Work Flow Engine Manuals

List

[Work Flow Engine Manuals 1](#_Toc480376696)

[1.1 Prepare software 2](#_Toc480376697)

[1.2 Install Eclipse Plugin 2](#_Toc480376698)

[1. Build Maven Activiti Project Integrate Spring and MYBATIS 2](#_Toc480376699)

[2.1 Build Maven POM 2](#_Toc480376702)

[2.2 Spring Integration 7](#_Toc480376703)

[2.2.1 The Object Mapper 8](#_Toc480376704)

[2.2.2 JobExecutor 8](#_Toc480376705)

[2.2.3 AsyncExecutor 8](#_Toc480376706)

[2.2.4 History configuration 9](#_Toc480376707)

[2.2.5 Mail configuration 9](#_Toc480376708)

[2.2.6 Deployment cache configuration 9](#_Toc480376709)

[2.2.7 Process Engine 10](#_Toc480376710)

[2.2.8 Repository Service 11](#_Toc480376711)

[2.2.9 Runtime Service 12](#_Toc480376712)

[2.2.10 Task Service 13](#_Toc480376713)

[2.2.11 History Service 13](#_Toc480376714)

[3. Table Declare 14](#_Toc480376715)

[4. Q&A 16](#_Toc480376716)

[4.1 Is task service support role or group 16](#_Toc480376717)

[4.2 Is process service support auto execute? 17](#_Toc480376718)

[4.3 Is Sub-Process support review image? 18](#_Toc480376719)

[4.4 How to invoke outer program interface? 19](#_Toc480376720)

[4.5 How to separate business data with process data? 20](#_Toc480376721)

[4.6 Is support sub-process? 21](#_Toc480376722)

[4.7 Is task service Support assign? 21](#_Toc480376723)

[4.8 Is service support out of time notify? 22](#_Toc480376724)

[5. Appendix 22](#_Toc480376725)

Prepare Environment

* 1. Prepare software

Runtime environment: JDK 6+;

Server: Tomcat 6+;

Database Server: MYSQL 5.6.4+ (lower than 5.6.4 has no support for timestamps or dates with millisecond precision);

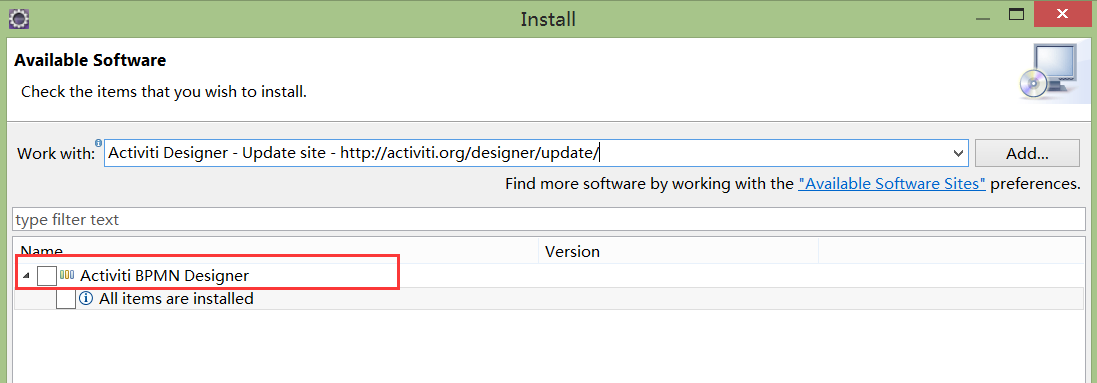
Activiti: 5.22.0 (latest stable version), download address: <https://github.com/Activiti/Activiti/releases/download/activiti-5.22.0/activiti-5.22.0.zip>

Library: Maven 3.x

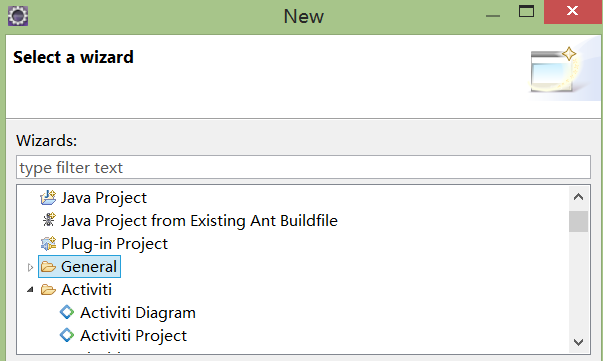
* 1. Install Eclipse Plugin

Open Eclipse and Go to Help → Install New Software and enter address:

Activiti Designer - Update site - http://activiti.org/designer/update/



Installed and new project in Eclipse had Activiti Diagram and Activiti Project choice (Install plugin also could use file link)



