

Java Activiti User Guide Digest

2018-07-03
2018
July
Week 27
Day 03
Tue

目次

frontPage
・さくらVPS
・Federal1
・SuSe18
・Docker
・Ansible
・Java
・Scala
・Python
・Ruby
・Lisp
・Computer
・GIS
・HTML
・Culture
・Random
・Link

訪問者

total: 3811
today: 21
yesterday: 11
now: 0

更新

最新010件

2018-07-02
Python
CuPy on
MacBookPro
Python
2018-06-28
Thought 近
は誰にでもIT
の環境
2018-06-25
Random
Memorandum
201801
2018-06-24
Mac 数式の入
った資料を作
る
QnigGraffleLaTeX
2018-06-23
Mac
2018-06-12
Books 深淵
7日 Once
Learning
2018-06-07
Thought 文
学211
2018-05-30
Culture
Python
BatuLatiB

1. Introduction
2. Getting Started
3. Configuration
 - 3.16. Loggin
 - 3.18. Event handlers
4. The Activiti API
 - 4.1~4.3 ワークフローを実行するコンソールアプリ
 - 4.3.4.プロセスの Suspend と Activate
 - 4.4 Query API
 - 4.5.Variables
 - 4.7 Unit Test
 - 4.9 The process engine in a web application
6. Deployment
- 7.標準 BPMN 2.0
- 8.Activiti 拡張 BPMN 2.0
 - 8.2.Event
 - 8.3 Sequence Flow
 - 8.4 Gateways
 - 8.5 Task
 - 8.15.12.Execution listener
 - 8.5.13. Task listener
 - 8.5.14 Multi-instance (for each).
 - 8.6. Sub-Processes and Call Activities
9. Form
- 11.History
13. Activiti Explorer 14. Modeler
15. REST API

本家 : <http://www.activiti.org/userguide/>

1. Introduction ¹

- Activiti 本体 Apache License V2
- Activiti Modeler LGPL 2.1
- <http://activiti.org/download.html>
- <https://github.com/Activiti/Activiti>
- JDK6+
- Eclipse designer plugin
 - <http://www.eclipse.org/downloads/> (最新の Mars でも OK だった)
 - <http://activiti.org/designer/update/> (Plugin Repository)

2. Getting Started ¹

- Tutorial
 - Java6 + Tomcat + Activiti-Explorer
- Demo database setup flag
 - Activiti Explorer や Rest API では、後述する標準の `activiti.cfg.xml` の代わりに `db.properties`, `engine.properties` でアプリケーションの設定を行う
 - `WEB-INF/classes/db.properties`

```
1 db=h2
2 jdbc.driver=org.h2.Driver
3 jdbc.url=jdbc:h2:mem:activiti;DB_CLOSE_DELAY=1000
4 jdbc.username=sa
5 jdbc.password=
```

- `WEB-INF/classes/engine.properties`

```
1 # demo data properties
2 create.demo.users=true
3 create.demo.definitions=true
4 create.demo.models=true
5 create.demo.reports=true
6
7 # engine properties
```

A 通知機能

FrontPage/activiti

Activiti

Eclipse/Activiti

Activiti/activiti

Jakarta

activiti-xml

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

Eclipse/Activiti

```

8 engine.schema.update=true
9 engine.activate.jobexecutor=false
10 engine.asyncexecutor.enabled=true
11 engine.asyncexecutor.activate=true
12 engine.history.level=full
13
14 # email properties
15 #engine.email.enabled=true
16 #engine.email.host=localhost
17 #engine.email.port=1025

```

3. Configuration

- 独自の Workflow アプリ ⇒ [ProcessEngine](#) を使ってワークフローを操作

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

- デフォルト設定では Classpath にある `activiti.cfg.xml` または `activiti-context.xml` の設定に基づいて [ProcessEngine](#) が作られる
(設定内容をプログラム内で明示的に設定することも可。後述)
- `activity.cfg.xml`

<https://github.com/kagyuu/ActivitiExam/blob/master/src/main/resources/activiti.cfg.xml>

```

1 <beans xmlns="http://www.springframework.org/schema/beans"
2     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans"
4
5     <!-- 3.2.processEngineConfiguration
6     Stand alone app      : org.activiti.engine.impl.cfg.StandaloneProcessEngineConfiguration
7     Test (Use in-memory h2) : org.activiti.engine.impl.cfg.StandaloneInMemProcessEngineConfiguration
8     Spring integration    : org.activiti.spring.SpringProcessEngineConfiguration
9     Stand alone app with JTA : org.activiti.engine.impl.cfg.JtaProcessEngineConfiguration
10    -->
11    <bean id="processEngineConfiguration"
12        class="org.activiti.engine.impl.cfg.StandaloneInMemProcessEngineConfiguration">
13
14        <!-- 3.3-3.8.database connection
15        support h2, mysql, oracle, postgres, db2, mssql
16        -->
17        <property name="jdbcUrl" value="jdbc:h2:mem:activiti;DB_CLOSE_DELAY=1000" />
18        <property name="jdbcDriver" value="org.h2.Driver" />
19        <property name="jdbcUsername" value="sa" />
20        <property name="jdbcPassword" value="" />
21        <!-- Optional database connection properties * The unit of XXXTime is millisec *
22        <property name="jdbcMaxActiveConnections" value="10" />
23        <property name="jdbcMaxIdleConnections" value="" />
24        <property name="jdbcMaxCheckoutTime" value="20000" />
25        <property name="jdbcMaxWaitTime" value="20000" />
26        -->
27        <!-- Schema Update
28        false (default) : If the db schema version != activiti lib version then abend
29        true             : If the db schema version != activiti lib version then update db schema
30        create-drop      : Always create and drop schema
31        <property name="databaseSchemaUpdate" value="false" />
32        -->
33
34        <!-- 3.9-3.11 Job Executor
35        By default, the JobExecutor is activated when the process engine boots.
36        By default, the AsyncExecutor is not enabled and the JobExecutor is used
37        due to legacy reasons.
38        It's however recommended to use the new AsyncExecutor instead.
39        -->
40        <!--
41        <property name="jobExecutorActivate" value="false" />
42        <property name="asyncExecutorEnabled" value="true" />
43        <property name="asyncExecutorActivate" value="true" />
44        -->
45
46        <!-- 3.12. Mail Configuration (for email task)
47        =====
48        property name      Default value
49        =====
50        mailServerHost     localhost
51        mailServerPort     25
52        mailServerDefaultFrom activiti@activiti.org
53        mailServerUsername (not set)
54        mailServerPassword (not set)
55        mailServerUseSSL   (not set)
56        mailServerUseTLS   (not set)
57        =====
58        -->
59        <property name="mailServerHost" value="localhost" />
60        <property name="mailServerPort" value="50025" />
61
62        <!-- 3.13. History (for logging task execution)
63        <property name="history" value="audit" />

```

```

64      -->
65
66      <!-- 3.15. Cache (default is no limit)
67      <property name="processDefinitionCacheLimit" value="10" />
68      -->
69    </bean>
70
71  </beans>

```

- Spring の bean.xml 形式だが、文法を借りているだけ。Activiti に Spring が必須ではない
- JNDI から DataSource を lookup して利用することも可能
- Activiti Explorer、Activity Rest API は、db.properties で接続先を設定するようになっている。→ 内部的には、[その他の ProcessEngine 作成方法](#)

• [その他の ProcessEngine 作成方法](#)

1. 設定ファイルを明示する

```

ProcessEngine processEngine
= ProcessEngineConfiguration
    .createProcessEngineConfigurationFromInputStream(inputStream).buildProcessEngine()

```

2. プログラム中で設定を行う (Activiti-Explorer、Activiti-Rest はこのやり方)

- デフォルト設定

```

ProcessEngineConfiguration.createStandaloneProcessEngineConfiguration();
ProcessEngineConfiguration.createStandaloneInMemProcessEngineConfiguration();

```

- デフォルト設定に個別設定を追加

```

ProcessEngine processEngine
= ProcessEngineConfiguration
    .createStandaloneInMemProcessEngineConfiguration()
    .setDatabaseSchemaUpdate(ProcessEngineConfiguration.DB_SCHEMA_UPDATE_FALSE)
    .setJdbcUrl("jdbc:h2:mem:my-own-db;DB_CLOSE_DELAY=1000")
    .setAsyncExecutorEnabled(true)
    .setAsyncExecutorActivate(false)
    .buildProcessEngine();

```

- [ProcessEngine2](#) は、Singleton。設定ファイルを読み直すには

```

ProcessEngine.destory();
ProcessEngine.init();

```

を実行する必要あり

3.16. Login [↑]

- Activiti は SLF4J にログを書いている
- SLF4J は、出力先がないときは NOP-logger にログを送っているため、結果的にログは出ない
 - SLF4J 推奨の logback を Classpath に追加すれば logback からログが出る

```

pom.xml
<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
</dependency>

```

- log4j でログを出力させるときには SLF4J-log4j ブリッジを Classpath に追加すれば良い

```

pom.xml
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-log4j12</artifactId>
</dependency>

```

Activiti Explorer、Activity Rest API はこの方式

- Spring (commons-logging) に統合したいときは

```

pom.xml
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>jcl-over-slf4j</artifactId>
</dependency>

```

- Activiti は、SLF4J の Mapped Diagnostic Contexts に、実行中のワークフローの情報を詰め込んでいるので、ログ出力の設定 (logback.xml など) で適宜拾うことができる → 【留意点あり】

```
<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
  <withJansi>true</withJansi>
  <encoder>
    <pattern>%date [%thread] %highlight(%-5level) %-20logger{20} %mdc{mdcProcessDefinitionID} %msg %n
  </encoder>
</appender>
```

- processDefinition Id as mdcProcessDefinitionID
- processInstance Id as mdcProcessInstanceId
- execution Id as mdcExecutionId
- 【留意点】
 - 実装は、<http://activiti.org/javadocs/org/activiti/engine/logging/LogMDC.html>
 - Activiti の本体コードを見てみると、タスク開始時に一律 MDC にタスク情報が設定されて、終了時にクリアされるわけではないらしい。最初そう思ってたまく行かず、色々調べてみた。
 - どうやら、例外発生時に MDC にプロセス情報が設定されるようだ。きっと、障害解析用の仕組みなんだろう
 - 業務ログとしてこういう情報が欲しければ、自前で 3.18. Event handler を作って MDC にタスク情報を格納する必要あり
 - LogMDC.putMDCExecution(ActivitiExecution e) の引き数は、Activiti の内部クラス org.activiti.engine.impl.pvm.delegate.ActivitiExecution であり、ユーザプログラムからは取得できない。LogMDC を強制初期化するようなユーザプログラム (イベントハンドラ) は作ることが出来ない

3.18. Event handlers [↑]

- [ProcessEngine?](#) の起動・停止、JOB の開始・終了イベントをフックできる
 - 3.18.6. Supported event types
 - [EventListener?](#) は [ProcessEngine?](#) のインスタンスのもちもの。DBを共有して同じJOBを二つの [ProcessEngine?](#) をまたいで実行した場合、それぞれの [EventListener?](#) はそれぞれの [ProcessEngine?](#) (の実行環境にある activiti.cfg.xml) に定義されているもの
- [EventListener?](#) 実装
 - インタフェース org.activiti.engine.delegate.event.[ActivitiEventListener?](#) を実装する
 - 3.18.1. Event listener implementation
- [EventListener?](#) 定義
 - activiti.cfg.xml に定義する
 - 全イベントを拾う 3.18.2. Configuration and setup

```
1 <bean id="processEngineConfiguration" class="org.activiti.engine.impl.cfg.StandaloneProcessEn
2 ...
3 <property name="eventListeners">
4   <list>
5     <bean class="org.activiti.engine.example.MyEventListener" />
6   </list>
7 </property>
8 </bean>
```

- 特定のイベントだけを拾う

```
1 <bean id="processEngineConfiguration" class="org.activiti.engine.impl.cfg.StandaloneProcessE
2 ...
3 <property name="typedEventListeners">
4   <map>
5     <entry key="JOB_EXECUTION_SUCCESS,JOB_EXECUTION_FAILURE" >
6       <list>
7         <bean class="org.activiti.engine.example.MyJobEventListener" />
8       </list>
9     </entry>
10  </map>
11 </property>
12 </bean>
```

- 処理中でイベントリスナを設定 3.18.3. Adding listeners at runtime
- 特定のプロセスだけ拾う 3.18.4. Adding listeners to process definitions

4. The Activiti API [↑]

- <http://activiti.org/javadocs/index.html>
- アプリの中では [ProcessEngine?](#) のインスタンスは一つだけにする。Thread Safe になっている。

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

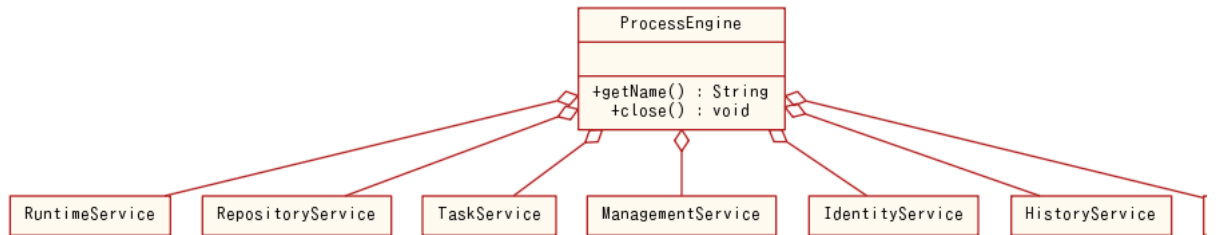
- 何回 [ProcessEngines?](#).[getDefaultProcessEngine?](#)(); を呼び出しても返ってくるインスタンスは同じ (設定ファイルを毎回読まない)
- 設定ファイルを読み直すには

```
ProcessEngine.destory();
ProcessEngine.init();
```

を実行する必要あり

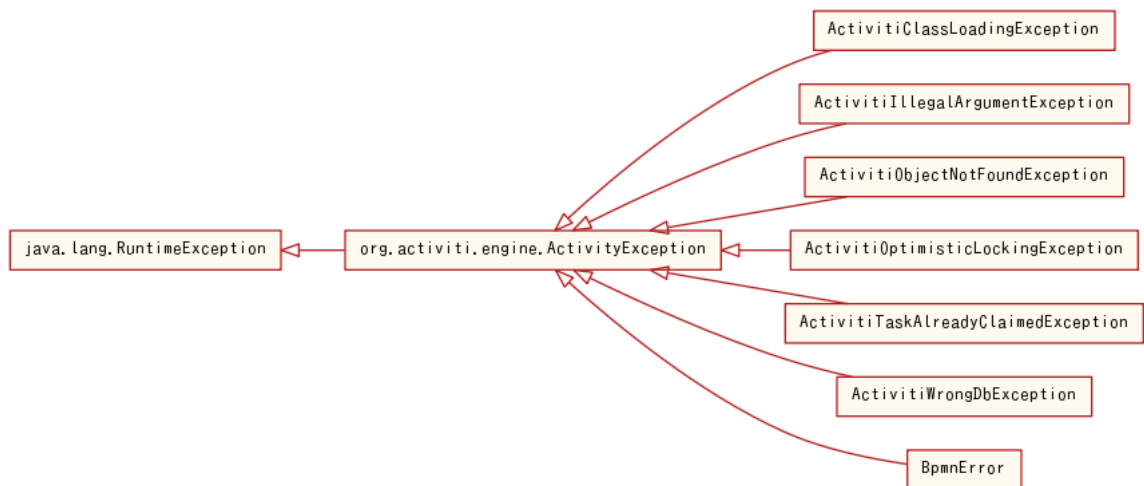
- [ProcessEngine?](#) から `getXXXService()` で、Service API を取得できる

| | | |
|------------------------------------|--|---------------------|
| RuntimeService? | <code>processEngine.getRuntimeService?</code> | プロセス管理 |
| RepositoryService? | <code>processEngine.getRepositoryService?</code> | BPMN 管理。配備とか |
| TaskService? | <code>processEngine.getTaskService?</code> | ユーザ操作の実現 (エミュレーション) |
| ManagementService? | <code>processEngine.getManagementService?</code> | Activiti 自体の管理 |
| IdentityService? | <code>processEngine.getIdentityService?</code> | 認証 |
| HistoryService? | <code>processEngine.getHistoryService?</code> | タスク処理の記録をDBに永続化 |
| FormService? | <code>processEngine.getFormService?</code> | タスク開始・終了時のユーザ確認 |



- <http://activiti.org/javadocs/org/activiti/engine/ProcessEngine.html>
- どのサービスもStateless ⇒ JOB管理DBが共有でもクラスタで同時実行可能

- Exception



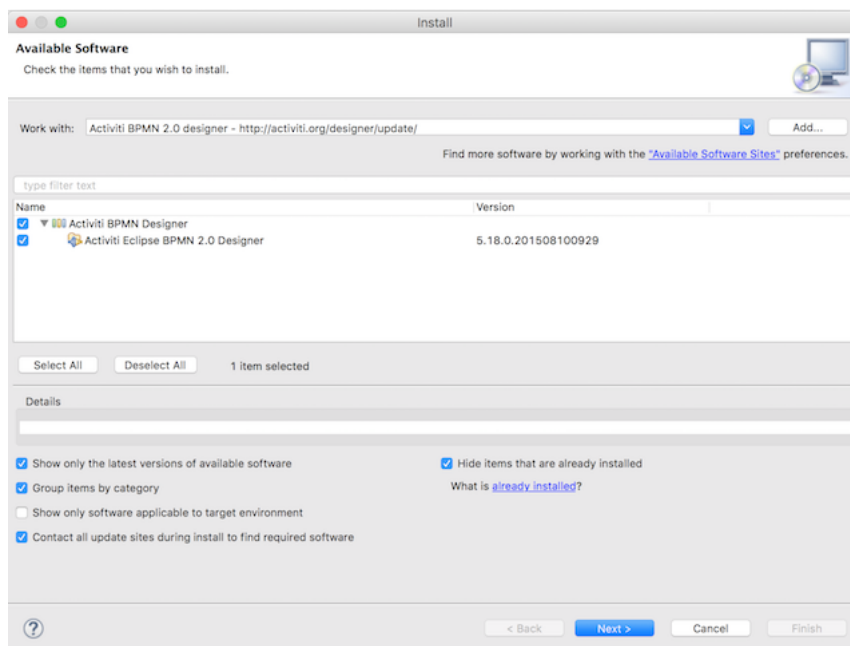
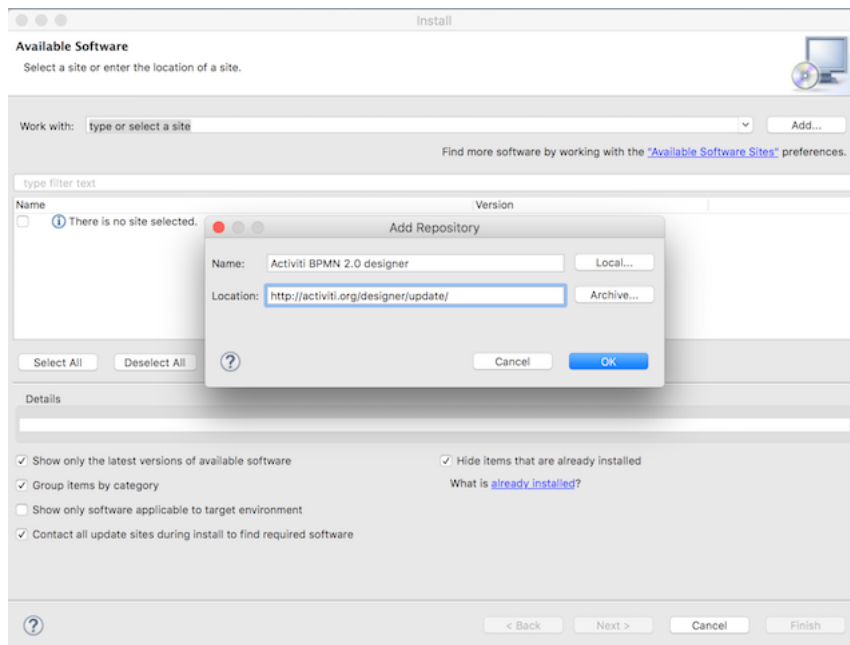
- `org.activiti.engine.ActivityException?` の子
- <http://activiti.org/javadocs/org/activiti/engine/ActivityException.html>
- It's `RuntimeException?`.

