Activiti5 学习笔记

hrj

目录

准备工作

JDK 5+

JDK1.5 以上版本

Ant 1.8.1+

Ant1.8.1 以上版本，运行自带的 Demo 必须。开发不要求。

Eclipse 3.6.2

Eclipse3.6.2 以上版本，Activiti5 可视化流程设计插件必须。开发不要求。

Activiti5

下载

http://activiti.org/download.html，当前使用版本 5.4。

包含的内容

下载的 Activiti 发布文件包含如下内容，先关注 doc、dependencies 下内容。

```
├──docs
|    ├──javadocs  API DOC
|    └──userguide 用户手册
├──setup 演示程序的配置脚本
|    └──files
|         ├──cfg.activiti
|         ├──cfg.cycle
|         ├──cfg.modeler
|         ├──demo
|         ├──dependencies 依赖 JAR 包说明
|         |    └──libs Activiti5 可能用到的所有 JAR 包，具体什么环境下用什么包参照上级目录的说明文件。
|         ├──h2
|         ├──tomcat
|         └──webapps
└──workspace 源代码
     ├──activiti-cxf-examples
     ├──activiti-cycle-examples
     ├──activiti-engine-examples
```

```
├──activiti-groovy-examples
├──activiti-modeler-examples
└──activiti-spring-examples
```

Activiti 的持久化方式

Activiti 使用 Mybatis3 做持久化工作，可以在配置中设置流程引擎启动时创建表。
Activiti 使用到的表都是 ACT_开头的。

ACT_RE_*:流程定义存储。
ACT_RU_*:流程执行记录，记录流程启动到结束的所有动作，流程结束后会清除相关记录。
ACT_ID_*:用户记录，流程中使用到的用户和组。
ACT_HI_*:流程执行的历史记录。
ACT_GE_*:通用数据及设置。

使用到的表：

ACT_GE_BYTEARRAY：流程部署的数据。
ACT_GE_PROPERTY：通用设置。
ACT_HI_ACTINST：流程活动的实例。
ACT_HI_ATTACHMENT：
ACT_HI_COMMENT：
ACT_HI_DETAIL：
ACT_HI_PROCINST：流程实例。
ACT_HI_TASKINST：任务实例。
ACT_ID_GROUP：用户组。
ACT_ID_INFO：
ACT_ID_MEMBERSHIP：
ACT_ID_USER：用户。
ACT_RE_DEPLOYMENT：部署记录。
ACT_RE_PROCDEF：流程定义。
ACT_RU_EXECUTION：流程执行记录。
ACT_RU_IDENTITYLINK：
ACT_RU_JOB：
ACT_RU_TASK：执行的任务记录。
ACT_RU_VARIABLE：执行中的变量记录。

Activiti 自带的几个组件简介和配置说明

activiti-administrator

自带的用户管理系统，维护用户和组，需要配置数据连接参数，在
activiti-administrator\WEB-INF\applicationContext.xml 中，并加入 JDBC 驱动包。

activiti-cycle

PVM 活动检测的，由 activiti-rest 提供服务，不需配置。

activiti-explorer

可以查看用户任务和启动流程，由 activiti-rest 提供服务，不需配置。

activiti-kickstart

简单的点对点流程定义维护工具，需要配置数据连接，把 activiti.cfg.xml 文件放在 classes 下，并加入驱动包。

activiti-modeler

在线编辑和维护流程定义的工具，最后以文件夹方式部署，需要配置
activiti-modeler\WEB-INF\classes\configuration.properties 文件。

activiti-probe

PVM 的观测服务，由 activiti-rest 提供服务，不需配置，可以查看 deployment、processdefinition、processinstance、database。

activiti-rest

其他几个应用的服务提供者，需要配置数据连接，把 activiti.cfg.xml 文件放在 classes 下，并加入驱动包。

数据库

Activiti 支持的数据库

Activiti database type	Versions tested	Notes
h2	1.2.132	Default configured database
mysql	5.1.11	
oracle	10.2.0	
postgres	8.4	
db2	DB2 9.7 using db2jcc4	**[EXPERIMENTAL]**
mssql	2008 using JDBC jtds-1.2.4	**[EXPERIMENTAL]**