1. Build Maven Activiti Project Integrate Spring and MYBATIS

2.1 Build Maven POM

Create Activiti Project (default Maven Project) and convert it to Web Project. Configure repository (The Activiti 5.22.0 support spring 4.0+):

Example:

|  |
| --- |
| <properties>  <activiti-version>5.22.0</activiti-version>  <org.springframework-version>4.3.7.RELEASE</org.springframework-version>  <!-- 启用log4j由slf4j路由 -->  <slf4j.version>1.6.4</slf4j.version>  </properties> |

Other than configure spring, web related, MYSQL, MYBATIS, JDBC connector, file upload, IO, we should add log slf4j (Activiti log System Default Support) and JSON parse jar (for store and parse BPMN XML or PNG File).

Example :

|  |
| --- |
| <dependency>  <groupId>org.codehaus.jackson</groupId>  <artifactId>jackson-mapper-asl</artifactId>  <version>1.9.2</version>  </dependency>  <dependency>  <groupId>org.codehaus.jackson</groupId>  <artifactId>jackson-core-asl</artifactId>  <version>1.9.2</version>  </dependency>  <dependency>  <groupId>org.codehaus.plexus</groupId>  <artifactId>plexus-interpolation</artifactId>  <version>1.19</version>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-api</artifactId>  <version>1.7.6</version>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-jdk14</artifactId>  <version>1.7.6</version>  </dependency>  <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>1.2.15</version>  <exclusions>  <exclusion>  <groupId>javax.mail</groupId>  <artifactId>mail</artifactId>  </exclusion>  <exclusion>  <groupId>javax.jms</groupId>  <artifactId>jms</artifactId>  </exclusion>  <exclusion>  <groupId>com.sun.jdmk</groupId>  <artifactId>jmxtools</artifactId>  </exclusion>  <exclusion>  <groupId>com.sun.jmx</groupId>  <artifactId>jmxri</artifactId>  </exclusion>  </exclusions>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>${slf4j.version}</version>  </dependency> |

Last, Some Activiti Support Jar should be added.

Example :

|  |
| --- |
| <!-- activity -->  <dependency>  <groupId>org.activiti</groupId>  <artifactId>activiti-engine</artifactId>  <version>${activiti-version}</version>  </dependency>  <dependency>  <groupId>org.activiti</groupId>  <artifactId>activiti-spring</artifactId>  <version>${activiti-version}</version>  </dependency>  <dependency>  <groupId>org.codehaus.groovy</groupId>  <artifactId>groovy-all</artifactId>  <version>2.4.3</version>  </dependency>  <!-- activity diagram-editor -->  <dependency>  <groupId>org.activiti</groupId>  <artifactId>activiti-diagram-rest</artifactId>  <version>${activiti-version}</version>  </dependency>  <!-- Batik的包主要是用来解析html中的svg的内容 -->  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-transcoder</artifactId>  <version>1.7</version>  </dependency>  <dependency>  <groupId>org.activiti</groupId>  <artifactId>activiti-json-converter</artifactId>  <version>${activiti-version}</version>  <exclusions>  <exclusion>  <artifactId>commons-collections</artifactId>  <groupId>commons-collections</groupId>  </exclusion>  </exclusions>  </dependency>  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-bridge</artifactId>  <version>1.7</version>  </dependency>  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-css</artifactId>  <version>1.7</version>  </dependency>  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-anim</artifactId>  <version>1.8</version>  </dependency>  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-codec</artifactId>  <version>1.7</version>  </dependency>  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-ext</artifactId>  <version>1.7</version>  </dependency>  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-gvt</artifactId>  <version>1.7</version>  </dependency>  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-script</artifactId>  <version>1.7</version>  </dependency>  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-parser</artifactId>  <version>1.7</version>  </dependency>  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-svg-dom</artifactId>  <version>1.7</version>  </dependency>  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-svggen</artifactId>  <version>1.7</version>  </dependency>  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-util</artifactId>  <version>1.7</version>  </dependency>  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-xml</artifactId>  <version>1.7</version>  </dependency>  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-js</artifactId>  <version>1.7</version>  </dependency>  <dependency>  <groupId>xml-apis</groupId>  <artifactId>xml-apis-ext</artifactId>  <version>1.3.04</version>  </dependency>  <dependency>  <groupId>xml-apis</groupId>  <artifactId>xml-apis</artifactId>  <version>1.3.04</version>  </dependency>  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>xmlgraphics-commons</artifactId>  <version>1.2</version>  </dependency>  <dependency>  <groupId>org.apache.xmlgraphics</groupId>  <artifactId>batik-awt-util</artifactId>  <version>1.7</version>  </dependency> |

