业环比变化) > 2){

System.out.print("变化较快");

}else{

System.out.print("变化缓慢");

}

System.out.println("。");

}

}

}

1. 操作及分析结果

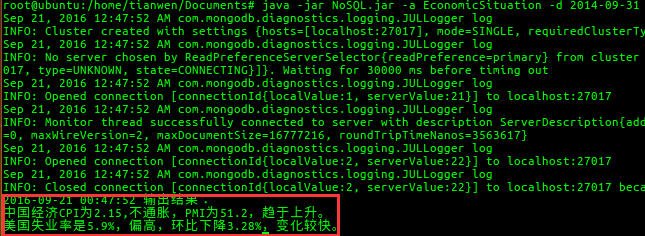
将csv数据导入mongodb操作：

java -jar NoSQL.jar -a ImportDataToMongodb –p /home/tianwen/bigdata



查询指定日期的经济数据操作：

java -jar NoSQL.jar -a EconomicSituation –d 2014-08-31



1. 微博数据分析
2. 问题描述

给出sql数据文件weibodatabase.sql，导入到本地mysql数据库中，过程如下：

* 部署mysql（细节略）
* 进入mysql，新建weibo数据库，SQL：create database weibo。
* 导入数据。SQL：Set names utf8;Source /home/tianwen/Documents/weibodatabase.sql;

数据描述如下：

|  |  |
| --- | --- |
| 数据表 | 字段 |
| 新浪微博用户信息 | 用户uid，用户昵称，用户姓名，用户所在地，用户主页url，  用户性别，用户粉丝数，用户关注数，用户微博数，用户收藏数，用户创建时间 |
| 新浪微博信息 | 微博mid，发布时间，微博内容，微博来源，微博转发数，微  博评论数，微博被赞数，发表用户uid，微博所属主题。  12个主题包括魅族，小米，火箭队，林书豪，恒大，韩剧，雾霾，房价，同桌的你，公务员，贪官，转基因。 |
| 新浪微博用户好友关系 | 每条记录由suid和tuid两个字段组成，表示suid关注tuid，由于新浪微博的限制，每个用户最多只能获取到200个关注人的信息，故好友关系不是很全； |
| 新浪微博转发关系 | 每条记录由smid和tmid两个字段组成，表示smid微博转发tmid微博。 |

根据作业要求和提供数据需要得出以下5个问题的结果：

* 找出发帖子最多的5个用户？
* 找出转发次数最多的20个帖子？
* 找出最热门的3个话题？
* 找出具有最多相同好友的5对用户？
* 找出转发最多相同帖子的5对用户？

1. 数据加载代码

ImportWeiboDataToRedis.java

package com.tianwen;

/\*\*

\* @author tianwen

\*

\*/

public class ImportWeiboDataToRedis {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

DataLoader dl = new DataLoader();

dl.loadAllData();

}

}

DataLoader.java

package com.tianwen;

import java.util.HashMap;

import java.util.List;

import java.util.Map.Entry;

public class DataLoader {

MysqlHandler mysqlHandler = null;

RedisHandler rh = null;

public DataLoader() {

mysqlHandler = MysqlHandler.getInstance();

rh = RedisHandler.getInstance();

}

public void loadAllData() {

// TODO Auto-generated method stub

mysqlHandler.connect();

rh.connect();

//导入用户信息

this.loadUsersInfo();

//导入微博信息

this.loadWeibosInfo();

//导入发帖最多的5个用户

this.loadFaTieUsersTop5();

//导入抓发微博最多的20个用户

this.loadZhuanFaWeibosTop20();

//导入最热门的3个主题

this.loadHotTopicsTop3();

//导入拥有相同朋友最多的5对用户

this.loadHaveSameFriendsUserPairs();

//导入转发相同帖子最多的5对用户

this.loadRepostSameWeibosUserPairs();

mysqlHandler.close();

rh.close();

}

public void loadUsersInfo() {

int userCount = mysqlHandler.getUsersCount();

int page = 1;

int pagesize = 100;

int pageCount = (int) Math.ceil(userCount / pagesize);

int start = 0;

List<HashMap<String, Object>> usersInfo = null;

while(page < pageCount) {

start = (page - 1) \* pagesize;

usersInfo = mysqlHandler.getUsersInfoByPage(start, pagesize);

rh.loadUsersInfo(usersInfo);

page++;

}

}

public void loadWeibosInfo() {

int weiboCount = mysqlHandler.getWeibosCount();

int page = 1;

int pagesize = 100;

int pageCount = (int) Math.ceil(weiboCount / pagesize);

int start = 0;

List<HashMap<String, Object>> weibosInfo = null;

while(page < pageCount) {

start = (page - 1) \* pagesize;

weibosInfo = mysqlHandler.getWeibosInfoByPage(start, pagesize);

rh.loadWeibosInfo(weibosInfo);

page++;

