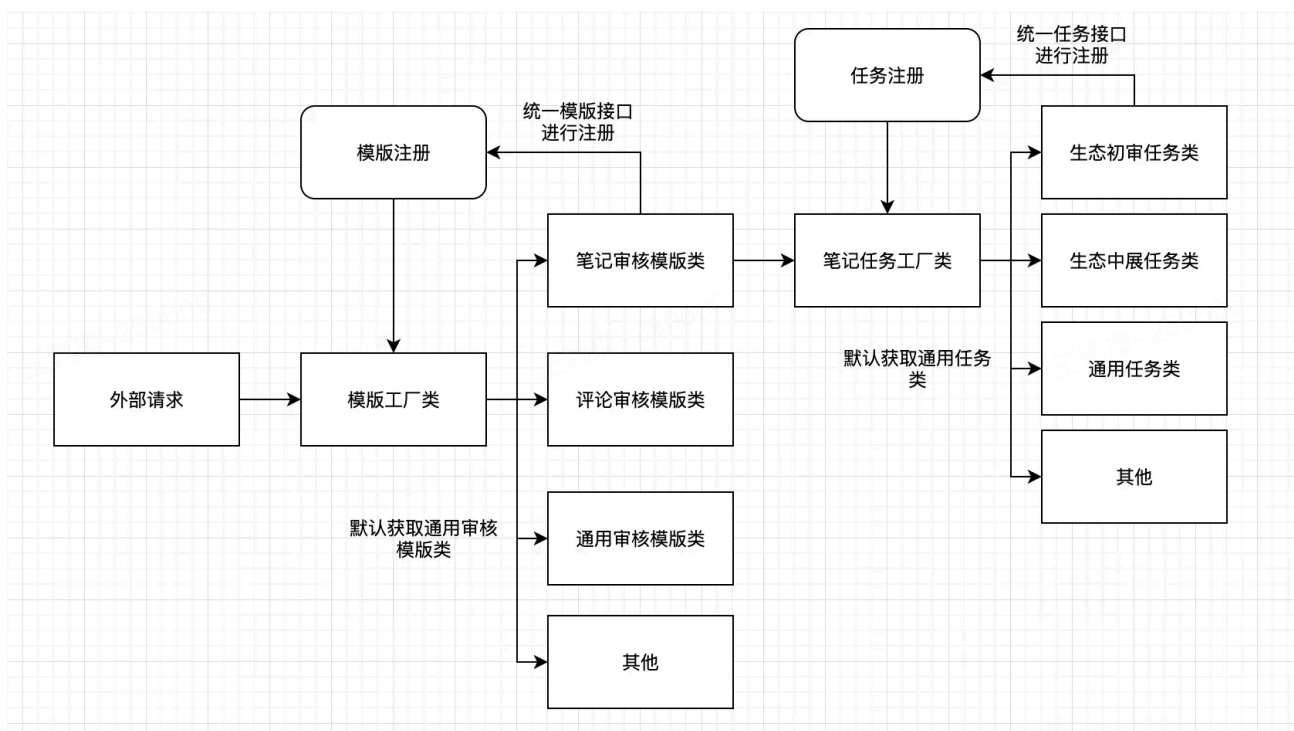


核心代码梳理

Mars

【代码逻辑】Mars核心代码讲解



请求传递流程：

1. 外部请求由MarsServiceImpl负责，调用 addTask，pullTask，finshTask等等

1. addTask：进审相关

2. pullTask：领取任务

3. disposalTask：处置相关，请求到这里以后先做校验，然后发异步mq返回。如果发mq失败，或者这个消息就是mq解析过来的，或者有强制处置标识，执行真正的disposal（子类实现

4. finishTask：完成任务相关，更新common_task_record信息，无特殊逻辑

5. allowTask 哪些队列的任务由哪个具体的service执行

2. 进入TaskManger，从HandleFactory获取handler，默认获取CommonHandler，通过allowHandler(String categoryType)匹配

