**Activiti 5.13 用户手册**

**Table of Contents**

[1. 简介](http://www.mossle.com/docs/activiti/#N10007)

[协议](http://www.mossle.com/docs/activiti/#license)

[下载](http://www.mossle.com/docs/activiti/#download)

[源码](http://www.mossle.com/docs/activiti/#sources)

[必要的软件](http://www.mossle.com/docs/activiti/#required.software)

[JDK 6+](http://www.mossle.com/docs/activiti/#N1002A)

[Eclipse Indigo 和 Juno](http://www.mossle.com/docs/activiti/#N10036)

[报告问题](http://www.mossle.com/docs/activiti/#reporting.problems)

[试验性功能](http://www.mossle.com/docs/activiti/#experimental)

[内部实现类](http://www.mossle.com/docs/activiti/#internal)

[2. 开始学习](http://www.mossle.com/docs/activiti/#N1007D)

[一分钟入门](http://www.mossle.com/docs/activiti/#demo.setup.one.minute.version)

[安装Activiti](http://www.mossle.com/docs/activiti/#activiti.setup)

[安装Activiti数据库](http://www.mossle.com/docs/activiti/#activiti.setup.database)

[引入Activiti jar和依赖](http://www.mossle.com/docs/activiti/#getting.started.including.libs)

[下一步](http://www.mossle.com/docs/activiti/#getting.started.next.steps)

[3. 配置](http://www.mossle.com/docs/activiti/#N10156)

[创建ProcessEngine](http://www.mossle.com/docs/activiti/#configuration)

[ProcessEngineConfiguration bean](http://www.mossle.com/docs/activiti/#configurationRoot)

[数据库配置](http://www.mossle.com/docs/activiti/#databaseConfiguration)

[支持的数据库](http://www.mossle.com/docs/activiti/#supporteddatabases)

[创建数据库表](http://www.mossle.com/docs/activiti/#creatingDatabaseTable)

[理解数据库表的命名](http://www.mossle.com/docs/activiti/#database.tables.explained)

[数据库升级](http://www.mossle.com/docs/activiti/#databaseUpgrade)

[启用Job执行器](http://www.mossle.com/docs/activiti/#jobExecutorConfiguration)

[配置邮件服务器](http://www.mossle.com/docs/activiti/#mailServerConfiguration)

[配置历史](http://www.mossle.com/docs/activiti/#historyConfiguration)

[为表达式和脚本暴露配置](http://www.mossle.com/docs/activiti/#exposingConfigurationBeans)

[配置部署缓存](http://www.mossle.com/docs/activiti/#processDefinitionCacheConfiguration)

[日志](http://www.mossle.com/docs/activiti/#loggingConfiguration)

[4. Activiti API](http://www.mossle.com/docs/activiti/#chapterApi)

[流程引擎的API和服务](http://www.mossle.com/docs/activiti/#apiEngine)

[异常策略](http://www.mossle.com/docs/activiti/#N10486)

[使用Activiti的服务](http://www.mossle.com/docs/activiti/#api.services)

[发布流程](http://www.mossle.com/docs/activiti/#api.services.deployment)

[启动一个流程实例](http://www.mossle.com/docs/activiti/#api.services.start.processinstance)

[完成任务](http://www.mossle.com/docs/activiti/#api.services.tasls)

[挂起，激活一个流程](http://www.mossle.com/docs/activiti/#api.services.suspend)

[更多知识](http://www.mossle.com/docs/activiti/#api.services)

[查询API](http://www.mossle.com/docs/activiti/#queryAPI)

[表达式](http://www.mossle.com/docs/activiti/#apiExpressions)

[单元测试](http://www.mossle.com/docs/activiti/#apiUnitTesting)

[调试单元测试](http://www.mossle.com/docs/activiti/#apiDebuggingUnitTest)

[web应用中的流程引擎](http://www.mossle.com/docs/activiti/#apiProcessEngineInWebApp)

[5. Spring集成](http://www.mossle.com/docs/activiti/#springintegration)

[ProcessEngineFactoryBean](http://www.mossle.com/docs/activiti/#N10680)

[事务](http://www.mossle.com/docs/activiti/#N10698)

[表达式](http://www.mossle.com/docs/activiti/#springExpressions)

[资源的自动部署](http://www.mossle.com/docs/activiti/#N106F6)

[单元测试](http://www.mossle.com/docs/activiti/#springUnitTest)

[6. 部署](http://www.mossle.com/docs/activiti/#chDeployment)

[业务文档](http://www.mossle.com/docs/activiti/#N1071A)

[编程式部署](http://www.mossle.com/docs/activiti/#N1072E)

[通过Activiti Explorer控制台部署](http://www.mossle.com/docs/activiti/#deploymentWithExplorer)

[外部资源](http://www.mossle.com/docs/activiti/#N10752)

[Java类](http://www.mossle.com/docs/activiti/#N10757)

[在流程中使用Spring beans](http://www.mossle.com/docs/activiti/#N10763)

[创建独立应用](http://www.mossle.com/docs/activiti/#N10775)

[流程定义的版本](http://www.mossle.com/docs/activiti/#versioningOfProcessDefinitions)

[提供流程图片](http://www.mossle.com/docs/activiti/#providingProcessDiagram)

[自动生成流程图片](http://www.mossle.com/docs/activiti/#generatingProcessDiagram)

[类别](http://www.mossle.com/docs/activiti/#deploymentCategory)

[7. BPMN 2.0介绍](http://www.mossle.com/docs/activiti/#bpmn20)

[啥是BPMN？](http://www.mossle.com/docs/activiti/#whatIsBpmn)

[定义一个流程](http://www.mossle.com/docs/activiti/#bpmnDefiningProcess)

[快速起步：10分钟教程](http://www.mossle.com/docs/activiti/#10minutetutorial)

[前提](http://www.mossle.com/docs/activiti/#bpmn10MinuteTutorialPrerequisites)

[目标](http://www.mossle.com/docs/activiti/#bpmn10MinuteTutorialGoal)

[用例](http://www.mossle.com/docs/activiti/#bpmnFirstExampleUseCase)

[流程图](http://www.mossle.com/docs/activiti/#bpmnFirstExampleDiagram)

[XML内容](http://www.mossle.com/docs/activiti/#bpmnFirstExampleXml)

[启动一个流程实例](http://www.mossle.com/docs/activiti/#bpmnFirstExamplStartProcess)

[任务列表](http://www.mossle.com/docs/activiti/#bpmnFirstExampleCandidateList)

[领取任务](http://www.mossle.com/docs/activiti/#bpmnFirstExampleClaimTask)

[完成任务](http://www.mossle.com/docs/activiti/#bpmnFirstExampleCompleteTask)

[结束流程](http://www.mossle.com/docs/activiti/#bpmnFirstExampleEndingProcess)

[代码总结](http://www.mossle.com/docs/activiti/#bpmnFirstExampleCode)

[更多思考](http://www.mossle.com/docs/activiti/#bpmnFirstExampleFutureEnhancements)

[8. BPMN 2.0结构](http://www.mossle.com/docs/activiti/#bpmnConstructs)

[自定义扩展](http://www.mossle.com/docs/activiti/#bpmnCustomExtensions)

[事件（Event）](http://www.mossle.com/docs/activiti/#bpmnEvents)

[事件定义](http://www.mossle.com/docs/activiti/#eventDefinitions)

[定时器事件定义](http://www.mossle.com/docs/activiti/#timerEventDefinitions)

[错误事件定义](http://www.mossle.com/docs/activiti/#bpmnErrorEventDefinition)

[信号事件定义](http://www.mossle.com/docs/activiti/#bpmnSignalEventDefinition)

[消息事件定义](http://www.mossle.com/docs/activiti/#bpmnMessageEventDefinition)

[开始事件](http://www.mossle.com/docs/activiti/#bpmnStartEvents)

[空开始事件](http://www.mossle.com/docs/activiti/#bpmnNoneStartEvent)

[定时开始事件](http://www.mossle.com/docs/activiti/#bpmnTimerStartEvent)

[消息开始事件](http://www.mossle.com/docs/activiti/#bpmnMessageStartEvent)

[错误开始事件](http://www.mossle.com/docs/activiti/#bpmnErrorStartEvent)

[结束事件](http://www.mossle.com/docs/activiti/#bpmnEndEvent)

[空结束事件](http://www.mossle.com/docs/activiti/#bpmnNoneEndEvent)

[错误结束事件](http://www.mossle.com/docs/activiti/#bpmnErrorEndEvent)

[取消结束事件](http://www.mossle.com/docs/activiti/#bpmnCancelEndEvent)

[边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryEvent)

[定时边界事件](http://www.mossle.com/docs/activiti/#bpmnTimerBoundaryEvent)

[错误边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryErrorEvent)

[信号边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundarySignalEvent)

[消息边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryMessageEvent)

[取消边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryCancelEvent)

[补偿边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryCompensationEvent)

[中间捕获事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateCatchingEvent)

[定时中间捕获事件](http://www.mossle.com/docs/activiti/#bpmnTimerIntermediateEvent)

[信号中间捕获事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateSignalEvent)

[消息中间捕获事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateMessageEvent)

[内部触发事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateThrowEvent)

[中间触发空事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateThrowNoneEvent)

[信号中间触发事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateThrowSignalEvent)

[补偿中间触发事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateThrowCompensationEvent)

[顺序流](http://www.mossle.com/docs/activiti/#bpmnSequenceFlow)

[描述](http://www.mossle.com/docs/activiti/#sequenceFlowDescription)

[图形标记](http://www.mossle.com/docs/activiti/#sequenceFlowGraphicalNotation)

[XML内容](http://www.mossle.com/docs/activiti/#sequenceFlowXml)

[条件顺序流](http://www.mossle.com/docs/activiti/#bpmnConditionalSequenceFlow)

[默认顺序流](http://www.mossle.com/docs/activiti/#bpmnDefaultSequenceFlow)

[网关](http://www.mossle.com/docs/activiti/#bpmnGateways)

[排他网关](http://www.mossle.com/docs/activiti/#bpmnExclusiveGateway)

[并行网关](http://www.mossle.com/docs/activiti/#bpmnParallelGateway)

[包含网关](http://www.mossle.com/docs/activiti/#bpmnInclusiveGateway)

[基于事件网关](http://www.mossle.com/docs/activiti/#bpmnEventbasedGateway)

[任务](http://www.mossle.com/docs/activiti/#bpmnTask)

[用户任务](http://www.mossle.com/docs/activiti/#bpmnUserTask)

[脚本任务](http://www.mossle.com/docs/activiti/#bpmnScriptTask)

[Java服务任务](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTask)

[Web Service Task](http://www.mossle.com/docs/activiti/#bpmnWebserviceTask)

[Business Rule Task](http://www.mossle.com/docs/activiti/#bpmnBusinessRuleTask)

[Email Task](http://www.mossle.com/docs/activiti/#bpmnEmailTask)

[Mule Task](http://www.mossle.com/docs/activiti/#bpmnMuleTask)

[Camel Task](http://www.mossle.com/docs/activiti/#bpmnCamelTask)

[Manual Task](http://www.mossle.com/docs/activiti/#bpmnManualTask)

[Java Receive Task](http://www.mossle.com/docs/activiti/#bpmnReceiveTask)

[Shell Task](http://www.mossle.com/docs/activiti/#bpmnShellTask)

[Execution listener](http://www.mossle.com/docs/activiti/#executionListeners)

[Task listener](http://www.mossle.com/docs/activiti/#taskListeners)

[Multi-instance (for each)](http://www.mossle.com/docs/activiti/#bpmnMultiInstance)

[Compensation Handlers](http://www.mossle.com/docs/activiti/#bpmnCompensationHandlers)

[Sub-Processes and Call Activities](http://www.mossle.com/docs/activiti/#bpmnSubprocessAndCallActivity)

[Sub-Process](http://www.mossle.com/docs/activiti/#bpmnSubProcess)

[Event Sub-Process](http://www.mossle.com/docs/activiti/#bpmnEventSubprocess)

[Transaction subprocess](http://www.mossle.com/docs/activiti/#bpmnTransactionSubprocess)

[Call activity (subprocess)](http://www.mossle.com/docs/activiti/#bpmnCallActivity)

[Transactions and Concurrency](http://www.mossle.com/docs/activiti/#bpmnConcurrencyAndTransactions)

[Asynchronous Continuations](http://www.mossle.com/docs/activiti/#asyncContinuations)

[Exclusive Jobs](http://www.mossle.com/docs/activiti/#exclusiveJobs)

[Process Initiation Authorization](http://www.mossle.com/docs/activiti/#security)

[9. 表单](http://www.mossle.com/docs/activiti/#forms)

[表单属性](http://www.mossle.com/docs/activiti/#formProperties)

[外置表单的渲染](http://www.mossle.com/docs/activiti/#externalFormRendering)

[10. JPA](http://www.mossle.com/docs/activiti/#N12115)