}

}

public void loadFaTieUsersTop5() {

List<HashMap<String, Object>> atUsersTop5 = mysqlHandler.getFaTieUsersTop5();

rh.loadFaTieUsersTop5(atUsersTop5);

}

public void loadZhuanFaWeibosTop20() {

List<HashMap<String, Object>> zfWeibosTop20 = mysqlHandler.getZhuanFaWeibosTop20();

rh.loadZhuanFaWeibosTop20(zfWeibosTop20);

}

public void loadHotTopicsTop3() {

List<HashMap<String, Object>> hotTopicsTop3 = mysqlHandler.getHotTopicsTop3();

rh.loadHotTopicsTop3(hotTopicsTop3);

}

public void loadHaveSameFriendsUserPairs() {

List<HashMap<String, Object>> list = mysqlHandler.getUserFriends();

HashMap<String, Integer> map = DataHandler.getHaveSameFriendsUserPairs(list);

List<Entry<String, Integer>> userPairs = DataHandler.getUserPairsTop5(map, "desc");

rh.loadHaveSameFriendsUserPairsTop5(userPairs);

}

public void loadRepostSameWeibosUserPairs() {

List<HashMap<String, Object>> list = mysqlHandler.getUserZFWeibos();

HashMap<String, Integer> map = DataHandler.getRepostSameWeibosUserPairs(list);

List<Entry<String, Integer>> userPairs = DataHandler.getUserPairsTop5(map, "desc");

rh.loadRepostSameWeibosUserPairsTop5(userPairs);

}

}

MysqlHandler.java

package com.tianwen;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.Collections;

import java.util.HashMap;

import java.util.HashSet;

import java.util.List;

import java.util.Set;

public class MysqlHandler{

private static Connection conn = null;

private MysqlHandler() {}

private static class LazyHolder {

private static final MysqlHandler INSTANCE = new MysqlHandler();

}

public static final MysqlHandler getInstance() {

return LazyHolder.INSTANCE;

}

public void connect() {

String url = "jdbc:mysql://"+Config.MYSQL\_HOST+":"+Config.MYSQL\_PORT+"/"+Config.MYSQL\_DBNAME+"?"

+ "user="+Config.MYSQL\_USER+"&password="+Config.MYSQL\_PASSWD+"&useUnicode=true&characterEncoding=UTF8";

try {

Class.forName("com.mysql.jdbc.Driver");

conn = DriverManager.getConnection(url);

} catch (SQLException e) {

System.out.println("MySQL连接失败");

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

}

public void close(){

if(conn != null) {

try {

conn.close();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

/\*\*

\* 取得发帖最多的5个用户

\*/

public List<HashMap<String, Object>> getFaTieUsersTop5() {

List<HashMap<String, Object>> list = new ArrayList<HashMap<String, Object>>();

try {

Statement stmt = conn.createStatement();

String sql = "select uid,count(\*) ftnum from weibo group by uid order by ftnum desc limit 5";

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

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

map.put("uid", rs.getString("uid"));

map.put("ftnum", rs.getInt("ftnum"));

list.add(map);

}

stmt.close();

} catch (SQLException e) {

System.out.println("MySQL操作错误");

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

return list;

}

/\*\*

\* 取得转发次数最多的20个帖子

\*/

public List<HashMap<String, Object>> getZhuanFaWeibosTop20() {

List<HashMap<String, Object>> list = new ArrayList<HashMap<String, Object>>();

try {

Statement stmt = conn.createStatement();

String sql = "select mid,sum(repostsnum) as zfnum from weibo group by mid order by zfnum desc limit 20";

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

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

map.put("mid", rs.getString("mid"));

map.put("zfnum", rs.getInt("zfnum"));

list.add(map);

}

stmt.close();

} catch (SQLException e) {

System.out.println("MySQL操作错误");

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

return list;

}

/\*\*

\* 取得最热门的3个话题

\*/

public List<HashMap<String, Object>> getHotTopicsTop3() {

List<HashMap<String, Object>> list = new ArrayList<HashMap<String, Object>>();

try {

Statement stmt = conn.createStatement();

String sql = "select topic,sum(commentsnum) as cnum from weibo group by topic order by cnum desc limit 3";

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

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

map.put("topic", rs.getString("topic"));

map.put("cnum", rs.getInt("cnum"));

list.add(map);

}

stmt.close();

} catch (SQLException e) {

System.out.println("MySQL操作错误");

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

return list;

}

/\*\*

\* 取得用户朋友数据

\*/

public List<HashMap<String, Object>> getUserFriends() {

List<HashMap<String, Object>> list = new ArrayList<HashMap<String, Object>>();

try {

Statement stmt = conn.createStatement();