2.2 Spring Integration

|  |
| --- |
| <bean id=*"objectMapper"* class=*"com.fasterxml.jackson.databind.ObjectMapper"* />  <!-- spring负责创建流程引擎的配置文件 -->  <bean id=*"processEngineConfiguration"* class=*"org.activiti.spring.SpringProcessEngineConfiguration"*>  <!-- 数据源 -->  <property name=*"dataSource"* ref=*"dataSource"* />  <!-- 配置事务管理器，统一事务 -->  <property name=*"transactionManager"* ref=*"transactionManager"* />  <!-- 设置建表策略，如果没有表，自动创建表 -->  <property name=*"databaseSchemaUpdate"* value=*"false"* />  <!-- 作业异步执行器 -->  <property name=*"jobExecutorActivate"* value=*"false"* />  <property name=*"asyncExecutorEnabled"* value=*"true"* />  <!-- 指定引擎启动启动Async执行程序线程池 -->  <property name=*"asyncExecutorActivate"* value=*"false"* />  <property name=*"history"* value=*"full"*/>  <!-- 邮件服务器 -->  <!-- <property name="mailServerHost" value="mail.my-corp.com" />  <property name="mailServerPort" value="5025" /> -->  <!-- 进程定义缓存限制，默认不限制 -->  <!-- <property name="processDefinitionCacheLimit" value="10" /> -->  <!-- <property name="processDefinitionCache">  <bean class="org.activiti.MyCache" />  </property> -->  <!-- knowledgeBaseCacheLimit，knowledgeBaseCache用于配置规则缓存 -->  </bean>    <!-- 创建流程引擎对象 -->  <bean id=*"processEngine"* class=*"org.activiti.spring.ProcessEngineFactoryBean"*>  <property name=*"processEngineConfiguration"* ref=*"processEngineConfiguration"* />  </bean>  <!-- 由流程引擎对象，提供的方法，创建项目中使用的Activiti工作流的Service -->  <bean id=*"repositoryService"* factory-bean=*"processEngine"* factory-method=*"getRepositoryService"* />  <bean id=*"runtimeService"* factory-bean=*"processEngine"* factory-method=*"getRuntimeService"* />  <bean id=*"taskService"* factory-bean=*"processEngine"* factory-method=*"getTaskService"* />  <bean id=*"identityService"* factory-bean=*"processEngine"* factory-method=*"getIdentityService"* />  <bean id=*"managementService"* factory-bean=*"processEngine"* factory-method=*"getManagementService"* />  <bean id=*"historyService"* factory-bean=*"processEngine"* factory-method=*"getHistoryService"* />  <bean id=*"formService"* factory-bean=*"processEngine"* factory-method=*"getFormService"* /> |

* + 1. The Object Mapper

The Object Mapper of Jackson jar is used to parse BPMN xml to JSON data store in Database with BLOB format.

* + 1. JobExecutor

JobExecutor is a component that manages a couple of threads to fire timers (and later also asynchronous messages).We can use API MangementService.createJobQuery or ManagementService.executeJob to control. By default, the JobExecutor is activated, set the value false if we don’t want it to be activated.

* + 1. AsyncExecutor

AsyncExecutor is a component that manages a thread poll to fire timers and other asynchronous tasks. By default, the AsyncExecutor is not enable, Usually, AysncExecutor is used recommended instead of JobExecutor. If we want to use it, configure like this:

|  |
| --- |
| <property name=*"asyncExecutorEnabled"* value=*"true"* />  <property name=*"asyncExecutorActivate"* value=*"true"* /> |

asyncExecutorEnable is to enable the Async executor instead of Old Job executor.

ayncExecutorActivate instructs the Activiti engine to startup the Async executor thread pool at startup.

Both executors deal with timers and asynchronous jobs in the Activiti Engine, so only one executor can be enabled.

* + 1. History configuration

History configuration is optional, using the enum org.activiti.engine.impl.history.HistoryLevel . Following history levels can be configured:

None: skips all history archiving. This is the most performances for runtime process execution, but no historical information will be available.

Activity: archives all process instances and activity instances, the latest values of the top level process instance variables will be copied to historic variable instances. No details will be archived.

Audit: This is the default. It archives all process instances, activity instances, keeps variable values continuously in sync and all form properties that are submitted can be audited.

Full: This is the highest level of history archiving and hence the slowest. It contain all other possible details, mostly this are process variable updates and contain all Audit level values.