[要求](http://www.mossle.com/docs/activiti/#N12128)

[配置](http://www.mossle.com/docs/activiti/#jpaconfiguration)

[用法](http://www.mossle.com/docs/activiti/#N121B8)

[简单例子](http://www.mossle.com/docs/activiti/#N121BB)

[查询JPA流程变量](http://www.mossle.com/docs/activiti/#N12202)

[使用Spring beans和JPA结合的高级例子](http://www.mossle.com/docs/activiti/#N1222F)

[11. 历史](http://www.mossle.com/docs/activiti/#history)

[查询历史](http://www.mossle.com/docs/activiti/#historyQuery)

[HistoricProcessInstanceQuery](http://www.mossle.com/docs/activiti/#historyQueryProcessInstance)

[HistoricVariableInstanceQuery](http://www.mossle.com/docs/activiti/#historyQueryVariableInstance)

[HistoricActivityInstanceQuery](http://www.mossle.com/docs/activiti/#historyQueryActivityInstance)

[HistoricDetailQuery](http://www.mossle.com/docs/activiti/#historyQueryDetail)

[HistoricTaskInstanceQuery](http://www.mossle.com/docs/activiti/#historyQueryTaskInstance)

[历史配置](http://www.mossle.com/docs/activiti/#historyConfig)

[审计目的的历史](http://www.mossle.com/docs/activiti/#historyFormAuditPurposes)

[12. Eclipse Designer](http://www.mossle.com/docs/activiti/#activitiDesigner)

[Installation](http://www.mossle.com/docs/activiti/#eclipseDesignerInstallation)

[Activiti Designer editor features](http://www.mossle.com/docs/activiti/#eclipseDesignerEditorFeatures)

[Activiti Designer BPMN features](http://www.mossle.com/docs/activiti/#eclipseDesignerBPMNFeatures)

[Activiti Designer deployment features](http://www.mossle.com/docs/activiti/#eclipseDesignerDeployment)

[Extending Activiti Designer](http://www.mossle.com/docs/activiti/#eclipseDesignerExtending)

[Customizing the palette](http://www.mossle.com/docs/activiti/#eclipseDesignerCustomizingPalette)

[Validating diagrams and exporting to custom output formats](http://www.mossle.com/docs/activiti/#N127EE)

[13. Activiti Explorer](http://www.mossle.com/docs/activiti/#activitiExplorer)

[流程图](http://www.mossle.com/docs/activiti/#N1291C)

[任务](http://www.mossle.com/docs/activiti/#N12938)

[启动流程实例](http://www.mossle.com/docs/activiti/#N12968)

[我的流程实例](http://www.mossle.com/docs/activiti/#N1297B)

[管理](http://www.mossle.com/docs/activiti/#N1298A)

[报表](http://www.mossle.com/docs/activiti/#explorer.reporting)

[报告数据JSON](http://www.mossle.com/docs/activiti/#explorer.reporting.json)

[实例流程](http://www.mossle.com/docs/activiti/#explorer.reporting.example)

[报告开始表单](http://www.mossle.com/docs/activiti/#explorer.reporting.start.form)

[流程例子](http://www.mossle.com/docs/activiti/#explorer.reporting.examples)

[修改数据库](http://www.mossle.com/docs/activiti/#explorer.change.database)

[14. Activiti Modeler](http://www.mossle.com/docs/activiti/#activitiModeler)

[编辑模型](http://www.mossle.com/docs/activiti/#N12AEF)

[导入模型](http://www.mossle.com/docs/activiti/#N12AFA)

[把发布的流程定义转换成可编辑的模型](http://www.mossle.com/docs/activiti/#N12B05)

[把模型导出成BPMN XML](http://www.mossle.com/docs/activiti/#N12B10)

[把模型部署到Activiti引擎中](http://www.mossle.com/docs/activiti/#N12B1B)

[15. REST API](http://www.mossle.com/docs/activiti/#N12B26)

[General Activiti REST principles](http://www.mossle.com/docs/activiti/#N12B29)

[Installation and Authentication](http://www.mossle.com/docs/activiti/#N12B2C)

[Usage in Tomcat](http://www.mossle.com/docs/activiti/#restUsageInTomcat)

[Methods and return-codes](http://www.mossle.com/docs/activiti/#N12B78)

[Error response body](http://www.mossle.com/docs/activiti/#N12BFB)

[Request parameters](http://www.mossle.com/docs/activiti/#N12C02)

[Deployment](http://www.mossle.com/docs/activiti/#N12E29)

[List of Deployments](http://www.mossle.com/docs/activiti/#N12E34)

[Get a deployment](http://www.mossle.com/docs/activiti/#N12EAB)

[Create a new deployment](http://www.mossle.com/docs/activiti/#N12EF1)

[Delete a deployment](http://www.mossle.com/docs/activiti/#N12F2C)

[List resources in a deployment](http://www.mossle.com/docs/activiti/#N12F69)

[Get a deployment resource](http://www.mossle.com/docs/activiti/#N12FD3)

[Get a deployment resource content](http://www.mossle.com/docs/activiti/#N13044)

[Process Definitions](http://www.mossle.com/docs/activiti/#N13095)

[List of process definitions](http://www.mossle.com/docs/activiti/#N13098)

[Get a process definition](http://www.mossle.com/docs/activiti/#N1317D)

[Update category for a process definition](http://www.mossle.com/docs/activiti/#N131D4)

[Get a process definition resource content](http://www.mossle.com/docs/activiti/#N1320C)

[Get a process definition BPMN model](http://www.mossle.com/docs/activiti/#N1323B)

[Suspend a process definition](http://www.mossle.com/docs/activiti/#N13286)

[Activate a process definition](http://www.mossle.com/docs/activiti/#N132E9)

[Get all candidate starters for a process-definition](http://www.mossle.com/docs/activiti/#N13325)

[Add a candidate starter to a process definition](http://www.mossle.com/docs/activiti/#N1336B)

[Delete a candidate starter from a process definition](http://www.mossle.com/docs/activiti/#N133C3)

[Get a candidate starter from a process definition](http://www.mossle.com/docs/activiti/#N13421)

[Models](http://www.mossle.com/docs/activiti/#N1347F)

[Get a list of models](http://www.mossle.com/docs/activiti/#N13482)

[Get a model](http://www.mossle.com/docs/activiti/#N1354A)

[Update a model](http://www.mossle.com/docs/activiti/#N13590)

[Create a model](http://www.mossle.com/docs/activiti/#N135C6)

[Delete a model](http://www.mossle.com/docs/activiti/#N135F4)

[Get the editor source for a model](http://www.mossle.com/docs/activiti/#N13631)

[Set the editor source for a model](http://www.mossle.com/docs/activiti/#N13677)

[Get the extra editor source for a model](http://www.mossle.com/docs/activiti/#N136C6)

[Set the extra editor source for a model](http://www.mossle.com/docs/activiti/#N1370C)

[Process Instances](http://www.mossle.com/docs/activiti/#N1375B)

[Get a process instance](http://www.mossle.com/docs/activiti/#N1375E)

[Delete a process instance](http://www.mossle.com/docs/activiti/#N137A4)

[Activate or suspend a process instance](http://www.mossle.com/docs/activiti/#N137E1)

[Start a process instance](http://www.mossle.com/docs/activiti/#N1383A)

[List of process instances](http://www.mossle.com/docs/activiti/#restProcessInstancesGet)

[Query process instances](http://www.mossle.com/docs/activiti/#N13951)

[Get diagram for a process instance](http://www.mossle.com/docs/activiti/#N13992)

[Get involved people for process instance](http://www.mossle.com/docs/activiti/#N139DD)

[Add an involved user to a process instance](http://www.mossle.com/docs/activiti/#N13A26)

[Remove an involved user to from process instance](http://www.mossle.com/docs/activiti/#N13A83)

[List of variables for a process instance](http://www.mossle.com/docs/activiti/#N13ADE)

[Get a variable for a process instance](http://www.mossle.com/docs/activiti/#N13B2D)

[Create (or update) variables on a process instance](http://www.mossle.com/docs/activiti/#N13B8A)

[Update a single variable on a process instance](http://www.mossle.com/docs/activiti/#N13BF3)

[Create a new binary variable on a process-instance](http://www.mossle.com/docs/activiti/#N13C5B)

[Update an existing binary variable on a process-instance](http://www.mossle.com/docs/activiti/#N13CC6)

[Executions](http://www.mossle.com/docs/activiti/#N13D2C)

[Get an execution](http://www.mossle.com/docs/activiti/#N13D2F)

[Execute an action on an execution](http://www.mossle.com/docs/activiti/#N13D75)

[Get active activities in an execution](http://www.mossle.com/docs/activiti/#N13DEC)

[List of executions](http://www.mossle.com/docs/activiti/#restExecutionsGet)

[Query executions](http://www.mossle.com/docs/activiti/#N13ECD)

[List of variables for an execution](http://www.mossle.com/docs/activiti/#N13F14)

[Get a variable for an execution](http://www.mossle.com/docs/activiti/#N13F6C)

[Create (or update) variables on an execution](http://www.mossle.com/docs/activiti/#N13FD2)

[Update a variable on an execution](http://www.mossle.com/docs/activiti/#N1403F)

[Create a new binary variable on an execution](http://www.mossle.com/docs/activiti/#N1409E)

[Update an existing binary variable on a process-instance](http://www.mossle.com/docs/activiti/#N14111)

[Tasks](http://www.mossle.com/docs/activiti/#N14188)

[Get a task](http://www.mossle.com/docs/activiti/#N1418B)

[List of tasks](http://www.mossle.com/docs/activiti/#restTasksGet)

[Query for tasks](http://www.mossle.com/docs/activiti/#N14330)

[Update a task](http://www.mossle.com/docs/activiti/#N14373)

[Task actions](http://www.mossle.com/docs/activiti/#N143AE)

[Delete a task](http://www.mossle.com/docs/activiti/#N14413)

[Get all variables for a task](http://www.mossle.com/docs/activiti/#N1446A)

[Get a variable from a task](http://www.mossle.com/docs/activiti/#N144C3)

[Get the binary data for a variable](http://www.mossle.com/docs/activiti/#N14525)

[Create new variables on a task](http://www.mossle.com/docs/activiti/#N14595)

[Create a new binary variable on a task](http://www.mossle.com/docs/activiti/#N1460C)

[Update an existing variable on a task](http://www.mossle.com/docs/activiti/#N14682)

[Updating a binary variable on a task](http://www.mossle.com/docs/activiti/#N146FD)

[Delete a variable on a task](http://www.mossle.com/docs/activiti/#N14777)

[Delete all local variables on a task](http://www.mossle.com/docs/activiti/#N147CF)

[Get all identity links for a task](http://www.mossle.com/docs/activiti/#N1480C)

[Get all identitylinks for a task for either groups or users](http://www.mossle.com/docs/activiti/#N14852)

[Get a single identity link on a task](http://www.mossle.com/docs/activiti/#N1485C)

[Create an identity link on a task](http://www.mossle.com/docs/activiti/#N148C3)

[Delete an identity link on a task](http://www.mossle.com/docs/activiti/#N1491B)

[Create a new comment on a task](http://www.mossle.com/docs/activiti/#N14979)

[Get all comments on a task](http://www.mossle.com/docs/activiti/#N149CB)

[Get a comment on a task](http://www.mossle.com/docs/activiti/#N14A0F)

[Delete a comment on a task](http://www.mossle.com/docs/activiti/#N14A5C)

[Get all events for a task](http://www.mossle.com/docs/activiti/#N14A9F)

[Get an event on a task](http://www.mossle.com/docs/activiti/#N14AE3)

[Create a new attachment on a task, containing a link to an external resource](http://www.mossle.com/docs/activiti/#N14B30)

[Create a new attachment on a task, with an attached file](http://www.mossle.com/docs/activiti/#N14B82)

[Get all attachments on a task](http://www.mossle.com/docs/activiti/#N14BE5)

[Get an attachment on a task](http://www.mossle.com/docs/activiti/#N14C29)

[Get the content for an attachment](http://www.mossle.com/docs/activiti/#N14C8C)

[Delete an attachment on a task](http://www.mossle.com/docs/activiti/#N14CDE)

[History](http://www.mossle.com/docs/activiti/#N14D21)

[Get a historic process instance](http://www.mossle.com/docs/activiti/#N14D24)

[List of historic process instances](http://www.mossle.com/docs/activiti/#restHistoricProcessInstancesGet)

[Query for historic process instances](http://www.mossle.com/docs/activiti/#N14E14)

[Delete a historic process instance](http://www.mossle.com/docs/activiti/#N14E54)

[Get the identity links of a historic process instance](http://www.mossle.com/docs/activiti/#N14E75)

[Get the binary data for a historic process instance variable](http://www.mossle.com/docs/activiti/#N14E9F)

[Get a single historic task instance](http://www.mossle.com/docs/activiti/#N14ED2)