4.1~4.3 ワークフローを実行するコンソールアプリ [↑]

開発環境 [↑]

- <https://eclipse.org/> 最新版(2015時点)のMarsでOK
- plugin

| | |
|----------|---|
| Name | Activiti BPMN 2.0 designer |
| Location | http://activiti.org/designer/update/ |



- BPMN を GUI で編集するために Activiti BPMN 2.0 designer が必要ないのであれば Eclipse でなくてもいい。プログラム本体を作るだけなら Java の Maven Project が扱えればどんな IDE でもいい。

サンプルプロジェクト [↑]

- 普通の Maven Project
- pom.xml

<https://github.com/kagyuu/ActivitiExam/blob/master/pom.xml>

```

1  <?xml version="1.0"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
4      <modelVersion>4.0.0</modelVersion>
5      <groupId>com.snail.exam</groupId>
6      <artifactId>ActivitiExam</artifactId>
7      <version>0.0.1-SNAPSHOT</version>
8      <dependencies>
9          <dependency>
10             <groupId>org.activiti</groupId>
11             <artifactId>activiti-engine</artifactId>
12             <version>5.19.0</version>
13          </dependency>
14          <dependency>
15             <groupId>ch.qos.logback</groupId>
16             <artifactId>logback-classic</artifactId>
17             <version>1.1.3</version>
18          </dependency>
19          <dependency>
20             <groupId>com.h2database</groupId>
21             <artifactId>h2</artifactId>

```

```

22     <version>1.4.190</version>
23   </dependency>
24 </dependencies>
25   <build>
26     <plugins>
27       <!-- Make JAR OSGi Bundle Ready -->
28       <plugin>
29         <groupId>org.apache.felix</groupId>
30         <artifactId>maven-bundle-plugin</artifactId>
31         <version>2.3.7</version>
32         <extensions>true</extensions>
33       </plugin>
34       <!-- Compile -->
35       <plugin>
36         <groupId>org.apache.maven.plugins</groupId>
37         <artifactId>maven-compiler-plugin</artifactId>
38         <version>3.3</version>
39         <configuration>
40           <source>1.8</source>
41           <target>1.8</target>
42         </configuration>
43       </plugin>
44     </plugins>
45   </build>
46   <repositories>
47     <repository>
48       <!-- http://www.activiti.org/community.html -->
49       <id>Alfresco Maven Repository</id>
50       <url>https://maven.alfresco.com/nexus/content/groups/public/</url>
51     </repository>
52   </repositories>
53 </project>

```

- レポジトリに Alfresco (Activiti開発元) のレポジトリを追加
- 依存ライブラリに Activiti Engine を設定する
- その他に、ログ出力用に logback、テスト用 DB に H2

activiti.cfg.xml ¹

ここでは、テスト用の Table をロードする設定(StandaloneInMemProcessEngineConfiguration²)を使う

<https://github.com/kagyuu/ActivitiExam/blob/master/src/main/resources/activiti.cfg.xml>

```

1  <beans xmlns="http://www.springframework.org/schema/beans"
2    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans.xsd">
4
5    <!-- 3.2.processEngineConfiguration
6    Stand alone app      : org.activiti.engine.impl.cfg.StandaloneProcessEngineConfiguration
7    Test (Use in-memory h2) : org.activiti.engine.impl.cfg.StandaloneInMemProcessEngineConfiguration
8    Spring integration   : org.activiti.spring.SpringProcessEngineConfiguration
9    Stand alone app with JTA : org.activiti.engine.impl.cfg.JtaProcessEngineConfiguration
10   -->
11   <bean id="processEngineConfiguration"
12     class="org.activiti.engine.impl.cfg.StandaloneInMemProcessEngineConfiguration">
13
14     <!-- 3.3-3.8.database connection
15     support h2, mysql, oracle, postgres, db2, mssql
16     -->
17     <property name="jdbcUrl" value="jdbc:h2:mem:activiti;DB_CLOSE_DELAY=1000" />
18     <property name="jdbcDriver" value="org.h2.Driver" />
19     <property name="jdbcUsername" value="sa" />
20     <property name="jdbcPassword" value="" />
21     <!-- Optional database connection properties * The unit of XXXTime is millisec *
22     <property name="jdbcMaxActiveConnections" value="10" />
23     <property name="jdbcMaxIdleConnections" value="" />
24     <property name="jdbcMaxCheckoutTime" value="20000" />
25     <property name="jdbcMaxWaitTime" value="20000" />
26     -->
27     <!-- Schema Update
28     false (default) : If the db schema version != activiti lib version then abend
29     true             : If the db schema version != activiti lib version then update db schema
30     create-drop      : Always create and drop schema
31     <property name="databaseSchemaUpdate" value="false" />
32     -->
33
34     <!-- 3.9-3.11 Job Executor
35     By default, the JobExecutor is activated when the process engine boots.
36     By default, the AsyncExecutor is not enabled and the JobExecutor is used
37     due to legacy reasons.
38     It?'s however recommended to use the new AsyncExecutor instead.
39     -->
40     <!--
41     <property name="jobExecutorActivate" value="false" />
42     <property name="asyncExecutorEnabled" value="true" />
43     <property name="asyncExecutorActivate" value="true" />
44     -->
45
46     <!-- 3.12. Mail Configuration (for email task)

```



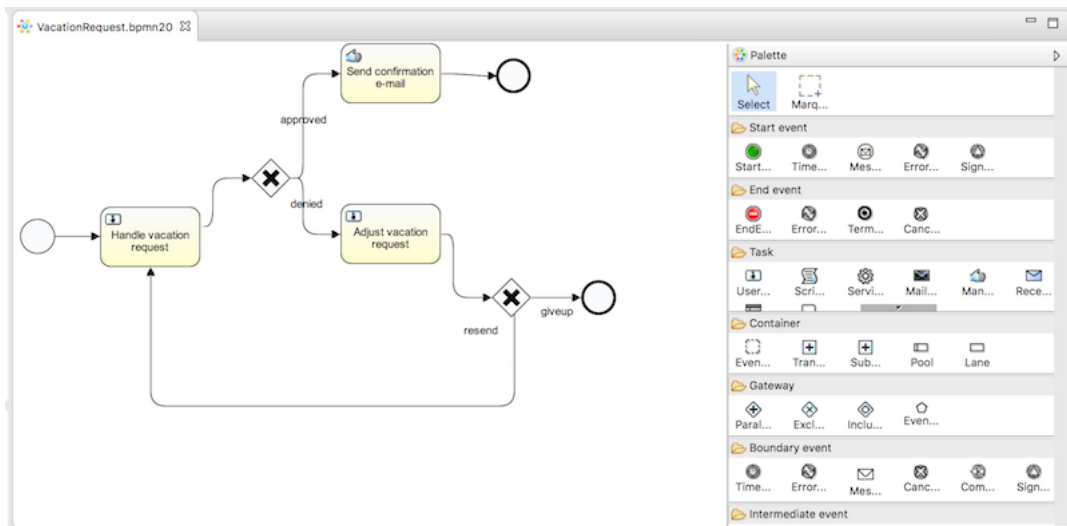
```

47 |=====|=====|
48 |property name      |Default value|
49 |=====|=====|
50 |mailServerHost     |localhost    |
51 |mailServerPort     |25           |
52 |mailServerDefaultFrom |activiti@activiti.org|
53 |mailServerUsername  |(not set)    |
54 |mailServerPassword  |(not set)    |
55 |mailServerUseSSL    |(not set)    |
56 |mailServerUseTLS    |(not set)    |
57 |=====|=====|
58 |-->
59 |<property name="mailServerHost" value="localhost" />
60 |<property name="mailServerPort" value="50025" />
61 |
62 |<!-- 3.13. History (for logging task execution)
63 |<property name="history" value="audit" />
64 |-->
65 |
66 |<!-- 3.15. Cache (default is no limit)
67 |<property name="processDefinitionCacheLimit" value="10" />
68 |-->
69 |</bean>
70 |
71 |</beans>

```

テスト用BPMN (VacationRequest?.bpmn20.xml) [↑]

[ActivitiExplorerのデモアプリ](#) で使った休暇申請のワークフローを /src/main/resources/org/activiti/test に配置



Java コンソールアプリ [↑]

BPMNの配備(DB格納) → プロセス開始 (休暇申請) → 否決 → 再申請の流をやる

<https://github.com/kagyu/ActivitiExam/blob/master/src/main/java/com/snail/exam/MyProcess.java>

```

1 | package com.snail.exam;
2 |
3 | import java.util.HashMap;
4 | import java.util.List;
5 | import java.util.Map;
6 |
7 | import org.activiti.engine.ProcessEngine;
8 | import org.activiti.engine.ProcessEngines;
9 | import org.activiti.engine.RepositoryService;
10 | import org.activiti.engine.RuntimeService;
11 | import org.activiti.engine.TaskService;
12 | import org.activiti.engine.repository.ProcessDefinition;
13 | import org.activiti.engine.runtime.ProcessInstance;
14 | import org.activiti.engine.task.Task;
15 | import org.apache.commons.lang3.time.DateUtils;
16 | import org.slf4j.Logger;
17 | import org.slf4j.LoggerFactory;
18 |
19 | public class MyProcess {
20 |
21 |     private static final Logger log = LoggerFactory.getLogger(MyProcess.class);
22 |
23 |     public static void main(String[] args) {
24 |         try {
25 |             ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine();

```



```

26
27 // ----- 4.3.1. Deploying the process
28 log.info("--- #1. Deploying the process");
29 RepositoryService repositoryService = processEngine.getRepositoryService();
30 repositoryService.createDeployment()
31     .addClasspathResource("org/activiti/test/VacationRequest.bpmn20.xml")
32     .deploy();
33
34 log.info("Number of process definitions {}", repositoryService.createProcessDefinitionQuery().list().size());
35 for (ProcessDefinition p : repositoryService.createProcessDefinitionQuery().list()) {
36     log.info("PROCESS DEF [id={},name={},key={}]", p.getId(), p.getName(), p.getKey());
37 }
38
39 // ----- 4.3.2. Starting a process instance
40 log.info("--- #2. Starting a process instance");
41
42 // VacationRequest.bpmn20.xml L3-10
43 // -----
44 // <process id="vacationRequest" name="Vacation request" isExecutable="true">
45 //     ^^^^^^^^^^^^^^^^^
46 //     <startEvent id="request" activiti:initiator="employeeName">
47 //         ^^^^^^^*1 ^^^^^^^^^^^^^^ $employeeName = the user
48 //     <extensionElements>
49 //         <activiti:formProperty id="numberOfDays" name="Number of days" type="long" required="true">
50 //             ^^^^^^^^^^^^^^
51 //         </activiti:formProperty>
52 //         <activiti:formProperty id="startDate" name="First day of holiday (dd-MM-yyyy)" type="date">
53 //             ^^^^^^^^^
54 //             datePattern="dd-MM-yyyy hh:mm" required="true"></activiti:formProperty>
55 //         <activiti:formProperty id="vacationMotivation" name="Motivation" type="string">
56 //             ^^^^^^^^^^^^^^^^^
57 //         </activiti:formProperty>
58 //     </extensionElements>
59 // </startEvent>
60 // <activiti:formProperty>
61 Map<String, Object> variables = new HashMap<String, Object>();
62 variables.put("employeeName", "Kermit");
63 variables.put("numberOfDays", new Integer(4));
64 variables.put("startDate", DateUtils.parseDate("1999-12-31", "yyyy-MM-dd"));
65 variables.put("vacationMotivation", "I'm really tired!");
66
67 // the process to run
68 // id : <process id="vacationRequest">
69 // arguments : <activiti:formProperty>
70 RuntimeService runtimeService = processEngine.getRuntimeService();
71 ProcessInstance processInstance = runtimeService.startProcessInstanceByKey("vacationRequest", variables);
72
73 log.info("Number of process instances: " + runtimeService.createProcessInstanceQuery().list().size());
74 for (ProcessInstance p : runtimeService.createProcessInstanceQuery().list()) {
75     log.info("PROCESS INSTANCE [id={},pid={},pname={},pkey={}]",
76         p.getId(),
77         p.getProcessDefinitionId(),
78         p.getProcessDefinitionName(),
79         p.getProcessDefinitionKey());
80 }
81
82 // ----- 4.3.3. Completing tasks (Reject Request)
83 // VacationRequest.bpmn20.xml L11-21
84 // -----
85 // <sequenceFlow id="flow1" sourceRef="request" targetRef="handleRequest"></sequenceFlow>
86 //     ^^^^^^^*1 ^^^^^^^^^^^^^^*2
87 // <userTask id="handleRequest" name="Handle vacation request" activiti:candidateGroups="{}">
88 //     ^^^^^^^^^^^^^^^^^*2
89 //     <documentation>${employeeName} would like to take ${numberOfDays} day(s) of vacation.
90 //     (Motivation: ${vacationMotivation}).</documentation>
91 //     <extensionElements>
92 //         <activiti:formProperty id="vacationApproved" name="Do you approve this vacation request?" type="boolean">
93 //             ^^^^^^^^^^^^^^^^^
94 //             required="true">
95 //                 <activiti:value id="true" name="Approve"></activiti:value>
96 //                 <activiti:value id="false" name="Reject"></activiti:value>
97 //             </activiti:formProperty>
98 //         <activiti:formProperty id="managerMotivation" name="Motivation" type="string">
99 //             ^^^^^^^^^^^^^^^^^
100 //     </extensionElements>
101 // </userTask>
102 log.info("--- #3. Completing tasks (Reject Request)");
103
104 // Fetch all tasks for the management group
105 TaskService taskService = processEngine.getTaskService();
106 List<Task> tasks = taskService.createTaskQuery().taskCandidateGroup("management").list();
107 for (Task task : tasks) {
108     if (task.getProcessDefinitionId().startsWith("vacationRequest")){
109         if (task.getTaskDefinitionKey().equals("handleRequest")) {
110             // Description is <documentation>.
111             log.info("TASK REJECT REQ [{}]", task.getDescription());
112
113             // Do task (reject application)
114             Map<String, Object> taskVariables = new HashMap<String, Object>();
115             taskVariables.put("vacationApproved", "false");
116             taskVariables.put("managerMotivation", "We have a tight deadline!");
117             taskService.complete(task.getId(), taskVariables);

```

```

118     }
119     }
120 }
121
122 // ----- 4.3.3. Completing tasks (Adjust rejected request)
123 // VacationRequest.bpmn20.xml L22-23
124 // -----
125 // <sequenceFlow id="flow2" sourceRef="handleRequest" targetRef="requestApprovedDecision"
126 //                                     ^^^^^^^^^^^^^^^^^*2 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
127 // <exclusiveGateway id="requestApprovedDecision" name="Request approved?"></exclusiveGateway>
128 //                                     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*3
129 //
130 // VacationRequest.bpmn20.xml L30-35
131 // -----
132 // <sequenceFlow id="flow5" name="denied" sourceRef="requestApprovedDecision" targetRef="adjustVacationRequestTask"
133 //                                     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*3
134 //                                     <conditionExpression xsi:type="tFormalExpression"><![CDATA[ ${vacationApp
135 //                                     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
136 //                                     </conditionExpression>
137 //                                     </sequenceFlow>
138 //                                     <userTask id="adjustVacationRequestTask" name="Adjust vacation request" activiti:
139 //                                     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*4
140 //                                     <documentation>Your manager has disapproved your vacation request for ${
141 //                                     Reason: ${managerMotivation}</documentation>
142 log.info("--- #4. Completing tasks (Adjust rejected request)");
143
144 taskService = processEngine.getTaskService();
145 tasks = taskService.createTaskQuery().active().list();
146 for (Task task : tasks) {
147     if (task.getProcessDefinitionId().startsWith("vacationRequest")){
148         if (task.getTaskDefinitionKey().equals("adjustVacationRequestTask")) {
149             // Description is <documentation>.
150             log.info("ADJUST REJECT REQ [{}]", task.getDescription());
151         }
152     }
153 }
154
155 } catch (Throwable th) {
156     log.error("ERROR", th);
157 }
158 }
159 }
160 }

```

• 実行ログ

```

1 16-03-24 00:24:21 [INFO ] Initializing process engine using configuration 'file:/Users/atsushi/E
2 16-03-24 00:24:21 [INFO ] initializing process engine for resource file:/Users/atsushi/EclipseWo
3 16-03-24 00:24:21 [INFO ] Loading XML bean definitions from resource loaded through InputStream
4 16-03-24 00:24:22 [INFO ] performing create on engine with resource org/activiti/db/create/activ
5 16-03-24 00:24:22 [INFO ] performing create on history with resource org/activiti/db/create/acti
6 16-03-24 00:24:22 [INFO ] performing create on identity with resource org/activiti/db/create/act
7 16-03-24 00:24:22 [INFO ] ProcessEngine default created
8 16-03-24 00:24:22 [INFO ] initialised process engine default
9 16-03-24 00:24:22 [INFO ] --- #1. Deploying the process
10 16-03-24 00:24:22 [INFO ] Processing resource org/activiti/test/VacationRequest.bpmn20.xml
11 16-03-24 00:24:23 [INFO ] Number of process definitions 1
12 16-03-24 00:24:23 [INFO ] PROCESS DEF [id=vacationRequest:1:4,name=Vacation request,key=vacation
13 16-03-24 00:24:23 [INFO ] --- #2. Starting a process instance
14 16-03-24 00:24:23 [INFO ] Number of process instances: 1
15 16-03-24 00:24:23 [INFO ] PROCESS INSTANCE [id=5,pid=vacationRequest:1:4,pname=Vacation request,
16 16-03-24 00:24:23 [INFO ] --- #3. Completing tasks (Reject Request)
17 16-03-24 00:24:23 [INFO ] TASK REJECT REQ [Kermit would like to take 4 day(s) of vacation (Motiv
18 16-03-24 00:24:23 [INFO ] --- #4. Completing tasks (Adjust rejected request)
19 16-03-24 00:24:23 [INFO ] ADJUST REJECT REQ [Your manager has disapproved your vacation request
20 Reason: We have a tight deadline!]

```

I got it.

BPMNの配備 (RepositoryService?) ↑

```

1 log.info("--- #1. Deploying the process");
2 RepositoryService repositoryService = processEngine.getRepositoryService();
3 repositoryService.createDeployment()
4     .addClasspathResource("org/activiti/test/VacationRequest.bpmn20.xml")
5     .deploy()

```

プロセス開始 (RuntimeService?) ↑

```

1 Map<String, Object> variables = new HashMap<String, Object>();
2 variables.put("employeeName", "Kermit");
3 variables.put("numberOfDays", new Integer(4));
4 variables.put("startDate", DateUtils.parseDate("1999-12-31", "yyyy-MM-dd"));
5 variables.put("vacationMotivation", "I'm really tired!");
6
7 RuntimeService runtimeService = processEngine.getRuntimeService();
8 ProcessInstance processInstance = runtimeService.startProcessInstanceByKey("vacationRequest", variab

```

- <http://activiti.org/javadocs/org/activiti/engine/RuntimeService.html>
- プロセスIDを指定してプロセスを開始する。入力値がある場合には Map 形式で投入する
- `RuntimeService.startProcessInstanceByKey(String processDefinitionId, Map<String, Object> variables)` でプロセスを開始する
 - `processDefinitionId` : BPMN の process id

```
<process id="vacationRequest" name="Vacation request" isExecutable="true">
```

- `variables` : BPMN の <startEvent> に定義された入力値

```
<startEvent id="request" activiti:initiator="employeeName">
  <extensionElements>
    <activiti:formProperty id="numberOfDays" name="Number of days" type="long" required="true"></activiti:formProperty>
    <activiti:formProperty id="startDate" name="First day of holiday (dd-MM-yyyy)" type="date"
      datePattern="dd-MM-yyyy hh:mm" required="true"></activiti:formProperty>
    <activiti:formProperty id="vacationMotivation" name="Motivation" type="string"></activiti:formProperty>
  </extensionElements>
</startEvent>
```

タスク実行 (TaskService?) ↑

```
1 TaskService taskService = processEngine.getTaskService();
2 List<Task> tasks = taskService.createTaskQuery().taskCandidateGroup("management").list();
3 for (Task task : tasks) {
4   if (task.getProcessDefinitionId().startsWith("vacationRequest")){
5     if (task.getTaskDefinitionKey().equals("handleRequest")) {
6       // Description is <documentation>.
7       log.info("TASK REJECT REQ [{}]", task.getDescription());
8
9       // Do task (reject application)
10      Map<String, Object> taskVariables = new HashMap<String, Object>();
11      taskVariables.put("vacationApproved", "false");
12      taskVariables.put("managerMotivation", "We have a tight deadline!");
13      taskService.complete(task.getId(), taskVariables);
14    }
15  }
16 }
```

- プロセスが開始されたり、タスクが実行されたら、後続処理は ユーザによる入力待ちまで `ProcessEngine` が実行する
- ユーザの入力待ちになっているタスクのリストから、今回実行するタスクを選んで実行する
 - manager 決済のタスクのうち
 - プロセスID が vacationRequest で始まり
 - タスク名が handleRequest なもの
 - ※ プロセスIDには、プロセス名に version と一意ID がつく。今回の場合 "vacationRequest:1:4" となる
→ cf. [6.Deployment](#)
- <http://activiti.org/javadocs/org/activiti/engine/TaskService.html>
- `TaskService.complete(String taskId, Map<String, Object> variables)` でタスクを実行する。入力値がある場合には Map 形式で投入する
 - `taskId` : `TaskService.createTaskQuery().list()` で取得したタスクのID
 - `variables` : BPMN の <userTask> に定義された入力値

```
<userTask id="handleRequest" name="Handle vacation request" activiti:candidateGroups="management">
  <documentation>${employeeName} would like to take ${numberOfDays} day(s) of vacation
    (Motivation: ${vacationMotivation}).</documentation>
  <extensionElements>
    <activiti:formProperty id="vacationApproved" name="Do you approve this vacation" type="enum" re
      <activiti:value id="true" name="Approve"></activiti:value>
      <activiti:value id="false" name="Reject"></activiti:value>
    </activiti:formProperty>
    <activiti:formProperty id="managerMotivation" name="Motivation" type="string"></activiti:formProperty>
  </extensionElements>
</userTask>
```

- <userTask> の <documentation> は `task.getDescription()` で取得できる

```
log.info("TASK REJECT REQ [{}]", task.getDescription());
```