String sql = "SET session group\_concat\_max\_len = 10240;";

stmt.execute(sql);

sql = "select suid,group\_concat(tuid) as friends,count(1) fnum from userrelation group by suid having fnum>50 order by fnum desc";

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

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

map.put("uid", rs.getString("suid"));

Set<String> friends = new HashSet<String>();

Collections.addAll(friends, rs.getString("friends").split(","));

map.put("friends", friends);

list.add(map);

}

stmt.close();

} catch (SQLException e) {

System.out.println("MySQL操作错误");

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

return list;

}

/\*\*

\* 取得用户转发贴数据

\*/

public List<HashMap<String, Object>> getUserZFWeibos() {

List<HashMap<String, Object>> list = new ArrayList<HashMap<String, Object>>();

try {

Statement stmt = conn.createStatement();

String sql = "SET session group\_concat\_max\_len = 20480;";

stmt.execute(sql);

sql = "select a.uid,group\_concat(b.tmid) as mids,count(1) as zfnum from weibo a inner join weiborelation b on a.mid=b.smid group by a.uid having zfnum > 4 order by zfnum desc";

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

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

map.put("uid", rs.getString("uid"));

Set<String> mids = new HashSet<String>();

Collections.addAll(mids, rs.getString("mids").split(","));

map.put("mids", mids);

list.add(map);

}

stmt.close();

} catch (SQLException e) {

System.out.println("MySQL操作错误");

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

return list;

}

/\*\*

\* 取得用户数量

\*/

public int getUsersCount() {

int users = 0;

try {

Statement stmt = conn.createStatement();

String sql = "select count(1) as users from user";

ResultSet rs = stmt.executeQuery(sql);

if (rs.next()) {

users = rs.getInt("users");

}

stmt.close();

} catch (SQLException e) {

System.out.println("MySQL操作错误");

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

return users;

}

/\*\*

\* 分页取得用户信息

\* @param start 开始位置

\* @param pagesize 取得分页大小

\*/

public List<HashMap<String, Object>> getUsersInfoByPage(int start, int pagesize) {

List<HashMap<String, Object>> list = new ArrayList<HashMap<String, Object>>();

try {

Statement stmt = conn.createStatement();

String sql = "select uid,name from user limit "+ start +","+ pagesize;

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

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

map.put("uid", rs.getString("uid"));

map.put("name", rs.getString("name"));

list.add(map);

}

stmt.close();

} catch (SQLException e) {

System.out.println("MySQL操作错误");

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

return list;

}

/\*\*

\* 取得微博信息数量

\*/

public int getWeibosCount() {

int weibos = 0;

try {

Statement stmt = conn.createStatement();

String sql = "select count(1) as weibos from weibo";

ResultSet rs = stmt.executeQuery(sql);

if (rs.next()) {

weibos = rs.getInt("weibos");

}

stmt.close();

} catch (SQLException e) {

System.out.println("MySQL操作错误");

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

return weibos;

}

/\*\*

\* 分页取得微博信息

\* @param start 开始位置

\* @param pagesize 取得分页大小

\*/

public List<HashMap<String, Object>> getWeibosInfoByPage(int start, int pagesize) {

List<HashMap<String, Object>> list = new ArrayList<HashMap<String, Object>>();

try {

Statement stmt = conn.createStatement();

String sql = "select mid,text from weibo limit "+ start +","+ pagesize;

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

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

map.put("mid", rs.getString("mid"));

map.put("text", rs.getString("text"));

list.add(map);

}

stmt.close();

} catch (SQLException e) {

System.out.println("MySQL操作错误");

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

return list;

}

}

RedisHandler.java

package com.tianwen;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

import redis.clients.jedis.Jedis;

public class RedisHandler {

private final static String USER\_KEY\_PREFIX = "U\_";

private final static String WEIBO\_KEY\_PREFIX = "M\_";

private final static String FATIE\_USERS\_TOP5 = "FT\_USER\_TOP5";

private final static String ZF\_WEIBO\_TOP20 = "ZF\_WEIBO\_TOP20";

private final static String HOT\_TOPIC\_TOP3 = "HOT\_TOPIC\_TOP3";

private final static String HAVE\_SAME\_FRIENDS\_USER\_PAIRS\_TOP5 = "HSFUP\_TOP5";

private final static String REPOST\_SAME\_WEIBOS\_USER\_PAIRS\_TOP5 = "RSWUP\_TOP5";

private static Jedis jedis = null;

private RedisHandler() {}

private static class LazyHolder {

private static final RedisHandler INSTANCE = new RedisHandler();

}

public static final RedisHandler getInstance() {

return LazyHolder.INSTANCE;

}

public void connect() {

jedis = new Jedis(Config.REDIS\_HOST, Config.REDIS\_PORT);