名词解释

关键对象

Deployment：流程部署对象，部署一个流程是创建。
ProcessDefinitions：流程定义，部署成功后自动创建。
ProcessInstances：流程实例，启动流程是创建。
Task：任务，在 Activiti 中的 Task 仅指有角色参与的任务，即定义中的 UserTask。
Execution：执行计划，流程实例和流程执行中的所有节点都是 Execution，如 UserTask、ServiceTask 等。

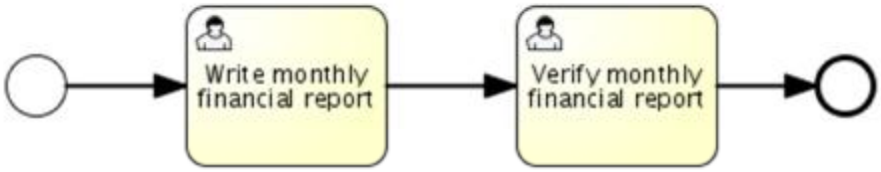
服务接口

ProcessEngine：流程引擎接口，提供流程管理和运作的所有接口。
RuntimeService：运行时服务接口，提供流程启动服务，运行中流程查询，运行变量设置和获取。
TaskService:用户任务接口（UserTask），提供运行时任务查询、领取、完成、删除及变量设置用户管理等服务。
IdentityService:用户和组管理接口。
ManagementService:流程引擎管理接口。
HistoryService:流程处理查询接口，包括执行中流程查询和历史流程查询。

Activiti 使用

流程定义

- 流程图如下：



- 流程定义如下：

```
<definitions id="definitions"
   targetNamespace="http://activiti.org/bpmn20"
   xmlns:activiti="http://activiti.org/bpmn"
   xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">
      <process id="financialReport" name="Monthly financial report reminder process">
         <startEvent id="theStart" />
         <sequenceFlow id='flow1' sourceRef='theStart' targetRef='writeReportTask' />
         <userTask id="writeReportTask" name="Write monthly financial report" >
            <documentation>
               Write monthly financial report for publication to shareholders.
            </documentation>
            <potentialOwner>
               <resourceAssignmentExpression>
                  <formalExpression>accountancy</formalExpression>
               </resourceAssignmentExpression>
            </potentialOwner>
         </userTask>
         <sequenceFlow id='flow2' sourceRef='writeReportTask' targetRef='verifyReportTask' />
         <userTask id="verifyReportTask" name="Verify monthly financial report" >
            <documentation>
               Verify monthly financial report composed by the accountancy department.
               This financial report is going to be sent to all the company shareholders.
            </documentation>
            <potentialOwner>
               <resourceAssignmentExpression>
                  <formalExpression>management</formalExpression>
               </resourceAssignmentExpression>
            </potentialOwner>
         </userTask>
         <sequenceFlow id='flow3' sourceRef='verifyReportTask' targetRef='theEnd' />
         <endEvent id="theEnd" />
```

```
    </process>
</definitions>
```

配置文件

默认文件名称：activiti.cfg.xml，放在 classpath 下。
内容如下：

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

  <bean id="processEngineConfiguration"
class="org.activiti.engine.impl.cfg.StandaloneProcessEngineConfiguration">

    <property name="jdbcUrl" value="jdbc:h2:mem:activiti;DB_CLOSE_DELAY=1000" />
    <property name="jdbcDriver" value="org.h2.Driver" />
    <property name="jdbcUsername" value="sa" />
    <property name="jdbcPassword" value="" />

    <property name="databaseSchemaUpdate" value="true" />

    <property name="jobExecutorActivate" value="false" />

    <property name="mailServerHost" value="mail.my-corp.com" />
    <property name="mailServerPort" value="5025" />
  </bean>