4.3.4.プロセスの Suspend と Activate ↑

<https://github.com/kagyuu/ActivitiExam/blob/master/src/main/java/com/snail/exam/MyProcess2.java>

```

1 package com.snail.exam;
2
3 import java.util.HashMap;
4 import java.util.Map;
5
6 import org.activiti.engine.ActivitiException;
7 import org.activiti.engine.ProcessEngine;
8 import org.activiti.engine.ProcessEngines;
9 import org.activiti.engine.RepositoryService;
10 import org.activiti.engine.RuntimeService;
11 import org.activiti.engine.runtime.ProcessInstance;
12 import org.apache.commons.lang3.time.DateUtils;
13 import org.slf4j.Logger;
14 import org.slf4j.LoggerFactory;
15
16 public class MyProcess2 {
17
18     private static final Logger log = LoggerFactory.getLogger(MyProcess2.class);
19
20     public static void main(String[] args) {
21         try {
22             ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine();
23
24             // ----- 4.3.1. Deploying the process
25             log.info("--- #1. Deploying the process");
26             RepositoryService repositoryService = processEngine.getRepositoryService();
27             repositoryService.createDeployment()
28                 .addClasspathResource("org/activiti/test/VacationRequest.bpmn20.xml")
29                 .deploy();
30
31             // ----- 4.3.4. Suspending and activating a process
32             log.info("--- #2. Suspend Process");
33             repositoryService.suspendProcessDefinitionByKey("vacationRequest");
34
35             // ----- 4.3.2. Starting a process instance
36             log.info("--- #3. Starting a process instance (will fail)");
37
38             Map<String, Object> variables = new HashMap<String, Object>();
39             variables.put("employeeName", "Kermit");
40             variables.put("numberOfDays", new Integer(4));
41             variables.put("startDate", DateUtils.parseDate("1999-12-31", "yyyy-MM-dd"));
42             variables.put("vacationMotivation", "I'm really tired!");
43
44             // the process to run
45             // id : <process id="vacationRequest">
46             // arguments : <activiti:formProperty>
47             RuntimeService runtimeService = processEngine.getRuntimeService();
48             try {
49                 ProcessInstance processInstance = runtimeService.startProcessInstanceByKey("vacationRequest", variables);
50             } catch (ActivitiException e) {
51                 log.error("ERROR", e);
52             }
53
54             // ----- 4.3.4. Suspending and activating a process
55             log.info("--- #5. Activate Process");
56             repositoryService.activateProcessDefinitionByKey("vacationRequest");
57
58             log.info("--- #6. Starting a process instance (will success)");
59             try {
60                 ProcessInstance processInstance = runtimeService.startProcessInstanceByKey("vacationRequest", variables);
61                 log.info("{} was started", processInstance.getProcessDefinitionId());
62             } catch (ActivitiException e) {
63                 log.error("ERROR", e);
64             }
65
66         } catch (Throwable th) {
67             log.error("ERROR", th);
68         }
69     }
70 }

```

• 実行ログ

```

1 16-03-25 01:12:03 [INFO ] Initializing process engine using configuration 'file:/Users/atsushi/E
2 16-03-25 01:12:03 [INFO ] initializing process engine for resource file:/Users/atsushi/EclipseWo
3 16-03-25 01:12:03 [INFO ] Loading XML bean definitions from resource loaded through InputStream
4 16-03-25 01:12:04 [INFO ] performing create on engine with resource org/activiti/db/create/activ
5 16-03-25 01:12:04 [INFO ] performing create on history with resource org/activiti/db/create/acti
6 16-03-25 01:12:05 [INFO ] performing create on identity with resource org/activiti/db/create/acti
7 16-03-25 01:12:05 [INFO ] ProcessEngine default created
8 16-03-25 01:12:05 [INFO ] initialised process engine default
9 16-03-25 01:12:05 [INFO ] --- #1. Deploying the process
10 16-03-25 01:12:05 [INFO ] Processing resource org/activiti/test/VacationRequest.bpmn20.xml
11 16-03-25 01:12:06 [INFO ] --- #2. Suspend Process
12 16-03-25 01:12:06 [INFO ] --- #3. Starting a process instance (will fail)
13 16-03-25 01:12:06 [INFO ] Processing resource org/activiti/test/VacationRequest.vacationRequest.
14 16-03-25 01:12:06 [INFO ] Processing resource org/activiti/test/VacationRequest.bpmn20.xml
15 16-03-25 01:12:06 [ERROR] ERROR
16 org.activiti.engine.ActivitiException: Cannot start process instance. Process definition Vacatio
17 at org.activiti.engine.impl.cmd.StartProcessInstanceCmd.execute(StartProcessInstanceCmd.java
18 at org.activiti.engine.impl.cmd.StartProcessInstanceCmd.execute(StartProcessInstanceCmd.java
19 at org.activiti.engine.impl.interceptor.CommandInvoker.execute(CommandInvoker.java:24) ~[act

```

```

20      at org.activiti.engine.impl.interceptor.CommandContextInterceptor.execute(CommandContextInte
21      at org.activiti.engine.impl.interceptor.LogInterceptor.execute(LogInterceptor.java:31) ~[act
22      at org.activiti.engine.impl.cfg.CommandExecutorImpl.execute(CommandExecutorImpl.java:40) ~[a
23      at org.activiti.engine.impl.cfg.CommandExecutorImpl.execute(CommandExecutorImpl.java:35) ~[a
24      at org.activiti.engine.impl.RuntimeServiceImpl.startProcessInstanceByKey(RuntimeServiceImpl.
25      at com.snail.exam.MyProcess2.main(MyProcess2.java:49) ~[classes/:na]
26 16-03-25 01:12:06 [INFO ] --- #5. Activate Process
27 16-03-25 01:12:06 [INFO ] --- #6. Starting a process instance (will success)
28 16-03-25 01:12:06 [INFO ] Processing resource org/activiti/test/VacationRequest.vacationRequest.
29 16-03-25 01:12:06 [INFO ] Processing resource org/activiti/test/VacationRequest.bpmn20.xml
30 16-03-25 01:12:06 [INFO ] vacationRequest:1:4 was started

```

4.4 Query API [↑]

- タスクの検索
- [TaskQuery](#) と [NativeTaskQuery](#) を使うことができる
 - [TaskQuery](#)

```

1 List<Task> tasks = taskService.createTaskQuery()
2   .taskAssignee("kermit")
3   .processVariableValueEquals("orderId", "0815")
4   .orderByDueDate().asc()
5   .list();

```

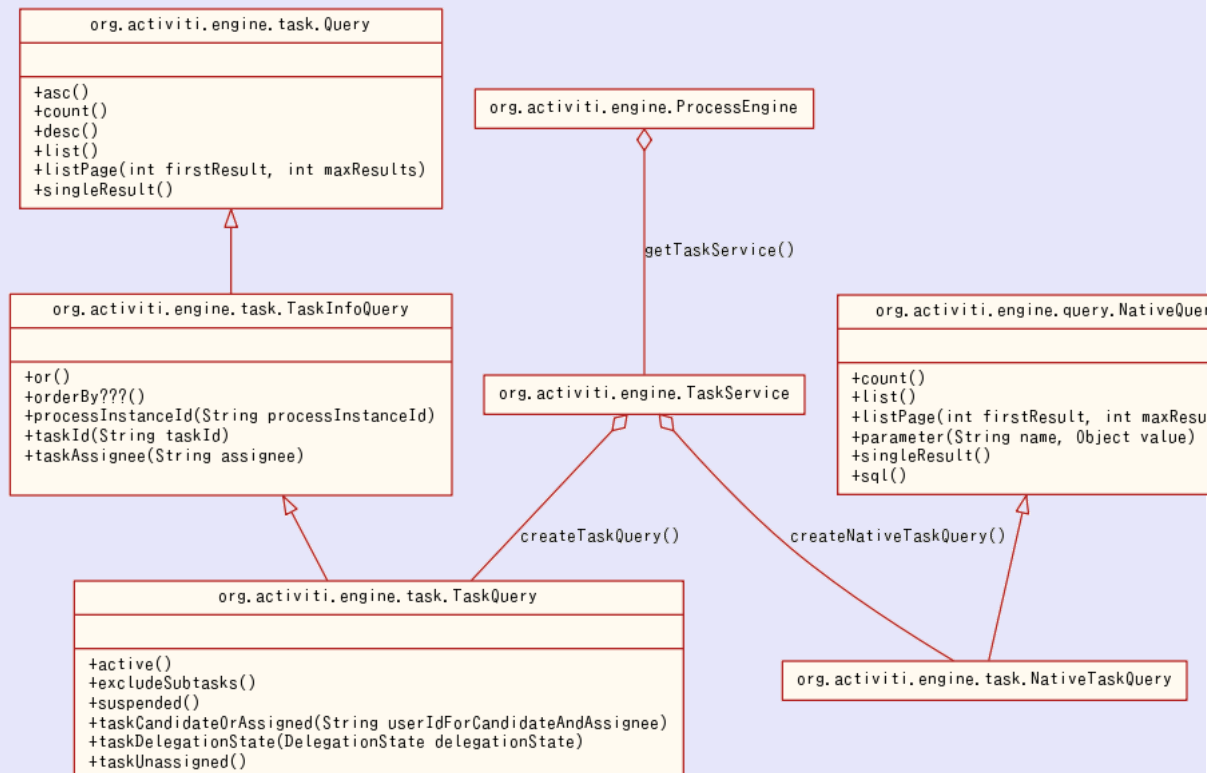
- [NativeTaskQuery](#)

```

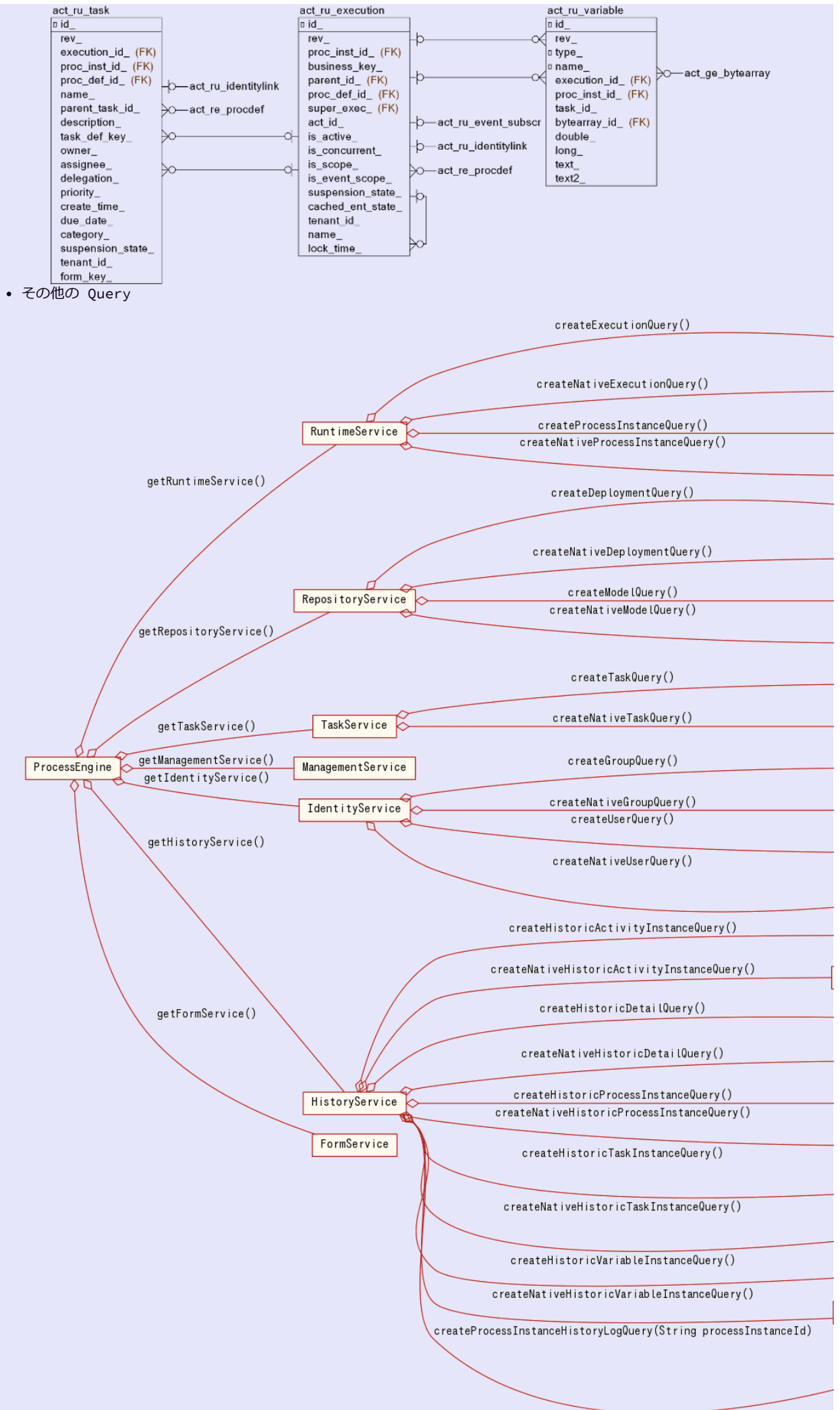
1 List<Task> tasks = taskService.createNativeTaskQuery()
2   .sql("SELECT count(*) FROM " + managementService.getTableName(Task.class) + " T WHERE T.NAM
3   .parameter("taskName", "gonzoTask")
4   .list();
5
6 long count = taskService.createNativeTaskQuery()
7   .sql("SELECT count(*) FROM " + managementService.getTableName(Task.class) + " T1, "
8   + managementService.getTableName(VariableInstanceEntity.class) + " V1 WHERE V1.TASK_ID
9   .count();

```

- クラス図



- [NativeTaskQuery](#) の列名



代表的な [TaskQuery](#) によるタスクの検索と同様に、[ProcessEngine](#) から Service を取得して、Service から Query を取得して、検索を行う。

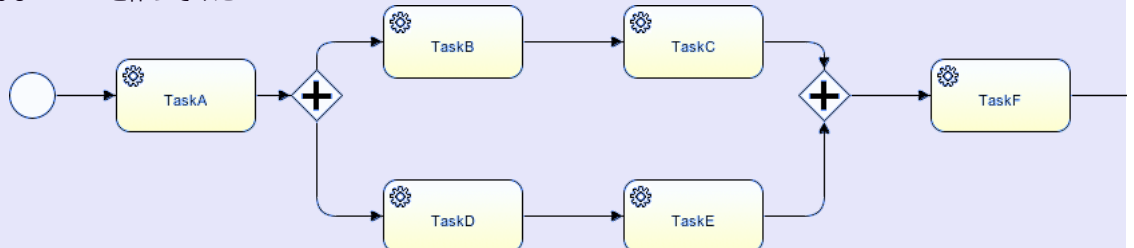
どんな検索ができるかは Javadoc を参照

<http://activiti.org/javadocs/org/activiti/engine/query/Query.html>

4.5.Variables [↑]

- プロセス変数は ACT_RU_VARIABLE に格納される
- アクセス方法
 - BPMN 中の EL 式
 - `${myVar}`
 - `${myBean.myProperty}` ⇔ Object も入れられるようだけど、文字列にしたいほうが無難だね
 - 暗黙オブジェクト

| | |
|--------------------------------------|-----------------------------------|
| <code>\${execution}</code> | DelegateExecution |
| <code>\${task}</code> | DelegateTask |
| <code>\${authenticatedUserId}</code> | String |
 - Java Service Task の execution
 - `JavaDelegate#execution(DelegateExecution execution)`
 - Object `execution.getVariable(String variableName);`
 - void `execution.setVariable(String variableName, Object value);`
 - Map<String, Object> `execution.getVariables();`
 - [RuntimeService](#)
 - Object `execution.getVariable(String executionId, String variableName);`
 - void `execution.setVariable(String executionId, String variableName, Object value);`
 - Map<String, Object> `execution.getVariables(String executionId);`
 - [TaskService](#)
 - Object `execution.getVariable(String taskId, String variableName);`
 - void `execution.setVariable(String taskId, String variableName, Object value);`
 - Map<String, Object> `execution.getVariables(String taskId);`
- 変数をたくさん取り出すときには、`Map<String, Object> getVariables()` を使って一気に取り出す。変数は RDB のテーブルから取り出しているの、一気に読み書きしたほうが効率的
- 変数スコープ
 - `getVariable()` と `getVariableLocal()` があって、変数スコープがあるようだけどよう分らない
 - こんな BPMN を作ってみた



B,C で `setVariablesLocal()` した変数は、D,E で見られななかりするのかな...と思ったけどそうでもないらしい

- 各 Task では、現在の Variable を表示して、自分のタスク名の変数を `setVariable()` `setVariableLocal()` する

<https://github.com/kagyuu/ActivitiExam/blob/master/src/main/java/com/snail/exam/VariableTask.java>

```

1 package com.snail.exam;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import org.activiti.engine.delegate.DelegateExecution;
7 import org.activiti.engine.delegate.JavaDelegate;
8 import org.slf4j.Logger;
9 import org.slf4j.LoggerFactory;
10
11 public class VariableTask implements JavaDelegate {
12     private static final Logger log = LoggerFactory.getLogger(VariableTask.class);
13     @Override
14     public void execute(DelegateExecution execution) throws Exception {
15         String id = execution.getCurrentActivityName();
16         log.info(id);
17
18         // Dump variables
19         List<String> keys = new ArrayList<String>(execution.getVariableNames());
20         keys.sort((o1,o2)->{return o1.compareTo(o2)});
21         keys.forEach((key)->{
22             log.info("@{} EXEC SCOPE {} = {}", id, key, execution.getVariable(key));
23         });
24
25         List<String> keysLocal = new ArrayList<String>(execution.getVariableNamesLocal());

```



```

26     keysLocal.sort((o1,o2)->{return o1.compareTo(o2)});
27     keysLocal.forEach((key)->{
28         log.info("@{} LOCAL SCOPE {} = {}", id, key, execution.getVariableLocal(key));
29     });
30
31     // Update variables
32     execution.getVariableNames().forEach((name)->{
33         execution.setVariable(name, id);
34     });
35     execution.getVariableNamesLocal().forEach((name)->{
36         execution.setVariableLocal(name, id);
37     });
38     execution.setVariable(id + "_EXEC", id);
39     execution.setVariable(id + "_LOCAL", id);
40 }
41 }

```

○ 実行結果

```

1 16-05-21 23:38:30 [INFO ] Initializing process engine using configuration 'file:/Users/atsus
2 16-05-21 23:38:30 [INFO ] initializing process engine for resource file:/Users/atsushi/Eclip
3 16-05-21 23:38:30 [INFO ] Loading XML bean definitions from resource loaded through InputStr
4 16-05-21 23:38:31 [INFO ] performing create on engine with resource org/activiti/db/create/a
5 16-05-21 23:38:31 [INFO ] performing create on history with resource org/activiti/db/create/
6 16-05-21 23:38:31 [INFO ] performing create on identity with resource org/activiti/db/create
7 16-05-21 23:38:31 [INFO ] ProcessEngine default created
8 16-05-21 23:38:31 [INFO ] initialised process engine default
9 16-05-21 23:38:31 [INFO ] --- #1. Deploying the process
10 16-05-21 23:38:31 [INFO ] Processing resource org/activiti/test/VariableProcess.bpmn
11 16-05-21 23:38:32 [INFO ] TaskA
12 16-05-21 23:38:32 [INFO ] @TaskA EXEC SCOPE start = Start
13 16-05-21 23:38:32 [INFO ] @TaskA LOCAL SCOPE start = Start
14 16-05-21 23:38:32 [INFO ] TaskB
15 16-05-21 23:38:32 [INFO ] @TaskB EXEC SCOPE TaskA_EXEC = TaskA
16 16-05-21 23:38:32 [INFO ] @TaskB EXEC SCOPE TaskA_LOCAL = TaskA
17 16-05-21 23:38:32 [INFO ] @TaskB EXEC SCOPE start = TaskA
18 16-05-21 23:38:32 [INFO ] TaskC
19 16-05-21 23:38:32 [INFO ] @TaskC EXEC SCOPE TaskA_EXEC = TaskB
20 16-05-21 23:38:32 [INFO ] @TaskC EXEC SCOPE TaskA_LOCAL = TaskB
21 16-05-21 23:38:32 [INFO ] @TaskC EXEC SCOPE TaskB_EXEC = TaskB
22 16-05-21 23:38:32 [INFO ] @TaskC EXEC SCOPE TaskB_LOCAL = TaskB
23 16-05-21 23:38:32 [INFO ] @TaskC EXEC SCOPE start = TaskB
24 16-05-21 23:38:32 [INFO ] TaskD
25 16-05-21 23:38:32 [INFO ] @TaskD EXEC SCOPE TaskA_EXEC = TaskC
26 16-05-21 23:38:32 [INFO ] @TaskD EXEC SCOPE TaskA_LOCAL = TaskC
27 16-05-21 23:38:32 [INFO ] @TaskD EXEC SCOPE TaskB_EXEC = TaskC
28 16-05-21 23:38:32 [INFO ] @TaskD EXEC SCOPE TaskB_LOCAL = TaskC
29 16-05-21 23:38:32 [INFO ] @TaskD EXEC SCOPE TaskC_EXEC = TaskC
30 16-05-21 23:38:32 [INFO ] @TaskD EXEC SCOPE TaskC_LOCAL = TaskC
31 16-05-21 23:38:32 [INFO ] @TaskD EXEC SCOPE start = TaskC
32 16-05-21 23:38:32 [INFO ] TaskE
33 16-05-21 23:38:32 [INFO ] @TaskE EXEC SCOPE TaskA_EXEC = TaskD
34 16-05-21 23:38:32 [INFO ] @TaskE EXEC SCOPE TaskA_LOCAL = TaskD
35 16-05-21 23:38:32 [INFO ] @TaskE EXEC SCOPE TaskB_EXEC = TaskD
36 16-05-21 23:38:32 [INFO ] @TaskE EXEC SCOPE TaskB_LOCAL = TaskD
37 16-05-21 23:38:32 [INFO ] @TaskE EXEC SCOPE TaskC_EXEC = TaskD
38 16-05-21 23:38:32 [INFO ] @TaskE EXEC SCOPE TaskC_LOCAL = TaskD
39 16-05-21 23:38:32 [INFO ] @TaskE EXEC SCOPE TaskD_EXEC = TaskD
40 16-05-21 23:38:32 [INFO ] @TaskE EXEC SCOPE TaskD_LOCAL = TaskD
41 16-05-21 23:38:32 [INFO ] @TaskE EXEC SCOPE start = TaskD
42 16-05-21 23:38:32 [INFO ] TaskF
43 16-05-21 23:38:32 [INFO ] @TaskF EXEC SCOPE TaskA_EXEC = TaskE
44 16-05-21 23:38:32 [INFO ] @TaskF EXEC SCOPE TaskA_LOCAL = TaskE
45 16-05-21 23:38:32 [INFO ] @TaskF EXEC SCOPE TaskB_EXEC = TaskE
46 16-05-21 23:38:32 [INFO ] @TaskF EXEC SCOPE TaskB_LOCAL = TaskE
47 16-05-21 23:38:32 [INFO ] @TaskF EXEC SCOPE TaskC_EXEC = TaskE
48 16-05-21 23:38:32 [INFO ] @TaskF EXEC SCOPE TaskC_LOCAL = TaskE
49 16-05-21 23:38:32 [INFO ] @TaskF EXEC SCOPE TaskD_EXEC = TaskE
50 16-05-21 23:38:32 [INFO ] @TaskF EXEC SCOPE TaskD_LOCAL = TaskE
51 16-05-21 23:38:32 [INFO ] @TaskF EXEC SCOPE TaskE_EXEC = TaskE
52 16-05-21 23:38:32 [INFO ] @TaskF EXEC SCOPE TaskE_LOCAL = TaskE
53 16-05-21 23:38:32 [INFO ] @TaskF EXEC SCOPE start = TaskE
54 16-05-21 23:38:32 [INFO ] @TaskF LOCAL SCOPE TaskA_EXEC = TaskE
55 16-05-21 23:38:32 [INFO ] @TaskF LOCAL SCOPE TaskA_LOCAL = TaskE
56 16-05-21 23:38:32 [INFO ] @TaskF LOCAL SCOPE TaskB_EXEC = TaskE
57 16-05-21 23:38:32 [INFO ] @TaskF LOCAL SCOPE TaskB_LOCAL = TaskE
58 16-05-21 23:38:32 [INFO ] @TaskF LOCAL SCOPE TaskC_EXEC = TaskE
59 16-05-21 23:38:32 [INFO ] @TaskF LOCAL SCOPE TaskC_LOCAL = TaskE
60 16-05-21 23:38:32 [INFO ] @TaskF LOCAL SCOPE TaskD_EXEC = TaskE
61 16-05-21 23:38:32 [INFO ] @TaskF LOCAL SCOPE TaskD_LOCAL = TaskE
62 16-05-21 23:38:32 [INFO ] @TaskF LOCAL SCOPE TaskE_EXEC = TaskE
63 16-05-21 23:38:32 [INFO ] @TaskF LOCAL SCOPE TaskE_LOCAL = TaskE
64 16-05-21 23:38:32 [INFO ] @TaskF LOCAL SCOPE start = TaskE