}

public void close(){

if(jedis != null) {

jedis = null;

}

}

public void loadUsersInfo(List<HashMap<String, Object>> usersInfo) {

// TODO Auto-generated constructor stub

for(int i=0;i<usersInfo.size();i++)

{

String uid = (String)usersInfo.get(i).get("uid");

String name = (String)usersInfo.get(i).get("name");

jedis.set(USER\_KEY\_PREFIX+uid, name);

//System.out.println(USER\_KEY\_PREFIX+":"+uid+"="+name);

}

}

public String getUser(String uid) {

return jedis.get(USER\_KEY\_PREFIX + uid);

}

public void loadWeibosInfo(List<HashMap<String, Object>> weibosInfo) {

// TODO Auto-generated constructor stub

for(int i=0;i<weibosInfo.size();i++)

{

String mid = (String)weibosInfo.get(i).get("mid");

String text = (String)weibosInfo.get(i).get("text");

jedis.set(WEIBO\_KEY\_PREFIX+mid, text);

//System.out.println(WEIBO\_KEY\_PREFIX+":"+mid+"="+text);

}

}

public String getWeibo(String mid) {

return jedis.get(WEIBO\_KEY\_PREFIX + mid);

}

public void loadFaTieUsersTop5(List<HashMap<String, Object>> faTieUsersTop5) {

// TODO Auto-generated method stub

for(int i=0;i<faTieUsersTop5.size();i++)

{

String uid = (String)faTieUsersTop5.get(i).get("uid");

int ftnum = (int)faTieUsersTop5.get(i).get("ftnum");

jedis.hset(FATIE\_USERS\_TOP5, uid, String.valueOf(ftnum));

//System.out.println(FATIE\_USERS\_TOP5+":"+uid+"="+ftnum);

}

}

public Map<String, String> getFaTieUsersTop5() {

return jedis.hgetAll(FATIE\_USERS\_TOP5);

}

public void loadZhuanFaWeibosTop20(List<HashMap<String, Object>> zfWeibosTop20) {

// TODO Auto-generated method stub

for(int i=0;i<zfWeibosTop20.size();i++)

{

String mid = (String)zfWeibosTop20.get(i).get("mid");

int zfnum = (int)zfWeibosTop20.get(i).get("zfnum");

jedis.hset(ZF\_WEIBO\_TOP20, mid, String.valueOf(zfnum));

//System.out.println(ZF\_WEIBO\_TOP20+":"+mid+"="+zfnum);

}

}

public Map<String, String> getZhuanFaWeibosTop20() {

return jedis.hgetAll(ZF\_WEIBO\_TOP20);

}

public void loadHotTopicsTop3(List<HashMap<String, Object>> hotTopicsTop3) {

// TODO Auto-generated method stub

for(int i=0;i<hotTopicsTop3.size();i++)

{

String topic = (String)hotTopicsTop3.get(i).get("topic");

int cnum = (int)hotTopicsTop3.get(i).get("cnum");

jedis.hset(HOT\_TOPIC\_TOP3, topic, String.valueOf(cnum));

//System.out.println(HOT\_TOPIC\_TOP3+":"+topic+"="+cnum);

}

}

public Map<String, String> getHotTopicsTop3() {

return jedis.hgetAll(HOT\_TOPIC\_TOP3);

}

public void loadHaveSameFriendsUserPairsTop5(List<Entry<String, Integer>> userPairs) {

// TODO Auto-generated method stub

for (Map.Entry<String, Integer> mapping : userPairs) {

String key = mapping.getKey();

int value = mapping.getValue();

jedis.hset(HAVE\_SAME\_FRIENDS\_USER\_PAIRS\_TOP5, key, String.valueOf(value));

//System.out.println(HAVE\_SAME\_FRIENDS\_USER\_PAIRS\_TOP5+":"+key+"="+value);

}

}

public Map<String, String> getHaveSameFriendsUserPairsTop5() {

return jedis.hgetAll(HAVE\_SAME\_FRIENDS\_USER\_PAIRS\_TOP5);

}

public void loadRepostSameWeibosUserPairsTop5(List<Entry<String, Integer>> userPairs) {

// TODO Auto-generated method stub

for (Map.Entry<String, Integer> mapping : userPairs) {

String key = mapping.getKey();

int value = mapping.getValue();

jedis.hset(REPOST\_SAME\_WEIBOS\_USER\_PAIRS\_TOP5, key, String.valueOf(value));

//System.out.println(REPOST\_SAME\_WEIBOS\_USER\_PAIRS\_TOP5+":"+key+"="+value);

}

}

public Map<String, String> getRepostSameWeibosUserPairsTop5() {

return jedis.hgetAll(REPOST\_SAME\_WEIBOS\_USER\_PAIRS\_TOP5);