The history level also can be configured programmatically, like this:

|  |
| --- |
| ProcessEngine processEngine = ProcessEngineConfiguration.*createProcessEngineConfigurationFromResourceDefault*().setHistoryLevel(HistoryLevel.*AUDIT*).buildProcessEngine(); |

* + 1. Mail configuration

Mail configuration is optional. To actually send an e-mail, a valid SMTP mail server configuration is required.

mailServerHost: Not required, Default is localhost.

mailServerPort: Required, Default is 25

mailServerDefaultFrom: Not Required, Default is [activiti@activiti.org](mailto:activiti@activiti.org)

…

* + 1. Deployment cache configuration

Process definition data doesn’t change. To avoid hitting the database every time, cached is needed. By default, there is no limit on this cache. To limit the process definition cache, set value in configuration.

|  |
| --- |
| <property name="processDefinitionCacheLimit" value="10" /> |

Setting this property will swap the default hashmap cache with a LRU cache that has the provided hard limit.

Also, we can inject own cache implementation, but must implements the org.activiti.engine.impl.persistence.deploy.DeploymentCache interface:

|  |
| --- |
| <property name=*"processDefinitionCache"*>  <bean class=*"org.activiti.MyCache"* />  </property> |

KnowledgeBaseCacheLimit, knowledgeBaseCache is used to configure the rules cache. This is only needed when use the rules task in processes.

* + 1. Process Engine

1. The easiest way get ProcessEngine:

|  |
| --- |
| ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine(); |

1. Now, if we used SPRING we can configure it into SPRING configuration to auto inject.
2. If we doesn’t used SPRING, We should Create ProcessEngineConfiguration first.

Example:

|  |
| --- |
| ProcessEngineConfiguration processEngineConfiguration = ProcessEngineConfiguration.*createProcessEngineConfigurationFromResourceDefault*();  ProcessEngineConfiguration processEngineConfiguration = ProcessEngineConfiguration.createProcessEngineConfigurationFrom Resource(String resource);  ProcessEngineConfiguration processEngineConfiguration = ProcessEngineConfiguration.createProcessEnginConfigurationFromResource(String resource,String beanName)  ProcessEngineConfiguration processEngineConfiguration = ProcessEngineConfiguration.createProcessEngineConfigurationFromInputStream(...) |

Also, There is several Default construct can be used:

|  |
| --- |
| ProcessEngineConfiguration processEngineConfiguration = ProcessEngineConfiguration.createStandaloneProcessEngineConfiguration() |

StandaloneProcessEngineConfiguration: is used in standalone way. Activiti will take care of transactions. The database will only be checked when engine boots.

|  |
| --- |
| ProcessEngineConfiguration processEngineConfiguration = ProcessEngineConfiguration.createStandoneInMemProcessEngineConfiguration()  ProcessEngineConfiguration processEngineConfiguration = ProcessEngineConfiguration.createStandoneInMemProcessEngineConfiguration().setDatabaseSchemaUpdate(ProcesEngineConfiguration.DB\_SCHEMA\_UPATE\_FALSE).setJdbcUrl(“jdbc:mysql://127.0.0.1/activiti“) |

StandoloneInMemProcessEngineConfiguration: is convenience for unit testing purposes, Activiti will take care of the transactions. By default, H2 in-Memory database is used. The database will be created and dropped when the engine boots and shuts down. When using Job executor or mail capabilities, other configuration is needed.

SpringProcessEngineConfiguration is used in a SPRING environment, like this text.

JtaProcessEngineConfiguration is used when engine runs in standalone mode, with JTA transactions.

At last we create ProcessEngine from calling method, like this:

|  |
| --- |
| ProcessEngine processEngine = processEngineConfiguration.buildProcessEngine() |

* + 1. Repository Service

Repository service is related with ACT\_RE\_\* tables, offers operations for managing and manipulation deployments and process definitions. (Static information)

Usually used API like this:

1. Create and deploy process definition.

|  |
| --- |
| repositoryService.createDeployment().name(deloyName).addZipInputStream(new ZipInputStream(new FileInputStream(“exampleFile”))).deploy();  repositoryService.createDeployment().addClasspathResource(“org/activity/test/example.bpmn.xml”); |

1. Delete process definition

|  |
| --- |
| repositoryService.deleteDeployment(deploymentId, true);  //delete model  repositoryService.deleteModel(modelId); |

1. Update process definition

When table ACT\_PROCDEF\_INFO exist KEY\_, deployed process contain the same key will update original and version update increment one.

|  |
| --- |
| repositoryService.createDeployment().name(“defineKey”).addString(processName, new String(new byte[])).deploy() |

1. Query process definition, deployment , Model, Identity or Resource

|  |
| --- |
| repositoryService.createDeploymentQuery().orderByDeploymenTime().asc().list();  repositoryService.createProcessDefinitionQuery().orderByProcessDefinitionVersion().asc().list();  repositoryService.getResourceAstream(deploymentId, imagName);  repositoryService.createModelQuery().list()  repositoryService.getIdentityLinksForProcessDefinition(“processDefinitionId”); |

1. Suspending and activating a process

When a process definition is suspended, new process instance can’t be created (an exception will be thrown).

|  |
| --- |
| repositoryService.suspendProcessDefinitionByKey(“defineKey”);  repositoryService.activateProcessDefinitionByKey(“defineKey”); |

* + 1. Runtime Service

Runtime Service is related with ACT\_RU\_\* tables, offers operations for start new process and create process instance at same time. Also, it used to retrieve and store process variables, query on process instances and executions… (Runtime information).