3. Handle类实现HandlerInterface接口自动实现注册,保存在handlerInterfaceList中，一般是继承AbstractHandler，复用代码逻辑，同样会自动注册

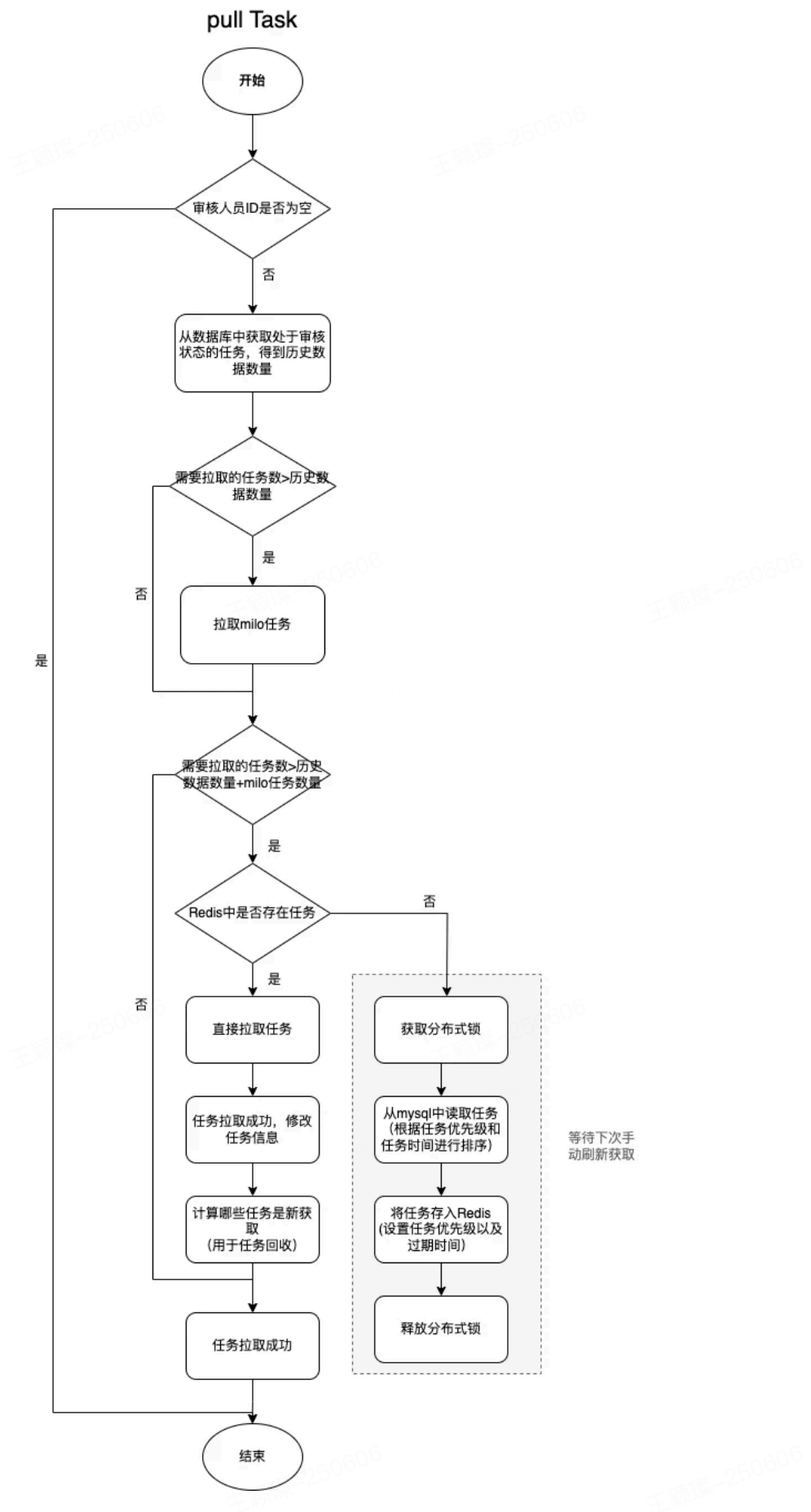
4. Handler中再调用taskFactory获取task：getCommonTask->getTask->getTaskByConfig