}

}

1. 数据统计代码

WeiboData.java

package com.tianwen;

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

public class WeiboData {

public static void main(String[] args) {

System.out.println("开始输出微博数据分析结果...");

System.out.println();

RedisHandler rh = RedisHandler.getInstance();

rh.connect();

//取得发帖最多用户Top5

Map<String, String> users = rh.getFaTieUsersTop5();

List<Entry<String, String>> newUsers = DataHandler.orderMapWithDesc(users);

System.out.println("发帖最多用户TOP5如下：");

System.out.println("用户ID\t用户名\t发帖数量");

for (Entry<String, String> mapping : newUsers) {

String uid = mapping.getKey();

String name = rh.getUser(uid);

String ftnum = mapping.getValue();

System.out.println(uid+"\t"+name+"\t"+ftnum);

}

/\*Iterator<Map.Entry<String, String>> userEntries = users.entrySet().iterator();

while (userEntries.hasNext()) {

Map.Entry<String, String> entry = userEntries.next();

String uid = entry.getKey();

String name = rh.getUser(uid);

String ftnum = entry.getValue();

System.out.println(uid+"\t"+name+"\t"+ftnum);

} \*/

System.out.println();

//取得转发最多的微博TOP20

Map<String, String> weibos = rh.getZhuanFaWeibosTop20();

List<Entry<String, String>> newWeibos = DataHandler.orderMapWithDesc(weibos);

System.out.println("转发最多的微博TOP20如下：");

System.out.println("微博ID\t微博内容\t转发次数");

for (Entry<String, String> mapping : newWeibos) {

String mid = mapping.getKey();

String text = rh.getWeibo(mid);

String rpnum = mapping.getValue();

System.out.println(mid+"\t"+text+"\t"+rpnum);

}

/\*Iterator<Map.Entry<String, String>> weiboEntries = weibos.entrySet().iterator();

while (weiboEntries.hasNext()) {

Map.Entry<String, String> entry = weiboEntries.next();

String mid = entry.getKey();

String text = rh.getWeibo(mid);

String rpnum = entry.getValue();

System.out.println(mid+"\t"+text+"\t"+rpnum);

}\*/

System.out.println();

//取得最热门主题TOP3

Map<String, String> topics = rh.getHotTopicsTop3();

List<Entry<String, String>> newTopics = DataHandler.orderMapWithDesc(topics);

System.out.println("最热门主题TOP3如下：");

System.out.println("主题\t评论数");

for (Entry<String, String> mapping : newTopics) {

String topic = mapping.getKey();

String cnum = mapping.getValue();

System.out.println(topic+"\t"+cnum);

}

/\*Iterator<Map.Entry<String, String>> topicEntries = topics.entrySet().iterator();

while (topicEntries.hasNext()) {

Map.Entry<String, String> entry = topicEntries.next();

String topic = entry.getKey();

String cnum = entry.getValue();

System.out.println(topic+"\t"+cnum);

}\*/

System.out.println();

//取得拥有相同朋友最多的用户对TOP5

Map<String, String> userPairs = rh.getHaveSameFriendsUserPairsTop5();

List<Entry<String, String>> newUserPairs = DataHandler.orderMapWithDesc(userPairs);

System.out.println("拥有相同朋友最多的用户对TOP5如下：");

System.out.println("用户1\t用户2\t拥有相同朋友数量");

for (Entry<String, String> mapping : newUserPairs) {

String userPair = mapping.getKey();

String[] userIds = userPair.split("\\|");

String user1 = rh.getUser(userIds[0]);

String user2 = rh.getUser(userIds[1]);

String fnum = mapping.getValue();

System.out.println(user1+"\t"+user2+"\t"+fnum);

}

/\*Iterator<Map.Entry<String, String>> userPairsEntries = userPairs.entrySet().iterator();

while (userPairsEntries.hasNext()) {

Map.Entry<String, String> entry = userPairsEntries.next();

String userPair = entry.getKey();

String[] userIds = userPair.split("\\|");

String user1 = rh.getUser(userIds[0]);

String user2 = rh.getUser(userIds[1]);

String fnum = entry.getValue();

System.out.println(user1+"\t"+user2+"\t"+fnum);

}\*/

System.out.println();

//取得转发相同微博最多的用户对TOP5

Map<String, String> userPairs1 = rh.getRepostSameWeibosUserPairsTop5();

List<Entry<String, String>> newUserPairs1 = DataHandler.orderMapWithDesc(userPairs1);

System.out.println("转发相同微博最多的用户对TOP5如下：");

System.out.println("用户1\t用户2\t转发相同微博数量");

for (Entry<String, String> mapping : newUserPairs1) {

String userPair = mapping.getKey();