```

- よくわからない。Local のことは忘れて、プロセス・インスタンのスコープで変数が保持されると思っておいたほうが良さそう

4.7 Unit Test [↑]

```

1 public class MyBusinessProcessTest {
2
3     @Rule
4     public ActivitiRule activitiRule = new ActivitiRule();
5
6     @Test
7     @Deployment
8     public void ruleUsageExample() {
9         RuntimeService runtimeService = activitiRule.getRuntimeService();
10        runtimeService.startProcessInstanceByKey("ruleUsage");
11
12        TaskService taskService = activitiRule.getTaskService();
13        Task task = taskService.createTaskQuery().singleResult();
14        assertEquals("My Task", task.getName());
15
16        taskService.complete(task.getId());
17        assertEquals(0, runtimeService.createProcessInstanceQuery().count());
18    }
19 }

```

@Deployment で BPMN を配備することもできる

```
@Deployment(resources = {"org/activiti/examples/bpmn/executionListener/ExecutionListenersFieldInjectionProc
```

4.9 The process engine in a web application [↑]

```

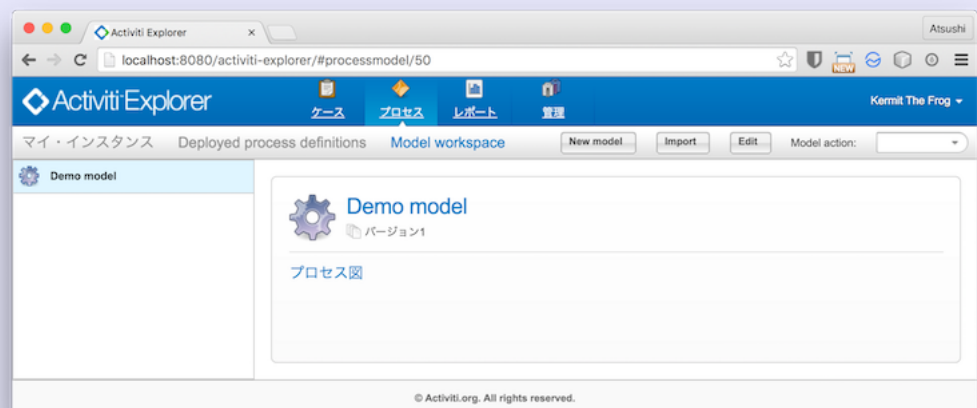
1 @WebServletContextListener
2 public class ProcessEnginesServletContextListener implements ServletContextListener {
3
4     public void contextInitialized(ServletContextEvent servletContextEvent) {
5         ProcessEngines.init();
6     }
7
8     public void contextDestroyed(ServletContextEvent servletContextEvent) {
9         ProcessEngines.destroy();
10    }
11 }
12

```

コンソールアプリと同じように `ProcessEngines2.getDefaultProcessEngine2()` で `ProcessEngine2` を取得するが、アプリ再起動で確実に設定ファイルが反映されるように `ServletContextListener2` で `ProcessEngine2.init()`, `ProcessEngine2.destroy()` を実行する

6. Deployment [↑]

- BPMN の配備
 - `ActivitiExplorer2` で [Import]



- `RepositoryService2` API を使う

```

repositoryService
    .createDeployment()
    .addClasspathResource("org/activiti/test/VacationRequest.bpmn20.xml")
    .deploy();

```

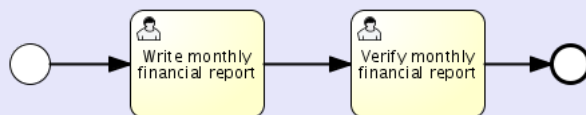
- 配備されたプロセス定義の名称

```
{processDefinitionKey} : {processDefinitionVersion} : {generated-id}
```

- processDefinitionKey? プロセス名
- processDefinitionVersion? バージョン
- generated-id は、自動採番の UID
- 独自 Java プログラム (Data Task、Listener) の配備
 - 実行環境の Classpath に入れる
 - tomcat なら \${tomcat.home}/lib とか

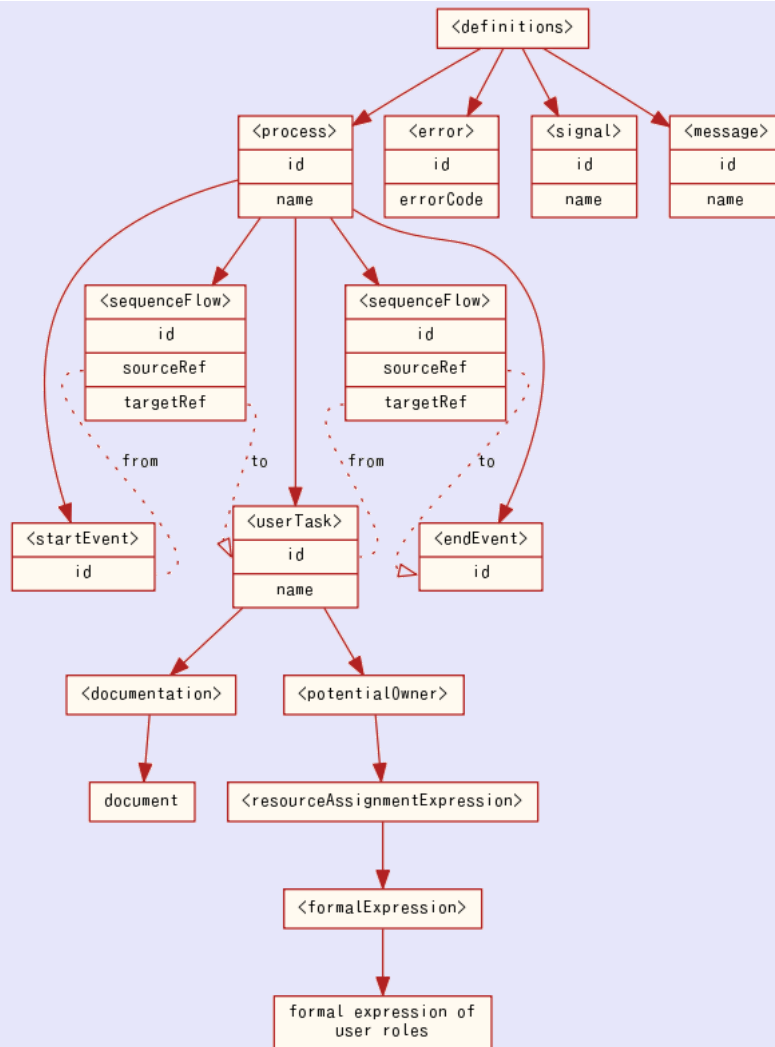
7. 標準 BPMN 2.0 ¹

- Eclipse の Activiti Designer または、Activiti Explorer で作る
- 拡張子 .bpmn20.xml または .bpmn
- BPMNの基本構造



```

1  <definitions id="definitions"
2    targetNamespace="http://activiti.org/bpmn20"
3    xmlns:activiti="http://activiti.org/bpmn"
4    xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">
5
6    <process id="financialReport" name="Monthly financial report reminder process">
7
8      <startEvent id="theStart" />
9
10     <sequenceFlow id="flow1" sourceRef="theStart" targetRef="writeReportTask" />
11
12     <userTask id="writeReportTask" name="Write monthly financial report" >
13       <documentation>
14         Write monthly financial report for publication to shareholders.
15       </documentation>
16       <potentialOwner>
17         <resourceAssignmentExpression>
18           <formalExpression>accountancy</formalExpression>
19         </resourceAssignmentExpression>
20       </potentialOwner>
21     </userTask>
22
23     <sequenceFlow id="flow2" sourceRef="writeReportTask" targetRef="verifyReportTask" />
24
25     <userTask id="verifyReportTask" name="Verify monthly financial report" >
26       <documentation>
27         Verify monthly financial report composed by the accountancy department.
28         This financial report is going to be sent to all the company shareholders.
29       </documentation>
30       <potentialOwner>
31         <resourceAssignmentExpression>
32           <formalExpression>management</formalExpression>
33         </resourceAssignmentExpression>
34       </potentialOwner>
35     </userTask>
36
37     <sequenceFlow id="flow3" sourceRef="verifyReportTask" targetRef="theEnd" />
38
39     <endEvent id="theEnd" />
40
41   </process>
42
43 </definitions>
  
```



- ルート要素は <definitions>
- その下に、<process> <error> <signal> <message> がくる
 - <process> 処理フロー定義
 - <error> エラー定義（<process>の中で、エラーが発生するときに指定する）
 - <signal> シグナル定義（<process>の中で、（全インスタンスに）シグナルを発行・受信するときに指定する。承認者が退職したので処理中フローを全部差し戻しとか）
 - <message> メッセージ定義（<process>の中で、メッセージを発行・受信するときに指定する。同時並行して実施中のサブフローを中止とか）
- <process> の下に <Event> <Task> <Flow> がくる
- <Event>
 - <startEvent> 開始点

```

ProcessInstance processInstance
    = runtimeService.startProcessInstanceByKey("financialReport");
  
```

でプロセスを開始すると、<startEvent> からプロセスが始まる

- <endEvent> 終了点
プロセスは <endEvent> までいくと終了する。終了したプロセスの実行履歴は History Service (processEngine.getHistoryService2()) で取得できる

- <Task>

- <userTask> ユーザタスク
ユーザ入力画面

<potentialOwner> で、実行可能なユーザ(または ロール)を指定できる。
実行権限ロールの人がタスクを個人指定で処理要求したいときには

```

List<Task> tasks = taskService.createTaskQuery().taskAssignee("fozzie").list();
for (Task task : tasks) {
    if (...) {
        taskService.claim(task.getId(), "fozzie");
    }
}
  
```

プログラムでタスクを終わらせたい場合には

```

List<Task> tasks = taskService.createTaskQuery().taskAssignee("fizzie").list();
for (Task task : tasks) {
    if (...) {
        taskService.complete(task.getId());
    }
}

```

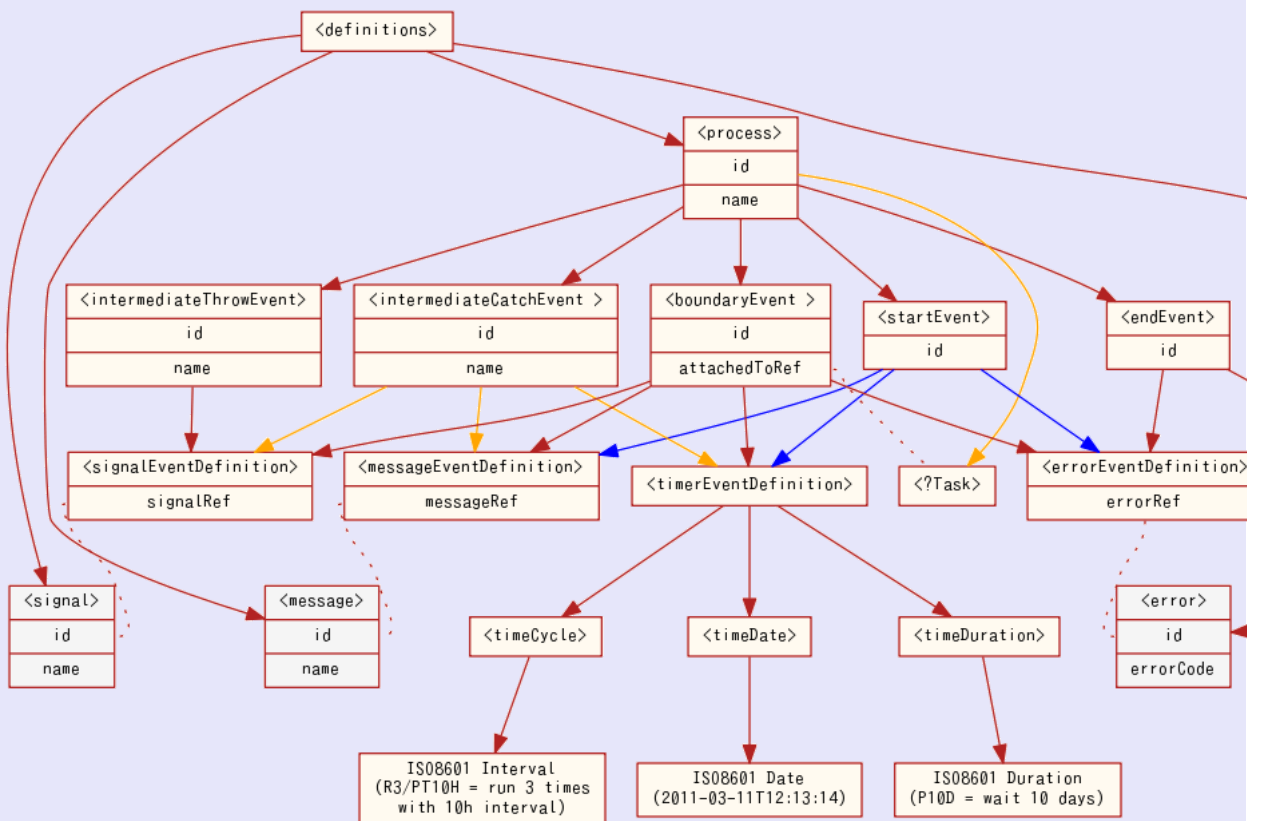
◦ <Flow>

- <sequenceFlow> フロー
sourceRef と targetRef に <Event> <Task> の id を設定して、処理の流れを作る
条件分岐は、sequenceFlow の一種

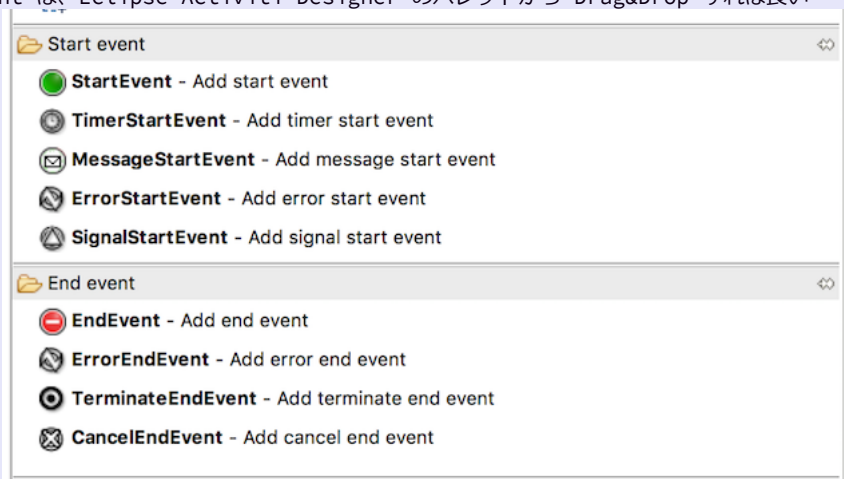
8. Activiti 拡張 BPMN 2.0

- BPMN 規格自体に、BPMN を拡張できる仕様が備わっている

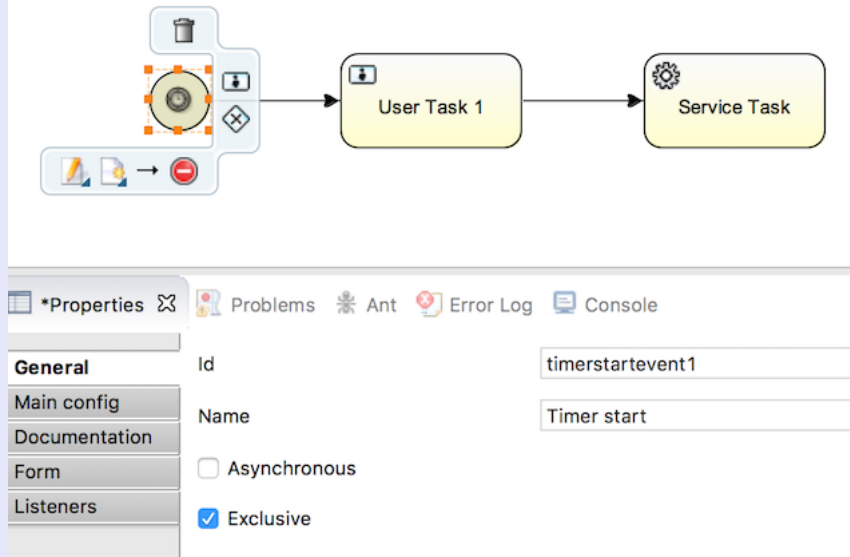
8.2. Event



- Event は、Eclipse Activiti Designer のパレットから Drag&Drop すれば良い



- Event の設定は、Activiti Designer 上の properties で編集すると BPMN (XML) に反映される



- `<timerEventDefinition?>`
 - `<?Event>` の子要素にすると、指定したタイマーによって Event が発効する
 - Note. timers are only fired when the job or async executor is enabled (i.e. `jobExecutorActivate?` or `asyncExecutorActivate?` needs to be set to true in the `activiti.cfg.xml`, since the job and async executor are disabled by default).
 - https://en.wikipedia.org/wiki/ISO_8601 (日本語Wikipedia には、Duration、Intervals の記述なし)
- `<errorEventDefinition?>`
 - `<startEvent>` の子要素にすると、エラーの発生を契機に Event が発効する
 - `<endEvent>` の子要素にすると、その `<endEvent>` に到達した時に指定されたエラーでフローを終了する
 - Java の Exception の発生を検知して `<endEvent>` に跳ぶのではなく、その `<endEvent>` に到達した時に指定されたエラーでフローを終了する
- `<signalEventDefinition?>`
 - シグナルは、全プロセスインスタンスに送られる。
 - 承認担当者が、不正処理で臽になったので、処理中のワークフローを全部差し戻しとか
 - `<intermediateThrowEvent?>` でシグナルを発効する。
 - `activiti:async="true"` をつけると非同期になる
 - シグナルが発行すると、全プロセスインスタンスで `<intermediateCatchEvent?>` からのフローが始まる
 - プログラムからシグナルを強制発効する

```
RuntimeService.signalEventReceived(String signalName);
RuntimeService.signalEventReceived(String signalName, String executionId);
```

- 引き数が二つある方は、特定のインスタンスのみにシグナルを発効する。テストとかで使うのかな
- シグナルを発行したプロセスインスタンスもプロセスを受信しちゃうんで、プロセスを発行した後に何かすることは出来ないことに留意する
- `<messageEventDefinition?>`
 - メッセージは、特定のプロセスに送られる。
 - BPMNのフローからメッセージを送ることはできない
 - `<startEvent>` の子要素にすると、メッセージの受信を契機に Event が発効する
 - `<intermediateCatchEvent?>` の子要素にすると、メッセージ受信を契機にそこからフローが始まる
 - プログラムからメッセージを発効する

```
RuntimeService.messageEventReceived(String messageName, String executionId);
RuntimeService.messageEventReceived(String messageName, String executionId, HashMap<String, Object>
```

- フローをはじめ (メッセージ受信を契機に実行できるフローがあるだけでは、勝手にフローは実行されないことに注意。どこかの Process Engine が実行する必要がある)

```
ProcessInstance.startProcessInstanceByMessage(String messageName);
ProcessInstance.startProcessInstanceByMessage(String messageName, Map<String, Object> processVariables);
ProcessInstance.startProcessInstanceByMessage(String messageName, String businessKey, Map<String, Object>
```

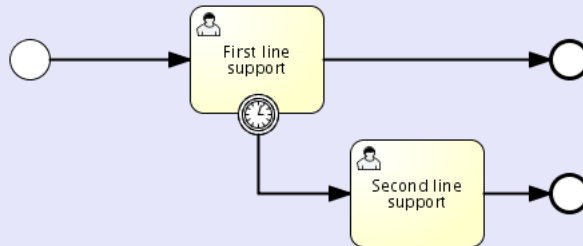
- あるメッセージを契機にはじまるプロセスの一覧 (`<startEvent>` がメッセージ契機なプロセスの一覧)

```
ProcessDefinition processDefinition = repositoryService.createProcessDefinitionQuery()
    .messageEventSubscription("newCallCenterBooking")
    .singleResult();
```

