批处理作业设计规范

结构组成：

1、ArafStatus 作业状态处理类。 不需要继承。

2、ArafReader 数据读取，通常包含检查及分块功能。　（reader. 想放弃数据需要在execute 后执行 renounce或直接清理数据库内容，exec中不能执行 renounce方法）

3、ArafProcessor 数据块处理，异步并行处理数据块。

4、ArafWriter 数据存储，异步并行存储数据。

5、用于查看某个特征码sign的处理状态

getArafStatus().getStatus(sign);

ArafStatus.getStatusMap(sign); 返回信息实例：

{"PROCESSOR_END":8,"WRITER_ALL":1,"PROCESSOR_SENDEND":1,"PROCESSOR_ALL":1,

"PROCESSOR_START":8,"READER_SENDEND":1,"ALL_END":0,"WRITER_START":8,"WRITER_END":8}

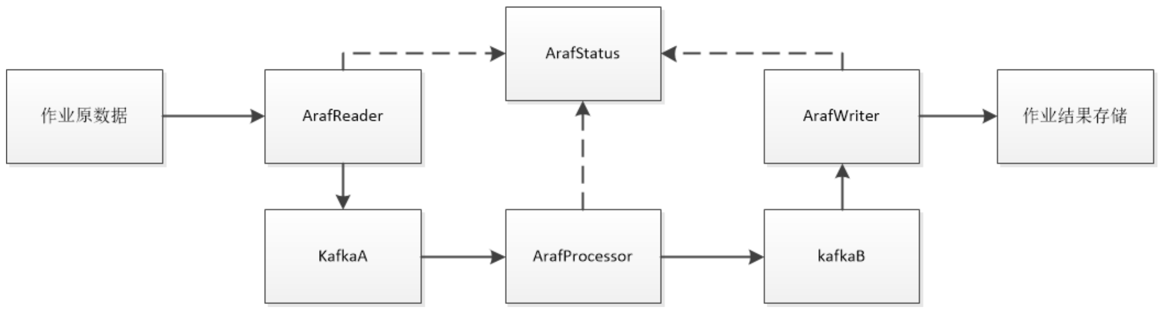
6、在特征码sign不可用的时候，宣布放弃状态，可以通知processor　和writer　放弃未接收的消息　getArafStatus().renounce(sign); 阻塞模式，方法执行完需要对数据库做处理。

7、获取异常信息　String []strs = dbBatchReader.getMsg(getJobInstanceId());

8、添加信息 arafStatus.addMessage(getJobType(), k.getSign(), "信息，信息保留 50小时");