5. 一般是根据categoryType和sourceType去匹配寻找

6. 具体的TaskService在TaskInterFace接口中注册，一般是直接继承AbstractXXTaskService

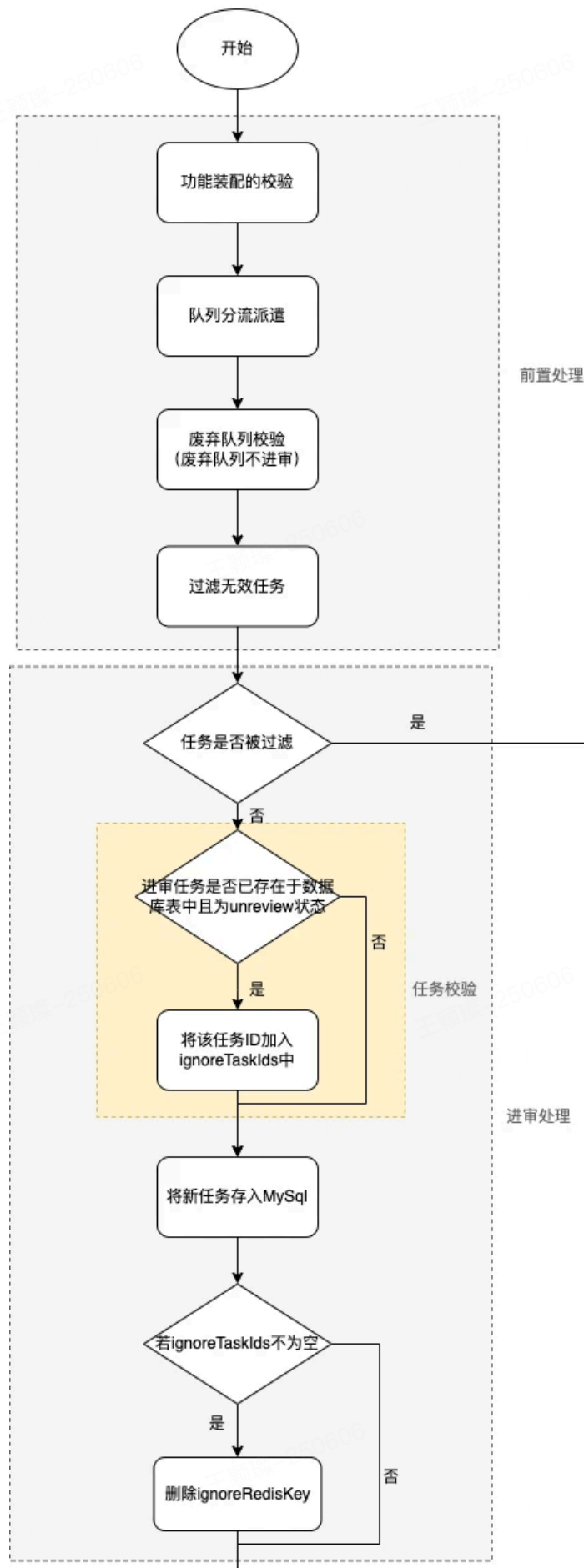
mars核心接口：pullTask、addTask、finishTask、disposalTask