</beans>
```

结合 Spring

- Spring 中定义数据连接及事务管理

```
  <bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="url"
  value="jdbc:mysql://localhost:3306/boss?autoReconnect=true&amp;characterEncoding=UTF-8&amp;characterSe
  tResults=UTF-8" />
    <property name="username" value="root" />
    <property name="password" value="" />
  </bean>

  <bean id="transactionManager"
    class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource" />
```

```
</bean>
```

- 定义 Activiti 配置

```
<bean id="processEngineConfiguration" class="org.activiti.spring.SpringProcessEngineConfiguration">
    <property name="dataSource" ref="dataSource" />
    <property name="transactionManager" ref="transactionManager" />
    <property name="databaseSchemaUpdate" value="true" />
    <property name="jobExecutorActivate" value="false" />
    <property name="mailServerHost" value="mail.xxxx.com" />
    <property name="mailServerPort" value="25" />
    <property name="mailServerDefaultFrom" value="hrj@xxxx.com" />
    <property name="mailServerUsername" value="xxxx" />
    <property name="mailServerPassword" value="xxxx" />
</bean>
```

- 定义流程引擎

```
<bean id="processEngine" class="org.activiti.spring.ProcessEngineFactoryBean">
    <property name="processEngineConfiguration" ref="processEngineConfiguration" />
</bean>
```

- 定义流程中使用的对象

```
<bean id="myServiceTask" class="hrj.activiti.MyServiceTask">
    <property name="processEngine" ref="processEngine" />
</bean>

<bean id="myActivityBehavior" class="hrj.activiti.MyActivityBehavior">
    <property name="processEngine" ref="processEngine" />
</bean>

<bean id="myExecutionListener" class="hrj.activiti.MyExecutionListener">
</bean>

<bean id="valueBean" class="hrj.activiti.ValueBean">
    <property name="value" value="张三李四" />
</bean>
```

创建流程引擎

- 根据默认的配置文件创建默认的流程引擎。

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

- 也可以通过代码由 ProcessEngineConfiguration 创建一个流程引擎，这种方式不需要配置文件，可以在 ProcessEngineConfiguration 中设置配置文件中有的所有参数。

```
ProcessEngine processEngine = ProcessEngineConfiguration
    .createProcessEngineConfigurationFromResourceDefault()
    .setMailServerHost("gmail.com")
    .setJdbcUsername("mickey")
    .setJdbcPassword("mouse")
    .buildProcessEngine();
```

- 由 Spring 创建流程引擎。见结合 Spring

部署流程

```
RepositoryService repositoryService = processEngine.getRepositoryService();
Deployment deployment = repositoryService.createDeployment()
   .addClasspathResource("FinancialReportProcess.bpmn20.xml")
   .deploy();
```

还可以通过字符串，zip 包，inputStream 等方式部署流程。

启动流程

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

启动流程使用流程定义中的 process id="financialReport"，可以绑定一个 String 类型的 businessKey 和 Map 类型的流程变量集合

查询任务

流程定义中，第一个任务是指定 accountancy 角色处理的。

```
        <potentialOwner>
          <resourceAssignmentExpression>
            <formalExpression>accountancy</formalExpression>
          </resourceAssignmentExpression>
        </potentialOwner>
```

取得任务接口

```
TaskService taskService = processEngine.getTaskService();
```

查询流转到 accountancy 的任务

```
List<Task> tasks = taskService.createTaskQuery().taskCandidateGroup("accountancy").list();
```

领取任务

如果 fozzie 是 accountancy 下的一个用户

```
for (Task task : tasks) {
    // fozzie 领取任务
    taskService.claim(task.getId(), "fozzie");
}
```

完成任务

查询用户 fozzie 可处理的任务

```
tasks = taskService.createTaskQuery().taskAssignee("fozzie").list();
for (Task task : tasks) {
```

```
    //完成任务
    taskService.complete(task.getId());
}
```

查询

流程部署查询

```
DeploymentQuery deploymentQuery = repositoryService.createDeploymentQuery();
List<Deployment> deploymentList = deploymentQuery.list();
```

流程定义查询