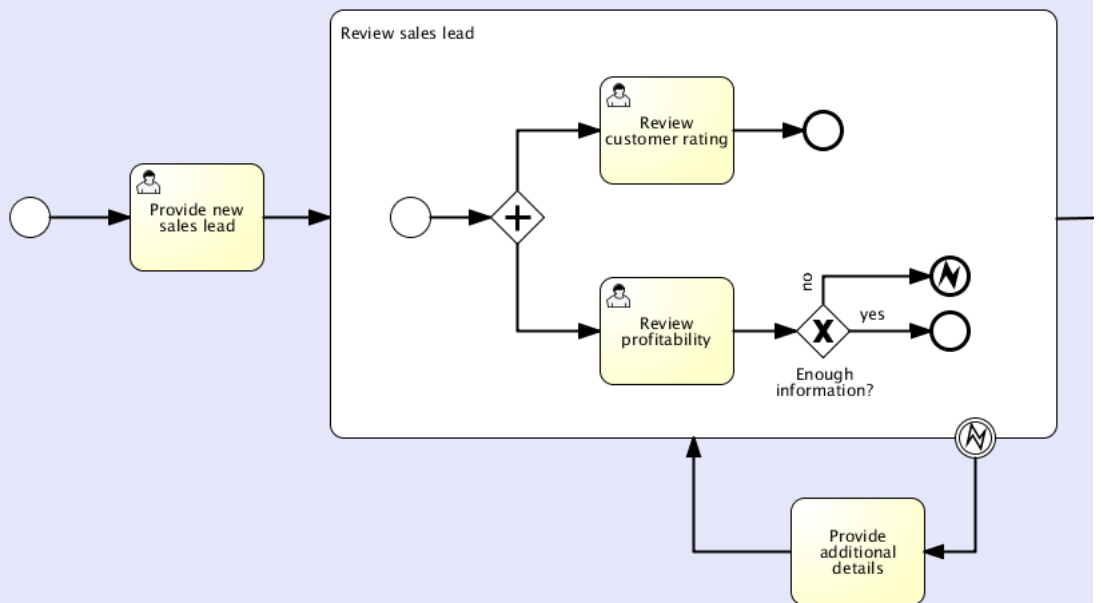
- あるメッセージを契機に特殊処理が走るプロセスの一覧 (<intermediateCatch> でメッセージを待っているプロセスインスタンスの一覧)

```
Execution execution = runtimeService.createExecutionQuery()
    .messageEventSubscriptionName("paymentReceived")
    .variableValueEquals("orderId", message.getOrderId())
    .singleResult();
```

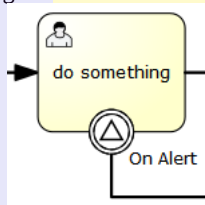
- JMSなどを使って、外部システムのイベントを契機にフローを動かしたい時とかに使う
- <terminateEventDefinition?>
 - <endEvent> の子要素
 - 平行して動いているフローが有った場合には強制終了して、プロセスインスタンスを完全に終了させる
- <boundaryEvent>
 - Task に張り付いて、Task 内のイベントに基づいてフローを開始する
 - <timerEventDefinition?> をつけると、主処理とは別に Task 終了後一定時間してから起動するフローを書ける



- <errorEventDefinition?> をつけると、Task / Sub process があるエラーで終わった場合に起動するフローを書ける



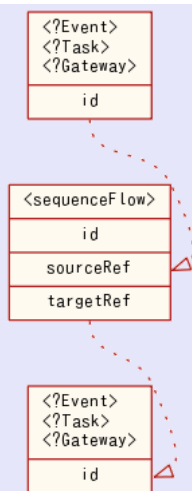
- <signalEventDefinition?> をつけると、当該タスクの実効待中にシグナルを受信した時のフローを書ける



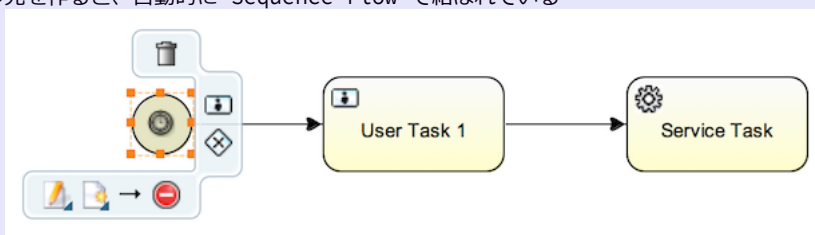
- <messageEventDefinition?> をつけると、当該タスクの実効待中にメッセージを受信した時のフローを書ける



8.3 Sequence Flow [↑]

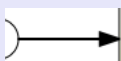


- Sequence Flow は、Eclipse Activiti Designer では、遷移元の Event、Task、Gatewayをシングルクリックした後、[→] (Create Connection) を遷移先に Drag すればできる。あるいは、シングルクリックメニューから遷移先を作ると、自動的に Sequence Flow で結ばれている



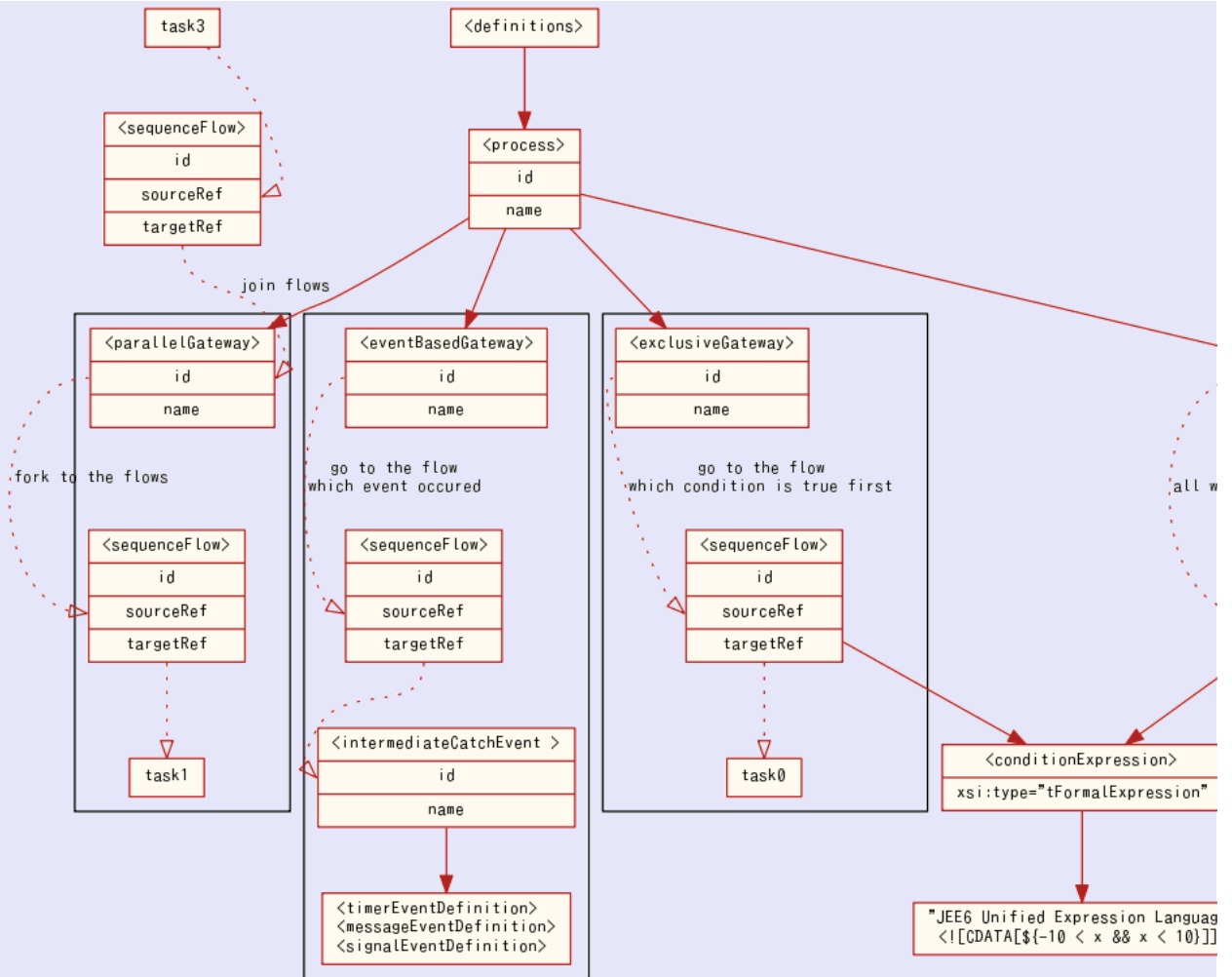
- Sequence Flow は、Event、Task、Gateway を結びつける

```
<sequenceFlow id="flow1" sourceRef="theStart" targetRef="theTask" />
```

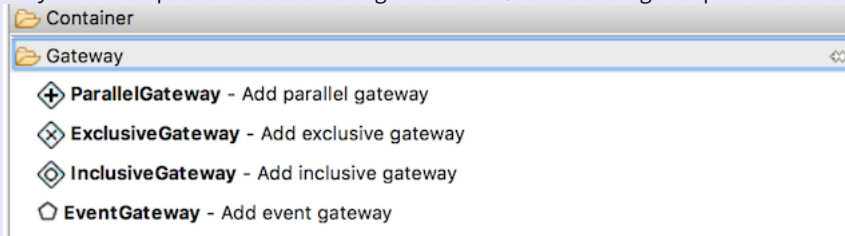


- sourceRef 属性 : 遷移元
- targetRef 属性 : 遷移先

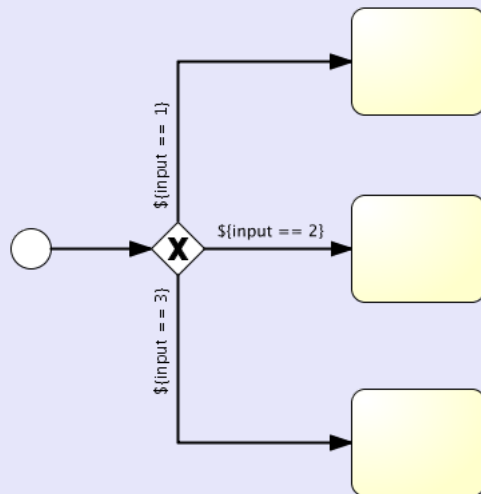
8.4 Gateways [↑](#)



- Gateway は、Eclipse Activiti Designer のパレットから Drag&Drop すれば良い



- <exclusiveGateway>



- 排他に分岐
- 行き先の <sequenceFlow> は、<conditionExpression> 子要素を持ち、評価式が true になる最初の <sequenceFlow> に進む (一つの Flow にしか進まない)
- <conditionExpression> の xsi:type は、現時点では [tFormalExpression](#) (EL式) しか設定できない

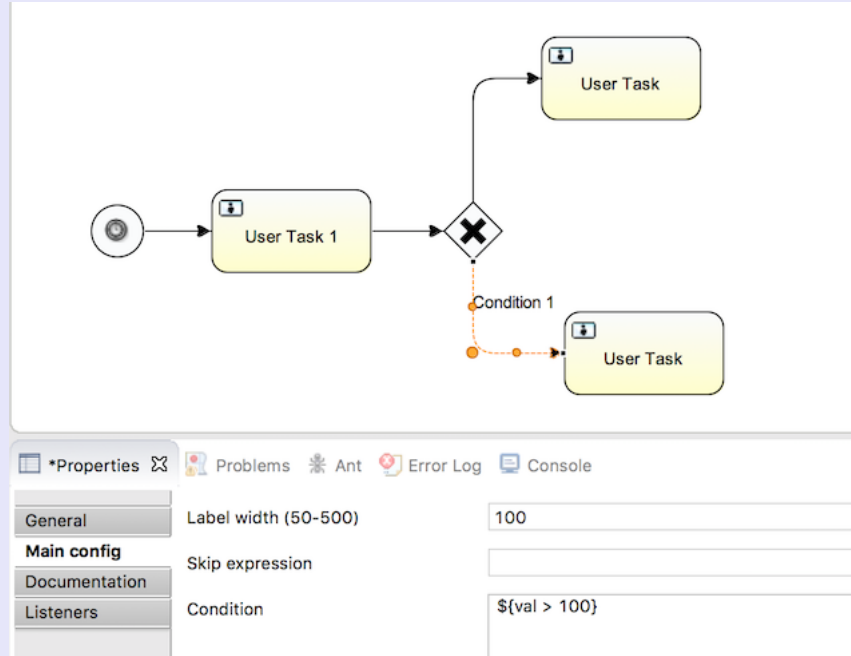
```
1 <exclusiveGateway id="exclusiveGw" name="Exclusive Gateway" />
2
```

```

3 <sequenceFlow id="flow2" sourceRef="exclusiveGw" targetRef="theTask1">
4   <conditionExpression xsi:type="tFormalExpression">${input == 1}</conditionExpression>
5 </sequenceFlow>
6
7 <sequenceFlow id="flow3" sourceRef="exclusiveGw" targetRef="theTask2">
8   <conditionExpression xsi:type="tFormalExpression">${input == 2}</conditionExpression>
9 </sequenceFlow>
10
11 <sequenceFlow id="flow4" sourceRef="exclusiveGw" targetRef="theTask3">
12   <conditionExpression xsi:type="tFormalExpression">${input == 3}</conditionExpression>
13 </sequenceFlow>

```

- 評価式は Eclipse Activiti Designer で、<sequenceFlow> をクリックして properties ペインで設定できる



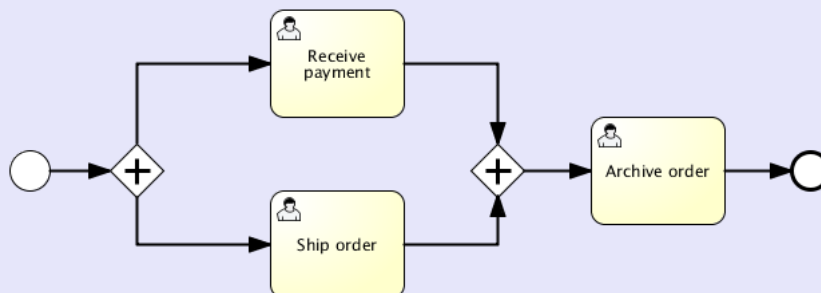
この設定から生成される bpmn (xml) は

```

<sequenceFlow id="flow7" sourceRef="exclusiveGateway1" targetRef="usertask3">
  <conditionExpression xsi:type="tFormalExpression"><![CDATA[${val > 100}]]</conditionExpression>
</sequenceFlow>

```

- <parallelGateway>



- 並行実行の分岐
- 行き先の <sequenceFlow> は、すべて同時に動き出す
- <parallelGateway> が行き先な場合は、並行実行されている処理の待ち合わせになる

```

1 <startEvent id="theStart" />
2 <sequenceFlow id="flow1" sourceRef="theStart" targetRef="fork" />
3
4 <parallelGateway id="fork" />
5 <sequenceFlow sourceRef="fork" targetRef="receivePayment" />
6 <sequenceFlow sourceRef="fork" targetRef="shipOrder" />
7
8 <userTask id="receivePayment" name="Receive Payment" />
9 <sequenceFlow sourceRef="receivePayment" targetRef="join" />
10
11 <userTask id="shipOrder" name="Ship Order" />
12 <sequenceFlow sourceRef="shipOrder" targetRef="join" />
13
14 <parallelGateway id="join" />
15 <sequenceFlow sourceRef="join" targetRef="archiveOrder" />
16
17 <userTask id="archiveOrder" name="Archive Order" />
18 <sequenceFlow sourceRef="archiveOrder" targetRef="theEnd" />

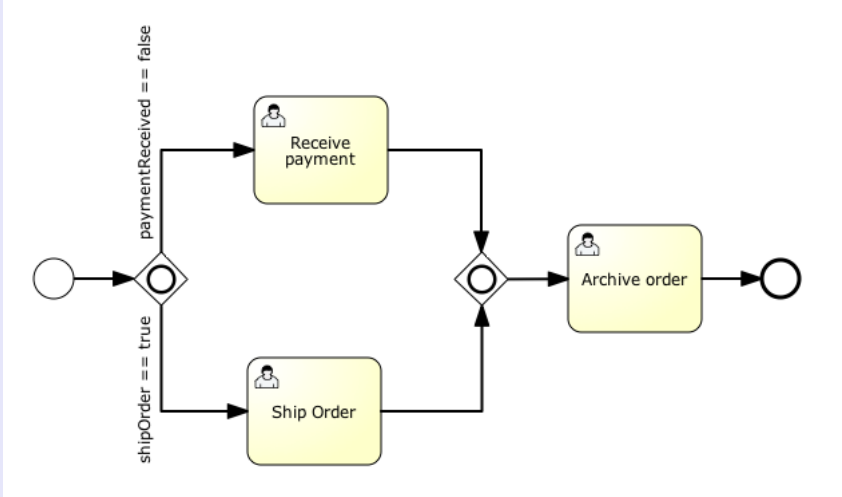
```

```

19 </endEvent id="theEnd" />
20

```

- <inclusiveGateway>



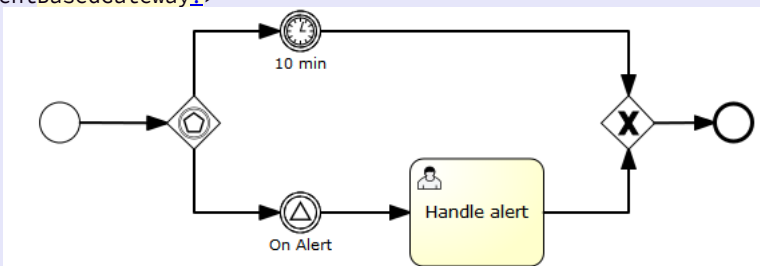
- 行き先の <sequenceFlow> は、<conditionExpression> 子要素を持ち、その評価式が true になる すべての <sequenceFlow> が同時に動き出す (複数の Flow が並行実行される)
- <inclusiveGateway> が行き先な場合は、並行実行されている処理の待ち合わせになる

```

1 <startEvent id="theStart" />
2 <sequenceFlow id="flow1" sourceRef="theStart" targetRef="fork" />
3
4 <inclusiveGateway id="fork" />
5 <sequenceFlow sourceRef="fork" targetRef="receivePayment" >
6   <conditionExpression xsi:type="tFormalExpression">${paymentReceived == false}</conditionEx
7 </sequenceFlow>
8 <sequenceFlow sourceRef="fork" targetRef="shipOrder" >
9   <conditionExpression xsi:type="tFormalExpression">${shipOrder == true}</conditionExpressio
10 </sequenceFlow>
11
12 <userTask id="receivePayment" name="Receive Payment" />
13 <sequenceFlow sourceRef="receivePayment" targetRef="join" />
14
15 <userTask id="shipOrder" name="Ship Order" />
16 <sequenceFlow sourceRef="shipOrder" targetRef="join" />
17
18 <inclusiveGateway id="join" />
19 <sequenceFlow sourceRef="join" targetRef="archiveOrder" />
20
21 <userTask id="archiveOrder" name="Archive Order" />
22 <sequenceFlow sourceRef="archiveOrder" targetRef="theEnd" />
23
24 <endEvent id="theEnd" />

```

- <eventBasedGateway?>



- 行き先の <sequenceFlow> の、さらに行先は、<intermediateCatchEvent?> になっており、最初に受信したイベントが行き先の <sequenceFlow> に進む (一つの Flow にしか進まない)
- 10分待つてシグナルが来なかったら次へ進むなど

```

1 <definitions id="definitions"
2   xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
3   xmlns:activiti="http://activiti.org/bpmn"
4   targetNamespace="Examples">
5
6   <signal id="alertSignal" name="alert" />
7
8   <process id="catchSignal">
9
10    <startEvent id="start" />
11
12    <sequenceFlow sourceRef="start" targetRef="gw1" />
13
14    <eventBasedGateway id="gw1" />
15
16    <sequenceFlow sourceRef="gw1" targetRef="signalEvent" />
17    <sequenceFlow sourceRef="gw1" targetRef="timerEvent" />
18

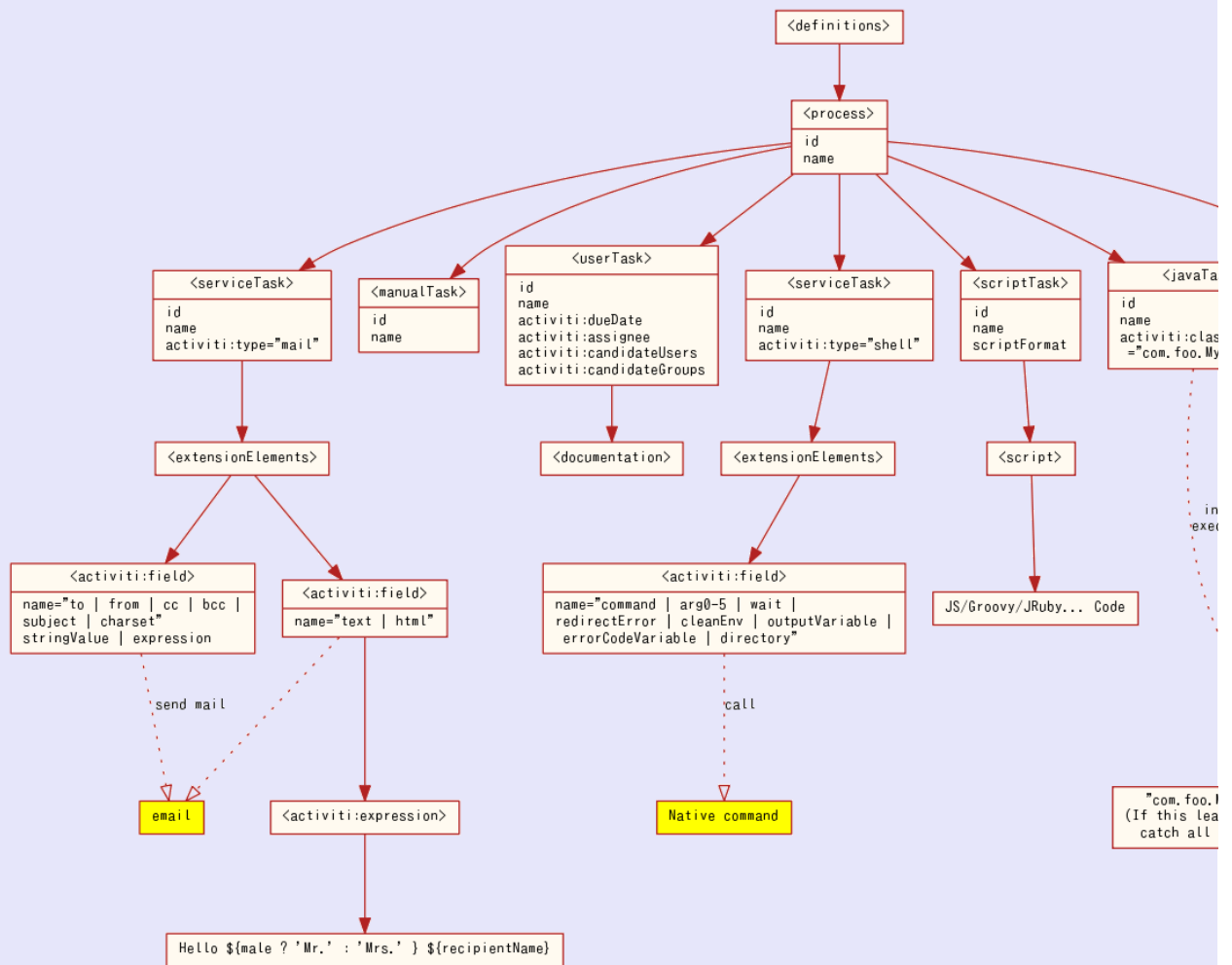
```