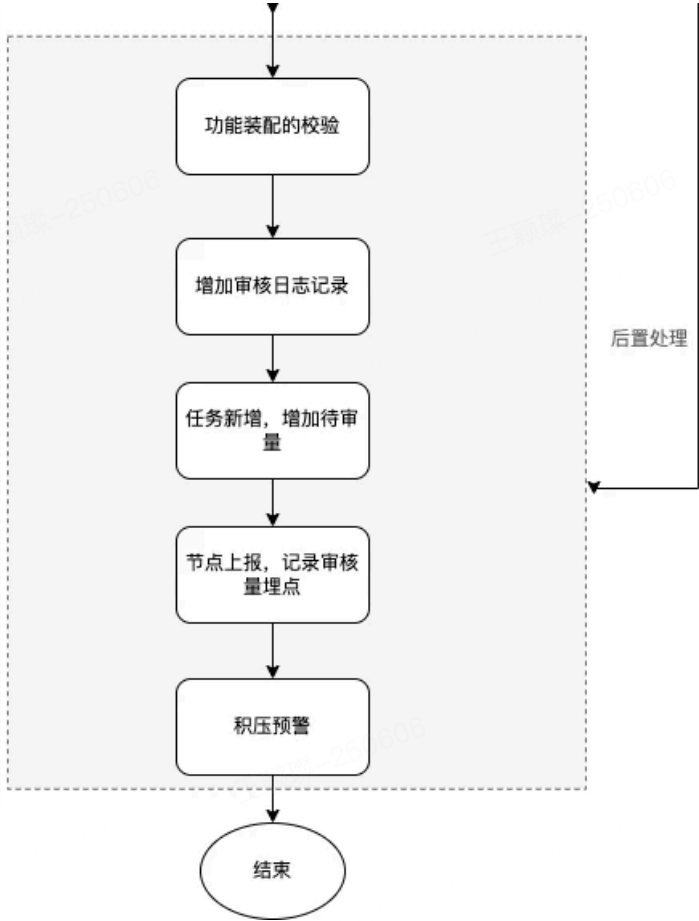
pullTask:



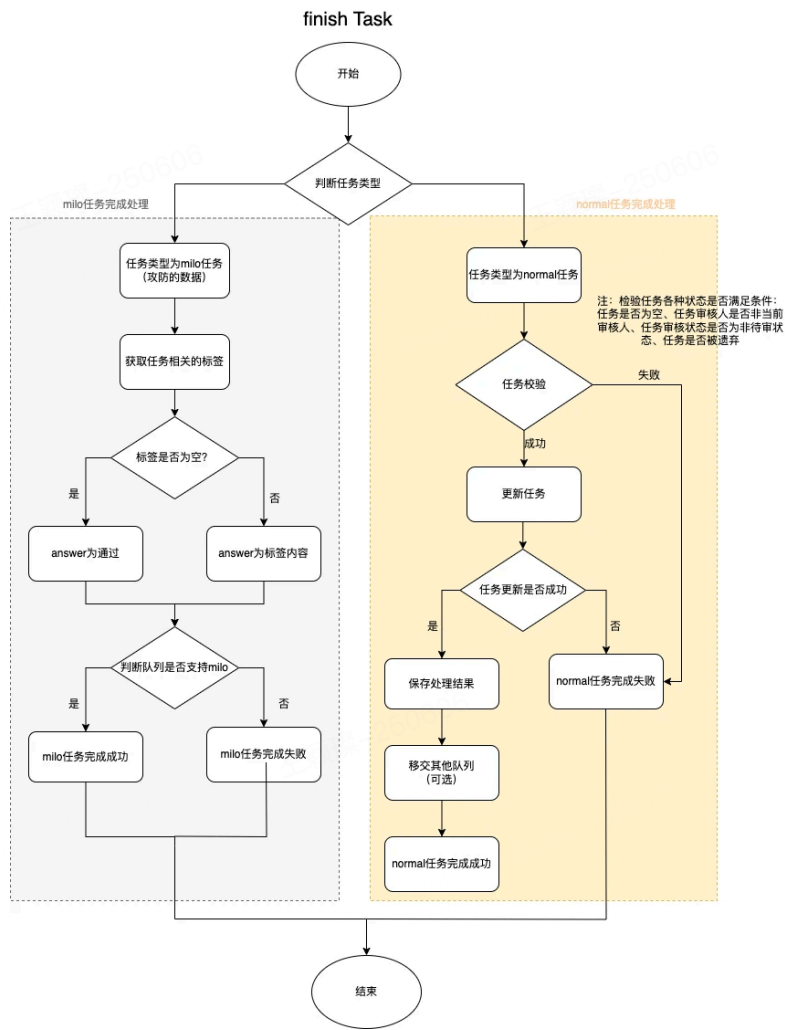
addTask:

add Task

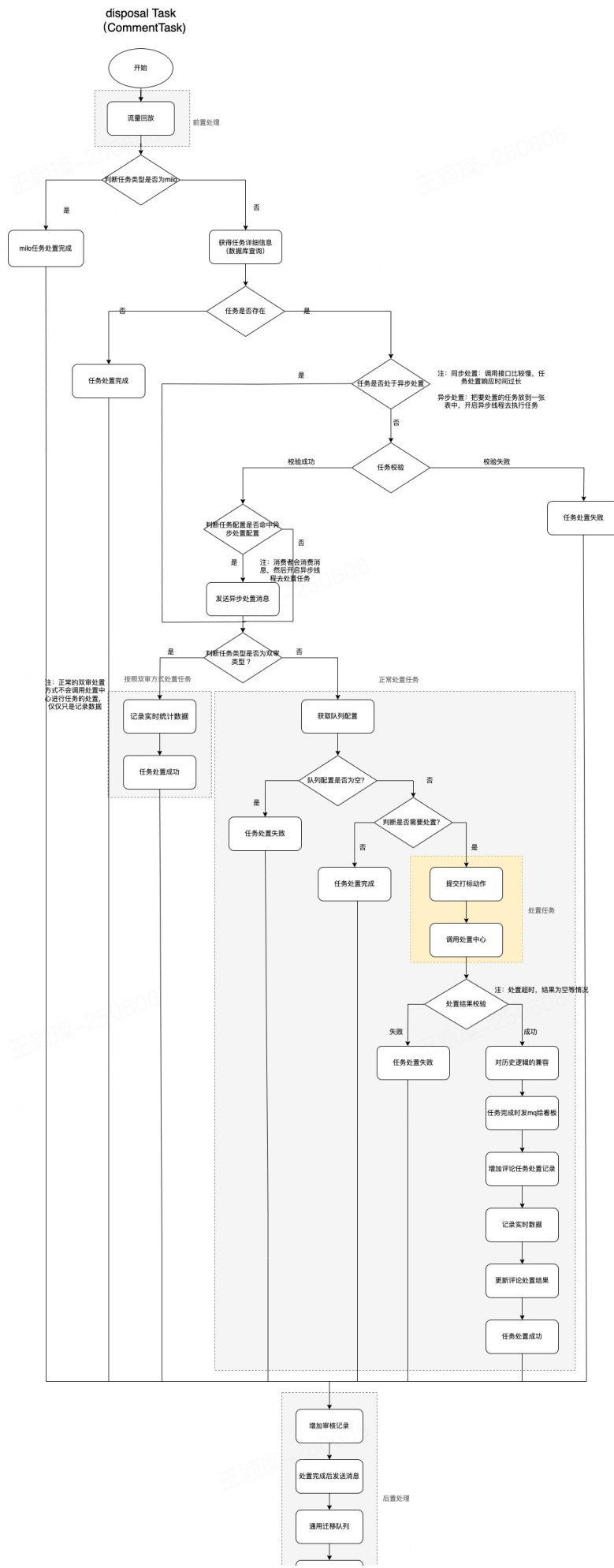


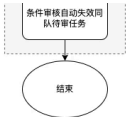