[Get historic task instances](http://www.mossle.com/docs/activiti/#restHistoricTaskInstancesGet)

[Query for historic task instances](http://www.mossle.com/docs/activiti/#N15049)

[Delete a historic task instance](http://www.mossle.com/docs/activiti/#N1508C)

[Get the identity links of a historic task instance](http://www.mossle.com/docs/activiti/#N150AD)

[Get the binary data for a historic task instance variable](http://www.mossle.com/docs/activiti/#N150D7)

[Get historic activity instances](http://www.mossle.com/docs/activiti/#restHistoricActivityInstancesGet)

[Query for historic activity instances](http://www.mossle.com/docs/activiti/#N151A3)

[List of historic variable instances](http://www.mossle.com/docs/activiti/#restHistoricVariableInstancesGet)

[Query for historic variable instances](http://www.mossle.com/docs/activiti/#N15251)

[Get the binary data for a historic task instance variable](http://www.mossle.com/docs/activiti/#N15291)

[Get historic detail](http://www.mossle.com/docs/activiti/#restHistoricDetailGet)

[Query for historic details](http://www.mossle.com/docs/activiti/#N1534B)

[Get the binary data for a historic detail variable](http://www.mossle.com/docs/activiti/#N15384)

[Forms](http://www.mossle.com/docs/activiti/#N153B7)

[Get form data](http://www.mossle.com/docs/activiti/#N153BA)

[Submit task form data](http://www.mossle.com/docs/activiti/#N15409)

[Database tables](http://www.mossle.com/docs/activiti/#N15445)

[List of tables](http://www.mossle.com/docs/activiti/#N15448)

[Get a single table](http://www.mossle.com/docs/activiti/#N1546D)

[Get column info for a single table](http://www.mossle.com/docs/activiti/#N154B1)

[Get row data for a single table](http://www.mossle.com/docs/activiti/#N154F5)

[Engine](http://www.mossle.com/docs/activiti/#N15573)

[Get engine properties](http://www.mossle.com/docs/activiti/#N15576)

[Get engine info](http://www.mossle.com/docs/activiti/#N1559B)

[Jobs](http://www.mossle.com/docs/activiti/#N155C0)

[Get a single job](http://www.mossle.com/docs/activiti/#N155C3)

[Delete a job](http://www.mossle.com/docs/activiti/#N15607)

[Execute a single job](http://www.mossle.com/docs/activiti/#N15644)

[Get the exception stacktrace for a job](http://www.mossle.com/docs/activiti/#N1568C)

[Get a list of jobs](http://www.mossle.com/docs/activiti/#N156C8)

[Users](http://www.mossle.com/docs/activiti/#N1578F)

[Get a single user](http://www.mossle.com/docs/activiti/#N15792)

[Get a list of users](http://www.mossle.com/docs/activiti/#N157D6)

[Update a user](http://www.mossle.com/docs/activiti/#N1586F)

[Create a user](http://www.mossle.com/docs/activiti/#N158AA)

[Delete a user](http://www.mossle.com/docs/activiti/#N158DD)

[Get a user's picture](http://www.mossle.com/docs/activiti/#N1591A)

[Updating a user's picture](http://www.mossle.com/docs/activiti/#N1595B)

[List a user's info](http://www.mossle.com/docs/activiti/#N159AB)

[Get a user's info](http://www.mossle.com/docs/activiti/#N159EF)

[Update a user's info](http://www.mossle.com/docs/activiti/#N15A3C)

[Create a new user's info entry](http://www.mossle.com/docs/activiti/#N15A97)

[Delete a user's info](http://www.mossle.com/docs/activiti/#N15AF1)

[Groups](http://www.mossle.com/docs/activiti/#N15B37)

[Get a single group](http://www.mossle.com/docs/activiti/#N15B3A)

[Get a list of groups](http://www.mossle.com/docs/activiti/#N15B7E)

[Update a group](http://www.mossle.com/docs/activiti/#N15BF9)

[Create a group](http://www.mossle.com/docs/activiti/#N15C31)

[Delete a group](http://www.mossle.com/docs/activiti/#N15C64)

[Get members in a group](http://www.mossle.com/docs/activiti/#N15CA1)

[Add a member to a group](http://www.mossle.com/docs/activiti/#N15CAC)

[Delete a member from a group](http://www.mossle.com/docs/activiti/#N15D05)

[Legacy REST - General Usage](http://www.mossle.com/docs/activiti/#N15D55)

[Repository](http://www.mossle.com/docs/activiti/#N15D6D)

[Upload Deployment](http://www.mossle.com/docs/activiti/#N15D71)

[Get Deployments](http://www.mossle.com/docs/activiti/#N15D95)

[Get Deployment Resources](http://www.mossle.com/docs/activiti/#N15DB9)

[Get Deployment Resource](http://www.mossle.com/docs/activiti/#N15DDD)

[Delete Deployment](http://www.mossle.com/docs/activiti/#N15E01)

[Delete Deployments](http://www.mossle.com/docs/activiti/#N15E25)

[Engine](http://www.mossle.com/docs/activiti/#N15E4D)

[Get Process Engine](http://www.mossle.com/docs/activiti/#N15E50)

[Processes](http://www.mossle.com/docs/activiti/#N15E75)

[List Process Definitions](http://www.mossle.com/docs/activiti/#N15E78)

[Get Process Definition Form Properties](http://www.mossle.com/docs/activiti/#N15E9C)

[Get Process Definition Form Resource](http://www.mossle.com/docs/activiti/#N15EC0)

[Get Process Definition Diagram](http://www.mossle.com/docs/activiti/#N15EE4)

[Start Process Instance](http://www.mossle.com/docs/activiti/#N15F08)

[List Process Instances](http://www.mossle.com/docs/activiti/#N15F2F)

[Get Process Instance Details](http://www.mossle.com/docs/activiti/#N15F53)

[Get Process Instance Diagram](http://www.mossle.com/docs/activiti/#N15F77)

[Get open tasks for a process instance](http://www.mossle.com/docs/activiti/#N15F9B)

[Signal an activity (receive task) for a specific process instance](http://www.mossle.com/docs/activiti/#N15FBF)

[Trigger a signal for a specific process instance](http://www.mossle.com/docs/activiti/#N15FE6)

[Tasks](http://www.mossle.com/docs/activiti/#N1600E)

[Get Task Summary](http://www.mossle.com/docs/activiti/#N16011)

[List Tasks](http://www.mossle.com/docs/activiti/#N16035)

[Get Task](http://www.mossle.com/docs/activiti/#N16063)

[Get Task Form](http://www.mossle.com/docs/activiti/#N16087)

[Perform Task Operation](http://www.mossle.com/docs/activiti/#N160AB)

[List Form Properties](http://www.mossle.com/docs/activiti/#N160D3)

[Add attachment to a task](http://www.mossle.com/docs/activiti/#N160F7)

[Get task attachment](http://www.mossle.com/docs/activiti/#N1611C)

[Add url to a task](http://www.mossle.com/docs/activiti/#N16140)

[Identity](http://www.mossle.com/docs/activiti/#N16166)

[Login](http://www.mossle.com/docs/activiti/#N16169)

[Get User](http://www.mossle.com/docs/activiti/#N16190)

[List User's Groups](http://www.mossle.com/docs/activiti/#N161B4)

[Search users](http://www.mossle.com/docs/activiti/#N161D6)

[Create user](http://www.mossle.com/docs/activiti/#N161FB)

[Add user to groups](http://www.mossle.com/docs/activiti/#N16225)

[Remove user from group](http://www.mossle.com/docs/activiti/#N1624C)

[Get user picture](http://www.mossle.com/docs/activiti/#N16270)

[Get Group](http://www.mossle.com/docs/activiti/#N16294)

[List Group Users](http://www.mossle.com/docs/activiti/#N162B8)

[Search groups](http://www.mossle.com/docs/activiti/#N162DA)

[Create group](http://www.mossle.com/docs/activiti/#N162FC)

[Add users to a group](http://www.mossle.com/docs/activiti/#N16326)

[Remove user from group](http://www.mossle.com/docs/activiti/#N1634D)

[Management](http://www.mossle.com/docs/activiti/#N16372)

[List Jobs](http://www.mossle.com/docs/activiti/#N16375)

[Get Job](http://www.mossle.com/docs/activiti/#N16399)

[Execute Job](http://www.mossle.com/docs/activiti/#N163BD)

[Execute Jobs](http://www.mossle.com/docs/activiti/#N163E1)

[List Database Tables](http://www.mossle.com/docs/activiti/#N16408)

[Get Table Meta Data](http://www.mossle.com/docs/activiti/#N1642C)

[Get Table Data](http://www.mossle.com/docs/activiti/#N1644E)

[16. 集成CDI](http://www.mossle.com/docs/activiti/#cdiintegration)

[设置activiti-cdi](http://www.mossle.com/docs/activiti/#N1648E)

[查找流程引擎](http://www.mossle.com/docs/activiti/#N16493)

[配置Process Engine](http://www.mossle.com/docs/activiti/#N164B7)

[发布流程](http://www.mossle.com/docs/activiti/#N164D5)

[基于CDI环境的流程执行](http://www.mossle.com/docs/activiti/#N164E3)

[与流程实例进行关联交互](http://www.mossle.com/docs/activiti/#N164E8)

[声明式流程控制](http://www.mossle.com/docs/activiti/#N16512)

[在流程中引用bean](http://www.mossle.com/docs/activiti/#N16529)

[使用@BusinessProcessScoped beans](http://www.mossle.com/docs/activiti/#N1653D)

[注入流程变量](http://www.mossle.com/docs/activiti/#N16547)

[接收流程事件](http://www.mossle.com/docs/activiti/#N16569)

[更多功能](http://www.mossle.com/docs/activiti/#N165A1)

[已知的问题](http://www.mossle.com/docs/activiti/#N165B9)

[17. 集成LDAP](http://www.mossle.com/docs/activiti/#N165BE)

[用法](http://www.mossle.com/docs/activiti/#ldap_usage)

[用例](http://www.mossle.com/docs/activiti/#ldap_usecases)

[配置](http://www.mossle.com/docs/activiti/#ldap_configuration)

[属性](http://www.mossle.com/docs/activiti/#ldap_properties)

[为Explorer集成LDAP](http://www.mossle.com/docs/activiti/#lda-explorer)

[18. 高级功能](http://www.mossle.com/docs/activiti/#N16705)

[监听流程解析](http://www.mossle.com/docs/activiti/#advanced_parseHandlers)

[支持高并发的UUID id生成器](http://www.mossle.com/docs/activiti/#advanced.uuid.generator)

[启用安全的BPMN 2.0 xml](http://www.mossle.com/docs/activiti/#advanced.safe.bpmn.xml)

**List of Tables**

2.1. [示例用户](http://www.mossle.com/docs/activiti/#N100BD)

2.2. [webapp工具](http://www.mossle.com/docs/activiti/#N100E2)

3.1. [支持的数据库](http://www.mossle.com/docs/activiti/#databaseTypes)

6.1.

6.2.

6.3.