Usually used API like this:

1. Start a new process

|  |
| --- |
| ProcessInstance processInstance = runtimeService.startProcessInstanceByKey(“processKey”, variables); |

1. Suspending and activating a process

When suspended, the process cannot be continue and no jobs will executed

|  |
| --- |
| runtimeService.suspendProcessInstance();  runtimeService.activateProcessInstanceByKey(); |

1. Add database change listener

By default, Event mechanism of the Activiti engine is disabled. This event logging mechanism is additional to the traditional history manager of Activiti. The real use case is audit trailing and feeding it into a big data store (like NoSQL such as MongoDB). If we enable the database logger like this, the table required created by default (ACT\_EVT\_LOG). By default, this table can be deleted.

|  |
| --- |
| EventLogger databaseEventLogger = new EventLogger(processEngineConfiguration.getClock(), objectMapper);  runtimeService.addEventListener(databaseEventLogger); |

1. Store and retrieve variables

|  |
| --- |
| runtimeService.setVariable();  runtimeService.setVariablesLocal();  runtimeService.getVariables();  runtimeService.getVariablesLocal(); |

1. Query process Instance or execution

|  |
| --- |
| ProcessInstance pi = runtimeService.createProcessInstanceQuery().processInstanceId(“”).list();  Execution execution = runtimeService.createExedcutionQuery().processInstanceId(“”).activityId(“”).singleResult(); |

* + 1. Task Service

Task Service is related with ACT\_RU\_\*, ACT\_HI\_\*, ACT\_RE\_\* tables. Usually used API like this.

1. Querying tasks comments information

|  |
| --- |
| List<Comment> list = taskService.getProcessInstanceComments(“”); |

2. Querying tasks assigned to users or groups

|  |
| --- |
| List<Task> tasks = taskService.createTaskQuery().taskCandidateGroup(“management”).list();  List<Task> tasks = taskService.createTaskQuery().taskAssignee(“”).processVariableValueEquals(“”,””).list(); |

3. Claiming and completing task

|  |
| --- |
| taskService.complete(task.getId(), taskVariables);  taskService.claim(task.getId(),””); |

* + 1. History Service

History service is related with ACT\_HI\_\* tables. The service exposes mainly query capabilities to access historical data gathered by Activiti engine.

Query API:

|  |
| --- |
| //Query table ACT\_HI\_PROCINST  HistoricProcessInstance historicProcessInstance = historyService.createHistoricProcessInstanceQuery().processInstanceId(processId).singleResult();  //Query table ACT\_HI\_VARINST  List<HistoricVariableInstance> historicVariableInstance = historyService.createHistoricVariableInstanceQuery().processInstanceId(“”).list();  //Query table ACT\_HI\_ACTINST  List<HistoricActivityInstance> historicActivityInstance = historyService.createHistoricActivityInstanceQuery().processDefinitionId(“”).activityType(“”).listPage(0,1);  //Query table ACT\_HI\_TASKINST  HistoricTaskInstanceEntity taskEntity = historyService.createHistoricTaskInstanceQuery(“”).taskId(“”).singleResult();  //Query table ACT\_HI\_DETAIL  List<HistoricDetail> historicDetails = historyService.createHistoricDetailQuery().activityInstanceId(“”).list(); |