```
ProcessDefinitionQuery processDefinitionQuery = repositoryService.createProcessDefinitionQuery();
List<ProcessDefinition> processDefinitionList = processDefinitionQuery.orderByProcessDefinitionId().asc().list();
```

处理中的流程实例查询

```
RuntimeService runtimeService = processEngine.getRuntimeService();
List<ProcessInstance> processInstanceList = runtimeService.createProcessInstanceQuery().list();
```

处理完成的流程实例查询

```
HistoricProcessInstanceQuery historicProcessInstanceQuery = processEngine.getHistoryService()
.createHistoricProcessInstanceQuery();
List<HistoricProcessInstance> historicProcessInstanceList =
    historicProcessInstanceQuery.finished().orderByProcessInstanceStartTime().asc().list();
```

流程处理记录查询

```
l   仅得到流程中的 UserTask 节点
HistoricTaskInstanceQuery historicTaskInstanceQuery = processEngine.getHistoryService()
        .createHistoricTaskInstanceQuery();
List<HistoricTaskInstance> historicTaskInstanceList = historicTaskInstanceQuery.processInstanceId(processInstanceId)
        .orderByHistoricActivityInstanceStartTime().asc().list();
l   查询流程中所有节点
HistoricActivityInstanceQuery
    historicActivityInstanceQuery=processEngine.getHistoryService().createHistoricActivityInstanceQuery();
List<HistoricActivityInstance> historicActivityInstanceList =
    historicActivityInstanceQuery.processInstanceId(processInstanceId).orderByHistoricActivityInstanceStartTime().
    asc().list();
```

常用的节点

Start events

流程的开始，必须。

Timer start event

用于定时启动的流程，可定时启动一次，或按时重复启动流程。

定时启动一次的流程：

```
<startEvent id="theStart">
      <timerEventDefinition>
            <timeDate>2011-03-11T12:13:14</timeDate>
      </timerEventDefinition>
</startEvent>
```

重复启动的流程：

```
<startEvent id="theStart">
      <timerEventDefinition>
            <timeCycle>R4/2011-03-11T12:13:00/PT5M</timeCycle>
      </timerEventDefinition>
</startEvent>
```

时间格式定义参照 ISO8601。
R4:重复 4 次。
2011-03-11T12:13:00：启动的时间。
P：重复。
T5M：每 5 分钟，T 表示时间，5M 是 5 分钟，05S 是 5 秒。

Sequence flow

顺序流

描述当前的节点（开始事件，任务，子流程、结束事件等）完成后流转到哪里。

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

有条件的顺序流

当前节点在满足定义条件后的流转方向。

```
<sequenceFlow id="flow" sourceRef="theStart" targetRef="theTask">
   <conditionExpression xsi:type="tFormalExpression">
      <![CDATA[${order.price > 100 && order.price < 250}]]>
   </conditionExpression>
</sequenceFlow>
```

默认的顺序流

用于关口（Gateway）之后，和条件顺序流同时存在，不满足所有条件的时候流转去哪里。

```
<exclusiveGateway id="exclusiveGw" name="Exclusive Gateway" default="flow2" />
<sequenceFlow id="flow1" sourceRef="exclusiveGw" targetRef="task1">
   <conditionExpression xsi:type="tFormalExpression">${conditionA}</conditionExpression>
</sequenceFlow>
<sequenceFlow id="flow2" sourceRef="exclusiveGw" targetRef="task2"/>
<sequenceFlow id="flow3" sourceRef="exclusiveGw" targetRef="task3">
   <conditionExpression xsi:type="tFormalExpression">${conditionB}</conditionExpression>
</sequenceFlow>
```

Gateways

Exclusive gateway

互斥关口，流程经过关口后只会走一个顺序流，即使关口后的顺序流都是无条件的。