8.1. [Mail server configuration](http://www.mossle.com/docs/activiti/#N118D2)

8.2. [Mail server configuration](http://www.mossle.com/docs/activiti/#N1192E)

8.3. [Mule server configuration](http://www.mossle.com/docs/activiti/#N119B2)

8.4. [Shell task parameter configuration](http://www.mossle.com/docs/activiti/#N11A9A)

15.1. [HTTP-methods and corresponding operations](http://www.mossle.com/docs/activiti/#N12B7B)

15.2. [HTTP-methods response codes](http://www.mossle.com/docs/activiti/#N12B9F)

15.3. [URL query parameter types](http://www.mossle.com/docs/activiti/#N12C19)

15.4. [JSON parameter types](http://www.mossle.com/docs/activiti/#N12C5E)

15.5. [Variable query JSON parameters](http://www.mossle.com/docs/activiti/#N12CA6)

15.6. [Variable query JSON parameters](http://www.mossle.com/docs/activiti/#N12CDA)

15.7. [Default query JSON types](http://www.mossle.com/docs/activiti/#N12D21)

15.8. [Variable JSON attributes](http://www.mossle.com/docs/activiti/#N12D6F)

15.9. [Variable Types](http://www.mossle.com/docs/activiti/#N12DC2)

15.10. [URL query parameters](http://www.mossle.com/docs/activiti/#N12E3E)

15.11. [REST Response codes](http://www.mossle.com/docs/activiti/#N12E90)

15.12. [Get a deployment - URL parameters](http://www.mossle.com/docs/activiti/#N12EB5)

15.13. [Get a deployment - Response codes](http://www.mossle.com/docs/activiti/#N12ED1)

15.14. [Create a new deployment - Response codes](http://www.mossle.com/docs/activiti/#N12F0C)

15.15. [Delete a deployment - URL parameters](http://www.mossle.com/docs/activiti/#N12F36)

15.16. [Delete a deployment - Response codes](http://www.mossle.com/docs/activiti/#N12F52)

15.17. [List resources in a deployment - URL parameters](http://www.mossle.com/docs/activiti/#N12F73)

15.18. [List resources in a deployment - Response codes](http://www.mossle.com/docs/activiti/#N12F8F)

15.19. [Get a deployment resource - URL parameters](http://www.mossle.com/docs/activiti/#N12FDD)

15.20. [Get a deployment resource - Response codes](http://www.mossle.com/docs/activiti/#N13005)

15.21. [Get a deployment resource content - URL parameters](http://www.mossle.com/docs/activiti/#N1304E)

15.22. [Get a deployment resource content - Response codes](http://www.mossle.com/docs/activiti/#N13076)

15.23. [List of process definitions - URL parameters](http://www.mossle.com/docs/activiti/#N130A2)

15.24. [List of process definitions - Response codes](http://www.mossle.com/docs/activiti/#N1314C)

15.25. [Get a process definition - URL parameters](http://www.mossle.com/docs/activiti/#N13187)

15.26. [Get a process definition - Response codes](http://www.mossle.com/docs/activiti/#N131A3)

15.27. [Update category for a process definition - Response codes](http://www.mossle.com/docs/activiti/#N131E7)

15.28. [Get a process definition resource content - URL parameters](http://www.mossle.com/docs/activiti/#N13216)

15.29. [Get a process definition BPMN model - URL parameters](http://www.mossle.com/docs/activiti/#N13245)

15.30. [Get a process definition BPMN model - Response codes](http://www.mossle.com/docs/activiti/#N13261)

15.31. [Suspend a process definition - JSON Body parameters](http://www.mossle.com/docs/activiti/#processDefinitionActionBodyParameters)

15.32. [Suspend a process definition - Response codes](http://www.mossle.com/docs/activiti/#N132C4)

15.33. [Activate a process definition - Response codes](http://www.mossle.com/docs/activiti/#N13300)

15.34. [Get all candidate starters for a process-definition - URL parameters](http://www.mossle.com/docs/activiti/#N1332F)

15.35. [Get all candidate starters for a process-definition - Response codes](http://www.mossle.com/docs/activiti/#N1334B)

15.36. [Add a candidate starter to a process definition - URL parameters](http://www.mossle.com/docs/activiti/#N13375)

15.37. [Add a candidate starter to a process definition - Response codes](http://www.mossle.com/docs/activiti/#N133A3)

15.38. [Delete a candidate starter from a process definition - URL parameters](http://www.mossle.com/docs/activiti/#N133CD)

15.39. [Delete a candidate starter from a process definition - Response codes](http://www.mossle.com/docs/activiti/#N13401)

15.40. [Get a candidate starter from a process definition - URL parameters](http://www.mossle.com/docs/activiti/#N1342B)

15.41. [Get a candidate starter from a process definition - Response codes](http://www.mossle.com/docs/activiti/#N1345F)

15.42. [Get a list of models - URL query parameters](http://www.mossle.com/docs/activiti/#N1348C)

15.43. [Get a list of models - Response codes](http://www.mossle.com/docs/activiti/#N1352A)

15.44. [Get a model - URL parameters](http://www.mossle.com/docs/activiti/#N13554)

15.45. [Get a model - Response codes](http://www.mossle.com/docs/activiti/#N13570)

15.46. [Update a model - Response codes](http://www.mossle.com/docs/activiti/#N135A6)

15.47. [Create a model - Response codes](http://www.mossle.com/docs/activiti/#N135D9)

15.48. [Delete a model - URL parameters](http://www.mossle.com/docs/activiti/#N135FE)

15.49. [Delete a model - Response codes](http://www.mossle.com/docs/activiti/#N1361A)

15.50. [Get the editor source for a model - URL parameters](http://www.mossle.com/docs/activiti/#N1363B)

15.51. [Get the editor source for a model - Response codes](http://www.mossle.com/docs/activiti/#N13657)

15.52. [Set the editor source for a model - URL parameters](http://www.mossle.com/docs/activiti/#N13681)

15.53. [Set the editor source for a model - Response codes](http://www.mossle.com/docs/activiti/#N136A6)

15.54. [Get the extra editor source for a model - URL parameters](http://www.mossle.com/docs/activiti/#N136D0)

15.55. [Get the extra editor source for a model - Response codes](http://www.mossle.com/docs/activiti/#N136EC)

15.56. [Set the extra editor source for a model - URL parameters](http://www.mossle.com/docs/activiti/#N13716)

15.57. [Set the extra editor source for a model - Response codes](http://www.mossle.com/docs/activiti/#N1373B)

15.58. [Get a process instance - URL parameters](http://www.mossle.com/docs/activiti/#N13768)

15.59. [Get a process instance - Response codes](http://www.mossle.com/docs/activiti/#N13784)

15.60. [Delete a process instance - URL parameters](http://www.mossle.com/docs/activiti/#N137AE)

15.61. [Delete a process instance - Response codes](http://www.mossle.com/docs/activiti/#N137CA)

15.62. [Activate or suspend a process instance - URL parameters](http://www.mossle.com/docs/activiti/#N137EB)

15.63. [Activate or suspend a process instance - Response codes](http://www.mossle.com/docs/activiti/#N13819)

15.64. [Start a process instance - Response codes](http://www.mossle.com/docs/activiti/#N13877)

15.65. [List of process instances - URL query parameters](http://www.mossle.com/docs/activiti/#N138A2)

15.66. [List of process instances - Response codes](http://www.mossle.com/docs/activiti/#N13931)

15.67. [Query process instances - Response codes](http://www.mossle.com/docs/activiti/#N13972)

15.68. [Get diagram for a process instance - URL parameters](http://www.mossle.com/docs/activiti/#N1399C)

15.69. [Get diagram for a process instance - Response codes](http://www.mossle.com/docs/activiti/#N139B8)

15.70. [Get involved people for process instance - URL parameters](http://www.mossle.com/docs/activiti/#N139E7)

15.71. [Get involved people for process instance - Response codes](http://www.mossle.com/docs/activiti/#N13A03)

15.72. [Add an involved user to a process instance - URL parameters](http://www.mossle.com/docs/activiti/#N13A30)

15.73. [Add an involved user to a process instance - Response codes](http://www.mossle.com/docs/activiti/#N13A5B)

15.74. [Remove an involved user to from process instance - URL parameters](http://www.mossle.com/docs/activiti/#N13A8D)

15.75. [Remove an involved user to from process instance - Response codes](http://www.mossle.com/docs/activiti/#N13ABB)

15.76. [List of variables for a process instance - URL parameters](http://www.mossle.com/docs/activiti/#N13AE8)

15.77. [List of variables for a process instance - Response codes](http://www.mossle.com/docs/activiti/#N13B04)

15.78. [Get a variable for a process instance - URL parameters](http://www.mossle.com/docs/activiti/#N13B37)

15.79. [Get a variable for a process instance - Response codes](http://www.mossle.com/docs/activiti/#N13B5C)

15.80. [Create (or update) variables on a process instance - URL parameters](http://www.mossle.com/docs/activiti/#N13B9D)

15.81. [Create (or update) variables on a process instance - Response codes](http://www.mossle.com/docs/activiti/#N13BC9)

15.82. [Update a single variable on a process instance - URL parameters](http://www.mossle.com/docs/activiti/#N13BFD)

15.83. [Update a single variable on a process instance - Response codes](http://www.mossle.com/docs/activiti/#N13C32)

15.84. [Create a new binary variable on a process-instance - URL parameters](http://www.mossle.com/docs/activiti/#N13C65)

15.85. [Create a new binary variable on a process-instance - Response codes](http://www.mossle.com/docs/activiti/#N13CA0)

15.86. [Update an existing binary variable on a process-instance - URL parameters](http://www.mossle.com/docs/activiti/#N13CD0)

15.87. [Update an existing binary variable on a process-instance - Response codes](http://www.mossle.com/docs/activiti/#N13D0B)

15.88. [Get an execution - URL parameters](http://www.mossle.com/docs/activiti/#N13D39)

15.89. [Get an execution - Response codes](http://www.mossle.com/docs/activiti/#N13D55)

15.90. [Execute an action on an execution - URL parameters](http://www.mossle.com/docs/activiti/#N13D7F)

15.91. [Execute an action on an execution - Response codes](http://www.mossle.com/docs/activiti/#N13DC2)

15.92. [Get active activities in an execution - URL parameters](http://www.mossle.com/docs/activiti/#N13DF8)

15.93. [Get active activities in an execution - Response codes](http://www.mossle.com/docs/activiti/#N13E14)

15.94. [List of executions - URL query parameters](http://www.mossle.com/docs/activiti/#N13E3F)

15.95. [List of executions - Response codes](http://www.mossle.com/docs/activiti/#N13EAD)

15.96. [Query executions - Response codes](http://www.mossle.com/docs/activiti/#N13EF4)

15.97. [List of variables for an execution - URL parameters](http://www.mossle.com/docs/activiti/#N13F1E)

15.98. [List of variables for an execution - Response codes](http://www.mossle.com/docs/activiti/#N13F49)

15.99. [Get a variable for an execution - URL parameters](http://www.mossle.com/docs/activiti/#N13F76)

15.100. [Get a variable for an execution - Response codes](http://www.mossle.com/docs/activiti/#N13FAA)

15.101. [Create (or update) variables on an execution - URL parameters](http://www.mossle.com/docs/activiti/#N13FE5)

15.102. [Create (or update) variables on an execution - Response codes](http://www.mossle.com/docs/activiti/#N14015)

15.103. [Update a variable on an execution - URL parameters](http://www.mossle.com/docs/activiti/#N14049)

15.104. [Update a variable on an execution - Response codes](http://www.mossle.com/docs/activiti/#N1407B)

15.105. [Create a new binary variable on an execution - URL parameters](http://www.mossle.com/docs/activiti/#N140A8)

15.106. [Create a new binary variable on an execution - Response codes](http://www.mossle.com/docs/activiti/#N140EB)

15.107. [Update an existing binary variable on a process-instance - URL parameters](http://www.mossle.com/docs/activiti/#N1411B)

15.108. [Update an existing binary variable on a process-instance - Response codes](http://www.mossle.com/docs/activiti/#N14167)

15.109. [Get a task - URL parameters](http://www.mossle.com/docs/activiti/#N14195)

15.110. [Get a task - Response codes](http://www.mossle.com/docs/activiti/#N141B1)

15.111. [List of tasks - URL query parameters](http://www.mossle.com/docs/activiti/#N141EB)

15.112. [List of tasks - Response codes](http://www.mossle.com/docs/activiti/#N14310)

15.113. [Query for tasks - Response codes](http://www.mossle.com/docs/activiti/#N14353)

15.114. [Update a task - Response codes](http://www.mossle.com/docs/activiti/#N14389)

15.115. [Task actions - Response codes](http://www.mossle.com/docs/activiti/#N143E6)

15.116. [>Delete a task - URL parameters](http://www.mossle.com/docs/activiti/#N1441D)

15.117. [>Delete a task - Response codes](http://www.mossle.com/docs/activiti/#N1444E)

15.118. [Get all variables for a task - URL parameters](http://www.mossle.com/docs/activiti/#N14474)

15.119. [Get all variables for a task - Response codes](http://www.mossle.com/docs/activiti/#N1449F)

15.120. [Get a variable from a task - URL parameters](http://www.mossle.com/docs/activiti/#N144CD)

15.121. [Get a variable from a task - Response codes](http://www.mossle.com/docs/activiti/#N14501)

15.122. [Get the binary data for a variable - URL parameters](http://www.mossle.com/docs/activiti/#N1452F)

15.123. [Get the binary data for a variable - Response codes](http://www.mossle.com/docs/activiti/#N1456C)

15.124. [Create new variables on a task - URL parameters](http://www.mossle.com/docs/activiti/#N1459F)

15.125. [Create new variables on a task - Response codes](http://www.mossle.com/docs/activiti/#N145E8)

15.126. [Create a new binary variable on a task - URL parameters](http://www.mossle.com/docs/activiti/#N14616)

15.127. [Create a new binary variable on a task - Response codes](http://www.mossle.com/docs/activiti/#N14659)

15.128. [Update an existing variable on a task - URL parameters](http://www.mossle.com/docs/activiti/#N1468C)

15.129. [Update an existing variable on a task - Response codes](http://www.mossle.com/docs/activiti/#N146DE)

15.130. [Updating a binary variable on a task - URL parameters](http://www.mossle.com/docs/activiti/#N14707)

15.131. [Updating a binary variable on a task - Response codes](http://www.mossle.com/docs/activiti/#N14753)

15.132. [Delete a variable on a task - URL parameters](http://www.mossle.com/docs/activiti/#N14781)

15.133. [Delete a variable on a task - Response codes](http://www.mossle.com/docs/activiti/#N147B8)

15.134. [Delete all local variables on a task - URL parameters](http://www.mossle.com/docs/activiti/#N147D9)