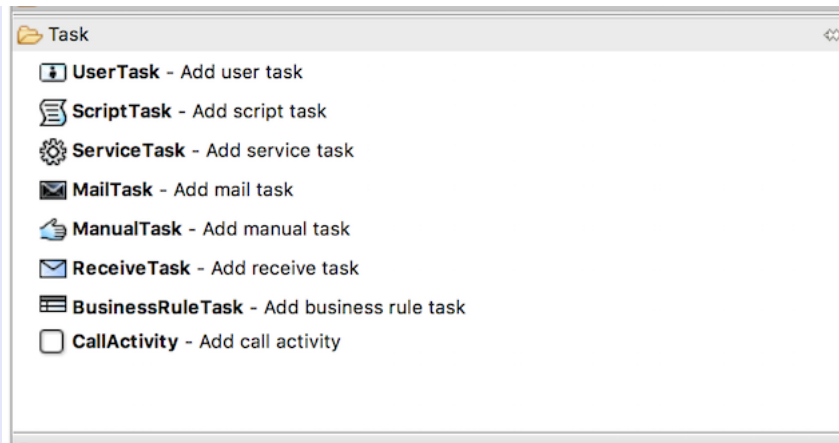
```

19 <intermediateCatchEvent id="signalEvent" name="Alert">
20   <signalEventDefinition signalRef="alertSignal" />
21 </intermediateCatchEvent>
22
23 <intermediateCatchEvent id="timerEvent" name="Alert">
24   <timerEventDefinition>
25     <timeDuration>PT10M</timeDuration>
26   </timerEventDefinition>
27 </intermediateCatchEvent>
28
29 <sequenceFlow sourceRef="timerEvent" targetRef="exGw1" />
30 <sequenceFlow sourceRef="signalEvent" targetRef="task" />
31
32 <userTask id="task" name="Handle alert"/>
33
34 <exclusiveGateway id="exGw1" />
35
36 <sequenceFlow sourceRef="task" targetRef="exGw1" />
37 <sequenceFlow sourceRef="exGw1" targetRef="end" />
38
39 <endEvent id="end" />
40 </process>
41 </definitions>

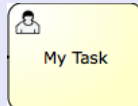
```

8.5 Task [↑]





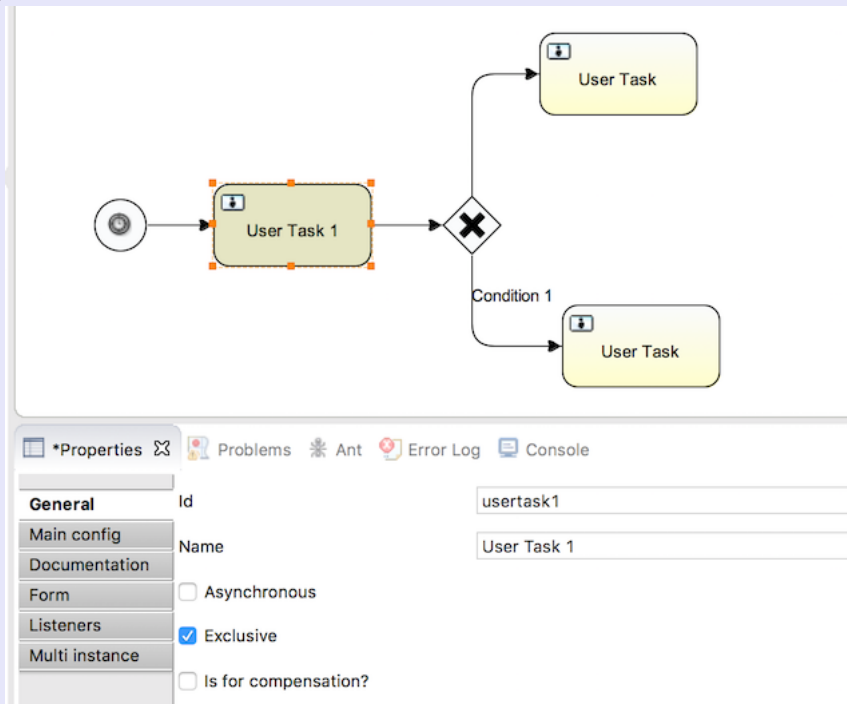
8.5.1. User Task [↑]



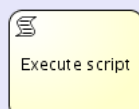
- <userTask>
- ユーザの実行するタスク

```
<userTask id="theTask" name="Important task" />
```

- 配下の document に説明を書く。プログラムからは task.getDescription(); で取得可能
- 期限: activiti:dueDate 要素に ISO8601 date-time 形式 (yyyy-MM-ddThh:mm:ss) または time-duration 形式 (PT50M) で設定する
- 実行権限:
 - BPMN2.0規格では、<userTask> タグの配下に、<humanPerformer> または <potentialOwner> タグを入れ子にして、そこでタスクを実行できるユーザやグループを定義する。この形式でも activiti は正しく動作する
 - activiti では、独自拡張として <userTask> タグに activiti:assignee、activiti:candidateUsers、activiti:candidateGroups を定義できる
- Eclipse Activiti Designer では、期限や権限は properties で編集できる。編集すると BPMN (XML) に反映される



8.5.2. Script Task [↑]



- JSR-223 (Java Scripting API) で、BPMN内に定義された文字列のスクリプトを実行する。

- 例

```

1 <scriptTask id="theScriptTask" name="Execute script" scriptFormat="groovy">
2   <script>
3     sum = 0
4     for ( i in inputArray ) {
5       sum += i
6     }
7   </script>
8 </scriptTask>

```

- フラグを立てるくらいなら使ってもいいかもね
- ここに、がっつり処理ロジックを書いてもなかなかテストできないし、実行時に Syntax Error が起きるかもしれんし
- プロセス変数の変更・設定

```

1 <script>
2   def scriptVar = "test123"
3   execution.setVariable("myVar", scriptVar)
4 </script>

```

- groovy を使う場合には、pom.xml に

```

1 <dependency>
2   <groupId>org.codehaus.groovy</groupId>
3   <artifactId>groovy-all</artifactId>
4   <version>2.x.x</version>
5 </dependency>

```

を追記する必要あり

8.5.3. Java Task [↑]



- Javaコードを実行する
- 例

```

1 <serviceTask id="javaService"
2   name="My Java Service Task"
3   activiti:class="org.activiti.MyJavaDelegate" />

```

(Activiti が Spring コンテナ上で動いている場合には、Spring が管理している Bean を lookup したり、その任意のメソッドを実行できる。

JavaSE環境で実行するときには上記のように、クラス名をパッケージ名付きで指定する)

- 呼び出される Java コード

```

1 public class MyJavaDelegate implements JavaDelegate {
2
3   public void execute(DelegateExecution execution) throws Exception {
4     String var = (String) execution.getVariable("input");
5     var = var.toUpperCase();
6     execution.setVariable("input", var);
7   }
8
9 }

```

- [JavaDelegate2](#) を実装する
- 処理ロジックは `void execute(DelegateExecution2 execution)` に記述する
- [DelegateExecution2](#) からプロセススコープの (ACT_RU_VARIABLE テーブルに格納されている) 変数を取得・変更できる cf. [4.5.Variables](#)

- パラメータの Injection

```

1 <serviceTask id="javaService" name="Java service invocation"
2   activiti:class="org.activiti.examples.bpmn.servicetask.ReverseStringsFieldInjected">
3
4   <extensionElements>
5     <activiti:field name="text1">
6       <activiti:expression>${genderBean.getGenderString(gender)}</activiti:expression>
7     </activiti:field>
8     <activiti:field name="text2">
9       <activiti:expression>Hello ${gender == 'male' ? 'Mr.' : 'Mrs.'} ${name}</activiti:expression>
10    </activiti:field>
11  </extensionElements>
12 </serviceTask>

```

フィールド変数 text1、text2 に BPMN で指定した値を設定できる

- エラー処理

```

1 <serviceTask id="servicetask1" name="Service Task" activiti:class="...">
2   <extensionElements>
3     <activiti:mapException errorCode="myErrorCode1"/>
4   </extensionElements>

```



```
5 </serviceTask>
```

例外が発生したら、エラーコード myErrorCode1 で終了

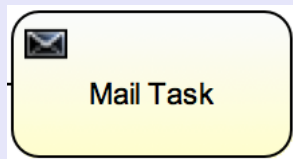
- Exception の種類によってエラーコードを変える

```
1 <serviceTask id="servicetask1" name="Service Task" activiti:class="...">
2   <extensionElements>
3     <activiti:mapException
4       errorCode="myErrorCode1">org.activiti.SomeException</activiti:mapException>
5   </extensionElements>
6 </serviceTask>
```

- Exception の親クラスを指定（どの子クラスの例外が起きても、指定されたエラーコードをで終了する）

```
1 <serviceTask id="servicetask1" name="Service Task" activiti:class="...">
2   <extensionElements>
3     <activiti:mapException errorCode="myErrorCode1"
4       includeChildExceptions="true">org.activiti.SomeException</activiti:mapException>
5   </extensionElements>
6 </serviceTask>
```

8.5.6. Email Task [↑]

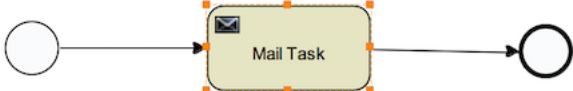


```
1 <serviceTask id="mailtask1" name="Mail Task" activiti:type="mail">
2   <extensionElements>
3     <activiti:field name="from">
4       <activiti:string><![CDATA[order-shipping@thecompany.com]]></activiti:string>
5     </activiti:field>
6     <activiti:field name="to">
7       <activiti:expression><![CDATA[foo@example.com]]></activiti:expression>
8     </activiti:field>
9     <activiti:field name="subject">
10      <activiti:string><![CDATA[ENGLISH SUBJECT 日本語の件名]]></activiti:string>
11    </activiti:field>
12    <activiti:field name="charset">
13      <activiti:string><![CDATA[iso-2022-jp]]></activiti:string>
14    </activiti:field>
15    <activiti:field name="text">
16      <activiti:string><![CDATA[ENGLISH MESSAGE
17 日本語の本文]]></activiti:string>
18    </activiti:field>
19  </extensionElements>
20 </serviceTask>
```

- [activiti.cfg.xml](#) で、SMTP サーバーの設定が必要
- Activiti Explorer / REST API では、[WEB-INF/classes/engine.properties](#)
- 設定できるのは

| activiti:field name="" | 設定内容 |
|------------------------------------|--|
| to | TO |
| from | FROM |
| subject | SUBJECT |
| cc | CC |
| bcc | BCC |
| charset | 日本語なら iso-2022-jp やね |
| html | HTML Mail 平文+EL式 で指定 |
| text | Text Mail 平文+EL式 で指定 |
| htmlVar | HTML Mail の全文が格納されているプロセス変数名 |
| textVar | Text Mail の全文が格納されているプロセス変数名 |
| ignoreException | 例外を無視 |
| exceptionVariableName ² | ignoreException=true のとき、発生した Exception を格納するプロセス変数名 |

- Eclipse Activiti Designer から設定可能



The diagram shows a BPMN process flow starting with a start circle, followed by a task circle labeled "Mail Task" with a mail icon, and ending with an end circle.

Properties panel for the Mail Task:

- General: To: foo@example.com
- Main config: From: order-shipping@thecompany.com
- Documentation: Subject: ENGLISH SUBJECT 日本語の件名
- Listeners: Cc: (empty), Bcc: (empty)
- Multi instance: Charset: iso-2022-jp
- Html: (empty)
- Non html: ENGLISH MESSAGE
日本語の本文

- 日本語メールもOK (charset = iso-2022-jp にしておけば、あとは (commons-mailが) やってくれる)





日本語部分は、iso-2022-jp（JISコード）の base64 になっている



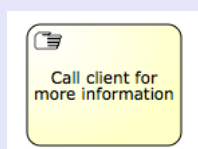
8.5.8. Camel Task (つかえない) [↑](#)

- 結局 Camel を実行する Java Class (Spring 管理下の Bean) を呼び出せるだけ
- こうだったらよかったのに <実際にはこうなっていません!>

```
1 <serviceTask id="i hope it" activiti:type="camel" >
2   <extensionElements>
3     <activiti:field name="from" stringValue="file:/a/b" />
4     <activiti:field name="to" stringValue="log:edi_send" />
5     <activiti:field name="to" stringValue="ftp:/edi" />
6   </extensionElements>
7 </serviceTask>
```

- → 普通の Java Task から Camel を呼び出せば良いんでないかい

8.5.9. Manual Task [↑](#)



- ユーザが Done を押すだけのタスク（ユーザがシステム外で何かやったことを確認するためのタスク）

```
1 <manualTask id="myManualTask" name="Call client for more information" />
```

8.5.10. Java Receive Task [↑](#)



- シグナル待ちタスク

```
1 <receiveTask id="waitState" name="wait" />
```

- シグナル発行

```
1 ProcessInstance pi = runtimeService.startProcessInstanceByKey("receiveTask");
2 Execution execution = runtimeService.createExecutionQuery()
3   .processInstanceId(pi.getId())
4   .activityId("waitState")
5   .singleResult();
6 assertNotNull(execution);
7
8 runtimeService.signal(execution.getId());
```

8.5.11. Shell Task (つかえない) ¹

```
1 <serviceTask id="shellEcho" activiti:type="shell" >
2   <extensionElements>
3     <activiti:field name="command" stringValue="cmd" />
4     <activiti:field name="arg1" stringValue="/c" />
5     <activiti:field name="arg2" stringValue="echo" />
6     <activiti:field name="arg3" stringValue="EchoTest" />
7     <activiti:field name="wait" stringValue="true" />
8     <activiti:field name="outputVariable" stringValue="resultVar" />
9   </extensionElements>
10 </serviceTask>
```

- ちゃんと動くけど、つかえない
- Activiti Designer が未対応で、対応予定もなし → <https://forums.activiti.org/content/eclipse-activiti-designer-and-shell-tasks>
- Activiti Designer で bpmn を編集すると **<serviceTask>** タグが消えてしまう!
- → Java Task で実装するのが良いだろう

8.15.12. Execution listener ¹

- プロセス開始、終了時の処理 (event="start/end")
 - bpmn

```
1 <process id="executionListenersProcess">
2   ...
3   <extensionElements>
4     <activiti:executionListener class="com.foo.ExecutionListenerOne" event="start" />
5   </extensionElements>
6   ...
7   <startEvent id="theStart">
8     <sequenceFlow sourceRef="theStart" ...
9   </sequenceFlow>
10 </process>
```

- com.foo.[ExecutionListenerOne2](#)

```
1 import org.activiti.engine.delegate.ExecutionListener;
2 import org.activiti.engine.delegate.ExecutionListenerExecution;
3
4 public class ExecutionListenerOne implements ExecutionListener {
5
6   public void notify(ExecutionListenerExecution execution) throws Exception {
7     execution.setVariable("variableSetInExecutionListener", "firstValue");
8     execution.setVariable("eventReceived", execution.getEventName());
9   }
10 }
```

旧バージョン (< 5.3) との互換性のため、

org.activiti.engine.impl.pvm.delegate.[ExecutionListener2](#) もあるが、そちらは使わない

- イベントハンドラには、[ExecutionListener2](#) ではなく、[JavaTask2](#) (org.activiti.engine.delegate.[JavaDelegate2](#)を継承したクラス) を指定してもよい
- Flow 通過時の処理 (event なし)
 - bpmn

```
1 <process id="executionListenersProcess">
2   ...
3   <userTask id="firstTask" />
4   <sequenceFlow sourceRef="firstTask" targetRef="secondTask">
5     <extensionElements>
6       <activiti:executionListener class="com.foo.ExecutionListenerTwo" />
7     </extensionElements>
8   </sequenceFlow>
9   ...
10 </process>
```

```

7     </extensionElements>
8     </sequenceFlow>
9     ...
10    </process>

```

- Flow には、event 属性がない（無視される）。Flow には start も end も無いので
- Flow のほかに、Gateway にも同様に Listener を設定できる
- Task開始、終了時の処理（event="start/end"）
 - bpmn

```

1    <process id="executionListenersProcess">
2      ...
3      <userTask id="secondTask" >
4        <extensionElements>
5          <activiti:executionListener expression="com.foo.ExecutionListenerThree" event="end" />
6        </extensionElements>
7      </userTask>
8      <sequenceFlow sourceRef="secondTask" targetRef="thirdTask" />
9      ...
10   </process>

```

- bpmn から executionListener への Field Injection
 - bpmn

```

1    <process id="executionListenersProcess">
2      <extensionElements>
3        <activiti:executionListener class="com.foo.FieldInjectedExecutionListener" event="start"
4          <activiti:field name="fixedValue" stringValue="Yes, I am " />
5          <activiti:field name="dynamicValue" expression="{myVar}" />
6        </activiti:executionListener>
7      </extensionElements>
8      ...
9      ...
10     ...
11   </process>

```

- com.foo.[FieldInjectedExecutionListener?](#)

```

1    import org.activiti.engine.delegate.ExecutionListener;
2    import org.activiti.engine.delegate.ExecutionListenerExecution;
3    import org.activiti.engine.delegate.Expression;
4
5    public class FieldInjectedExecutionListener implements ExecutionListener {
6
7        private Expression fixedValue;
8
9        private Expression dynamicValue;
10
11        public void notify(ExecutionListenerExecution execution) throws Exception {
12            execution.setVariable("var", fixedValue.getValue(execution).toString() + dynamicValue.ge
13        }
14    }

```

旧バージョン（< 5.3）との互換性のため、org.activiti.engine.impl.pvm.delegate.Expression もあるが、そちらは使わない

- bpmn からプロセススコープの変数を Injection できる（ACT_RU_VARIABLE テーブルに格納されているプロセス固有の変数 cf. [4.5.Variable](#)
- Unit Test では、[RuntimeService?](#) でプロセスを開始するときに、変数を設定してあげればいい

```

1    @Deployment(resources = {"org/activiti/examples/bpmn/executionListener/ExecutionListenersFieldIn
2    public void testExecutionListenerFieldInjection() {
3        Map<String, Object> variables = new HashMap<String, Object>();
4        variables.put("myVar", "listening!");
5
6        ProcessInstance processInstance = runtimeService.startProcessInstanceByKey("executionListeners
7
8        Object varSetByListener = runtimeService.getVariable(processInstance.getId(), "var");
9        assertNotNull(varSetByListener);
10       assertTrue(varSetByListener instanceof String);
11
12       // Result is a concatenation of fixed injected field and injected expression
13       assertEquals("Yes, I am listening!", varSetByListener);
14    }

```

8.5.13. Task listener [↑]

- User Task のイベント処理
 - event="create"（タスクが作られたとき）
 - event="assignment"（誰かにアサインされたとき）
 - event="complete"（完了したとき）
 - event="delete"（棄却されたとき）
- bpmn

```

1    <process id="executionListenersProcess">

```

```

2    ...
3    <userTask id="myTask" name="My Task" >
4        <extensionElements>
5            <activiti:taskListener class="com.foo.MyTaskCreateListener" event="create"/>
6        </extensionElements>
7    </userTask>
8    ...
9
10 </process>

```

- [com.foo.MyTaskCreateListener?](#)

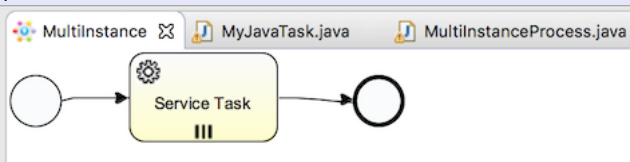
```

1 import org.activiti.engine.delegate.DelegateTask;
2 import org.activiti.engine.delegate.TaskListener;
3
4 public class MyTaskCreateListener implements TaskListener {
5
6     public void notify(DelegateTask delegateTask) {
7         // Custom logic goes here
8     }
9 }

```

8.5.14 Multi-instance (for each) [↑]

- Task/Subprocessの繰り返し実行
 - まあ、使うとしても [JavaTask?](#) だけだろうね
 - User Task とかで使うと複雑になりすぎる
 - Mail Task にも使えるけど... 大量の嫌がらせメール送るとか ...
- bpmn



縦線は並列実行、横線はシーケンシャル実行

```

1 <process id="MultiExam" name="Multi Exam" isExecutable="true">
2     <startEvent id="startevent1" name="Start"></startEvent>
3     <endEvent id="endevent1" name="End"></endEvent>
4     <serviceTask id="servicetask1" name="Service Task" activiti:class="com.snail.exam.MyJavaTask">
5         <multiInstanceLoopCharacteristics isSequential="false">
6             <loopCardinality>10</loopCardinality>
7         </multiInstanceLoopCharacteristics>
8     </serviceTask>
9     <sequenceFlow id="flow1" sourceRef="startevent1" targetRef="servicetask1"></sequenceFlow>
10    <sequenceFlow id="flow2" sourceRef="servicetask1" targetRef="endevent1"></sequenceFlow>
11 </process>