# 3. Table Declare

|  |  |  |
| --- | --- | --- |
| Table name | Service | Remark |
| ACT\_RE\_DEPLOYMENT | Repository Service  Example:  repositoryService.createDeployment().addClasspathResource(“org/activity/test/example.bpmn.xml”); | Deployment Information. Create data when deployed, static information |
| ACT\_RE\_PROCDEF | Repository Service  Example:  repositoryService.createDeployment().addClasspathResource(“org/activity/test/example.bpmn.xml”); | Deployment Information. Create data when deployed, static information |
| ACT\_RE\_MODEL | Repository Service  Example:  repositoryService.saveModel(repositoryService.newModel); | Model Information. Create data when save model, static information. |
| ACT\_RU\_VARIABLE | Runtime Service  Example:  runtimeService.getVariables();  runtimeService.startProcessInstanceByKey(“processKey”, variables);  taskService.complete(task.getId(), taskVariables); | Runtime Information.  Create data when store Variable. |
| ACT\_RU\_TASK | Runtime Service  Example:  runtimeService.startProcessInstanceByKey(“processKey”, variables);  List<Task> tasks = taskService.createTaskQuery().taskAssignee(“”).processVariableValueEquals(“”,””).list(); | Runtime Information.  Create data when Process start and BMPN.xml contain user task. |
| ACT\_RU\_JOB | Management Service  Example:  managementService.createJobQuery().list(); | Runtime Information.  Create data when Process start and bpmn contain Job executor/Async Executor/Service task/timer boundary event/ timer event |
| ACT\_RU\_EXECUTION | Runtime Service  Example:  runtimeService.createExecutionQuery().activityId(“”).list(); | Runtime Information.  Create data when Process start and create at least one record in current node. |
| ACT\_RU\_IDENTITYLINK | Runtime Service  Example:  runtimeService.addUserIdentityLink(processInstanceId,userId,identityLinkType); | Runtime Information.  create data when task service assigned |
| ACT\_RU\_EVENT\_SUBSCR | Management Service |  |
| ACT\_ID\_GROUP | Identity Service | Activity privilege System |
| ACT\_ID\_INFO | Identity Service | Activity privilege System |
| ACT\_ID\_MEMBERSHIP | Identity Service | Activity privilege System |
| ACT\_ID\_USER | Identity Service | Activity privilege System |
| ACT\_HI\_ACTINST | History Service  Example  historyService.createHistoricVariableInstanceQuery().processInstanceId(“”).list(); | Runtime/Complete Process. Create data when process store or update variable. |
| ACT\_HI\_ATTACHMENT | History Service | Runtime/Complete Process. Create data when process store attachment file |
| ACT\_HI\_COMMMENT | Task Service  Example  taskService.addComment(taskId,processInstanceId,message); | Runtime/Complete  Process. Create data when process store comment info. |
| ACT\_HI\_DETAIL | History Service  Example  historyService.createHistoricDetailQuery(); | Process Runtime or Completed create full detail information contain variables, activities, executions… |
| ACT\_HI\_IDENTITYLINK | History Service  Example:  historyService.getHistoricIdentityLinksForProcessInstance(processIntanceId); | Historic privilege information related with process |
| ACT\_HI\_PROCINST | History Service  Example:  historyService.createHistoricProcessInstanceQuery().processInstanceId(processId).singleResult(); | Historic process instance information |
| ACT\_HI\_TASKINST | History Service  Example:  historyService.createHistoricTaskInstanceQuery(“”).taskId(“”).singleResult(); | Historic process task information |
| ACT\_HI\_VARINST | History Service  Example:  historyService.createHistoricVariableInstanceQuery().processInstanceId(“”).list(); | Historic process variables |
| ACT\_GE\_BYTEARRARY | Repository Service | BPMN Model serialized data or PNG byte data and other static information |
| ACT\_GE\_PROPERTY | Management Service | Project Start and Activiti Initialized Information contain Activiti version… |
| ACT\_EVT\_LOG | Runtime Service | Database change information |

# 4. Q&A

## 4.1 Is task service support role or group

Yes, In Activiti, There is exist its own privilege, so ACT\_ID\_GROUP table correlated with ACT\_ID\_USER table should be used.

The Activiti identity component is used.

Example 01, code implementation

|  |
| --- |
| taskService.addCandidateGroup(“taskId”,”groupId”);  taskService.createTaskQuery().taskCandidateGroup(“groupId”); |

Example 02, bpmn.xml implementation

|  |
| --- |
| <userTask id='theTask' name='important task' >  <potentialOwner>  <resourceAssignmentExpression>  <formalExpression>user(kermit), group(management)</formalExpression><!-- the engine defaults to group -->  </resourceAssignmentExpression>  </potentialOwner>  </userTask> |

Example 03, bpmn.xml implementation

|  |
| --- |
| <userTask id="theTask" name="my task" activiti:candidateGroups="management, accountancy" /> |

The Activiti identity component is not used

Example 04, Code Implementation, task listener implement

|  |
| --- |
| <userTask id="task1" name="My task" >  <extensionElements>  <activiti:taskListener event="create" class="org.activiti.MyAssignmentHandler" />  </extensionElements>  </userTask> |
| public class MyAssignmentHandler implements TaskListener {  public void notify(DelegateTask delegateTask) {  // Execute custom identity lookups here  // and then for example call following methods:  delegateTask.addCandidateGroup("management");  ...  }  } |

Example 05, Spring Integration implementation

|  |
| --- |
| <userTask id="task" name="My Task" activiti:candidateUsers="${ldapService.findAllSales()}"/> |
| public class FakeLdapService {  public String findManagerForEmployee(String employee) {  return "Kermit The Frog";  }  public List<String> findAllSales() {  return Arrays.asList("kermit", "gonzo", "fozzie");  }  } |

## 4.2 Is process service support auto execute?

Yes, In Activiti, Script Task, service Task is support automatic execute.