15.135. [Delete all local variables on a task - Response codes](http://www.mossle.com/docs/activiti/#N147F5)

15.136. [Get all identity links for a task - URL parameters](http://www.mossle.com/docs/activiti/#N14816)

15.137. [Get all identity links for a task - Response codes](http://www.mossle.com/docs/activiti/#N14832)

15.138. [Get all identitylinks for a task for either groups or users - URL parameters](http://www.mossle.com/docs/activiti/#N14866)

15.139. [Get all identitylinks for a task for either groups or users - Response codes](http://www.mossle.com/docs/activiti/#N148A3)

15.140. [Create an identity link on a task - URL parameters](http://www.mossle.com/docs/activiti/#N148CD)

15.141. [Create an identity link on a task - Response codes](http://www.mossle.com/docs/activiti/#N148FB)

15.142. [Delete an identity link on a task - URL parameters](http://www.mossle.com/docs/activiti/#N14925)

15.143. [Delete an identity link on a task - Response codes](http://www.mossle.com/docs/activiti/#N14962)

15.144. [Create a new comment on a task - URL parameters](http://www.mossle.com/docs/activiti/#N14983)

15.145. [Create a new comment on a task - Response codes](http://www.mossle.com/docs/activiti/#N149AF)

15.146. [Get all comments on a task - URL parameters](http://www.mossle.com/docs/activiti/#N149D5)

15.147. [Get all comments on a task - Response codes](http://www.mossle.com/docs/activiti/#N149F8)

15.148. [Get a comment on a task - URL parameters](http://www.mossle.com/docs/activiti/#N14A19)

15.149. [Get a comment on a task - Response codes](http://www.mossle.com/docs/activiti/#N14A45)

15.150. [Delete a comment on a task - URL parameters](http://www.mossle.com/docs/activiti/#N14A66)

15.151. [Delete a comment on a task - Response codes](http://www.mossle.com/docs/activiti/#N14A89)

15.152. [Get all events for a task - URL parameters](http://www.mossle.com/docs/activiti/#N14AA9)

15.153. [Get all events for a task - Response codes](http://www.mossle.com/docs/activiti/#N14ACC)

15.154. [Get an event on a task - URL parameters](http://www.mossle.com/docs/activiti/#N14AED)

15.155. [Get an event on a task - Response codes](http://www.mossle.com/docs/activiti/#N14B19)

15.156. [Create a new attachment on a task, containing a link to an external resource - URL parameters](http://www.mossle.com/docs/activiti/#N14B3A)

15.157. [Create a new attachment on a task, containing a link to an external resource - Response codes](http://www.mossle.com/docs/activiti/#N14B66)

15.158. [Create a new attachment on a task, with an attached file - URL parameters](http://www.mossle.com/docs/activiti/#N14B8C)

15.159. [Create a new attachment on a task, with an attached file - Response codes](http://www.mossle.com/docs/activiti/#N14BC9)

15.160. [Get all attachments on a task - URL parameters](http://www.mossle.com/docs/activiti/#N14BEF)

15.161. [Get all attachments on a task - Response codes](http://www.mossle.com/docs/activiti/#N14C12)

15.162. [Get an attachment on a task - URL parameters](http://www.mossle.com/docs/activiti/#N14C33)

15.163. [Get an attachment on a task - Response codes](http://www.mossle.com/docs/activiti/#N14C75)

15.164. [Get the content for an attachment - URL parameters](http://www.mossle.com/docs/activiti/#N14C96)

15.165. [Get the content for an attachment - Response codes](http://www.mossle.com/docs/activiti/#N14CBE)

15.166. [Delete an attachment on a task - URL parameters](http://www.mossle.com/docs/activiti/#N14CE8)

15.167. [Delete an attachment on a task - Response codes](http://www.mossle.com/docs/activiti/#N14D0B)

15.168. [Get a historic process instance - Response codes](http://www.mossle.com/docs/activiti/#N14D2E)

15.169. [List of historic process instances - URL parameters](http://www.mossle.com/docs/activiti/#N14D59)

15.170. [List of historic process instances - Response codes](http://www.mossle.com/docs/activiti/#N14DF4)

15.171. [Query for historic process instances - Response codes](http://www.mossle.com/docs/activiti/#N14E34)

15.172. [Response codes](http://www.mossle.com/docs/activiti/#N14E5E)

15.173. [Response codes](http://www.mossle.com/docs/activiti/#N14E7F)

15.174. [Get the binary data for a historic process instance variable - Response codes](http://www.mossle.com/docs/activiti/#N14EA9)

15.175. [Get a single historic task instance - Response codes](http://www.mossle.com/docs/activiti/#N14EDC)

15.176. [Get historic task instances - URL parameters](http://www.mossle.com/docs/activiti/#N14F07)

15.177. [Get historic task instances - Response codes](http://www.mossle.com/docs/activiti/#N15029)

15.178. [Query for historic task instances - Response codes](http://www.mossle.com/docs/activiti/#N1506C)

15.179. [Response codes](http://www.mossle.com/docs/activiti/#N15096)

15.180. [Response codes](http://www.mossle.com/docs/activiti/#N150B7)

15.181. [Get the binary data for a historic task instance variable - Response codes](http://www.mossle.com/docs/activiti/#N150E1)

15.182. [Get historic activity instances - URL parameters](http://www.mossle.com/docs/activiti/#N15115)

15.183. [Get historic activity instances - Response codes](http://www.mossle.com/docs/activiti/#N15183)

15.184. [Query for historic activity instances - Response codes](http://www.mossle.com/docs/activiti/#N151BC)

15.185. [List of historic variable instances - URL parameters](http://www.mossle.com/docs/activiti/#N151E7)

15.186. [List of historic variable instances - Response codes](http://www.mossle.com/docs/activiti/#N15231)

15.187. [Query for historic variable instances - Response codes](http://www.mossle.com/docs/activiti/#N15271)

15.188. [Get the binary data for a historic task instance variable - Response codes](http://www.mossle.com/docs/activiti/#N1529B)

15.189. [Get historic detail - URL parameters](http://www.mossle.com/docs/activiti/#N152CF)

15.190. [Get historic detail - Response codes](http://www.mossle.com/docs/activiti/#N1532B)

15.191. [Query for historic details - Response codes](http://www.mossle.com/docs/activiti/#N15364)

15.192. [Get the binary data for a historic detail variable - Response codes](http://www.mossle.com/docs/activiti/#N1538E)

15.193. [Get form data - URL parameters](http://www.mossle.com/docs/activiti/#N153C4)

15.194. [Get form data - Response codes](http://www.mossle.com/docs/activiti/#N153E9)

15.195. [Submit task form data - Response codes](http://www.mossle.com/docs/activiti/#N15425)

15.196. [List of tables - Response codes](http://www.mossle.com/docs/activiti/#N15452)

15.197. [Get a single table - URL parameters](http://www.mossle.com/docs/activiti/#N15477)

15.198. [Get a single table - Response codes](http://www.mossle.com/docs/activiti/#N1549A)

15.199. [Get column info for a single table - URL parameters](http://www.mossle.com/docs/activiti/#N154BB)

15.200. [Get column info for a single table - Response codes](http://www.mossle.com/docs/activiti/#N154DE)

15.201. [Get row data for a single table - URL parameters](http://www.mossle.com/docs/activiti/#N154FF)

15.202. [Get row data for a single table - URL query parameters](http://www.mossle.com/docs/activiti/#N1551B)

15.203. [Get row data for a single table - Response codes](http://www.mossle.com/docs/activiti/#N1555C)

15.204. [Get engine properties - Response codes](http://www.mossle.com/docs/activiti/#N15589)

15.205. [Get engine info - Response codes](http://www.mossle.com/docs/activiti/#N155AE)

15.206. [Get a single job - URL parameters](http://www.mossle.com/docs/activiti/#N155CD)

15.207. [Get a single job - Response codes](http://www.mossle.com/docs/activiti/#N155F0)

15.208. [Delete a job - URL parameters](http://www.mossle.com/docs/activiti/#N15611)

15.209. [Delete a job - Response codes](http://www.mossle.com/docs/activiti/#N1562D)

15.210. [Execute a single job - JSON Body parameters](http://www.mossle.com/docs/activiti/#N15655)

15.211. [Execute a single job - Response codes](http://www.mossle.com/docs/activiti/#N15670)

15.212. [Get the exception stacktrace for a job - URL parameters](http://www.mossle.com/docs/activiti/#N15696)

15.213. [Get the exception stacktrace for a job - Response codes](http://www.mossle.com/docs/activiti/#N156AE)

15.214. [Get a list of jobs - URL query parameters](http://www.mossle.com/docs/activiti/#N156D2)

15.215. [Get a list of jobs - Response codes](http://www.mossle.com/docs/activiti/#N15772)

15.216. [Get a single user - URL parameters](http://www.mossle.com/docs/activiti/#N1579C)

15.217. [Get a single user - Response codes](http://www.mossle.com/docs/activiti/#N157BF)

15.218. [Get a list of users - URL query parameters](http://www.mossle.com/docs/activiti/#N157E0)

15.219. [Get a list of users - Response codes](http://www.mossle.com/docs/activiti/#N1585D)

15.220. [Update a user - Response codes](http://www.mossle.com/docs/activiti/#N15885)

15.221. [Create a user - Response codes](http://www.mossle.com/docs/activiti/#N158BD)

15.222. [Delete a user - URL parameters](http://www.mossle.com/docs/activiti/#N158E7)

15.223. [Delete a user - Response codes](http://www.mossle.com/docs/activiti/#N15903)

15.224. [Get a user's picture - URL parameters](http://www.mossle.com/docs/activiti/#N15924)

15.225. [Get a user's picture - Response codes](http://www.mossle.com/docs/activiti/#N15944)

15.226. [Updating a user's picture - URL parameters](http://www.mossle.com/docs/activiti/#N15965)

15.227. [Updating a user's picture - Response codes](http://www.mossle.com/docs/activiti/#N15994)

15.228. [List a user's info - URL parameters](http://www.mossle.com/docs/activiti/#N159B5)

15.229. [List a user's info - Response codes](http://www.mossle.com/docs/activiti/#N159D8)

15.230. [Get a user's info - URL parameters](http://www.mossle.com/docs/activiti/#N159F9)

15.231. [Get a user's info - Response codes](http://www.mossle.com/docs/activiti/#N15A25)

15.232. [Update a user's info - URL parameters](http://www.mossle.com/docs/activiti/#N15A46)

15.233. [Update a user's info - Response codes](http://www.mossle.com/docs/activiti/#N15A7B)

15.234. [Create a new user's info entry - URL parameters](http://www.mossle.com/docs/activiti/#N15AA1)

15.235. [Create a new user's info entry - Response codes](http://www.mossle.com/docs/activiti/#N15ACD)

15.236. [Delete a user's info - URL parameters](http://www.mossle.com/docs/activiti/#N15AFB)

15.237. [Delete a user's info - Response codes](http://www.mossle.com/docs/activiti/#N15B20)

15.238. [Get a single group - URL parameters](http://www.mossle.com/docs/activiti/#N15B44)

15.239. [Get a single group - Response codes](http://www.mossle.com/docs/activiti/#N15B67)

15.240. [Get a list of groups - URL query parameters](http://www.mossle.com/docs/activiti/#N15B88)

15.241. [Get a list of groups - Response codes](http://www.mossle.com/docs/activiti/#N15BE7)

15.242. [Update a group - Response codes](http://www.mossle.com/docs/activiti/#N15C0C)

15.243. [Create a group - Response codes](http://www.mossle.com/docs/activiti/#N15C44)

15.244. [Delete a group - URL parameters](http://www.mossle.com/docs/activiti/#N15C6E)

15.245. [Delete a group - Response codes](http://www.mossle.com/docs/activiti/#N15C8A)

15.246. [Add a member to a group - URL parameters](http://www.mossle.com/docs/activiti/#N15CB6)

15.247. [Add a member to a group - Response codes](http://www.mossle.com/docs/activiti/#N15CDB)

15.248. [Delete a member from a group - URL parameters](http://www.mossle.com/docs/activiti/#N15D0F)

15.249. [Delete a member from a group - Response codes](http://www.mossle.com/docs/activiti/#N15D34)

17.1. [LDAP配置属性](http://www.mossle.com/docs/activiti/#N165F6)

17.2. [高级属性](http://www.mossle.com/docs/activiti/#N1669C)

**Chapter 1. 简介**

**Table of Contents**

[协议](http://www.mossle.com/docs/activiti/#license)

[下载](http://www.mossle.com/docs/activiti/#download)

[源码](http://www.mossle.com/docs/activiti/#sources)

[必要的软件](http://www.mossle.com/docs/activiti/#required.software)

[JDK 6+](http://www.mossle.com/docs/activiti/#N1002A)

[Eclipse Indigo 和 Juno](http://www.mossle.com/docs/activiti/#N10036)