```

- [<multiInstanceLoopCharacteristics?>](#) の属性、子要素

| property | note |
|---------------------|------------------------------|
| isSequential | true(並列実行可) / false(逐次実行) |
| loopCardinality | 繰り返し数 (\${ } でプロセス変数の参照も可能) |
| completionCondition | 終了条件 \${EL式} でプロセス変数を評価する |

- タスク側では、自分が何番目の繰り返しなのかをプロセス変数で参照可能

| property | note |
|---|----------------|
| nrOfInstancefcs? | インスタンス数 |
| nrOfActiveInstances? | 起動中のインスタンス数 |
| nrOfCompletedInstances? | 終了したインスタンス数 |
| loopCounter | 現在実行中のインスタンス番号 |

- サンプル実装

<https://github.com/kagyuu/ActivitiExam/blob/master/src/main/java/com/snail/exam/MultiInstanceProcess.java>

```

1 package com.snail.exam;
2
3 import org.activiti.engine.ProcessEngine;
4 import org.activiti.engine.ProcessEngines;
5 import org.activiti.engine.RepositoryService;
6 import org.activiti.engine.RuntimeService;
7 import org.activiti.engine.runtime.ProcessInstance;
8 import org.slf4j.Logger;
9 import org.slf4j.LoggerFactory;
10
11 public class MultiInstanceProcess {
12
13     private static final Logger log = LoggerFactory.getLogger(MyProcess.class);
14 }

```

```

15     public static void main(String[] args) {
16         try {
17             ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine();
18
19             RepositoryService repositoryService = processEngine.getRepositoryService();
20             repositoryService.createDeployment()
21                 .addClasspathResource("org/activiti/test/MultiInstance.bpmn")
22                 .deploy();
23
24             log.info("Deployed");
25
26             RuntimeService runtimeService = processEngine.getRuntimeService();
27             ProcessInstance processInstance = runtimeService.startProcessInstanceByKey("MultiExa
28
29             log.info("Complete");
30
31         } catch (RuntimeException th) {
32             log.error("ERROR", th);
33         }
34     }
35 }

```

<https://github.com/kagyuu/ActivitiExam/blob/master/src/main/java/com/snail/exam/MyJavaTask.java>

```

1  package com.snail.exam;
2
3  import java.util.Map;
4
5  import org.activiti.engine.delegate.DelegateExecution;
6  import org.activiti.engine.delegate.JavaDelegate;
7  import org.slf4j.Logger;
8  import org.slf4j.LoggerFactory;
9
10 public class MyJavaTask implements JavaDelegate {
11
12     private static final Logger log = LoggerFactory.getLogger(MyJavaTask.class);
13
14     public void execute(DelegateExecution execution) throws Exception {
15         log.info("This is MyJavaTask");
16         for(Map.Entry entry : execution.getVariables().entrySet()) {
17             log.info("{}={}", entry.getKey(), entry.getValue());
18         }
19     }
20 }

```

実行結果 (10回実行された)

```

1  16-05-18 23:49:36 [INFO ] Initializing process engine using configuration 'file:/Users/atsushi/E
2  16-05-18 23:49:36 [INFO ] initializing process engine for resource file:/Users/atsushi/EclipseWo
3  16-05-18 23:49:36 [INFO ] Loading XML bean definitions from resource loaded through InputStream
4  16-05-18 23:49:37 [INFO ] performing create on engine with resource org/activiti/db/create/activ
5  16-05-18 23:49:37 [INFO ] performing create on history with resource org/activiti/db/create/acti
6  16-05-18 23:49:37 [INFO ] performing create on identity with resource org/activiti/db/create/act
7  16-05-18 23:49:37 [INFO ] ProcessEngine default created
8  16-05-18 23:49:37 [INFO ] initialised process engine default
9  16-05-18 23:49:37 [INFO ] Processing resource org/activiti/test/MultiInstance.bpmn
10 16-05-18 23:49:37 [INFO ] Deployed
11 16-05-18 23:49:37 [INFO ] This is MyJavaTask
12 16-05-18 23:49:37 [INFO ] nrOfActiveInstances=10
13 16-05-18 23:49:37 [INFO ] loopCounter=0
14 16-05-18 23:49:37 [INFO ] nrOfInstances=10
15 16-05-18 23:49:37 [INFO ] nrOfCompletedInstances=0
16 16-05-18 23:49:38 [INFO ] This is MyJavaTask
17 16-05-18 23:49:38 [INFO ] nrOfActiveInstances=9
18 16-05-18 23:49:38 [INFO ] loopCounter=1
19 16-05-18 23:49:38 [INFO ] nrOfInstances=10
20 16-05-18 23:49:38 [INFO ] nrOfCompletedInstances=1
21 16-05-18 23:49:38 [INFO ] This is MyJavaTask
22 16-05-18 23:49:38 [INFO ] nrOfActiveInstances=8
23 16-05-18 23:49:38 [INFO ] loopCounter=2
24 16-05-18 23:49:38 [INFO ] nrOfInstances=10
25 16-05-18 23:49:38 [INFO ] nrOfCompletedInstances=2
26 16-05-18 23:49:38 [INFO ] This is MyJavaTask
27 16-05-18 23:49:38 [INFO ] nrOfActiveInstances=7
28 16-05-18 23:49:38 [INFO ] loopCounter=3
29 16-05-18 23:49:38 [INFO ] nrOfInstances=10
30 16-05-18 23:49:38 [INFO ] nrOfCompletedInstances=3
31 16-05-18 23:49:38 [INFO ] This is MyJavaTask
32 16-05-18 23:49:38 [INFO ] nrOfActiveInstances=6
33 16-05-18 23:49:38 [INFO ] loopCounter=4
34 16-05-18 23:49:38 [INFO ] nrOfInstances=10
35 16-05-18 23:49:38 [INFO ] nrOfCompletedInstances=4
36 16-05-18 23:49:38 [INFO ] This is MyJavaTask
37 16-05-18 23:49:38 [INFO ] nrOfActiveInstances=5
38 16-05-18 23:49:38 [INFO ] loopCounter=5
39 16-05-18 23:49:38 [INFO ] nrOfInstances=10
40 16-05-18 23:49:38 [INFO ] nrOfCompletedInstances=5
41 16-05-18 23:49:38 [INFO ] This is MyJavaTask
42 16-05-18 23:49:38 [INFO ] nrOfActiveInstances=4
43 16-05-18 23:49:38 [INFO ] loopCounter=6

```



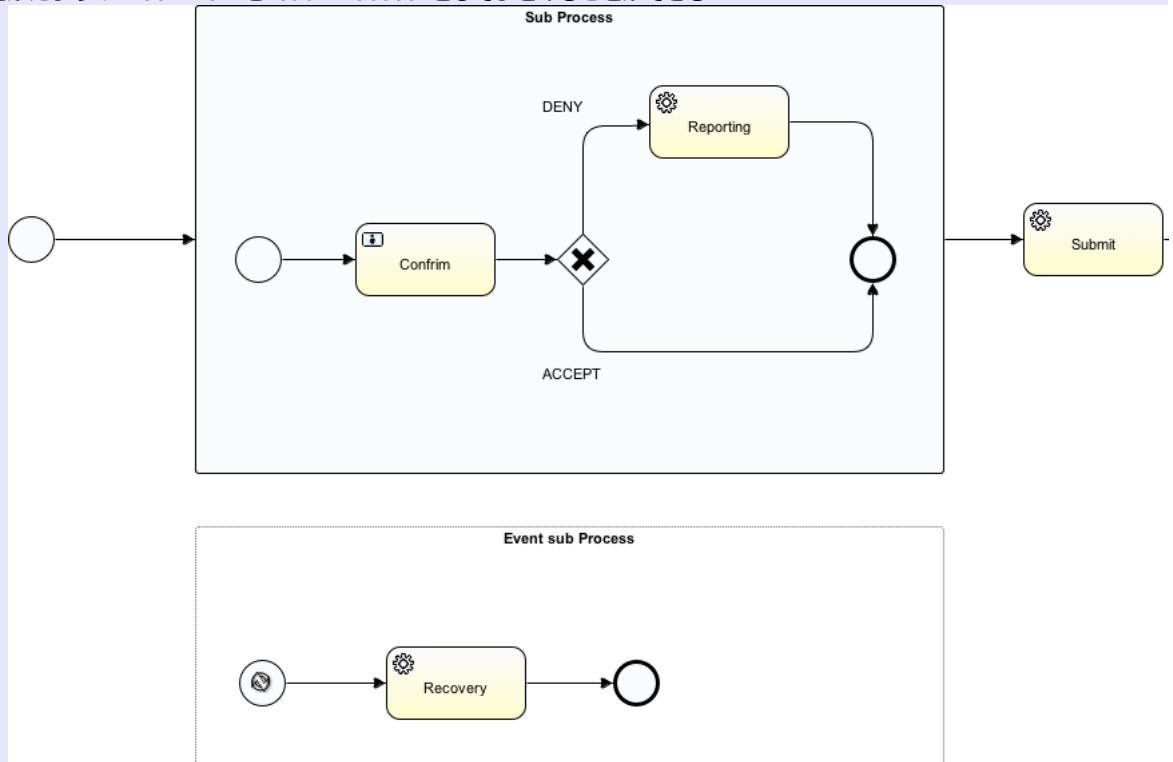
```

44 16-05-18 23:49:38 [INFO ] nrOfInstances=10
45 16-05-18 23:49:38 [INFO ] nrOfCompletedInstances=6
46 16-05-18 23:49:38 [INFO ] This is MyJavaTask
47 16-05-18 23:49:38 [INFO ] nrOfActiveInstances=3
48 16-05-18 23:49:38 [INFO ] loopCounter=7
49 16-05-18 23:49:38 [INFO ] nrOfInstances=10
50 16-05-18 23:49:38 [INFO ] nrOfCompletedInstances=7
51 16-05-18 23:49:38 [INFO ] This is MyJavaTask
52 16-05-18 23:49:38 [INFO ] nrOfActiveInstances=2
53 16-05-18 23:49:38 [INFO ] loopCounter=8
54 16-05-18 23:49:38 [INFO ] nrOfInstances=10
55 16-05-18 23:49:38 [INFO ] nrOfCompletedInstances=8
56 16-05-18 23:49:38 [INFO ] This is MyJavaTask
57 16-05-18 23:49:38 [INFO ] nrOfActiveInstances=1
58 16-05-18 23:49:38 [INFO ] loopCounter=9
59 16-05-18 23:49:38 [INFO ] nrOfInstances=10
60 16-05-18 23:49:38 [INFO ] nrOfCompletedInstances=9
61 16-05-18 23:49:38 [INFO ] Complete

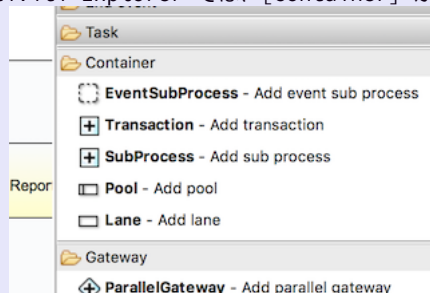
```

8.6. Sub-Processes and Call Activities

- ひとかたまりの Activiti を Sub-Process としてまとめることができる



- Activiti Explorer では、[Container] から [SubProcess?](#) と [EventSubProcess?](#) を選ぶことができる



- 通常の sub process は、Task と同じように扱うことができる。sub process 全体にエラーイベントを貼り付けることもできる
- event sub process は、Process 中のどこかで発生したイベントを契機に開始する
- 現時点では sub process にする意味はあまりない。
 - 各業務共通の Sub-Processes を「共通bpmnファイル」にくくりだして、各業務bpmnファイル から参照する ... というのなら意味がありそう
 - 現時点ではそういう使い方に対応する予定なし ⇒ <https://forums.activiti.org/content/how-include-other-bpmn-files>

9. Form

- Activiti Explorer は、bpmn 上の User Task や Start Event からユーザ入力画面を作ってくれる

- User Task や Start Event には、入力項目を指定できる

```

1  <startEvent>
2    <extensionElements>
3      <activiti:formProperty id="numberOfDays" name="Number of days"
4        value="${numberOfDays}" type="long" required="true"/>
5      <activiti:formProperty id="startDate" name="First day of holiday (dd-MM-yyy)"
6        value="${startDate}" datePattern="dd-MM-yyyy hh:mm" type="date" required="true"/>
7      <activiti:formProperty id="vacationMotivation" name="Motivation"
8        value="${vacationMotivation}" type="string" />
9    </extensionElements>
10 </userTask>

```

| property | note |
|----------|--|
| id | key項目。入力値は id をキーにしてプロセススコープの変数として保持される ((ACT_RU_VARIABLE テーブルに格納される)) |
| name | 表示するラベル |
| type | "string", "long", "enum", "date", "boolean" |
| value | 初期値。プロセススコープの変数 (ACT_RU_VARIABLE テーブルに格納されているプロセス固有の変数) を EL式で参照可能 |
| readable | "true"/"false" |
| writable | "true"/"false" |
| required | "true"/"false" |

- type="enum" の時には次のように候補を定義する

```

1  <userTask id="usertask1" name="User Task">
2    <extensionElements>
3      <activiti:formProperty id="sex" name="YOUR SEX" type="enum">
4        <activiti:value id="male" name="Male" />
5        <activiti:value id="female" name="Female" />
6      </activiti:formProperty>
7    </extensionElements>
8  </userTask>

```

GUI は Activiti Explorer 前提なら From の話はここまでで OK

- 自前のワークフローGUIを作りたい時、bpmn 上の Form 入出力定義から動的に画面を作りたいだろう
- [FormService2](#) にはそのための API がある

```

StartFormData FormService.getStartFormData(String processDefinitionId)
TaskFormdata FormService.getTaskFormData(String taskId)

```

- [StartFromData2](#) を検索するキーは、[processDefinitionId2](#) (プロセス定義のID)。 [TaskFormdata2](#) を検索するキーは [taskId](#) (実行中のタスクのID)。考えて見れば当たり前。
- [StartFormData2.getStartFormProperties2\(\)](#)、[TaskFormdata2.getFormProperties2\(\)](#) で [List<FormProperty2>](#) が返ってくる

```

1  public interface FormProperty {
2    FormService#submitStartFormData(String, java.util.Map)}
3    String getId();
4    String getName();
5    FormType getType();
6    String getValue();
7    boolean isReadable();
8    boolean isWritable();
9    boolean isRequired();
10 }

```

- <startEvent> や <userTask> に定義できる入出力項目 <activiti:formProperty> の属性と同じね
- 入出力項目の型は [FormType2](#) 型で返される [FormType2.getName\(\)](#) を使えば型名を文字列で取得できる
 - string (org.activiti.engine.impl.form.StringFormType2)
 - long (org.activiti.engine.impl.form.LongFormType2)
 - enum (org.activiti.engine.impl.form.EnumFormType2)
 - date (org.activiti.engine.impl.form.DateFormType2)
 - boolean (org.activiti.engine.impl.form.BooleanFormType2)

- Vacation Request を読み込むサンプル

<https://github.com/kagyuu/ActivitiExam/blob/master/src/main/java/com/snail/exam/FormProcess.java>

```

1  package com.snail.exam;
2
3  import java.util.HashMap;
4  import java.util.List;
5  import java.util.Map;
6
7  import org.activiti.engine.ProcessEngine;
8  import org.activiti.engine.ProcessEngines;
9  import org.activiti.engine.RepositoryService;
10 import org.activiti.engine.RuntimeService;
11 import org.activiti.engine.TaskService;
12 import org.activiti.engine.form.FormProperty;
13 import org.activiti.engine.form.StartFormData;
14 import org.activiti.engine.form.TaskFormData;
15 import org.activiti.engine.repository.ProcessDefinition;
16 import org.activiti.engine.runtime.ProcessInstance;
17 import org.activiti.engine.task.Task;
18 import org.apache.commons.lang3.time.DateUtils;
19 import org.slf4j.Logger;
20 import org.slf4j.LoggerFactory;
21
22 public class FormProcess {
23
24     private static final Logger log = LoggerFactory.getLogger(FormProcess.class);
25
26     public static void main(String[] args) {
27         try {
28             ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine();
29
30             // ----- 4.3.1. Deploying the process
31             log.info("--- #1. Deploying the process");
32             RepositoryService repositoryService = processEngine.getRepositoryService();
33             repositoryService.createDeployment()
34                 .addClasspathResource("org/activiti/test/VacationRequest.bpmn20.xml")
35                 .deploy();
36
37             // ----- 9. Forms (Read Start Form)
38             ProcessDefinition processDef = processEngine.getRepositoryService().createProcessDef
39                 .processDefinitionKey("vacationRequest").singleResult();
40
41             log.info("Vaction Request PID={}", processDef.getId());
42
43             StartFormData startForm = processEngine.getFormService().getStartFormData(processDef
44             for (FormProperty prop : startForm.getFormProperties()) {
45                 log.info("id={}, name={}, type={}, value={}, readable={}, required={}, writable=
46                     , prop.getId(), prop.getName(), prop.getType(), prop.getValue()
47                     , prop.isReadable(), prop.isRequired(), prop.isWritable());
48             }
49
50             // ----- 4.3.2. Starting a process instance
51             log.info("--- #3. Starting a process instance");
52
53             Map<String, Object> variables = new HashMap<String, Object>();
54             variables.put("employeeName", "Kermit");
55             variables.put("numberOfDays", new Integer(4));
56             variables.put("startDate", DateUtils.parseDate("1999-12-31", "yyyy-MM-dd"));
57             variables.put("vacationMotivation", "I'm really tired!");
58
59             // the process to run
60             // id      : <process id="vacationRequest">
61             // arguments : <activiti:formProperty>
62             RuntimeService runtimeService = processEngine.getRuntimeService();
63             ProcessInstance processInstance = runtimeService.startProcessInstanceByKey("vacation
64
65             // Fetch all tasks for the management group
66             TaskService taskService = processEngine.getTaskService();
67             List<Task> tasks = taskService.createTaskQuery().processDefinitionKeyLike("vacation
68             for (Task task : tasks) {
69                 TaskFormData taskForm = processEngine.getFormService().getTaskFormData(task.getI
70                 for (FormProperty prop : taskForm.getFormProperties()) {
71                     log.info("id={}, name={}, type={}, value={}, readable={}, required={}, writ
72                         , prop.getId(), prop.getName(), prop.getType(), prop.getValue()
73                         , prop.isReadable(), prop.isRequired(), prop.isWritable());
74                 }
75
76                 // Do task (reject application)
77                 Map<String, Object> taskVariables = new HashMap<String, Object>();
78                 taskVariables.put("vacationApproved", "false");
79                 taskVariables.put("managerMotivation", "We have a tight deadline!");
80                 taskService.complete(task.getId(), taskVariables);
81             }
82
83
84
85         } catch (Throwable th) {
86             log.error("ERROR", th);
87         }
88     }
89 }

```

```

88     }
89 }

```

- 実行結果

```

1 16-05-19 01:22:08 [INFO ] Initializing process engine using configuration 'file:/Users/atsushi/E
2 16-05-19 01:22:08 [INFO ] initializing process engine for resource file:/Users/atsushi/EclipseWo
3 16-05-19 01:22:08 [INFO ] Loading XML bean definitions from resource loaded through InputStream
4 16-05-19 01:22:09 [INFO ] performing create on engine with resource org/activiti/db/create/activ
5 16-05-19 01:22:09 [INFO ] performing create on history with resource org/activiti/db/create/acti
6 16-05-19 01:22:09 [INFO ] performing create on identity with resource org/activiti/db/create/acti
7 16-05-19 01:22:09 [INFO ] ProcessEngine default created
8 16-05-19 01:22:09 [INFO ] initialised process engine default
9 16-05-19 01:22:09 [INFO ] --- #1. Deploying the process
10 16-05-19 01:22:09 [INFO ] Processing resource org/activiti/test/VacationRequest.bpmn20.xml
11 16-05-19 01:22:10 [INFO ] Vacation Request PID=vacationRequest:1:4
12 16-05-19 01:22:10 [INFO ] id=numberOfDays, name=Number of days, type=org.activiti.engine.impl.f
13 value=null, readable=true, required=true, writable=true,
14 16-05-19 01:22:10 [INFO ] id=startDate, name=First day of holiday (dd-MM-yyy), type=org.activiti
15 value=null, readable=true, required=true, writable=true,
16 16-05-19 01:22:10 [INFO ] id=vacationMotivation, name=Motivation, type=org.activiti.engine.impl.
17 readable=true, required=false, writable=true,
18 16-05-19 01:22:10 [INFO ] --- #3. Starting a process instance
19 16-05-19 01:22:10 [INFO ] id=vacationApproved, name=Do you approve this vacation, type=org.activ
20 value=null, readable=true, required=true, writable=true,
21 16-05-19 01:22:10 [INFO ] id=managerMotivation, name=Motivation, type=org.activiti.engine.impl.f
22 value=null, readable=true, required=false, writable=true,

```

11. History ¹

- History Entities

| | |
|---|---------------------------|
| HistoricProcessInstance? | プロセスに関する情報 |
| HistoricVariableInstance? | プロセス変数に関する情報 |
| HistoricActivityInstance? | 操作に関する情報 |
| HistoricTaskInstance? | タスク情報 |
| HistoricDetail? | Form Submit項目、プロセス変数の変更履歴 |

- 設定 ⇒ [activiti.cfg.xml](#)

```

1 <bean id="processEngineConfiguration"
2 class="org.activiti.engine.impl.cfg.StandaloneInMemProcessEngineConfiguration">
3   <property name="history" value="audit" />
4   ...
5 </bean>