Example 01, Script Task

|  |
| --- |
| <scriptTask id="theScriptTask" name="Execute script" scriptFormat="groovy">  <script>  sum = 0  for ( i in inputArray ) {  sum += i  }  </script>  </scriptTask> |

Example 02, Service Task

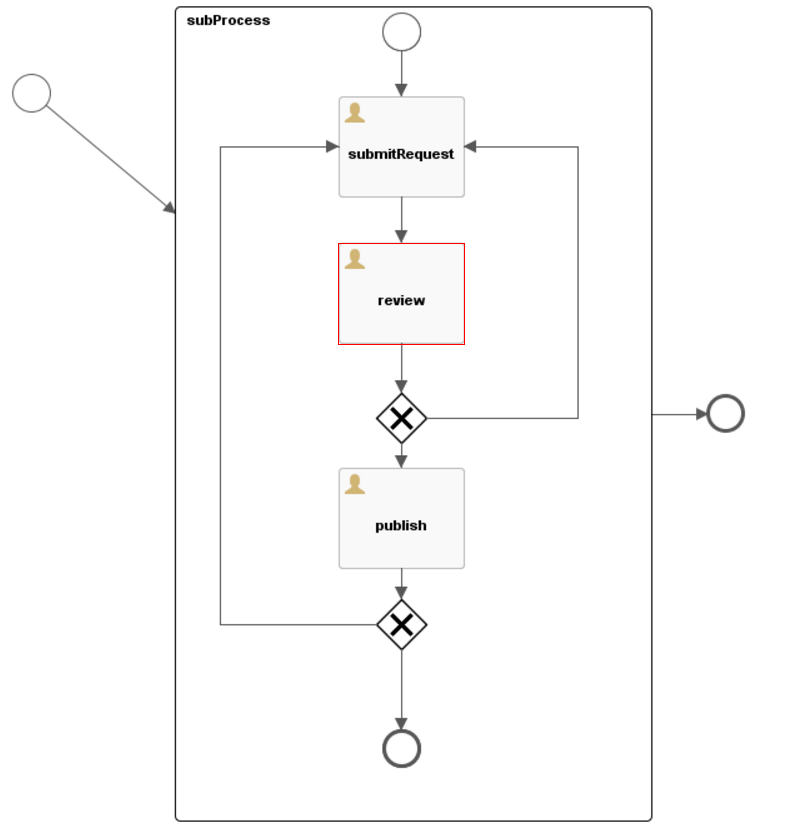
|  |
| --- |
| <serviceTask id="javaService"  name="My Java Service Task"  activiti:class="org.activiti.ToUppercase" /> |
| public class ToUppercase implements JavaDelegate {  public void execute(DelegateExecution execution) throws Exception {  String var = (String) execution.getVariable("input");  var = var.toUpperCase();  execution.setVariable("input", var);  }  } |

## 4.3 Is Sub-Process support review image?

Yes, Get current activity X, Y, Width and Height value and draw it to image.

Example:

|  |
| --- |
| String activityId = runtimeService.createExecutionQuery().executionId(task.getExecutionId()).singleResult().getActivityId();  ActivityImpl activityImpl = processDefinitionEntity.findActivity(activityId);  String x = activityImpl.getX();  String y = activityImpl.getY();  String width = activityImpl.getWidth();  String height = activityImpl.getHeight(); |



## 4.4 How to invoke outer program interface?

We can use Activiti call other program on the different application scenarios.

If called progress in different application with Activiti.

A Web Service task is used to synchronously invoke an external Web service.

Example:

|  |
| --- |
| <import importType="http://schemas.xmlsoap.org/wsdl/"  location="http://localhost:63081/counter?wsdl"  namespace="http://webservice.activiti.org/" />  <message id="prettyPrintCountRequestMessage" itemRef="tns:prettyPrintCountRequestItem" />  <message id="prettyPrintCountResponseMessage" itemRef="tns:prettyPrintCountResponseItem" />  <itemDefinition id="prettyPrintCountRequestItem" structureRef="counter:prettyPrintCount" />  <itemDefinition id="prettyPrintCountResponseItem" structureRef="counter:prettyPrintCountResponse" /> |

Defined BPMN interfaces and operations that actually reference the Web Service

|  |
| --- |
| <interface name="Counter Interface" implementationRef="counter:Counter">  <operation id="prettyPrintCountOperation" name="prettyPrintCount Operation"  implementationRef="counter:prettyPrintCount">  <inMessageRef>tns:prettyPrintCountRequestMessage</inMessageRef>  <outMessageRef>tns:prettyPrintCountResponseMessage</outMessageRef>  </operation>  </interface> |

Then declare a Web Service Task by using the ##WebService implementation and reference to the Web Service operation.

|  |
| --- |
| <serviceTask id="webService"  name="Web service invocation"  implementation="##WebService"  operationRef="tns:prettyPrintCountOperation"> |

If called progress in the same application with Activiti.

