Camunda Optimizations

Saturday, 15 April 2023 9:25 AM

https://camunda.com/bpmn/reference/#activities-call-activity
https://docs.camunda.org/manual/7.18/user-guide/process-engine/process-engine-api/

Use Optimistic locking QUERY API built by the team -> it has lot of cool features

Job executors thread - reduce time to make timer trigger accurate

https://camunda.com/blog/2019/10/job-executor-what-is-going-on-in-my-process-en
https://docs.camunda.org/manual/latest/user-guide/spring-boot-integration/configur

- Job acquisition thread
- Job execution thread
- Wait time
- Lock time
- Min - max thread size in pool

- Make the history state as **activity** to avoid storing more data in history tables
- Decouple notifications using asynchronous before/after checkbox in modeler.
- Use run time table itself for fetching any data when it is in active state
- Terminate event to close any active instances with timer of 14 days
- Multi instance cardinality
- increase Job executor pool size to make more threads available to pick all jobs
- Make Asynchronous before and after to make it as individual transaction

Camunda External task Client

https://medium.com/@dashedsouvik/camunda-external-task-pattern-fd84a29d9d3e

```
<dependency>
<groupId>org.camunda.bpm.springboot</groupId>
<artifactId>camunda-bpm-spring-boot-starter-external-task-client</artifa
<version>7.15.0-alpha5</version>
</dependency>
```

Create process instance:

```
ProcessInstanceprocessInstance=runtimeService.startProcessInstanceByKey(pr
requestEvent.getBusinessKey(),requestEvent.getVariables());
```

getActiveInstanceByBusinessKey:

ngine/
ration/

ctId>

ocessDefKey,

getActiveInstanceByBusinessKey:

```
ProcessInstanceQueryinstanceQuery=runtimeService.createProcessInstanceQu
.processInstanceBusinessKey(businessKey).processDefinitionKey(processDefini
processInstance=instanceQuery.active().singleResult();
```

User task claim:

```
TaskQuerytaskQuery=taskService.createTaskQuery().processInstanceId(process
.taskDefinitionKey(userTaskName).active();
if(assigned){
userTask=taskQuery.taskAssigned().singleResult();
}else{
userTask=taskQuery.taskUnassigned().singleResult();
//claim/unclaim
taskService.claim(task.getId(),userId));
taskService.claim(task.getId(),null));
```

User task complete:

```
taskService.complete(task.getId(),variables)
```

Message event subscription:

```
Optional<EventSubscription>
eventSubscription=Optional.ofNullable(runtimeService.createEventSubscriptio
.processInstanceId(processInstanceId).eventName(msgName).eventType(EVEN
AGE_TYPE).singleResult());

eventSubscription.ifPresentOrElse(msg->runtimeService.messageEventReceive
msg.getExecutionId(),messageEvent.getVariables()),
```

Message event receive:

```
longmsgCount=runtimeService.createEventSubscriptionQuery()
.processInstanceId(processInstanceId).eventName(msgName).eventType(EVEN
AGE_TYPE).count();

if(msgCount>0){
runtimeService.createMessageCorrelation(msgName).processInstanceId(proce
es(variables).correlateAll();
}.
```

```
uery()
tionKey);


sInstanceId)


nQuery()
ITSUBSCRIPTION_MESS

ed(msg.getEventName(),


ITSUBSCRIPTION_MESS


ssInstanceId).setVariabl
```

```
};
```

History activity instance:

```
historicActivityInstance=historyService.createHistoricActivityInstanceQuery()
.processInstanceId(processInstanceId)
.activityId(activityId).list().stream().reduce((first,second)->second).orElse(null);
```

fixedValue/Expression

```
Private FixedValue taskStatus;
Private Expression taskStatusExpression;

taskStatusExpression.getExpressionText())){
taskStatusValue=String.valueOf(taskStatusExpression.getValue(execution)
```

ExecutionListeners/UserTaskListeners:

```
TaskStatusEventProducerExecutionTaskListener
implementsExecutionListener,TaskListener{

@Override
Public void notify(DelegateTask delegateTask){
```

Assignment types: asignment/complete/create/delete/update

```
@Override
Public void notify(DelegateExecution execution){
```

Service Task/Java Delegate Class

```
Public class NotificationServiceDelegate implements JavaDelegate{

@Override
Public void execute(DelegateExecution delegateExecution){
```

Timer:

Duration: PT5M/P45D

Cycle: R10/PT1M

Placeholder: ${timerExpiryValue}

Listener type:

Java

Service Task/Java Delegate Class

```
Public class NotificationServiceDelegate implements JavaDelegate{

@Override
Public void execute(DelegateExecution delegateExecution){
```

Timer:
Duration: PT5M/P45D
Cycle: R10/PT1M
Placeholder: ${timerExpiryValue}

Listener type:
Java
Expression: ${execution.setVariable('completionStatus', 'COMPLETED')}
Delegate expression:

Pom dependency:

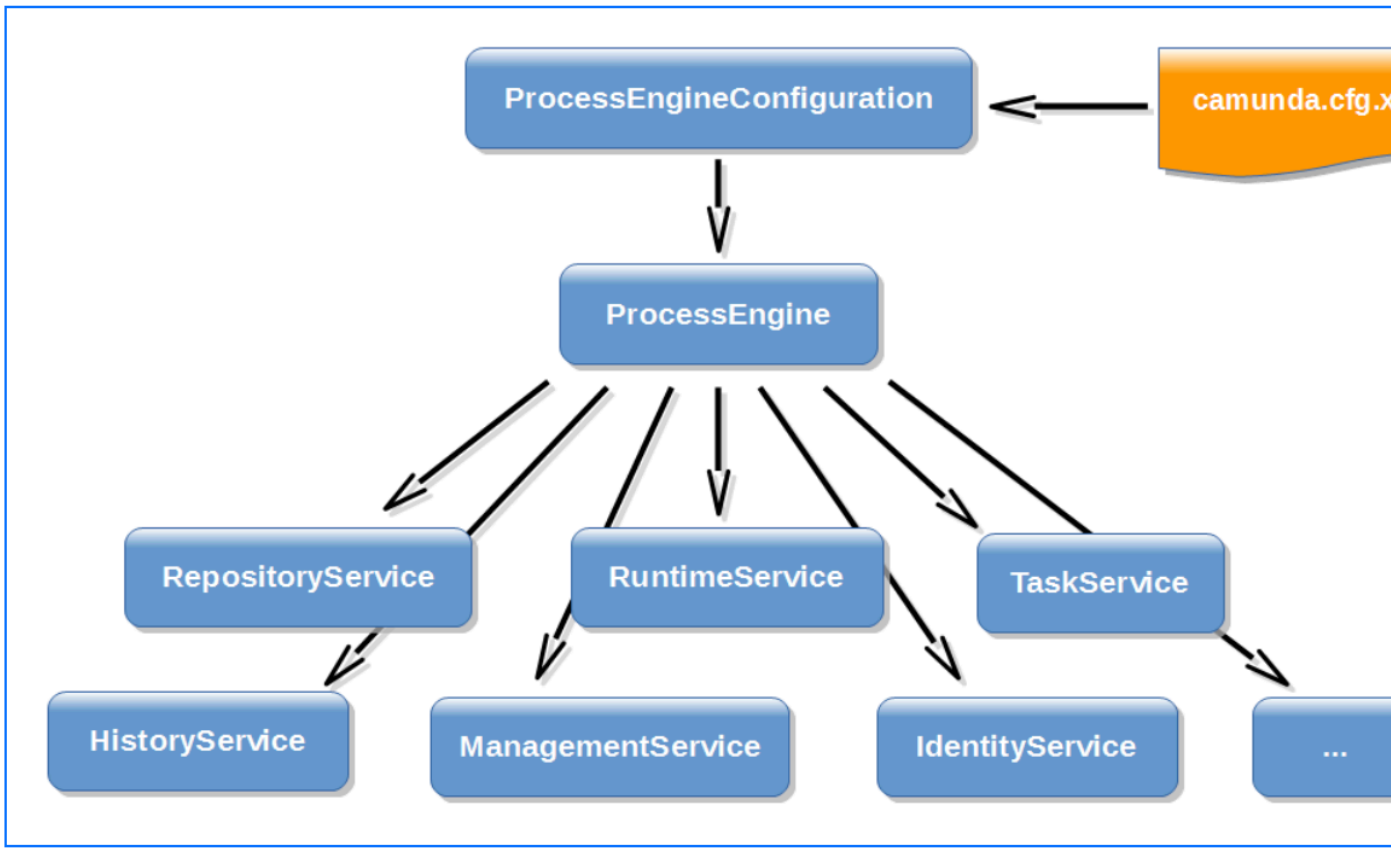
```
<dependency>
<groupId>org.camunda.bpm.springboot</groupId>
<artifactId>camunda-bpm-spring-boot-starter-webapp</artifactId>
<version>7.13.0</version>
</dependency>
<dependency>
<groupId>org.camunda.bpm.springboot</groupId>
<artifactId>camunda-bpm-spring-boot-starter-rest</artifactId>
<version>7.13.0</version>
</dependency>
```

BPMN error throw/catch event:

```
if(!
ObjectUtils.isEmpty(suppressException)&&"true".equals(suppressExcep
tion.getExpressionText())){
Throw new
BpmnError(STAMP_ERROR_NOTIFICATION_SERVICE_API,"errorinsendin
gnotification",baseException);
```

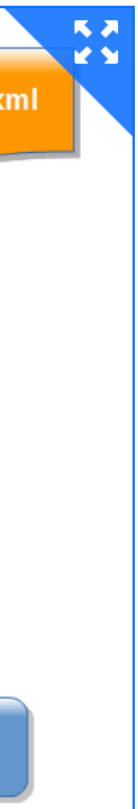
Conditional intermediate catch event:
#{statusCode=='1100.70.06.10' }

Interrupting boundary events - message/timers
Non-Interrupting boundary events - message/timers




```
ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine();

RepositoryService repositoryService = processEngine.getRepositoryService();
RuntimeService runtimeService = processEngine.getRuntimeService();
TaskService taskService = processEngine.getTaskService();
IdentityService identityService = processEngine.getIdentityService();
FormService formService = processEngine.getFormService();
HistoryService historyService = processEngine.getHistoryService();
ManagementService managementService = processEngine.getManagementService();
FilterService filterService = processEngine.getFilterService();
ExternalTaskService externalTaskService = processEngine.getExternalTaskServi
CaseService caseService = processEngine.getCaseService();
DecisionService decisionService = processEngine.getDecisionService();
```




```
Lce();
```