String[] userIds = userPair.split("\\|");

String user1 = rh.getUser(userIds[0]);

String user2 = rh.getUser(userIds[1]);

String num = mapping.getValue();

System.out.println(user1+"\t"+user2+"\t"+num);

}

/\*Iterator<Map.Entry<String, String>> userPairs1Entries = userPairs1.entrySet().iterator();

while (userPairs1Entries.hasNext()) {

Map.Entry<String, String> entry = userPairs1Entries.next();

String userPair = entry.getKey();

String[] userIds = userPair.split("\\|");

String user1 = rh.getUser(userIds[0]);

String user2 = rh.getUser(userIds[1]);

String num = entry.getValue();

System.out.println(user1+"\t"+user2+"\t"+num);

}\*/

System.out.println("输出结束。");

rh.close();

}

}

RedisHandler.java（同上略）

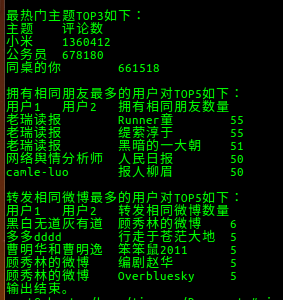
1. 分析结果及解释

命令行执行：

示例：Java –jar NoSQL.jar –a FaTieUsersTop5 (输出发帖最多的5个用户)

结果截图如下：





1. 上面两个数据结果通过Web界面查询
2. 问题描述

基于上面工作，制作web页面，能通过页面上的表单查询经济情况和微博数据。

1. 数据加载代码

数据已加载，此处不需要另外加载。

1. 数据统计代码

index.jsp (数据查询页面)

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>NoSQL作业结果查询</title>

<script type="text/javascript" src="js/jquery-1.11.1.min.js"></script>

</head>

<body>

<center>

<table width="1000px" style="background-color:aliceblue;">

<tr height="20px"><td align="left" style="background-color:aquamarine;">经济情况查询：</td></tr>

<tr>

<td>

<form action="#" name="form1">

<table>

<tr>

<td>日期：

<input type="text" id="data" name="date" size="20" value="2014-08-31"/>

<input type="button" id="btn1" name="btn1" value="查询"/>

</td>

</tr>

<tr>

<td>

<div id="economicSituation" style="width:1000px;height;100px;background-color:beige">

无数据

</div>

</td>

</tr>

</table>

</form>

</td>

</tr>

<tr height="50px"><td>&nbsp;</td></tr>

<tr height="20px"><td align="left" style="background-color:aquamarine;">微博数据查询：</td></tr>

<tr>

<td>

<form action="#" name="form2">

<table>

<tr>

<td>数据项：

<select id="act" name="act">

<option value="">请选择查询数据项...</option>

<option value="FaTieUsersTop5">发帖最多的5个用户</option>

<option value="ZhuanFaWeibosTop20">转发次数最多的20个微博</option>

<option value="HotTopicsTop3">最热门的3个主题</option>

<option value="HaveSameFriendsUserPairsTop5">拥有相同朋友最多的5对用户</option>

<option value="RepostSameWeibosUserPairsTop5">转发相同的微博最多的5对用户</option>

</select>

<input type="button" id="btn2" name="btn2" value="查询"/>

</td>

</tr>

<tr>

<td>

<div id="weiboData" style="width:1000px;height;100px;background-color:beige">

无数据

</div>

</td>

</tr>

</table>

</form>

</td>

</tr>

</table>

</center>

<script type="text/javascript">

$("#btn1").click(function(){

var date = $("#data").val();

if(date == ""){

alert("请输入查询日期!");

}else{

//发送查询请求

$.getJSON("form.do?act=EconomicSituation&date="+date, function(req) {

console.log(req);

if(req.error == 1){

alert(req.msg);

} else {

var result = "中国经济CPI为" + req.data.cpi + "，<font color='red'>";

if(req.data.cpi > 5){

result = result + "严重通胀";

} else if(req.data.cpi > 3) {

result = result + "通胀";

} else {

result = result + "不通胀";

}

result = result + "</font>，PMI为" + req.data.pmi + "，趋于<font color='red'>";

if(req.data.pmi > 50) {

result = result + "上升";

} else {

result = result + "下降";

}

result = result + "</font>。<br>";

result = result + "美国失业率是"+ req.data.失业率 +"%，<font color='red'>";

if(req.data.失业率 > 5) {

result = result + "偏高";

} else if(req.data.失业率 > 4){

result = result + "略高";

} else {

result = result + "正常";

}

result = result + "</font>，环比";

if(req.data.失业环比变化 == 0){

result = result + "<font color='red'>无变化</font>。";

}else{

if(req.data.失业环比变化>0){

result = result + "<font color='red'>上涨"+ (Math.round(req.data.失业环比变化\*100)/100)+"%</font>,";