```
<exclusiveGateway id="exclusiveGw" name="Exclusive Gateway" />
<sequenceFlow id="flow2" sourceRef="exclusiveGw" targetRef="theTask1">
   <conditionExpression xsi:type="tFormalExpression">${input == 1}</conditionExpression>
</sequenceFlow>
<sequenceFlow id="flow3" sourceRef="exclusiveGw" targetRef="theTask2">
   <conditionExpression xsi:type="tFormalExpression">${input == 2}</conditionExpression>
</sequenceFlow>
<sequenceFlow id="flow4" sourceRef="exclusiveGw" targetRef="theTask3">
   <conditionExpression xsi:type="tFormalExpression">${input == 3}</conditionExpression>
</sequenceFlow>
```

Parallel Gateway

并行关口，用在开头，流程经过关口后会同时经过所有顺序流，用在结尾，所有流程完成后会一起通过并行关口。

```
    <startEvent id="theStart" />
    <sequenceFlow id="flow1" sourceRef="theStart" targetRef="fork" />
    <parallelGateway id="fork" />
    <sequenceFlow sourceRef="fork" targetRef="receivePayment" />
    <sequenceFlow sourceRef="fork" targetRef="shipOrder" />
    <userTask id="receivePayment" name="Receive Payment" />
    <sequenceFlow sourceRef="receivePayment" targetRef="join" />
```

```
    <userTask id="shipOrder" name="Ship Order" />
    <sequenceFlow sourceRef="shipOrder" targetRef="join" />
    <parallelGateway id="join" />
    <sequenceFlow sourceRef="join" targetRef="archiveOrder" />
    <userTask id="archiveOrder" name="Archive Order" />
    <sequenceFlow sourceRef="archiveOrder" targetRef="theEnd" />
    <endEvent id="theEnd" />
```

User task

人工任务，必须要有人或人的组参与。

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

Script Task

脚本任务，可以执行一段脚本，Javascript，grove 都可以使用，可以在脚本中定义或修改变量，来控制流程的流转。

```
<scriptTask id="theScriptTask" name="Execute script" scriptFormat="groovy">
  <script>
    sum = 0
    for ( i in inputArray ) {
      sum += i
   }
    def scriptVar = "test123"  //局部变量，当前脚本可用。
    execution.setVariable("myVar", scriptVar)  //设置变量，整个流程可用。
  </script>
</scriptTask>
```

Java Service Task

流程示例

- 下面的流程定义，使用了 UserTask，简单模拟了下工单处理流程。

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<definitions id="definitions" targetNamespace="http://activiti.org/bpmn20"
          xmlns:activiti="http://activiti.org/bpmn" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">

          <process id="WorkOrderProcess" name="工单流程">

                    <startEvent id="theStart" name="开始事件" />

                    <sequenceFlow id='start_booking' sourceRef='theStart'
                              targetRef='booking' />

                    <userTask id="booking" name="上门预约">
                              <documentation>
                                        工单施工前安装时间预约，预约成功开始上门施工，预约失败重新预约，或直接
取消工单。
                              </documentation>
                              <humanPerformer>
                                        <resourceAssignmentExpression>
                                                  <formalExpression>A</formalExpression>
                                        </resourceAssignmentExpression>
                              </humanPerformer>
                    </userTask>

                    <sequenceFlow id='booking_booking' sourceRef='booking'
                              targetRef='booking' name="预约失败重新预约">
                              <conditionExpression xsi:type="tFormalExpression">${result != "true"}
                              </conditionExpression>
                    </sequenceFlow>

                    <sequenceFlow id='booking_install' sourceRef='booking'
                              targetRef='install' name="预约完成上门施工">
                              <conditionExpression xsi:type="tFormalExpression">${result == "true"}
                              </conditionExpression>
                    </sequenceFlow>

                    <userTask id="install" name="工单施工">
                              <documentation>
                                        工单上门施工
                              </documentation>
                              <humanPerformer>
                                        <resourceAssignmentExpression>
                                                  <formalExpression>C</formalExpression>
                                        </resourceAssignmentExpression>
                              </humanPerformer>
                              <potentialOwner>
                                        <resourceAssignmentExpression>
                                                  <formalExpression>user(B)</formalExpression>
                                        </resourceAssignmentExpression>
                              </potentialOwner>
```

```
                </userTask>

                <sequenceFlow id='install_installConfirm' sourceRef='install'
                        targetRef='installConfirm' />

                <userTask id="installConfirm" name="施工确认">
                        <documentation>
                                施工完成确认施工结果，施工成功进入流程终点，施工失败重新预约施工，或结束施工。
                        </documentation>
                        <potentialOwner>
                                <resourceAssignmentExpression>
                                        <formalExpression>user(B),group(g)</formalExpression>
                                </resourceAssignmentExpression>
                        </potentialOwner>
                </userTask>

                <sequenceFlow id='installConfirm_end' sourceRef='installConfirm'
                        targetRef='theEnd' name="施工成功完成流程、结束施工">
                        <conditionExpression xsi:type="tFormalExpression">${result == "true"}
                        </conditionExpression>
                </sequenceFlow>

                <sequenceFlow id='installConfirm_booking' sourceRef='installConfirm'
                        targetRef='booking' name="施工失败重新预约施工">
                        <conditionExpression xsi:type="tFormalExpression">${result != "true"}
                        </conditionExpression>
                </sequenceFlow>

                <endEvent id="theEnd" name="结束事件" />

        </process>

</definitions>
```