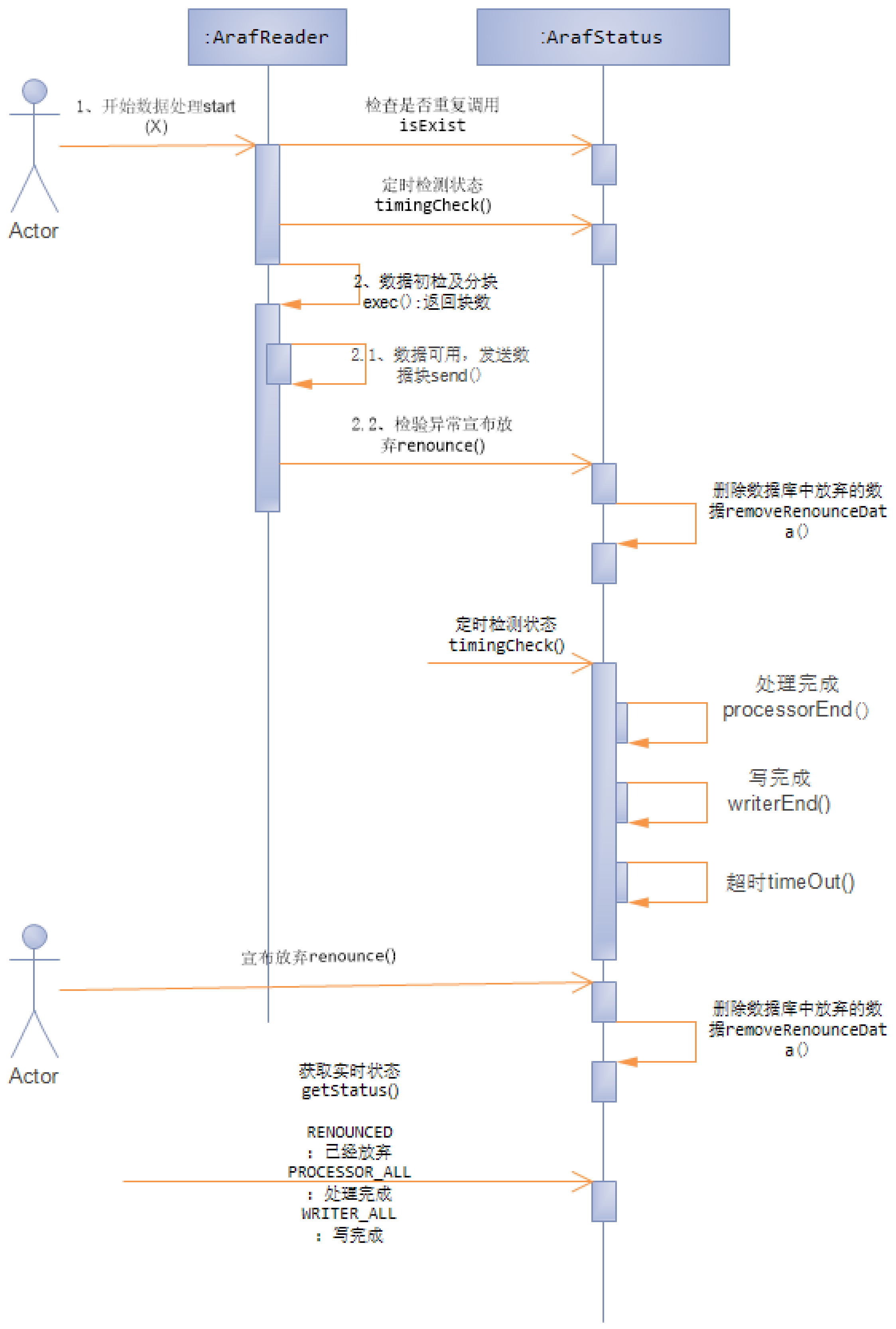
9、计划放弃状态 planRenounce()　isPlanRenounce()　方便开发者在processor 和writer产生放弃是传递给reader做业务处理。

数据流说明图

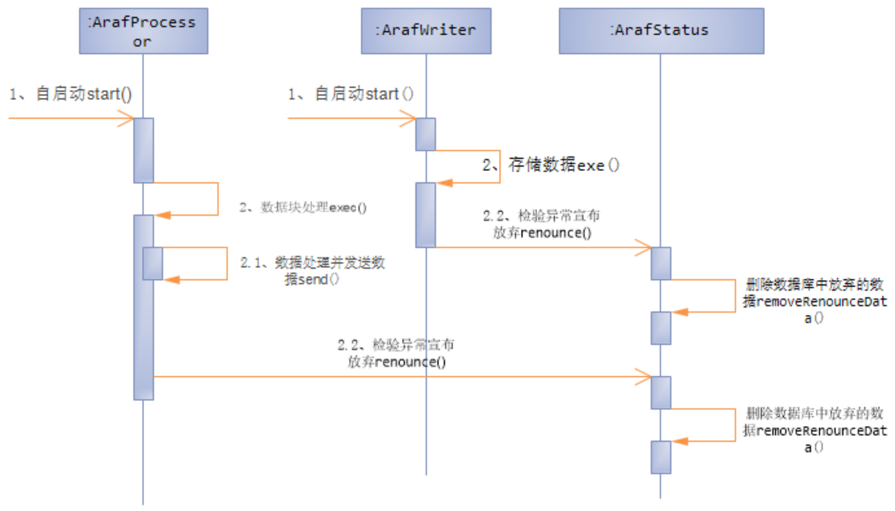


读、状态跟踪及作业结束序列图

自动启动、处理及存储序列图



自动启动、处理及存储序列图


:ArafProcessor
:ArafWriter
:ArafStatus
1、自启动start()
1、自启动start()
2、存储数据exe()
2、数据块处理exec()
2.2、检验异常宣布放弃renounce()
2.1、数据处理并发送数据send()
删除数据库中放弃的数据removeRenounceData()
2.2、检验异常宣布放弃renounce()
删除数据库中放弃的数据removeRenounceData()