}else{

result = result + "<font color='red'>下降"+ (Math.round(Math.abs(req.data.失业环比变化)\*100)/100) +"%</font>,";

}

if(Math.abs(req.data.失业环比变化) > 2){

result = result + "变化<font color='red'>较快</font>";

}else{

result = result + "变化<font color='red'>缓慢</font>";

}

result = result +"。";

}

$("#economicSituation").html(result);

}

});

}

});

$("#btn2").click(function(){

var act = $("#act").val();

if(act == ""){

alert("请选择查询数据项!");

}else{

//发送查询请求

$.getJSON("form.do?act="+act, function(req) {

console.log(req);

if(req.error == 1){

alert(req.msg);

} else {

var contentHtml = "<table>";

if(act == "FaTieUsersTop5"){

contentHtml = contentHtml + "<tr><td>用户ID</td><td>用户名</td><td>发帖数量</td></tr>";

$.each(req.data, function(i, item) {

contentHtml = contentHtml + "<tr><td>"+item.uid+"</td><td>"+item.name+"</td><td>"+item.num+"</td></tr>";

});

} else if (act == "ZhuanFaWeibosTop20") {

contentHtml = contentHtml + "<tr><td>微博ID</td><td>微博内容</td><td>转发次数</td></tr></tr>";

$.each(req.data, function(i, item) {

contentHtml = contentHtml + "<tr><td>"+item.mid+"</td><td>"+item.text+"</td><td>"+item.num+"</td></tr>";

});

} else if (act == "HotTopicsTop3") {

contentHtml = contentHtml + "<tr><td>话题</td><td>评论数</td></tr>";

$.each(req.data, function(i, item) {

contentHtml = contentHtml + "<tr><td>"+item.topic+"</td><td>"+item.num+"</td></tr>";

});

} else if (act == "HaveSameFriendsUserPairsTop5") {

contentHtml = contentHtml + "<tr><td>用户1</td><td>用户2</td><td>拥有相同朋友数量</td></tr>";

$.each(req.data, function(i, item) {

contentHtml = contentHtml + "<tr><td>"+item.user1+"</td><td>"+item.user2+"</td><td>"+item.num+"</td></tr>";

});

} else if (act == "RepostSameWeibosUserPairsTop5") {

contentHtml = contentHtml + "<tr><td>用户1</td><td>用户2</td><td>转发相同微博数量</td></tr>";

$.each(req.data, function(i, item) {

contentHtml = contentHtml + "<tr><td>"+item.user1+"</td><td>"+item.user2+"</td><td>"+item.num+"</td></tr>";

});

}

contentHtml = contentHtml + "</table>";

$("#weiboData").html(contentHtml);

}

});

}

});

</script>

</body>

</html>

FormHandle.java（接受查询的Servlet类）

package com.tianwen;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

import javax.json.Json;

import javax.json.JsonArrayBuilder;

import javax.json.JsonObject;

import javax.json.JsonObjectBuilder;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class FormHandle

\*/

@WebServlet("/FormHandle")

public class FormHandle extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* Default constructor.

\*/

public FormHandle() {

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {

// TODO Auto-generated method stub

//response.getWriter().append("Served at: ").append(request.getContextPath());

res.setContentType("text/json; charset=UTF-8");

PrintWriter out = res.getWriter();

String act = req.getParameter("act");

if(act.equals("EconomicSituation")){

String date = req.getParameter("date");

if(date == null){

JsonObject jsonOut = Json.createObjectBuilder()

.add("error", "1")

.add("msg", "参数错误")

.build();

out.print(jsonOut.toString());

} else {

String host = Config.MONGODB\_HOST;

int port = Config.MONGODB\_PORT;

String dbName = Config.MONGODB\_DBNAME;

HashMap<String, Float> ret = EconomicSituation.getEconomicSituation(host, port, dbName, date);

Float cpi = ret.get("CPI");

Float pmi = ret.get("PMI");

Float 失业率 = ret.get("失业率");

Float 失业环比变化 = ret.get("失业环比变化");

JsonObject jsonOut = Json.createObjectBuilder()

.add("error", "0")

.add("msg", "")

.add("data", Json.createObjectBuilder()

.add("cpi", String.valueOf(cpi))

.add("pmi", String.valueOf(pmi))

.add("失业率", String.valueOf(失业率))

.add("失业环比变化", String.valueOf(失业环比变化)))

.build();

out.print(jsonOut.toString());

}

} else if(act.equals("FaTieUsersTop5")){

RedisHandler rh = RedisHandler.getInstance();

rh.connect();

JsonArrayBuilder jsonData = Json.createArrayBuilder();

Map<String, String> users = rh.getFaTieUsersTop5();