- 下面的流程定义，包含了 UserTask、ServiceTask、MailService、scriptTask，ExecutionListener、TaskListener、TaskListener 的使用。
  需要结合 Spring 使用，有些表达式引用了 Spring 的定义。

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions id="definitions" targetNamespace="http://activiti.org/bpmn20"
        xmlns:activiti="http://activiti.org/bpmn" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">

        <process id="ServiceTaskTest" name="自定义任务测试">

<!--                配置的监听器，-->
<!--                可以用于 process 流程实例启动、结束-->
```

```
<!--                可以用于 sequenceFlow 流转发生-->
                    <extensionElements>
                              <activiti:executionListener delegateExpression="${myExecutionListener}"
event="start" />
                              <activiti:executionListener class="hrj.activiti.MyExecutionListener"
event="end"/>
                    </extensionElements>

                    <startEvent id="theStart" name="开始事件" />

                    <sequenceFlow id='start_serviceTaskSetVal' sourceRef='theStart'
                              targetRef='serviceTaskSetVal' />

                    <!--直接取 ValueBean 的属性值赋给 result -->
                    <serviceTask id="serviceTaskSetVal" name="serviceTask 设置变量值。"
                              activiti:resultVariable="result" activiti:expression="#{valueBean.value}" />

                    <!-- 配置的监听器，流转时触发，触发时调用一个自定义方法 -->
                    <sequenceFlow id='serviceTaskSetVal_scriptTask'
                              sourceRef='serviceTaskSetVal' targetRef='scriptTask' >
                              <extensionElements>
                                        <activiti:executionListener
expression="${myExecutionListener.myMethod(execution)}"/>
                              </extensionElements>
                    </sequenceFlow>

                    <!-- 执行一段脚本，直接修改变量值或流程行为 -->
                    <scriptTask id="scriptTask" name="执行脚本" scriptFormat="groovy">
                              <script>
                                        def a=""
                                        for ( i in 0..10 ) {
                                        a += result + "__"
                                        }
                                        execution.setVariable("scriptTask", a)
                                        println a
                              </script>
                    </scriptTask>

                    <sequenceFlow id='scriptTask_serviceTaskExecute'
                              sourceRef='scriptTask' targetRef='serviceTaskExecute' />

                    <!--使用 activiti:delegateExpression="${myServiceTask}"方式可以调用 execute 方法，但不能注入
值 -->
                    <!-- <serviceTask id="serviceTaskExecute" -->
                    <!-- activiti:delegateExpression="${myServiceTask}" name="自定义 ServiceTask，调用默认的
Execute 方法"> -->