[报告问题](http://www.mossle.com/docs/activiti/#reporting.problems)

[试验性功能](http://www.mossle.com/docs/activiti/#experimental)

[内部实现类](http://www.mossle.com/docs/activiti/#internal)

**协议**

Activiti是基于[Apache V2协议](http://www.mossle.com/license.txt)发布的。

**下载**

<http://activiti.org/download.html>

**源码**

从发布包里可以找到已经打好jar包的源码。 如果想从代码重新构建，请参考 [wiki “构建发布包”](http://docs.codehaus.org/display/ACT/Developers+Guide#DevelopersGuide-Buildingthedistribution)。

**必要的软件**

**JDK 6+**

Activiti需要运行在JDK 6或以上版本上。 进入 [Oracle Java SE 下载页](http://www.oracle.com/technetwork/java/javase/downloads/index.html) 点击 "下载 JDK"按钮。页面上也提供了安装的方法。 要想确认安装是否成功，可以在命令行中执行 java -version。 这就会打印出安装的JDK的版本。

**Eclipse Indigo 和 Juno**

在[Eclipse下载页](http://www.eclipse.org/downloads/)下载你选择的eclipse发布包。 解压下载文件，你就可以通过eclipse目录下的eclipse文件启动它。 在这个文档后面，专门有一章介绍[安装eclipse设计器插件](http://www.mossle.com/docs/activiti/#eclipseDesignerInstallation)。

**报告问题**

任何一个自觉的开发者都应该看看 [如果聪明的提问](http://www.catb.org/~esr/faqs/smart-questions.html)。

看完之后，你可以在[用户论坛](http://forums.activiti.org/en/viewforum.php?f=3)上进行提问和评论， 或者在[JIRA问题跟踪系统](http://jira.codehaus.org/browse/ACT)中创建问题。

**Note**

虽然Activiti已经托管在GitHub上了，但是问题不应该提交到GitHub的问题跟踪系统上。如果你想报告一个问题， 不要创建一个GitHub的问题，而是应该使用[JIRA](http://jira.codehaus.org/browse/ACT)。

**试验性功能**

那些标记着 **[EXPERIMENTAL]** 的章节表示功能尚未稳定。

包名中包含 .impl. 的类都是内部实现类，都是不保证稳定的。 不过，如果用户手册把哪些类列为配置项，那么它们可以认为是稳定不变的。

**内部实现类**

在jar包中，所有包名中包含.impl.（比如：org.activiti.engine.impl.pvm.delegate）的类都是实现类， 它们都是内部类。这些类和接口都是不保证稳定不变的。

**Chapter 2. 开始学习**

**Table of Contents**

[一分钟入门](http://www.mossle.com/docs/activiti/#demo.setup.one.minute.version)

[安装Activiti](http://www.mossle.com/docs/activiti/#activiti.setup)

[安装Activiti数据库](http://www.mossle.com/docs/activiti/#activiti.setup.database)

[引入Activiti jar和依赖](http://www.mossle.com/docs/activiti/#getting.started.including.libs)

[下一步](http://www.mossle.com/docs/activiti/#getting.started.next.steps)

**一分钟入门**

从[Activiti网站](http://www.activiti.org)下载Activiti Explorer的WAR文件后， 可以按照下列步骤以默认配置运行样例。 你需要一个[Java 运行环境](http://java.sun.com/javase/downloads/index.jsp)和 [Apache Tomcat](http://tomcat.apache.org/download-70.cgi) （其实，任何提供了servlet功能的web容器都可以正常运行。但是我们主要是使用tomcat进行的测试）。

* 把下载的activiti-explorer.war复制到Tomcat的webapps目录下。
* 执行Tomcat的bin目录下的startup.bat或startup.sh启动服务器。
* Tomcat启动后，打开浏览器访问<http://localhost:8080/activiti-explorer>。 使用kermit/kermit登录。

这样就好了！Activiti Explorer默认使用H2内存数据库，如果你想使用其他数据库 请参考[这里](http://www.mossle.com/docs/activiti/#activiti.setup)。

**安装Activiti**

要安装Activiti你需要一个 [Java运行环境](http://java.sun.com/javase/downloads/index.jsp) 和 [Apache Tomcat](http://tomcat.apache.org/download-70.cgi)。 还要确认设置好*JAVA\_HOME*系统变量。 不同的操作系统下的设置方法是不同的。

要运行Activiti Explorer和REST web应用，你要从Activiti的下载页下载WAR文件， 复制到Tomcat安装目录下webapps目录下。 默认Explorer应用使用的内存数据库已经包含了示例流程，用户和群组信息。

下面是示例中可以使用的用户：

**Table 2.1. 示例用户**

| 账号 | 密码 | 角色 |
| --- | --- | --- |
| kermit | kermit | admin |
| gonzo | gonzo | manager |
| fozzie | fozzie | user |

现在，你可以访问下列web应用：

**Table 2.2. webapp工具**

| Webapp名称 | URL | 描述 |  |
| --- | --- | --- | --- |
| Activiti Explorer | <http://localhost:8080/activiti-explorer> | 流程引擎的用户控制台。使用它来启动新流程，分配任务， 查看并认领任务，等等。这个工具也可以用来管理Activiti引擎。 |  |

注意Activiti Explorer演示实例只是一种简单快速展示Activiti的功能的方式。 但是并不是说只能使用这种方式使用Activiti。 Activiti只是一个jar， 可以内嵌到任何Java环境中：swing或者Tomcat, JBoss, WebSphere等等。 也可以把Activiti作为一个典型的单独运行的BPM服务器运行。 只要java可以做的，Activiti也可以。

**安装Activiti数据库**

就像在一分钟入门里说过的，Activiti Explorer默认使用H2内存数据库。 要让Activiti使用独立运行的H2数据库或者其他数据库， 可以修改Activiti Explorer web应用WEB-INF/classes目录下的db.properties。

另外，注意Activiti Explorer自动生成了演示用的默认用户和群组，流程定义，数据模型。 要想禁用这个功能，要修改WEB-INF目录下的activiti-standalone-context.xml。 可以使用下面的demoDataGenerator bean定义代码完全禁用安装默认数据。从代码中也可以看出，我们可以单独启用或禁用每一项功能。

      <bean id="demoDataGenerator" class="org.activiti.explorer.demo.DemoDataGenerator">

        <property name="processEngine" ref="processEngine" />

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

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

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

      </bean>

**引入Activiti jar和依赖**

为了引用Activiti jar和依赖，我们推荐使用 [Maven](http://maven.apache.org/)（或[Ivy](http://ant.apache.org/ivy/)）， 它简化了我们之间的依赖管理。 参考<http://www.activiti.org/community.html#maven.repository> 来为你的项目引入必须的jar包。

如果不想用Maven，你也可以自己把这些jar引入到你的项目中。 Activiti下载zip包包含了一个libs目录， 包含了所有Activiti的jar包（和源代码jar包）。依赖没有用这种方式发布。 Activiti引擎必须的依赖如下所示（通过mvn dependency:tree生成）：

org.activiti:activiti-engine:jar:5.12.1

+- org.apache.commons:commons-email:jar:1.2:compile

|  +- javax.mail:mail:jar:1.4.1:compile

|  \- javax.activation:activation:jar:1.1:compile

+- commons-lang:commons-lang:jar:2.4:compile

+- org.mybatis:mybatis:jar:3.1.1:compile

+- org.springframework:spring-beans:jar:3.1.2.RELEASE:compile

|  \- org.springframework:spring-core:jar:3.1.2.RELEASE:compile

|     +- org.springframework:spring-asm:jar:3.1.2.RELEASE:compile

|     \- commons-logging:commons-logging:jar:1.1.1:compile

\- joda-time:joda-time:jar:2.1:compile

注意：只有使用了[mail service task](http://www.mossle.com/docs/activiti/#bpmnEmailTask)才必须引入mail依赖jar。

所有依赖可以在[Activiti 源码](https://github.com/Activiti/Activiti)的模块中， 通过mvn dependency:copy-dependencies下载。

**下一步**

使用[Activiti Explorer](http://www.mossle.com/docs/activiti/#activitiExplorer) web应用 是一个熟悉Activiti概念和功能的好办法。但是， Activiti的主要目标是为你自己的应用添加强大的BPM和工作流功能。 下面的章节会帮助你熟悉 如何在你的环境中使用Activiti进行编程：

* [配置章节](http://www.mossle.com/docs/activiti/#configuration) 会教你如何设置Activiti， 如何获得ProcessEngine类的实例， 它是说有Activiti引擎功能的中心入口。
* [API章节](http://www.mossle.com/docs/activiti/#chapterApi)会带领你了解建立Activiti API的服务。 这些服务用简便的方法提供了Activiti引擎的强大功能， 它们可以使用在任何Java环境下。
* 对深入了解BPMN 2.0，Activiti引擎中流程的编写结构感兴趣吗？ 请继续浏览[BPMN 2.0 章节](http://www.mossle.com/docs/activiti/#bpmn20)。

**Chapter 3. 配置**

**Table of Contents**

[创建ProcessEngine](http://www.mossle.com/docs/activiti/#configuration)

[ProcessEngineConfiguration bean](http://www.mossle.com/docs/activiti/#configurationRoot)

[数据库配置](http://www.mossle.com/docs/activiti/#databaseConfiguration)

[支持的数据库](http://www.mossle.com/docs/activiti/#supporteddatabases)

[创建数据库表](http://www.mossle.com/docs/activiti/#creatingDatabaseTable)

[理解数据库表的命名](http://www.mossle.com/docs/activiti/#database.tables.explained)

[数据库升级](http://www.mossle.com/docs/activiti/#databaseUpgrade)

[启用Job执行器](http://www.mossle.com/docs/activiti/#jobExecutorConfiguration)

[配置邮件服务器](http://www.mossle.com/docs/activiti/#mailServerConfiguration)

[配置历史](http://www.mossle.com/docs/activiti/#historyConfiguration)

[为表达式和脚本暴露配置](http://www.mossle.com/docs/activiti/#exposingConfigurationBeans)

[配置部署缓存](http://www.mossle.com/docs/activiti/#processDefinitionCacheConfiguration)

[日志](http://www.mossle.com/docs/activiti/#loggingConfiguration)

**创建ProcessEngine**

Activiti流程引擎的配置文件是名为activiti.cfg.xml的XML文件。 注意这与使用[Spring方式创建流程引擎](http://www.mossle.com/docs/activiti/#springintegration) 是**不**一样的。

获得ProcessEngine最简单的办法是 使用org.activiti.engine.ProcessEngines类：

ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine()

它会在classpath下搜索activiti.cfg.xml， 并基于这个文件中的配置构建引擎。 下面代码展示了实例配置。 后面的章节会给出配置参数的详细介绍。

<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>

注意配置XML文件其实是一个spring的配置文件。 **但不是说Activiti只能用在Spring环境中！** 我们只是利用了Spring的解析和依赖注入功能 来构建引擎。

配置文件中使用的ProcessEngineConfiguration可以通过编程方式创建。 可以配置不同的bean id（比如，第三行）。

ProcessEngineConfiguration.createProcessEngineConfigurationFromResourceDefault();

ProcessEngineConfiguration.createProcessEngineConfigurationFromResource(String resource);

ProcessEngineConfiguration.createProcessEngineConfigurationFromResource(String resource, String beanName);

ProcessEngineConfiguration.createProcessEngineConfigurationFromInputStream(InputStream inputStream);

ProcessEngineConfiguration.createProcessEngineConfigurationFromInputStream(InputStream inputStream, String beanName);

也可以不使用配置文件，基于默认创建配置 （参考[各种支持类](http://www.mossle.com/docs/activiti/#configurationClasses)）

ProcessEngineConfiguration.createStandaloneProcessEngineConfiguration();

ProcessEngineConfiguration.createStandaloneInMemProcessEngineConfiguration();

所有这些ProcessEngineConfiguration.createXXX()方法都返回 ProcessEngineConfiguration，后续可以调整成所需的对象。 在调用buildProcessEngine()后， 就会创建一个ProcessEngine：

ProcessEngine processEngine = ProcessEngineConfiguration.createStandaloneInMemProcessEngineConfiguration()

  .setDatabaseSchemaUpdate(ProcessEngineConfiguration.DB\_SCHEMA\_UPDATE\_FALSE)

  .setJdbcUrl("jdbc:h2:mem:my-own-db;DB\_CLOSE\_DELAY=1000")

  .setJobExecutorActivate(true)

  .buildProcessEngine();

**ProcessEngineConfiguration bean**

activiti.cfg.xml必须包含一个id为'processEngineConfiguration'的bean。

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

这个bean会用来构建ProcessEngine。 有多个类可以用来定义processEngineConfiguration。 这些类对应不同的环境，并设置了对应的默认值。 最好选择（最）适用于你的环境的类， 这样可以少配置几个引擎的参数。 下面是目前可以使用的类（以后会包含更多）：