```

| history level | note | Prcess Instance | Activity Instance | From Properties | Variable(最終状態) | Variable(変更履歴) | その他 |
|---------------|---|-----------------|-------------------|-----------------|----------------|----------------|-----|
| none | 何も記録しない | | | | | | |
| activity | ユーザ操作をすべて査証 ^{※1} 可能 | ✓ | ✓ | ✓ | ✓ | | |
| audit | デフォルト設定。プロセス変数の変更履歴を査証 ^{※2} 可能 | ✓ | ✓ | ✓ | ✓ | ✓ | |
| full | すべてを記録 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

- API

- [HistoricProcessInstance?](#)

| | | |
|--------------------|--|--|
| String | getBusinessKey? () | The user provided unique reference to this process instance. |
| String | getDeleteReason? () | Obtains the reason for the process instance's deletion. |
| Long | getDurationInMillis? () | The difference between?getEndTime?() and?getStartTime?()? |
| Date | getEndTime? () | The time the process was ended. |
| String | getId() | The process instance id (== as the id for the run21:28:09process instance). |
| String | getName() | The name for the process instance. |
| String | getProcessDefinitionId? () | The process definition reference. |
| Map<String,Object> | getProcessVariables? () | Returns the process variables if requested in the process instance query. ※ なんか何にも入っていない Map が返される。プロセス変数は |

| | | |
|--------|--|--|
| | | HistoricVariableInstance2 からとればいいだろう |
| String | getStartActivitId2() | The start activit. |
| Date | getStartTime2() | The time the process was started. |
| String | getStartUserId2() | The authenticated user that started this process instance. |
| String | getSuperProcessInstanceId2() | The process instance id of a potential super process instance or null if no super process instance exists. |
| String | getTenantId2() | The tenant identifier for the process instance. |

◦ [HistoricVariableInstance2](#)

| | | |
|--------|---|---|
| Date | getCreateTime2() | Returns the time when the variable was created. |
| String | getId() | The unique DB id |
| Date | getLastUpdatedTime2() | Returns the time when the value of the variable was last updated. |
| String | getProcessInstanceId2() | The process instance reference. |
| String | getTaskId2() | |
| Object | getValue() | |
| String | getVariableName2() | |
| String | getVariableTypeName2() | |

◦ [HistoricActivityInstance2](#)

| | | |
|--------|---|---|
| String | getActivityId2() | The unique identifier of the activity in the process |
| String | getActivityName2() | The display name for the activity |
| String | getActivityType2() | The XML tag of the activity as in the process file |
| String | getAssignee() | Assignee in case of user task activity |
| String | getCalledProcessInstanceId2() | The called process instance in case of call activity |
| Long | getDurationInMillis2() | Difference between? getEndTime2() ?and? getStartTime2() . |
| Date | getEndTime2() | Time when the activity instance ended |
| String | getExecutionId2() | Execution reference |
| String | getId() | The unique identifier of this historic activity instance. |
| String | getProcessDefinitionId2() | Process definition reference |
| String | getProcessInstanceId2() | Process instance reference |
| Date | getStartTime2() | Time when the activity instance started |
| String | getTaskId2() | The corresponding task in case of task activity |
| String | getTenantId2() | Returns the tenant identifier for the historic activity |

◦ [HistoricDetail2](#)

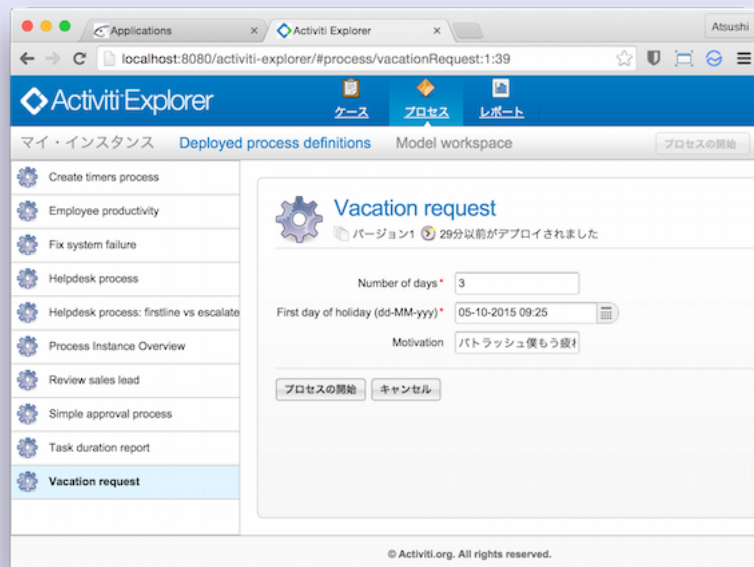
| | | |
|--------|--|---|
| String | getActivityInstanceId2() | The activity reference in case this detail is related to an activity instance. |
| String | getExecutionId2() | The identifier for the path of execution. |
| String | getId() | The unique DB id for this historic detail |
| String | getProcessInstanceId2() | The process instance reference. |
| String | getTaskId2() | The identifier for the task. |
| Date | getTime() | The time when this detail occurred |
| int | getRevision()? | (Query の検索結果が HistoricVariableUpdate2 extends HistoricDetail2 型のとき)the revision of Process variable |
| Object | getValue()? | (Query の検索結果が HistoricVariableUpdate2 extends HistoricDetail2 型のとき)Process variable |
| String | getVariableName2()? | (Query の検索結果が HistoricVariableUpdate2 extends HistoricDetail2 型のとき) |

| | | |
|--------|---------------------------------------|--|
| String | <code>getVariableTypeName2()</code> ? | (Query の検索結果が <code>HistoricVariableUpdate2</code> extends <code>HistoricDetail2</code> 型るとき) |
| String | <code>getPropertyId2()</code> | (Query の検索結果が <code>HistoricFormProperty2</code> extends <code>HistoricDetail2</code> 型るとき) the id or key of the Form property |
| String | <code>getPropertyValue2()</code> | (Query の検索結果が <code>HistoricFormProperty2</code> extends <code>HistoricDetail2</code> 型るとき) the Formsubmitted value |

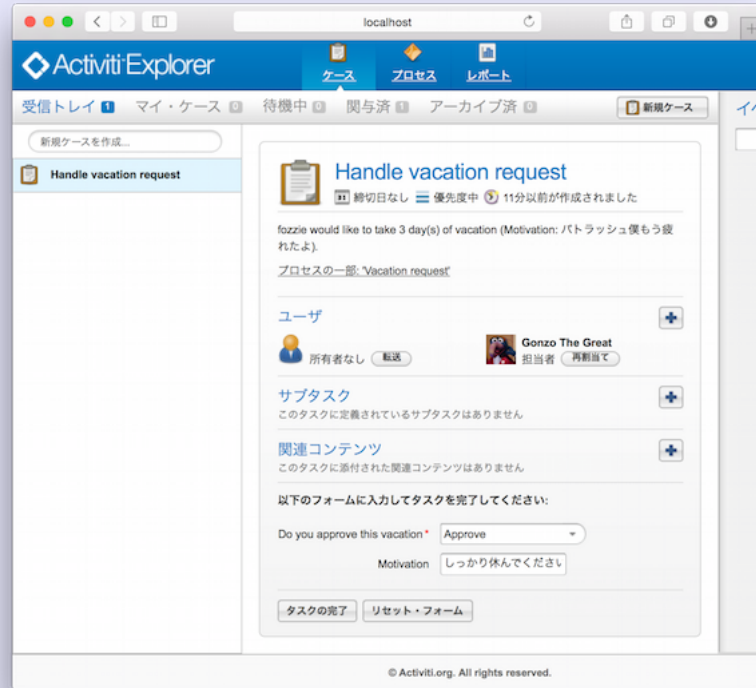
- `HistoricTaskInstance2`

| | | |
|--------|-------------------------------------|---|
| Date | <code>getClaimTime2()</code> | Time when the task was claimed. |
| String | <code>getDeleteReason2()</code> | The reason why this task was deleted {'completed', 'deleted', any other user defined string }. |
| Long | <code>getDurationInMillis2()</code> | Difference between <code>getEndTime2()</code> and <code>getStartTime2()</code> in milliseconds. |
| Date | <code>getEndTime2()</code> | Time when the task was deleted or completed. |
| Date | <code>getStartTime2()</code> | Time when the task started. |
| Long | <code>getWorkTimeInMillis2()</code> | Difference between <code>getEndTime2()</code> and <code>getClaimTime2()</code> in milliseconds. |

- サンプルプログラム (Vacation Request)
おなじみの Vacation Request
 - start で休暇申請



- manager が承認する



。サンプルコード

<https://github.com/kagyuu/ActivitiExam/blob/master/src/main/java/com/snail/exam/HistoryProcess.java>

```

1 package com.snail.exam;
2
3 import java.util.HashMap;
4 import java.util.List;
5 import java.util.Map;
6
7 import org.activiti.engine.HistoryService;
8 import org.activiti.engine.ProcessEngine;
9 import org.activiti.engine.ProcessEngines;
10 import org.activiti.engine.RepositoryService;
11 import org.activiti.engine.RuntimeService;
12 import org.activiti.engine.TaskService;
13 import org.activiti.engine.history.HistoricActivityInstance;
14 import org.activiti.engine.history.HistoricDetail;
15 import org.activiti.engine.history.HistoricFormProperty;
16 import org.activiti.engine.history.HistoricProcessInstance;
17 import org.activiti.engine.history.HistoricTaskInstance;
18 import org.activiti.engine.history.HistoricVariableInstance;
19 import org.activiti.engine.history.HistoricVariableUpdate;
20 import org.activiti.engine.repository.ProcessDefinition;
21 import org.activiti.engine.runtime.ProcessInstance;
22 import org.activiti.engine.task.Task;
23 import org.apache.commons.lang3.time.DateUtils;
24 import org.slf4j.Logger;
25 import org.slf4j.LoggerFactory;
26
27 public class HistoryProcess {
28
29     private static final Logger log = LoggerFactory.getLogger(HistoryProcess.class);
30
31     public static void main(String[] args) {
32         try {
33             ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine();
34
35             // ----- 4.3.1. Deploying the process
36             log.info("--- #1. Deploying the process");
37             RepositoryService repositoryService = processEngine.getRepositoryService();
38             repositoryService.createDeployment().addClasspathResource("org/activiti/test/Va
39                 .deploy());
40
41             log.info("Number of process definitions {}", repositoryService.createProcessDef
42                 for (ProcessDefinition p : repositoryService.createProcessDefinitionQuery().lis
43                     log.info("PROCESS DEF [id={},name={},key={}]", p.getId(), p.getName(), p.ge
44                 }
45
46             // ----- 4.3.2. Starting a process instance
47             log.info("--- #2. Starting a process instance");
48
49             Map<String, Object> variables = new HashMap<String, Object>();
50             variables.put("employeeName", "Kermit");
51             variables.put("numberOfDays", new Integer(4));
52             variables.put("startDate", DateUtils.parseDate("1999-12-31", "yyyy-MM-dd"));

```



```

53     variables.put("vacationMotivation", "I'm really tired!");
54
55     RuntimeService runtimeService = processEngine.getRuntimeService();
56     ProcessInstance processInstance = runtimeService.startProcessInstanceByKey("vacationRequest");
57
58     log.info("Number of process instances: " + runtimeService.createProcessInstanceQuery().list().size());
59     for (ProcessInstance p : runtimeService.createProcessInstanceQuery().list()) {
60         log.info("PROCESS INSTANCE [id={},pid={},pname={},pkey={}]", p.getId(), p.getProcessDefinitionId(),
61             p.getProcessDefinitionName(), p.getProcessDefinitionKey());
62     }
63
64     log.info("--- #3. Completing tasks (Accept Request)");
65
66     // Fetch all tasks for the management group
67     TaskService taskService = processEngine.getTaskService();
68     List<Task> tasks = taskService.createTaskQuery().taskCandidateGroup("management").processDefinitionKey("vacationRequest").list();
69     for (Task task : tasks) {
70         if (task.getTaskDefinitionKey().equals("handleRequest")) {
71             // Description is <documentation>.
72             log.info("TASK ACCEPT REQ [{}]", task.getDescription());
73
74             // Do task (reject application)
75             Map<String, Object> taskVariables = new HashMap<String, Object>();
76             taskVariables.put("vacationApproved", "true");
77             taskVariables.put("managerMotivation", "Okay, refresh yourself!");
78             taskService.complete(task.getId(), taskVariables);
79         }
80     }
81
82     // ----- 11. History
83     log.info("--- #4. History");
84     HistoryService historyService = processEngine.getHistoryService();
85
86     // 11.1.1. HistoricProcessInstanceQuery
87     List<HistoricProcessInstance> historicProcessInstances = historyService.createHistoricProcessInstanceQuery()
88         .finished().orderByProcessInstanceDuration().desc().listPage(0, 10);
89     log.info("TERMINATED PROCESS = {}", historicProcessInstances.size());
90     historicProcessInstances.forEach(p -> {
91         log.info("PROCESS INSTANCE [id={},pid={},name={},variables_size={}]",
92             p.getId(),
93             p.getProcessDefinitionId(),
94             p.getName(),
95             p.getProcessVariables().size());
96         p.getProcessVariables().forEach((k,v) -> {
97             log.info("{} = {}", k, v);
98         });
99     });
100
101     // 11.1.2. HistoricVariableInstanceQuery
102     List<HistoricVariableInstance> historicVariableInstances = historyService.createHistoricVariableInstanceQuery()
103         .orderByVariableName().asc().list();
104     log.info("VARIABLE SIZE = {}", historicVariableInstances.size());
105     historicVariableInstances.forEach(p -> {
106         log.info("VARIABLE [id={},pid={},task={},type={},name={},value={}]",
107             p.getId(),
108             p.getProcessInstanceId(),
109             p.getTaskId(),
110             p.getVariableTypeName(),
111             p.getVariableName(),
112             p.getValue());
113     });
114
115     // 11.1.3. HistoricActivityInstanceQuery
116     List<HistoricActivityInstance> historicActivityInstances = historyService.createHistoricActivityInstanceQuery()
117         .finished().orderByHistoricActivityInstanceEndTime().desc().list();
118     log.info("ACTIVITY SIZE = {}", historicActivityInstances.size());
119     historicActivityInstances.forEach(p -> {
120         log.info("ACTIVITY [id={},name={},task={},assignee={}]",
121             p.getActivityId(),
122             p.getActivityName(),
123             p.getTaskId(),
124             p.getAssignee());
125     });
126
127     // 11.1.4. HistoricDetailQuery
128     List<HistoricDetail> historicDetails = historyService.createHistoricDetailQuery()
129         .finished().list();
130     log.info("DETAIL SIZE = {}", historicDetails.size());
131     historicDetails.forEach(p -> {
132         if (p instanceof HistoricFormProperty) {
133             log.info("FORM [pid={},taskid={},property_id={},value={}]",
134                 p.getProcessInstanceId(),
135                 p.getTaskId(),
136                 ((HistoricFormProperty)p).getPropertyId(),
137                 ((HistoricFormProperty)p).getPropertyValue());
138         } else if (p instanceof HistoricVariableUpdate) {
139             log.info("VARIABLE [pid={},taskid={},revision={},name={},value={}]",
140                 p.getProcessInstanceId(),
141                 p.getTaskId(),
142                 p.getRevision(),
143                 p.getName(),
144                 p.getValue());
145         }
146     });

```

```

145         , ((HistoricVariableUpdate)p).getRevision()
146         , ((HistoricVariableUpdate)p).getVariableName()
147         , ((HistoricVariableUpdate)p).getValue()
148     );
149     }
150     });
151
152     // 11.1.5. HistoricTaskInstanceQuery
153     List<HistoricTaskInstance> historicTaskInstances = historyService.createHistoricTaskInstanceQuery()
154         .finished()
155         .orderByHistoricTaskInstanceDuration().desc()
156         .listPage(0, 10);
157     log.info("TASK SIZE = {}", historicTaskInstances.size());
158     historicTaskInstances.forEach(p -> {
159         log.info("TASK [id={},name={},time={},assignee={}]"
160             , p.getId()
161             , p.getName()
162             , p.getCreateTime()
163             , p.getAssignee()
164         );
165     });
166 } catch (Throwable th) {
167     log.error("ERROR", th);
168 }
169 }
170 }

```

。実行結果

```

1 16-05-23 21:34:18 [INFO ] Initializing process engine using configuration 'file:/Users/atsus
2 16-05-23 21:34:18 [INFO ] initializing process engine for resource file:/Users/atsushi/Eclip
3 16-05-23 21:34:18 [INFO ] Loading XML bean definitions from resource loaded through InputStr
4 16-05-23 21:34:19 [INFO ] performing create on engine with resource org/activiti/db/create/a
5 16-05-23 21:34:19 [INFO ] performing create on history with resource org/activiti/db/create/
6 16-05-23 21:34:19 [INFO ] performing create on identity with resource org/activiti/db/create
7 16-05-23 21:34:19 [INFO ] ProcessEngine default created
8 16-05-23 21:34:19 [INFO ] initialised process engine default
9 16-05-23 21:34:19 [INFO ] --- #1. Deploying the process
10 16-05-23 21:34:19 [INFO ] Processing resource org/activiti/test/VacationRequest.bpmn20.xml
11 16-05-23 21:34:21 [INFO ] Number of process definitions 1
12 16-05-23 21:34:21 [INFO ] PROCESS DEF [id=vacationRequest:1:4,name=Vacation request,key=vaca
13 16-05-23 21:34:21 [INFO ] --- #2. Starting a process instance
14 16-05-23 21:34:21 [INFO ] Number of process instances: 1
15 16-05-23 21:34:21 [INFO ] PROCESS INSTANCE [id=5,pid=vacationRequest:1:4,pname=Vacation requ
16 16-05-23 21:34:21 [INFO ] --- #3. Completing tasks (Accept Request)
17 16-05-23 21:34:21 [INFO ] TASK ACCEPT REQ [Kermit would like to take 4 day(s) of vacation (M
18 16-05-23 21:34:21 [INFO ] --- #4. History
19 16-05-23 21:34:21 [INFO ] TERMINATED PROCESS = 1
20 16-05-23 21:34:21 [INFO ] PROCESS INSTANCE [id=5,pid=vacationRequest:1:4,name=null,variables
21 16-05-23 21:34:21 [INFO ] VARIABLE SIZE = 6
22 16-05-23 21:34:21 [INFO ] VARIABLE [id=6,pid=5,task=null,type=string,name=employeeName,value
23 16-05-23 21:34:21 [INFO ] VARIABLE [id=15,pid=5,task=null,type=string,name=managerMotivation
24 16-05-23 21:34:21 [INFO ] VARIABLE [id=9,pid=5,task=null,type=integer,name=numberOfDays,valu
25 16-05-23 21:34:21 [INFO ] VARIABLE [id=10,pid=5,task=null,type=date,name=startDate,value=Fri
26 16-05-23 21:34:21 [INFO ] VARIABLE [id=14,pid=5,task=null,type=string,name=vacationApproved,
27 16-05-23 21:34:21 [INFO ] VARIABLE [id=8,pid=5,task=null,type=string,name=vacationMotivation
28 16-05-23 21:34:21 [INFO ] ACTIVITY SIZE = 5
29 16-05-23 21:34:21 [INFO ] ACTIVITY [id=theEnd1,name=null,task=null,assignee=null]
30 16-05-23 21:34:21 [INFO ] ACTIVITY [id=sendApprovalMail,name=Send confirmation e-mail,task=n
31 16-05-23 21:34:21 [INFO ] ACTIVITY [id=requestApprovedDecision,name=Request approved?,task=n
32 16-05-23 21:34:21 [INFO ] ACTIVITY [id=handleRequest,name=Handle vacation request,task=12,as
33 16-05-23 21:34:21 [INFO ] ACTIVITY [id=request,name=null,task=null,assignee=null]
34 16-05-23 21:34:21 [INFO ] DETAIL SIZE = 0
35 16-05-23 21:34:21 [INFO ] TASK SIZE = 1
36 16-05-23 21:34:21 [INFO ] TASK [id=12,name=Handle vacation request,time=Mon May 23 21:34:21

```

HistoricProcessInstance から Variable が取れないとか、謎仕様になっているけどまあいいか
 activiti.cfg.xml で <property name="history" value="full" /> にすると DETAIL が表示される。
 でも Form Property ではなくプロセス変数

```

1 16-05-23 22:06:05 [INFO ] DETAIL SIZE = 7
2 16-05-23 22:06:05 [INFO ] VARIABLE [pid=5,taskid=null,revision=0,name=vacationMotivation,valu
3 16-05-23 22:06:05 [INFO ] VARIABLE [pid=5,taskid=null,revision=0,name=numberOfDays,value=4]
4 16-05-23 22:06:05 [INFO ] VARIABLE [pid=5,taskid=null,revision=0,name=startDate,value=Fri Dec
5 16-05-23 22:06:05 [INFO ] VARIABLE [pid=5,taskid=null,revision=0,name=vacationApproved,value=
6 16-05-23 22:06:05 [INFO ] VARIABLE [pid=5,taskid=null,revision=0,name=managerMotivation,value
7 16-05-23 22:06:05 [INFO ] VARIABLE [pid=5,taskid=null,revision=0,name=employeeName,value=null
8 16-05-23 22:06:05 [INFO ] VARIABLE [pid=5,taskid=null,revision=0,name=employeeName,value=Kerm

```

プロセス変数周りは、細かい所色々謎仕様が多いけどまあいいか。一通りのことはできるし、必要になったら(プロセス監視画面をつくらうとか思い立ったら)もうちょっと細かい所検証する必要あり。まあその時は REST API 使うだろうけど

13. Activiti Explorer 14. Modeler

→ [Java Activiti Explorer](#)

15. REST API [↑]

→ [Java Activiti REST API](#)


[Java#Activiti](#)

 audit
 audit

Last-modified: 2018-01-30 (火) 01:18:32 (154d)

Site admin: [kagyuu](#)

PukiWiki 1.4.6 Copyright © 2001-2005 [PukiWiki Developers Team](#). License is [GPL](#).
Based on "PukiWiki" 1.3 by [yu-ji](#). Powered by PHP 5.3.3. HTML convert time: 3.017 sec.

| | | |
|---|---|---------------|
| ISBN10 | | ISBN13 |
| |  | |
| <input type="text" value="4061426060"/> | | 9784061426061 |