                    <!--使用 activiti:class 方法可以注入值，但 string 和 expression 好像一样的，注入的字段都要是
expression 的，不能使用 String。 -->
```

```
                <serviceTask id="serviceTaskExecute" name="自定义 ServiceTask，调用默认的 Execute 方法"
                        activiti:class="hrj.activiti.MyServiceTask">
                        <extensionElements>
                                <activiti:field name="text">
                                        <activiti:string>Hello World</activiti:string>
                                </activiti:field>
                                <activiti:field name="myVar">
                                        <activiti:expression>Hello
${scriptTask}</activiti:expression>
                                </activiti:field>
                        </extensionElements>
                </serviceTask>

                <sequenceFlow id='serviceTaskExecute_sendMail' sourceRef='serviceTaskExecute'
                        targetRef='sendMail' />

                <!-- 发送 Email 的 ServiceTask typeEmail 或 SendMail 任务定义都不处理中文，所以自己重写了下
MailActivityBehavior，使支持中文。 -->
                <!-- 另注意 5.3 版本之前的 MailActivityBehavior 都不能流转 -->
                <serviceTask id="sendMail" name="发送 Email"
                        activiti:class="hrj.activiti.MyMailActivityBehavior">
                        <extensionElements>
                                <activiti:field name="from" stringValue="hrj@xxxx.com" />
                                <activiti:field name="to" expression="hrj@xxxx.com" />
                                <activiti:field name="charset" expression="UTF-8" />
                                <activiti:field name="subject" expression="${result}该吃" />
                                <activiti:field name="html">
                                        <activiti:expression>
                                <![CDATA[
                                  <html>
                                    <body>
                                                Hello ${scriptTask},<br/><br/>
                                                你该吃饭了.
                                    </body>
                                  </html>
                                ]]>
                                        </activiti:expression>
                                </activiti:field>
                        </extensionElements>
                </serviceTask>

                <sequenceFlow id='sendMail_serviceTaskPrintMessage'
                        sourceRef='sendMail' targetRef='serviceTaskPrintMessage' />

                <!-- 调用一个自定义的方法， -->
                <serviceTask id="serviceTaskPrintMessage" name="自定义 ServiceTask，打印变量值"
                        activiti:expression="#{myServiceTask.printMessage(execution)}" />

                <sequenceFlow id='serviceTaskPrintMessage_userTaskSetResult'
```

```
                        sourceRef='serviceTaskPrintMessage' targetRef='userTaskSetResult' />

                <!-- 配置一个 TaskListener,可以用于 userTask 活动的创建、分配、完成,使用 activiti:taskListener
定义。-->
                <userTask id="userTaskSetResult" name="userTask，由前端设置变量">
                        <documentation></documentation>
                        <extensionElements>
                                <activiti:taskListener event="complete"
class="hrj.activiti.MyTaskListener" />
                        </extensionElements>
                        <humanPerformer>
                                <resourceAssignmentExpression>
                                        <formalExpression>A</formalExpression>
                                </resourceAssignmentExpression>
                        </humanPerformer>
                </userTask>

                <sequenceFlow id='userTaskSetResult_myActivityBehavior'
                        sourceRef='userTaskSetResult' targetRef='myActivityBehavior' />

                <!-- 自定义的流程控制器，自己实现方法决定流程走向。  -->
                <serviceTask id="myActivityBehavior" name="自定义的 ActivityBehavior，根据上下文条件决定流
程走向。"
                        activiti:delegateExpression="${myActivityBehavior}">
                </serviceTask>

                <sequenceFlow id="myActivityBehavior_end" sourceRef="myActivityBehavior"
                        targetRef="theEnd" />
                <sequenceFlow id="myActivityBehavior_serviceTaskPrintMessage"
                        sourceRef="myActivityBehavior" targetRef="serviceTaskPrintMessage" />

                <endEvent id="theEnd" name="结束事件" />

        </process>

</definitions>
```