finishTask

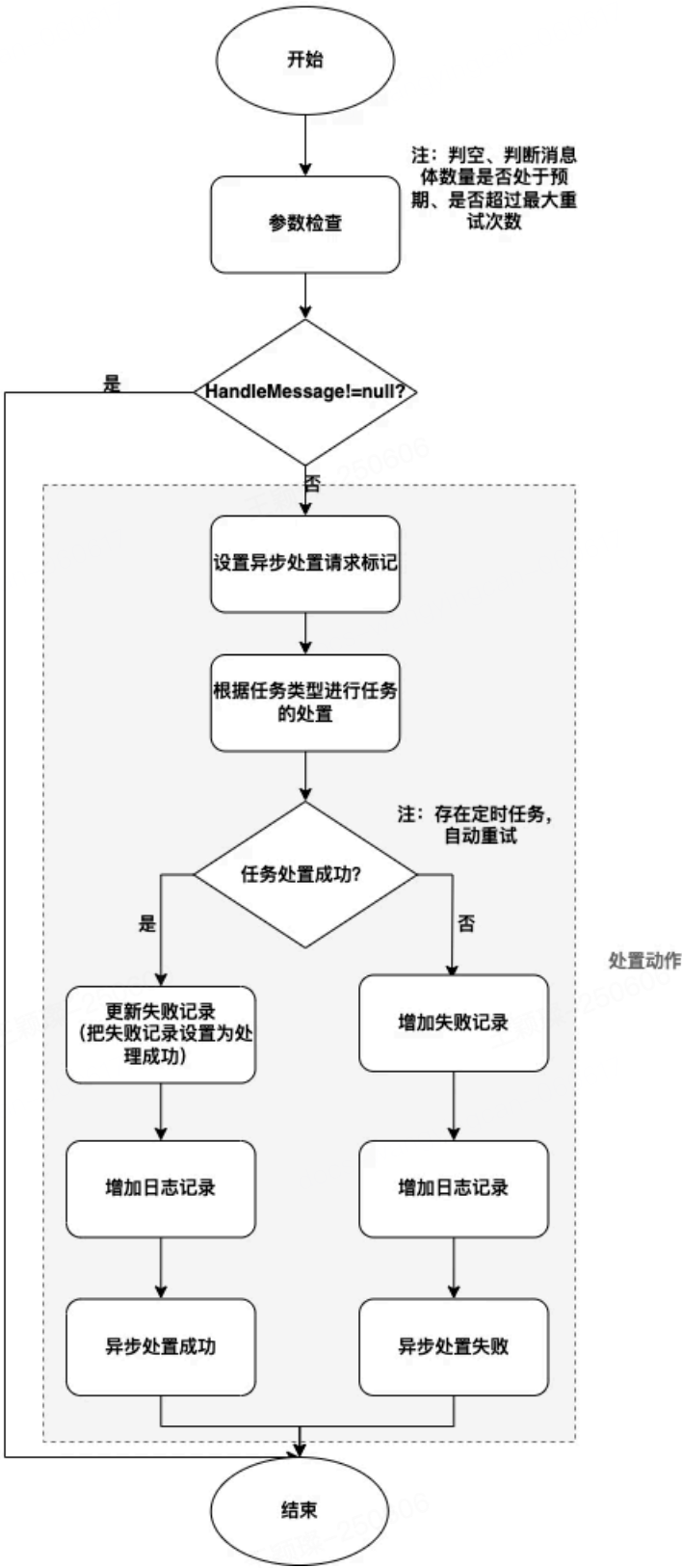


disposalTask:

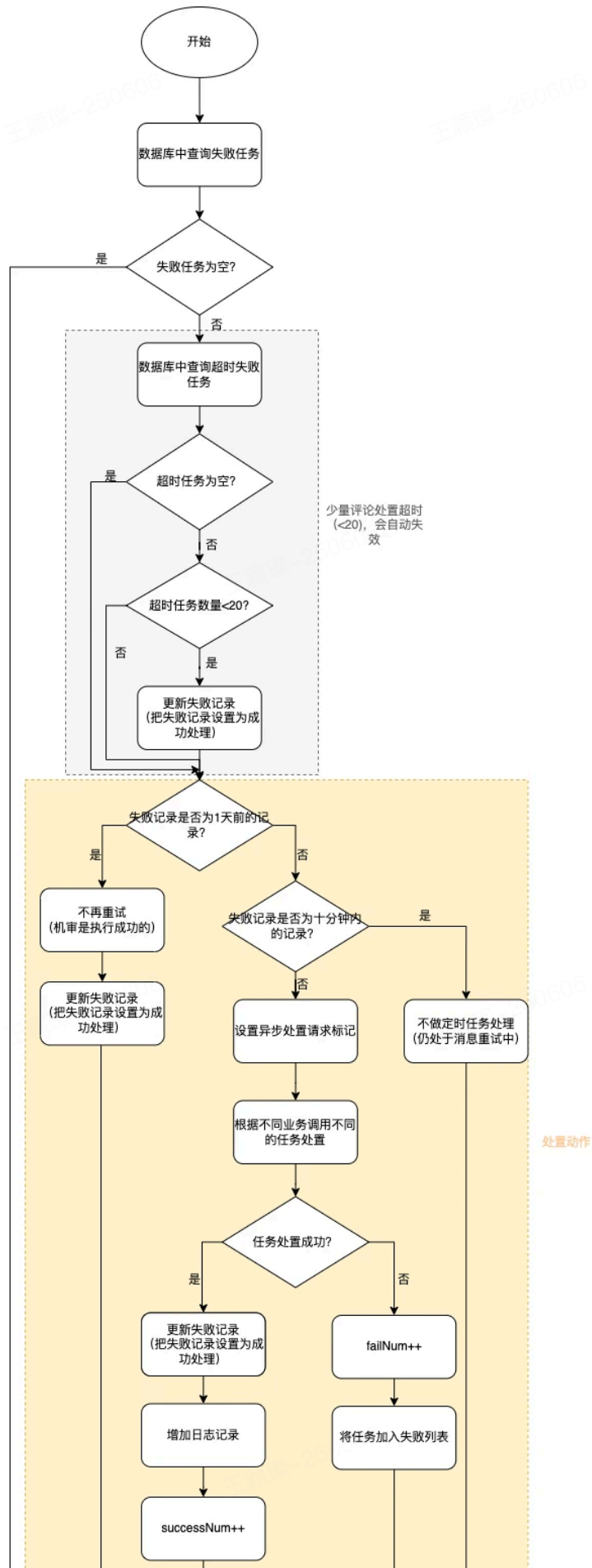


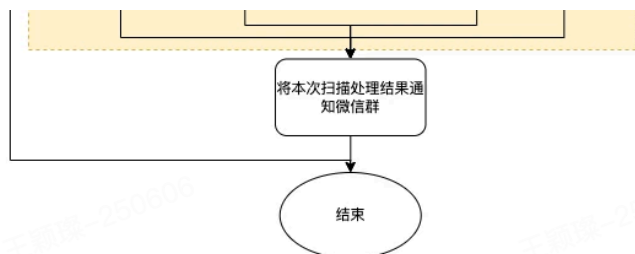


CommentAsyncDisposal
(异步处置消息)



AsyncDisposalRetryJobHandler (定时任务 异步处置失败重试)





QA @邢彩霄（实习）

1. pullTask 中如果想通过任务优先级拉取任务，是怎么实现的？

答：如上图所示，pullTask阶段，处于Review状态的任务优先级最高，其次是milo任务，最后是Redis中存储的任务，根据需要拉取的任务数量来判断是否要拉取milo任务或者从Redis中拉取任务。

2. addTask 如何实现任务的前置处理，后置处理逻辑？

答：如上图所示，addTask阶段，前置处理中，主要的操作包括队列分流、废弃队列的检验以及无效任务的过滤。在后置处理中，主要的操作包括增加审核日志记录、增加任务待审量、节点上报、记录审核量埋点以及积压预警。