* **org.activiti.engine.impl.cfg.StandaloneProcessEngineConfiguration**: 单独运行的流程引擎。Activiti会自己处理事务。 默认，数据库只在引擎启动时检测 （如果没有Activiti的表或者表结构不正确就会抛出异常）。
* **org.activiti.engine.impl.cfg.StandaloneInMemProcessEngineConfiguration**: 单元测试时的辅助类。Activiti会自己控制事务。 默认使用H2内存数据库。数据库表会在引擎启动时创建，关闭时删除。 使用它时，不需要其他配置（除非使用job执行器或邮件功能）。
* **org.activiti.spring.SpringProcessEngineConfiguration**: 在Spring环境下使用流程引擎。 参考[Spring集成章节](http://www.mossle.com/docs/activiti/#springintegration)。
* **org.activiti.engine.impl.cfg.JtaProcessEngineConfiguration**: 单独运行流程引擎，并使用JTA事务。

**数据库配置**

Activiti可能使用两种方式配置数据库。 第一种方式是定义数据库配置参数：

* **jdbcUrl**: 数据库的JDBC URL。
* **jdbcDriver**: 对应不同数据库类型的驱动。
* **jdbcUsername**: 连接数据库的用户名。
* **jdbcPassword**: 连接数据库的密码。

基于JDBC参数配置的数据库连接 会使用默认的[MyBatis](http://www.mybatis.org/)连接池。 下面的参数可以用来配置连接池（来自MyBatis参数）：

* **jdbcMaxActiveConnections**: 连接池中处于被使用状态的连接的最大值。默认为10。
* **jdbcMaxIdleConnections**: 连接池中处于空闲状态的连接的最大值。
* **jdbcMaxCheckoutTime**: 连接被取出使用的最长时间，超过时间会被强制回收。 默认为20000（20秒）。
* **jdbcMaxWaitTime**: 这是一个底层配置，让连接池可以在长时间无法获得连接时， 打印一条日志，并重新尝试获取一个连接。（避免因为错误配置导致沉默的操作失败）。 默认为20000（20秒）。

示例数据库配置：

<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="" />

也可以使用javax.sql.DataSource。 （比如，[Apache Commons](http://commons.apache.org/dbcp/)的DBCP）：

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" >

  <property name="driverClassName" value="com.mysql.jdbc.Driver" />

  <property name="url" value="jdbc:mysql://localhost:3306/activiti" />

  <property name="username" value="activiti" />

  <property name="password" value="activiti" />

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

</bean>

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

    <property name="dataSource" ref="dataSource" />

    ...

注意，Activiti的发布包中没有这些类。 你要自己把对应的类（比如，从DBCP里）放到你的classpath下。

无论你使用JDBC还是DataSource的方式，都可以设置下面的配置：

* **databaseType**: 一般不用设置，因为可以自动通过数据库连接的元数据获取。 只有自动检测失败时才需要设置。 可能的值有：{h2, mysql, oracle, postgres, mssql, db2}。 **如果没使用默认的H2数据库就必须设置这项**。 这个配置会决定使用哪些创建/删除脚本和查询语句。 参考[支持数据库章节](http://www.mossle.com/docs/activiti/#supporteddatabases) 了解支持哪些类型。
* **databaseSchemaUpdate**: 设置流程引擎启动和关闭时如何处理数据库表。
  + false（默认）：检查数据库表的版本和依赖库的版本， 如果版本不匹配就抛出异常。
  + true: 构建流程引擎时，执行检查，如果需要就执行更新。 如果表不存在，就创建。
  + create-drop: 构建流程引擎时创建数据库表， 关闭流程引擎时删除这些表。

**支持的数据库**

下面列出Activiti使用的数据库类型（大小写敏感）。

**Table 3.1. 支持的数据库**

| Activiti数据库类型 | 测试版本 | JDBC URL实例 | 备注 |
| --- | --- | --- | --- |
| h2 | 1.3.168 | jdbc:h2:tcp://localhost/activiti | 默认配置的数据库 |
| mysql | 5.1.21 | jdbc:mysql://localhost:3306/activiti?autoReconnect=true | 使用mysql-connector-java驱动测试 |
| oracle | 11.2.0.1.0 | jdbc:oracle:thin:@localhost:1521:xe |  |
| postgres | 8.1 | jdbc:postgresql://localhost:5432/activiti |  |
| db2 | DB2 10.1 using db2jcc4 | jdbc:db2://localhost:50000/activiti |  |
| mssql | 2008 using sqljdbc4 | jdbc:sqlserver://localhost:1433/activiti |  |

**创建数据库表**

下面是创建数据库表最简单的办法：

* 把activiti-engine的jar放到classpath下
* 添加对应的数据库驱动
* 把Activiti配置文件 (*activiti.cfg.xml*) 放到 classpath下， 指向你的数据库（参考[数据库配置章节](http://www.mossle.com/docs/activiti/#databaseConfiguration)）
* 执行 *DbSchemaCreate* 类的main方法

然而，只有数据库管理员可以执行DDL语句。 SQL DDL语句可以从Activiti下载页或Activiti发布目录里找到，在database子目录下。 脚本也包含在引擎的jar中(*activiti-engine-x.jar*)， 在*org/activiti/db/create*包下（*drop*目录里是删除语句）。 SQL文件的命名方式如下

activiti.{db}.{create|drop}.{type}.sql

其中 *db* 是 [支持的数据库](http://www.mossle.com/docs/activiti/#supporteddatabases)， *type* 是

* **engine:** 引擎执行的表。必须。
* **identity:** 包含用户，群组，用户与组之间的关系的表。 这些表是可选的，只有使用引擎自带的默认身份管理时才需要。
* **history:** 包含历史和审计信息的表。可选的：历史级别设为*none*时不会使用。 注意这也会引用一些需要把数据保存到历史表中的功能（比如任务的评论）。

**理解数据库表的命名**

Activiti的表都以**ACT\_**开头。 第二部分是表示表的用途的两个字母标识。 用途也和服务的API对应。

* **ACT\_RE\_\***: 'RE'表示repository。 这个前缀的表包含了流程定义和流程静态资源 （图片，规则，等等）。
* **ACT\_RU\_\***: 'RU'表示runtime。 这些运行时的表，包含流程实例，任务，变量，异步任务，等运行中的数据。 Activiti只在流程实例执行过程中保存这些数据， 在流程结束时就会删除这些记录。 这样运行时表可以一直很小速度很快。
* **ACT\_ID\_\***: 'ID'表示identity。 这些表包含身份信息，比如用户，组等等。
* **ACT\_HI\_\***: 'HI'表示history。 这些表包含历史数据，比如历史流程实例， 变量，任务等等。
* **ACT\_GE\_\***: 通用数据， 用于不同场景下。

**数据库升级**

在执行更新之前要先备份数据库 （使用数据库的备份功能）

默认，每次构建流程引擎时都会还行版本检测。 这一版都在应用启动或Activiti webapp启动时发生。 如果Activiti发现数据库表的版本与依赖库的版本不同， 就会抛出异常。

要升级，你要把下面的配置 放到activiti.cfg.xml配置文件里：

<beans ... >

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

    <!-- ... -->

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

    <!-- ... -->

  </bean>

</beans>

**依然，把对应的数据库驱动放到classpath里。** 升级应用的Activiti依赖。启动一个新版本的Activiti 指向包含旧版本的数据库。将databaseSchemaUpdate设置为true， Activiti会自动将数据库表升级到新版本， 当发现依赖和数据库表版本不通过时。

**也可以执行更新升级DDL语句。** 也可以执行数据库脚本，可以在Activiti下载页找到。

**启用Job执行器**

JobExecutor是管理一系列线程的组件，可以触发定时器（也包含后续的异步消息）。 在单元测试场景下，很难使用多线程。因此API允许查询(ManagementService.createJobQuery)和执行job (ManagementService.executeJob)，所以job可以在单元测试中控制。 要避免与job执行器冲突，可以关闭它。

默认，JobExecutor在流程引擎启动时就会激活。 如果不想在流程引擎启动后自动激活JobExecutor，可以设置

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

**配置邮件服务器**

可以选择配置邮件服务器。Activiti支持在业务流程中发送邮件。 想真正的发送一个email，必须配置一个真实的SMTP邮件服务器。 参考[e-mail任务](http://www.mossle.com/docs/activiti/#bpmnEmailTaskServerConfiguration)。

**配置历史**

可以选择定制历史存储的配置。你可以通过配置影响引擎的[历史功能](http://www.mossle.com/docs/activiti/#history)。 参考[历史配置](http://www.mossle.com/docs/activiti/#historyConfig)。

<property name="history" value="audit" />

**为表达式和脚本暴露配置**

默认，activiti.cfg.xml和你自己的Spring配置文件中所有bean 都可以在表达式和脚本中使用。 如果你想限制配置文件中的bean的可见性， 可以配置流程引擎配置的beans配置。 ProcessEngineConfiguration的beans是一个map。当你指定了这个参数， 只有包含这个map中的bean可以在表达式和脚本中使用。 通过在map中指定的名称来决定暴露的bean。

**配置部署缓存**

所有流程定义都被缓存了（解析之后）避免每次使用前都要访问数据库， 因为流程定义数据是不会改变的。 默认，不会限制这个缓存。如果想限制流程定义缓存，可以添加如下配置

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

这个配置会把默认的hashmap缓存替换成LRU缓存，来提供限制。 当然，这个配置的最佳值跟流程定义的总数有关， 实际使用中会具体使用多少流程定义也有关。

也你可以注入自己的缓存实现。这个bean必须实现 org.activiti.engine.impl.persistence.deploy.DeploymentCache接口：

<property name="processDefinitionCache">

  <bean class="org.activiti.MyCache" />

</property>

有一个类似的配置叫knowledgeBaseCacheLimit和knowledgeBaseCache， 它们是配置规则缓存的。只有流程中使用规则任务时才会用到。

**日志**

从Activiti 5.12开始，SLF4J被用作日志框架，替换了之前使用java.util.logging。 所有日志（activiti, spring, mybatis等等）都转发给SLF4J 允许使用你选择的日志实现。

**默认activiti-engine依赖中没有提供SLF4J绑定的jar， 需要根据你的实际需要使用日志框架。**如果没有添加任何实现jar，SLF4J会使用NOP-logger，不使用任何日志，不会发出警告，而且什么日志都不会记录。 可以通过<http://www.slf4j.org/codes.html#StaticLoggerBinder>了解这些实现。

使用Maven，比如使用一个依赖（这里使用log4j），注意你还需要添加一个version：

<dependency>

  <groupId>org.slf4j</groupId>

  <artifactId>slf4j-log4j12</artifactId>

</dependency>

activiti-explorer和activiti-rest应用都使用了Log4j绑定。执行所有activiti-\*模块的单元测试页使用了Log4j。

**特别提醒如果容器classpath中存在commons-logging：** 为了把spring日志转发给SLF4J，需要使用桥接（参考<http://www.slf4j.org/legacy.html#jclOverSLF4J>）。 如果你的容器提供了commons-logging实现，请参考下面网页：<http://www.slf4j.org/codes.html#release>来确保稳定性。

使用Maven的实例（忽略版本）：

<dependency>

  <groupId>org.slf4j</groupId>

  <artifactId>jcl-over-slf4j</artifactId>

</dependency>

**Chapter 4. Activiti API**

**Table of Contents**

[流程引擎的API和服务](http://www.mossle.com/docs/activiti/#apiEngine)

[异常策略](http://www.mossle.com/docs/activiti/#N10486)

[使用Activiti的服务](http://www.mossle.com/docs/activiti/#api.services)

[发布流程](http://www.mossle.com/docs/activiti/#api.services.deployment)

[启动一个流程实例](http://www.mossle.com/docs/activiti/#api.services.start.processinstance)

[完成任务](http://www.mossle.com/docs/activiti/#api.services.tasls)

[挂起，激活一个流程](http://www.mossle.com/docs/activiti/#api.services.suspend)

[更多知识](http://www.mossle.com/docs/activiti/#api.services)

[查询API](http://www.mossle.com/docs/activiti/#queryAPI)

[表达式](http://www.mossle.com/docs/activiti/#apiExpressions)

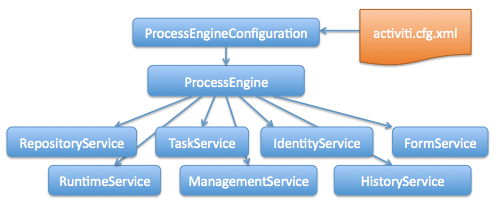
[单元测试](http://www.mossle.com/docs/activiti/#apiUnitTesting)

[调试单元测试](http://www.mossle.com/docs/activiti/#apiDebuggingUnitTest)

[web应用中的流程引擎](http://www.mossle.com/docs/activiti/#apiProcessEngineInWebApp)

