交割结算价项目

1.项目描述

根据配置信息，读取当日行情数据，计算出交割结算价，并生成文件。

2.项目需求

(1)配置格式：

指数名称:交易时间段,收盘时间,间隔时间

CES120:13:35-15:00,0,5

CESG10:9:35-12:00,13:00-16:00,16:00,5

(2)如果收盘时间为0，无须获取收盘价即可计算结果

(3)如果收盘时间不为0，

当收盘时间等于最末交易时间时，只取一条收盘价，不取最末交易时间的实时价

当收盘时间不等于最末交易时间时，既取最末交易时间的实时价，也取收盘时间的收盘价

(4)当到达指定收盘时间，仍未获取到收盘价，则不计算，不等待。

3.工作流程内容

(1)了解行情数据文件txt内部数据格式，以及要生成的文件txt内部数据格式。

(2)规划配置文件，开发配置文件读取模块。设计对应配置文件信息的对象类Index，TimeDuration类，并配置关联关系。

(3)根据取到的配置信息，按照顺序逐条计算。利用BufferdReader读取行情数据文件信息，再利用BufferdWriter生成计算结果文件。再对结果文件生成MD5码。

public class MD5Util {

private static final String[] HEX\_MAP = {"0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F"};

public static String getMD5(File file){

String value = null;

FileInputStream fis = null;

try {

fis = new FileInputStream(file);

MappedByteBuffer byteBuffer = fis.getChannel().map(FileChannel.MapMode.READ\_ONLY, 0, file.length());

MessageDigest md5 = MessageDigest.getInstance("MD5");

md5.update(byteBuffer);

value = byteToHex(md5.digest());

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

}finally {

if (null != fis) {

try {

fis.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

return value;

}

private static String byteToHex(byte[] digest) {

StringBuilder sb = new StringBuilder();

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

sb.append(HEX\_MAP[((digest[i]>>4) & 0x0F)]);

sb.append(HEX\_MAP[(digest[i] & 0x0F)]);

}

return sb.toString();

}

public static void main(String[] args) {

File file = new File("files/rsfp/csiRFSPCES12020160513.txt");

String md5 = getMD5(file);

System.out.println(md5);

}

}

(4)最后追加了定时自动计算功能。利用Timer类设计定时器。

/\*\*

\* 初始化定时器

\*/

public void initTimer() {

for (Entry<String, List<Index>> entry : exeMap.entrySet()) {

MyTimeTask mtt = new MyTimeTask(entry.getKey(),entry.getValue(), ms);

Timer newTimer = new Timer(entry.getKey());

MyCalendar mc = new MyCalendar(new Date());

try {

mc.setHHmm(entry.getKey());

long period = 24 \* 60 \* 60 \* 1000;

newTimer.scheduleAtFixedRate(mtt, mc.getDate(), period);

timerMap.put(entry.getKey(), newTimer);

} catch (ParseException e) {

e.printStackTrace();

}

}

}

public class MyTimeTask extends TimerTask {

private String time;

private List<Index> list;

private MainService service;

public MyTimeTask(String time,List<Index> list, MainService mainService) {

this.time = time;

this.list = list;

this.service = mainService;

}

@Override

public void run() {

service.loadData(list);

}

}

4.工作挑战

时间段处理：时间段按时间间隔分成指定个数的取值时间点，每个时间点要取到数据。行情数据文件不会准时，需要延后提取。