List<Entry<String, String>> newUsers = DataHandler.orderMapWithDesc(users);

for (Entry<String, String> mapping : newUsers) {

String uid = mapping.getKey();

String name = rh.getUser(uid);

String ftnum = mapping.getValue();

JsonObjectBuilder item = Json.createObjectBuilder();

item.add("uid", uid);

item.add("name", name);

item.add("num", ftnum);

jsonData.add(item);

}

JsonObject jsonOut = Json.createObjectBuilder()

.add("error", "0")

.add("msg", "")

.add("data",jsonData)

.build();

out.print(jsonOut.toString());

rh.close();

} else if(act.equals("ZhuanFaWeibosTop20")){

RedisHandler rh = RedisHandler.getInstance();

rh.connect();

JsonArrayBuilder jsonData = Json.createArrayBuilder();

Map<String, String> weibos = rh.getZhuanFaWeibosTop20();

List<Entry<String, String>> newWeibos = DataHandler.orderMapWithDesc(weibos);

for (Entry<String, String> mapping : newWeibos) {

String mid = mapping.getKey();

String text = rh.getWeibo(mid);

String rpnum = mapping.getValue();

JsonObjectBuilder item = Json.createObjectBuilder();

item.add("mid", mid);

item.add("text", text);

item.add("num", rpnum);

jsonData.add(item);

}

JsonObject jsonOut = Json.createObjectBuilder()

.add("error", "0")

.add("msg", "")

.add("data",jsonData)

.build();

out.print(jsonOut.toString());

rh.close();

} else if(act.equals("HotTopicsTop3")){

RedisHandler rh = RedisHandler.getInstance();

rh.connect();

JsonArrayBuilder jsonData = Json.createArrayBuilder();

Map<String, String> topics = rh.getHotTopicsTop3();

List<Entry<String, String>> newTopics = DataHandler.orderMapWithDesc(topics);

for (Entry<String, String> mapping : newTopics) {

String topic = mapping.getKey();

String cnum = mapping.getValue();

JsonObjectBuilder item = Json.createObjectBuilder();

item.add("topic", topic);

item.add("num", cnum);

jsonData.add(item);

}

JsonObject jsonOut = Json.createObjectBuilder()

.add("error", "0")

.add("msg", "")

.add("data",jsonData)

.build();

out.print(jsonOut.toString());

rh.close();

} else if(act.equals("HaveSameFriendsUserPairsTop5")){

RedisHandler rh = RedisHandler.getInstance();

rh.connect();

JsonArrayBuilder jsonData = Json.createArrayBuilder();

Map<String, String> userPairs = rh.getHaveSameFriendsUserPairsTop5();

List<Entry<String, String>> newUserPairs = DataHandler.orderMapWithDesc(userPairs);

for (Entry<String, String> mapping : newUserPairs) {

String userPair = mapping.getKey();

String[] userIds = userPair.split("\\|");

String user1 = rh.getUser(userIds[0]);

String user2 = rh.getUser(userIds[1]);

String fnum = mapping.getValue();

JsonObjectBuilder item = Json.createObjectBuilder();

item.add("user1", user1);

item.add("user2", user2);

item.add("num", fnum);

jsonData.add(item);

}

JsonObject jsonOut = Json.createObjectBuilder()

.add("error", "0")

.add("msg", "")

.add("data",jsonData)

.build();

out.print(jsonOut.toString());

rh.close();

} else if(act.equals("RepostSameWeibosUserPairsTop5")){

RedisHandler rh = RedisHandler.getInstance();

rh.connect();

JsonArrayBuilder jsonData = Json.createArrayBuilder();

Map<String, String> userPairs = rh.getRepostSameWeibosUserPairsTop5();

List<Entry<String, String>> newUserPairs = DataHandler.orderMapWithDesc(userPairs);

for (Entry<String, String> mapping : newUserPairs) {

String userPair = mapping.getKey();

String[] userIds = userPair.split("\\|");

String user1 = rh.getUser(userIds[0]);

String user2 = rh.getUser(userIds[1]);

String num = mapping.getValue();

JsonObjectBuilder item = Json.createObjectBuilder();

item.add("user1", user1);

item.add("user2", user2);

item.add("num", num);

jsonData.add(item);

}

JsonObject jsonOut = Json.createObjectBuilder()

.add("error", "0")

.add("msg", "")

.add("data",jsonData)

.build();

out.print(jsonOut.toString());

rh.close();

} else {

JsonObject jsonOut = Json.createObjectBuilder()

.add("error", "1")

.add("msg", "参数错误")

.build();

out.print(jsonOut.toString());

}

out.close();

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

}

}

1. 分析结果及解释

结果截图如下：



源代码已放到github上，地址如下：

https://github.com/tianwensun/NoSQL-Practice.git