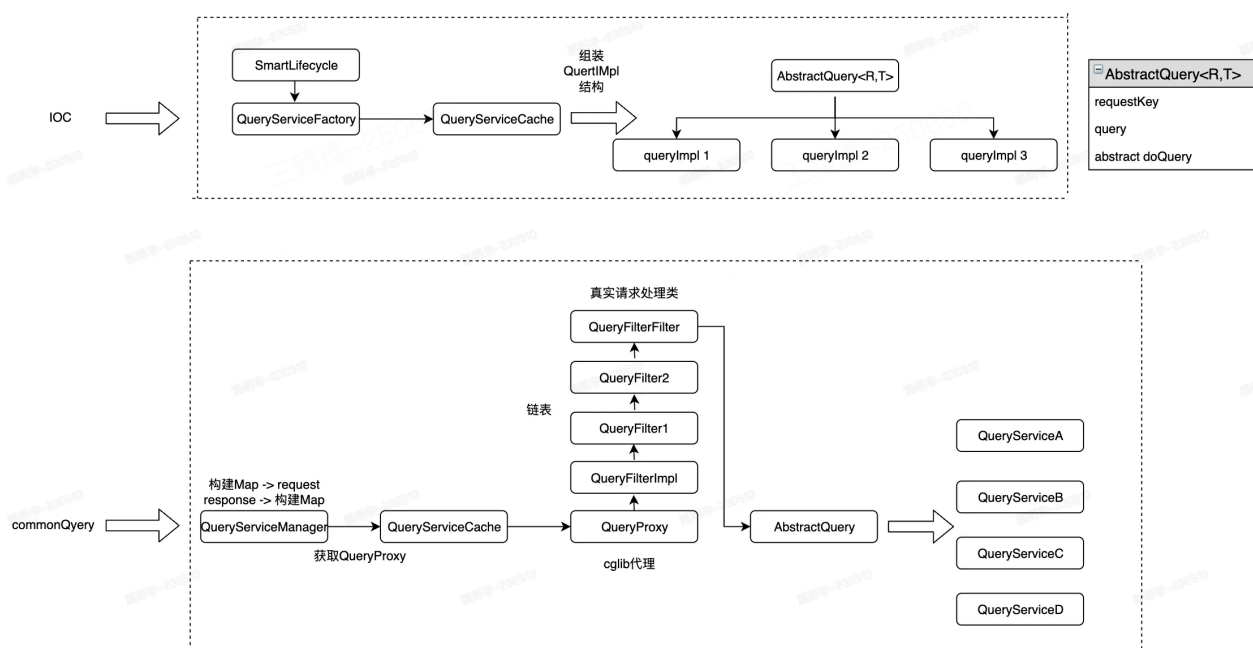
3. disposalTask 处置最终调用的什么方法，怎么实现审核结果的生效？

答：disposalTask中，主要做的操作是提交打标动作，然后通过调用处置中心来实现审核结果的生效。

4. disposalTask、finishTask 的区别是什么？ 分别用来做什么？

答：disposalTask中所做的操作主要包括提交打标动作和调用处置中心。finishTask中所做的操作主要包括任务表的更新以及处置结果的保存，比如新增记录到业务处置记录表以及属性类型处置记录表。

Volcan



VolcanServiceImpl对外暴露接口，前端主要调用commonQuery/commonObjectQuery方法进行查询
调用QueryServiceManager进行objectQuery/query


Java ▾

 复制代码

```
1 queryServiceManager.query(commonObjectQueryRequest.getRequestKey(), params);
```

objectQuery/query从QueryServiceCache中获取到queryService以及查询配置CommonQueryBO

Java ▾


 复制代码

```
1 CommonQueryBO commonQueryBO = queryServiceCache.getCommonQueryBOMap().get(requestKey);
2 Object request = buildObjectQueryRequest(commonQueryBO, params);
3 //使用CGILB动态代理获取服务
4 AbstractQuery queryService = (AbstractQuery)
    queryServiceCache.getQueryProxyMap().get(requestKey);
```

内部的impl中的query类继承AbstractQuery之后实现具体查询逻辑

要写新的查询业务直接继承 Abstractquery即可,在enity包中编写自定义的CommonQueryResponse子类以及request类

Java ▾

 复制代码

```
1 public abstract class AbstractQuery<R, T extends CommonQueryResponse> {
    public abstract T doQuery(R request) throws Exception;
    public abstract String requestKey();//这个必须要写,用来查找这个服务
    public T query(R request) throws Exception {
        return doQuery(request);
    }
    public T objectQuery(R request, Map<String, Object> originParams) throws Exception
    {return query(rebuildRequest(request, originParams));}
    public R rebuildRequest(R request, Map<String, Object> originParams) {
        return request;
    }
}
```