日志分析实用类

作者：Andy
原文链接：https://ld246.com/article/1538128777255
来源网站：链滴

1、从日志中提取需要的信息，并计算，方法如下：

```
/**
     * 数据提取计算
     * @param filepath
     */
    public static void Txt(String filepath) {
        String encoding = "gbk";//txt一般默认编码为gbk
        File file = new File(filepath);
        if (file.exists() && file.isFile()) {
            try {
                InputStreamReader read = new InputStreamReader(new FileInputStream(file), encod
ng);
                BufferedReader bufferedReader = new BufferedReader(read);
                Map<String, List<Integer>> resultMapTotal = new HashMap<>();
                Map<String, Integer> resultMap = new HashMap<>();

                String txtLine = "";
                while ((txtLine = bufferedReader.readLine()) != null) {
                    String[] methodAndTime = txtLine.split("====")[1].split(" ");
                    //包含key，追加，不包含则写入
                    if (resultMapTotal.containsKey(methodAndTime[1])) {
                        List list = resultMapTotal.get(methodAndTime[1]);
                        list.add(Integer.valueOf(methodAndTime[5]));
                        resultMapTotal.put(methodAndTime[1], list);
                    } else {
                        List<Integer> list = new ArrayList();
                        list.add(Integer.valueOf(methodAndTime[5]));
                        resultMapTotal.put(methodAndTime[1], list);
                    }

                }
                Iterator<String> it = resultMapTotal.keySet().iterator();
                while (it.hasNext()) {
                    String key = it.next();
                    List listSort = resultMapTotal.get(key);
                    List<Integer> resultList = new ArrayList<>();
                    //降序排序
                    Collections.sort(listSort, Collections.reverseOrder());
                    //最大时间
                    String maxTime = Integer.toString((int) listSort.get(0));
                    //最小时间
                    String minTime = Integer.toString((int) listSort.get(listSort.size() - 1));
                    //调用次数
                    String totalNum = Integer.toString(listSort.size());
                    //中间值
                    String median = Integer.toString((int) listSort.get(listSort.size() / 2));
                    //100毫秒内占比
                    int proportionNum100 = 0;
                    //200毫秒内占比
                    int proportionNum200 = 0;
                    //300毫秒内占比
                    int proportionNum300 = 0;
                    //平均调用时间
```

```
                int totalavg = 0;
                for (int i = 0; i < listSort.size(); i++) {
                    totalavg = (int) listSort.get(i) + totalavg;
                    if ((int) listSort.get(i) <= 100) {
                        proportionNum100++;
                    }
                    if ((int) listSort.get(i) <= 200) {
                        proportionNum200++;
                    }
                    if ((int) listSort.get(i) <= 300) {
                        proportionNum300++;
                    }
                }
                //平均调用时间
                int numavg = totalavg / listSort.size();
                //100毫秒内占比
                String proportion100 = (float) proportionNum100 * 100 / listSort.size() + "%";
                //200毫秒内占比
                String proportion200 = (float) proportionNum200 * 100 / listSort.size() + "%";
                //300毫秒内占比
                String proportion300 = (float) proportionNum300 * 100 / listSort.size() + "%";

                //90%调用时间
                int total = (int) listSort.get(new BigDecimal(listSort.size() * 0.1).setScale(0, BigDec
mal.ROUND_HALF_UP).intValue());
                //90%调用时间
                String num90 = Integer.toString(total/* / listSort.size()*/);
                //System.out.println(key+"."+maxTime+"."+minTime);
                resultMap.put(key + " " + num90 + " " + maxTime + " " + minTime + " " + media
 + " " + proportion100 + " " + proportion200 + " " + proportion300 + " " + totalNum, numav
);

            }
            deriveTable(resultMap);
            System.out.println("导出完成");
            read.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

    }
}
```

2、将提取的数据放入Map中，根据value进行排序，并导出excel表格，方法如下：

```
/**
 * Map类型数据按value排序，并导出excel
 *
 * @param map
 */
public static void deriveTable(Map<String, Integer> map) {
    //Map<String,Integer> map1 =
    //排序规则
    Comparator<Map.Entry<String, Integer>> valueComparator = new Comparator<Map.En
```

```
ry<String, Integer>>() {
            @Override
            public int compare(Map.Entry<String, Integer> o1, Map.Entry<String, Integer> o2) {
                return o2.getValue() - o1.getValue();
            }
        };

        // map转换成list进行排序
        List<Map.Entry<String, Integer>> list = new ArrayList<>(map.entrySet());

        // 排序
        Collections.sort(list, valueComparator);

        // 创建Excel文件对应的对象
        HSSFWorkbook hwk = new HSSFWorkbook();
        // 创建一个sheet表名
        HSSFSheet hssfSheet = hwk.createSheet("接口平均速度统计");

        // 默认情况下，TreeMap对key进行降序排序
        System.out.println("------------map按照value降序排序--------------------");
        // 通过sheet创建一盒row（行） 范围0-65535
        HSSFRow hssfRowHead1 = hssfSheet.createRow(0);

        String[] heads = {"接口名称", "平均调用耗时（ms）", "90% Line", "最大调用时间(ms)", "最
调用时间(ms)", "中间值(ms)", "100毫秒内占比", "200毫秒内占比", "300毫秒内占比", "调用次数"};

        for (int i = 0; i < heads.length; i++) {
            HSSFCell cellHead = hssfRowHead1.createCell(i);
            cellHead.setCellValue(heads[i]);
        }

        int index = 0;
        for (Map.Entry<String, Integer> entry : list) {
            index++;
            System.out.println(entry.getKey());
            String[] subString = entry.getKey().split(" ");
            HSSFRow hssfRow = hssfSheet.createRow(index);

            HSSFCell cell = hssfRow.createCell(0);
            cell.setCellValue(subString[0]);

            HSSFCell cell1 = hssfRow.createCell(1);
            cell1.setCellValue(entry.getValue());
            System.out.println("subString.length"+subString.length);
            for (int a = 1; a < subString.length; a++) {
                HSSFCell cell3 = hssfRow.createCell(a+1        );
                cell3.setCellValue(subString[a]);
            }

        }

        FileOutputStream fos = null;
        try {
            File file = new File("e:/接口平均时间统计.xls");
```

```
            if (file.exists()) {
                file.delete();
            }
            fos = new FileOutputStream("e:/接口平均时间统计.xls");
            hwk.write(fos);
        }catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                fos.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```