实现

系统中添加依赖：

```
<dependency>
        <groupId>com.acca</groupId>
        <artifactId>araf-job-spring-boot-starter</artifactId>
</dependency>
```

继承基础类

```
@Component
public class OagProcessor extends ArafProcessor<SsimFlightDTO, SsimDTO> {

    @Autowired
    private OagSsimService oagSsimService;

    @Override
    public SsimDTO process(AdpKey k, SsimFlightDTO v) {
        return this.processData(v);
    }

    private SsimDTO processData(SsimFlightDTO t) {
        return oagSsimService.processSsim(t);
    }

    @Override
    public String getJobType() {
        return Constants.JOB_OAG_IMPORT;
    }

    @Override
    protected void onSendFailure(AdpKey k, SsimDTO v, Throwable ex) {
        // TODO Auto-generated method stub
        super.onSendFailure(k, v, ex);
    }

}
```

如果文件中发送0条数据。processLines（）　需要返回　-1

```
@Component
@Slf4j
public class OagReader extends AbstractFileReader<SsimFlightDTO> {

    @Autowired
    private MasOagSsimVersionService masOagSsimVersionService;

    @Override
    protected long processLines(String filePath, BufferedReader reader)
throws IOException {
        String fileName = FileUtils.getFileName(filePath);
        String line = "";
        SsimFlightDTO dto = null; //
        String flightNumberAndItinerary = ""; // , 3flightDTO.
        SsimRecord2 seasonRecord = null; // season record
        SsimLegDTO legRecord = null; // record3record4
        long count = 0;
        long sendnum = 0; //
        while ((line = reader.readLine()) != null) {
            count++; //

            String firstChar = ArafStringUtils.substring(line, 0, 1);

            if ("3".equals(firstChar)) {
                String currentFlight = ArafStringUtils.substring(line, 2, 
11);
                if (dto != null) {
                    if (currentFlight.equals(flightNumberAndItinerary)) {
                        // legflightDTO.
                        legRecord = new SsimLegDTO();
                        legRecord.getLines().add(new SsimLine(count, 
line));
                        dto.getLegDTOs().add(legRecord);
                        continue; //
                    } else {
                        // scheduleflightDTO
                        //
                        //
                        sendnum ++;
                        this.send(AdpKey.of(filePath, sendnum), dto);
                        dto = null;
                    }

                }

                //
                flightNumberAndItinerary = currentFlight;
                dto = new SsimFlightDTO();
                dto.setRecord2(seasonRecord);
                legRecord = new SsimLegDTO();
                legRecord.getLines().add(new SsimLine(count, line));
                dto.getLegDTOs().add(legRecord);

            } else if ("4".equals(firstChar) && legRecord != null) {
                legRecord.getLines().add(new SsimLine(count, line));
            } else if ("5".equals(firstChar)) {
                //
                sendnum ++;
                this.send(AdpKey.of(filePath, sendnum), dto);
                dto = null;
            } else if ("1".equals(firstChar)) {
            } else if ("2".equals(firstChar)) {
                seasonRecord = SegmentUtils.formatBySegment(SsimRecord2.
class, line);
                seasonRecord.setFileName(fileName);
                //
                seasonRecord.setPartition(Integer.valueOf(FileUtils.
regStrNum(fileName)));
                masOagSsimVersionService.processType2(seasonRecord);
            } else {
                // 0
                }

            }

            return sendnum;
        }

}
```

```
@Component
@Slf4j
public class OagWirter extends ArafWriter<SsimDTO> {

    @Autowired
    private MasOagSsimType3Service masOagSsimType3Service;

    @Override
    public void persist(AdpKey k, SsimDTO v) {
        log.debug("persist {} {} " , k , v);
        this.persist(Arrays.asList(v));
    }

    public int persist(List<SsimDTO> t) {
        return masOagSsimType3Service.saveOagSsimEntity(t);
    }

    @Override
    public String getJobType() {
        return Constants.JOB_OAG_IMPORT;
    }

}
```

ArafBatchWriter 继承这个类，实现批量持久化， 需要注意 key 中的sign 有可能不一样， 可能是多个文件的数据 。

@刘树友