We can invoke the program directly with API, The example like this:

Example01, Execution Listeners, Task Listeners and Java Service tasks can directly be used.

If we have used SPRING to integrate.

|  |
| --- |
| <userTask id="myTask" name="My Task" >  <extensionElements>  <activiti:taskListener event="create" class="org.activiti.MyTaskCreateListener" />  </extensionElements>  </userTask> |
| public class MyTaskCreateListener implements TaskListener {  @Resource  private MyProgressService myProgressService;  public void notify(DelegateTask delegateTask) {  // Custom logic goes here  }  } |

In other case, we can use expression inject own service into BPMN.xml

|  |
| --- |
| <bean id="printer" class="org.activiti.examples.spring.Printer" />  <serviceTask id="print" activiti:expression="#{printer.printMessage()}" />  <serviceTask id="serviceTask" activiti:delegateExpression="${delegateExpressionBean}" /> |

Otherwise, if outer System use Activiti interface,

We can convenient use Activiti Engine with Activiti REST API.

## 4.5 How to separate business data with process data?

Yes, Activiti Engine has separate business and process. Using Business Key in ACT\_RU\_EXECUTION to the association.

## 4.6 Is support sub-process?

Yes, BPMN 2.0 allows sub-process, as well as Activiti Engine.

Also Event Sub-Process, Transaction sub-process was supported by Activiti Engine.

## 4.7 Is task service Support assign?

Yes, we can configure in the BPMN.xml or defined in Program.

Example01: HumanPerformer

|  |
| --- |
| <userTask id='theTask' name='important task' >  <humanPerformer>  <resourceAssignmentExpression>  <formalExpression>kermit</formalExpression>  </resourceAssignmentExpression>  </humanPerformer>  </userTask> |

Example02: candidate task

|  |
| --- |
| <userTask id='theTask' name='important task' >  <potentialOwner>  <resourceAssignmentExpression>  <formalExpression>user(kermit), group(management)</formalExpression>  </resourceAssignmentExpression>  </potentialOwner> |

Example03: assignee

|  |
| --- |
| <userTask id="theTask" name="my task" activiti:assignee="kermit" />  <userTask id="task" name="My Task" activiti:assignee="${ldapService.findManagerForEmployee(emp)}"/> |

Example04: candidateUsers

|  |
| --- |
| <userTask id="theTask" name="my task" activiti:candidateUsers="kermit, gonzo" /> |

Example06: TaskListener

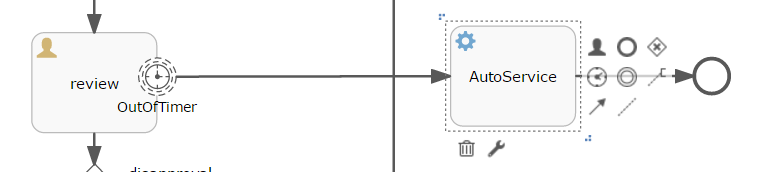
|  |
| --- |
| public class MyAssignmentHandler implements TaskListener {  public void notify(DelegateTask delegateTask) {  // Execute custom identity lookups here  // and then for example call following methods:  delegateTask.setAssignee("kermit");  delegateTask.addCandidateUser("fozzie");  delegateTask.addCandidateGroup("management");  ...  }  } |

## 4.8 Is service support out of time notify?

Yes. A timer boundary event could acts as alarm clock. When an execution arrives in the activity where the boundary event is attached to, a timer is started.

Example:

|  |
| --- |
| <boundaryEvent id="OutOfTimer" name="OutOfTimer" attachedToRef="review" cancelActivity="false">  <timerEventDefinition>  <timeDuration>PT12H</timeDuration>  </timerEventDefinition>  </boundaryEvent> |



1. Appendix

|  |  |  |
| --- | --- | --- |
| name | JBPM6 | Activity5 |
| engine | Drools | PVM |
| database | DB2,derby, H2,MYSQL5,ORACLE,SQLSERVER… | H2, MYSQL, ORACLE, DB2, MSSQL… |
| persistent | Hibernate, JPA | MyBatis, JPA, Hibernate |
| transaction | Bitronix, JTA | Mybatis, Spring |
| support | BPMN 2.0  JDK | BPMN 2.0  JDK  Support LDAP |
| extension | 37 tables | 23 tables |
| Remote interface | JMS,REST API | REST API, webservice |
| document | http://docs.jboss.org/jbpm/v6.3/userguide/ | https://www.activiti.org/userguide/ |

Comment: The document is just for your reference. If you have any question or find mistakes, please point them out and let me know.