**流程引擎的API和服务**

引擎API是与Activiti打交道的最常用方式。 我们从ProcessEngine开始， 创建它的很多种方法都已经在 [配置章节](http://www.mossle.com/docs/activiti/#configuration)中有所涉及。 从ProcessEngine中，你可以获得很多囊括工作流/BPM方法的服务。 ProcessEngine和服务类都是线程安全的。 你可以在整个服务器中仅保持它们的一个引用就可以了。



ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine();

RuntimeService runtimeService = processEngine.getRuntimeService();

RepositoryService repositoryService = processEngine.getRepositoryService();

TaskService taskService = processEngine.getTaskService();

ManagementService managementService = processEngine.getManagementService();

IdentityService identityService = processEngine.getIdentityService();

HistoryService historyService = processEngine.getHistoryService();

FormService formService = processEngine.getFormService();

ProcessEngines.getDefaultProcessEngine()会在第一次调用时 初始化并创建一个流程引擎，以后再调用就会返回相同的流程引擎。 使用对应的方法可以创建和关闭所有流程引擎：ProcessEngines.init() 和 ProcessEngines.destroy()。

ProcessEngines会扫描所有activiti.cfg.xml 和 activiti-context.xml 文件。 对于activiti.cfg.xml文件，流程引擎会使用Activiti的经典方式构建： ProcessEngineConfiguration.createProcessEngineConfigurationFromInputStream(inputStream).buildProcessEngine(). 对于activiti-context.xml文件，流程引擎会使用Spring方法构建：先创建一个Spring的环境， 然后通过环境获得流程引擎。

所有服务都是无状态的。这意味着可以在多节点集群环境下运行Activiti，每个节点都指向同一个数据库， 不用担心哪个机器实际执行前端的调用。 无论在哪里执行服务都没有问题。

**RepositoryService**可能是使用Activiti引擎时最先接触的服务。 它提供了管理和控制发布包和流程定义的操作。 这里不涉及太多细节，流程定义是BPMN 2.0流程的java实现。 它包含了一个流程每个环节的结构和行为。 发布包是Activiti引擎的打包单位。一个发布包可以包含多个BPMN 2.0 xml文件和其他资源。 开发者可以自由选择把任意资源包含到发布包中。 既可以把一个单独的BPMN 2.0 xml文件放到发布包里，也可以把整个流程和相关资源都放在一起。 （比如，'hr-processes'实例可以包含hr流程相关的任何资源）。 可以通过RepositoryService来部署这种发布包。 发布一个发布包，意味着把它上传到引擎中，所有流程都会在保存进数据库之前分析解析好。 从这点来说，系统知道这个发布包的存在，发布包中包含的流程就已经可以启动了。

除此之外，服务可以

* 查询引擎中的发布包和流程定义。
* 暂停或激活发布包，对应全部和特定流程定义。 暂停意味着它们不能再执行任何操作了，激活是对应的反向操作。
* 获得多种资源，像是包含在发布包里的文件， 或引擎自动生成的流程图。
* 获得流程定义的pojo版本， 可以用来通过java解析流程，而不必通过xml。

正如RepositoryService负责静态信息（比如，不会改变的数据，至少是不怎么改变的）， **RuntimeService**正好是完全相反的。它负责启动一个流程定义的新实例。 如上所述，流程定义定义了流程各个节点的结构和行为。 流程实例就是这样一个流程定义的实例。对每个流程定义来说，同一时间会有很多实例在执行。 RuntimeService也可以用来获取和保存流程变量。 这些数据是特定于某个流程实例的，并会被很多流程中的节点使用 （比如，一个排他网关常常使用流程变量来决定选择哪条路径继续流程）。 Runtimeservice也能查询流程实例和执行。 执行对应BPMN 2.0中的'token'。基本上执行指向流程实例当前在哪里。 最后，RuntimeService可以在流程实例等待外部触发时使用，这时可以用来继续流程实例。 流程实例可以有很多暂停状态，而服务提供了多种方法来'触发'实例， 接受外部触发后，流程实例就会继续向下执行。

任务是由系统中真实人员执行的，它是Activiti这类BPMN引擎的核心功能之一。 所有与任务有关的功能都包含在**TaskService**中：

* 查询分配给用户或组的任务
* 创建*独立运行*任务。这些任务与流程实例无关。
* 手工设置任务的执行者，或者这些用户通过何种方式与任务关联。
* 认领并完成一个任务。认领意味着一个人期望成为任务的执行者， 即这个用户会完成这个任务。完成意味着“做这个任务要求的事情”。 通常来说会有很多种处理形式。

**IdentityService**非常简单。它可以管理（创建，更新，删除，查询...）群组和用户。 请注意， Activiti执行时并没有对用户进行检查。 例如，任务可以分配给任何人，但是引擎不会校验系统中是否存在这个用户。 这是Activiti引擎也可以使用外部服务，比如ldap，活动目录，等等。

**FormService**是一个可选服务。即使不使用它，Activiti也可以完美运行， 不会损失任何功能。这个服务提供了*启动表单*和*任务表单*两个概念。 *启动表单*会在流程实例启动之前展示给用户， *任务表单*会在用户完成任务时展示。Activiti支持在BPMN 2.0流程定义中设置这些表单。 这个服务以一种简单的方式将数据暴露出来。再次重申，它时可选的， 表单也不一定要嵌入到流程定义中。

**HistoryService**提供了Activiti引擎手机的所有历史数据。 在执行流程时，引擎会保存很多数据（根据配置），比如流程实例启动时间，任务的参与者， 完成任务的时间，每个流程实例的执行路径，等等。 这个服务主要通过查询功能来获得这些数据。

**ManagementService**在使用Activiti的定制环境中基本上不会用到。 它可以查询数据库的表和表的元数据。另外，它提供了查询和管理异步操作的功能。 Activiti的异步操作用途很多，比如定时器，异步操作， 延迟暂停、激活，等等。后续，会讨论这些功能的更多细节。

可以从[javadocs](http://www.mossle.com/docs/javadocs/index.html)中获得这些服务和引擎API的更多信息。

**异常策略**

Activiti中的基础异常为org.activiti.engine.ActivitiException，一个非检查异常。 这个异常可以在任何时候被API抛出，不过特定方法抛出的“特定”的异常都记录在 [javadocs](http://www.mossle.com/docs/javadocs/index.html)中。 例如，下面的TaskService：

/\*\*

 \* Called when the task is successfully executed.

 \* @param taskId the id of the task to complete, cannot be null.

 \* @throws ActivitiObjectNotFoundException when no task exists with the given id.

 \*/

 void complete(String taskId);

在上面的例子中，当传入一个不存在的任务的id时，就会抛出异常。 同时，javadoc**明确指出taskId不能为null，如果传入null， 就会抛出ActivitiIllegalArgumentException**。

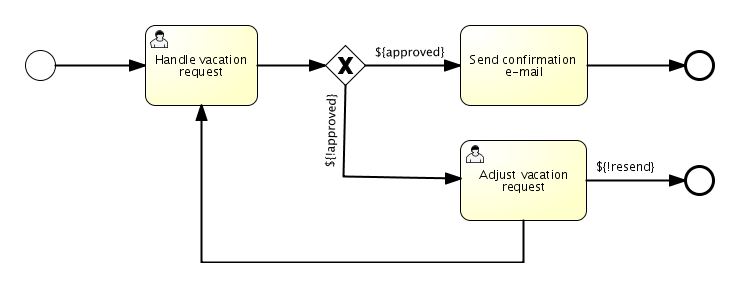
我们希望避免过多的异常继承，下面的子类用于特定的场合。 流程引擎和API调用的其他场合不会使用下面的异常， 它们会抛出一个普通的ActivitiExceptions。

* ActivitiWrongDbException：当Activiti引擎发现数据库版本号和引擎版本号不一致时抛出。
* ActivitiOptimisticLockingException：对同一数据进行并发方法并出现乐观锁时抛出。
* ActivitiClassLoadingException：当无法找到需要加载的类或在加载类时出现了错误（比如，JavaDelegate，TaskListener等。
* ActivitiObjectNotFoundException：当请求或操作的对应不存在时抛出。
* ActivitiIllegalArgumentException：这个异常表示调用Activiti API时传入了一个非法的参数，可能是引擎配置中的非法值，或提供了一个非法制，或流程定义中使用的非法值。
* ActivitiTaskAlreadyClaimedException：当任务已经被认领了，再调用taskService.claim(...)就会抛出。

**使用Activiti的服务**

像上面介绍的那样，要想操作Activiti引擎，需要通过 org.activiti.engine.ProcessEngine实例暴露的服务。 下面的代码假设你已经拥有了一个可以运行的Activiti环境。 你就可以操作一个org.activiti.engine.ProcessEngine。 如果只想简单尝试一下代码， 可以下载或者clone[Activiti单元测试模板](https://github.com/Activiti/activiti-unit-test-template)， 导入到IDE中，把testUserguideCode()方法添加到 org.activiti.MyUnitTest中。

这个小例子的最终目标是做一个工作业务流程， 演示公司中简单的请假申请：



**发布流程**

任何与“静态”资源有关的数据（比如流程定义）都可以通过 **RepositoryService**访问。 从概念上讲，所以静态数据都是Activiti的资源内容。

在src/test/resources/org/activiti/test目录下创建一个新的xml文件 VacationRequest.bpmn20.xml（如果不使用单元测试模板，你也可以在任何地方创建）， 内容如下。注意这一章不会解释例子中使用的xml结构。 如果有需要可以先阅读[bpmn 2.0章](http://www.mossle.com/docs/activiti/#bpmn20)来了解这些。

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

<definitions id="definitions"

             targetNamespace="http://activiti.org/bpmn20"

             xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"

             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

             xmlns:activiti="http://activiti.org/bpmn">

  <process id="vacationRequest" name="Vacation request">

    <startEvent id="request" activiti:initiator="employeeName">

      <extensionElements>

        <activiti:formProperty id="numberOfDays" name="Number of days" type="long" value="1" required="true"/>

        <activiti:formProperty id="startDate" name="First day of holiday (dd-MM-yyy)" datePattern="dd-MM-yyyy hh:mm" type="date" required="true" />

        <activiti:formProperty id="vacationMotivation" name="Motivation" type="string" />

      </extensionElements>

    </startEvent>

    <sequenceFlow id="flow1" sourceRef="request" targetRef="handleRequest" />

    <userTask id="handleRequest" name="Handle vacation request" >

      <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" required="true">

          <activiti:value id="true" name="Approve" />

          <activiti:value id="false" name="Reject" />

        </activiti:formProperty>

        <activiti:formProperty id="managerMotivation" name="Motivation" type="string" />

      </extensionElements>

      <potentialOwner>

        <resourceAssignmentExpression>

          <formalExpression>management</formalExpression>

        </resourceAssignmentExpression>

      </potentialOwner>

    </userTask>

    <sequenceFlow id="flow2" sourceRef="handleRequest" targetRef="requestApprovedDecision" />

    <exclusiveGateway id="requestApprovedDecision" name="Request approved?" />

    <sequenceFlow id="flow3" sourceRef="requestApprovedDecision" targetRef="sendApprovalMail">

      <conditionExpression xsi:type="tFormalExpression">${vacationApproved == 'true'}</conditionExpression>

    </sequenceFlow>

    <task id="sendApprovalMail" name="Send confirmation e-mail" />

    <sequenceFlow id="flow4" sourceRef="sendApprovalMail" targetRef="theEnd1" />

    <endEvent id="theEnd1" />

    <sequenceFlow id="flow5" sourceRef="requestApprovedDecision" targetRef="adjustVacationRequestTask">

      <conditionExpression xsi:type="tFormalExpression">${vacationApproved == 'false'}</conditionExpression>

    </sequenceFlow>

    <userTask id="adjustVacationRequestTask" name="Adjust vacation request">

      <documentation>

        Your manager has disapproved your vacation request for ${numberOfDays} days.

        Reason: ${managerMotivation}

      </documentation>

      <extensionElements>

        <activiti:formProperty id="numberOfDays" name="Number of days" value="${numberOfDays}" type="long" required="true"/>

        <activiti:formProperty id="startDate" name="First day of holiday (dd-MM-yyy)" value="${startDate}" datePattern="dd-MM-yyyy hh:mm" type="date" required="true" />

        <activiti:formProperty id="vacationMotivation" name="Motivation" value="${vacationMotivation}" type="string" />

        <activiti:formProperty id="resendRequest" name="Resend vacation request to manager?" type="enum" required="true">

          <activiti:value id="true" name="Yes" />

          <activiti:value id="false" name="No" />

        </activiti:formProperty>

      </extensionElements>

      <humanPerformer>

        <resourceAssignmentExpression>

          <formalExpression>${employeeName}</formalExpression>

        </resourceAssignmentExpression>

      </humanPerformer>

    </userTask>

    <sequenceFlow id="flow6" sourceRef="adjustVacationRequestTask" targetRef="resendRequestDecision" />

    <exclusiveGateway id="resendRequestDecision" name="Resend request?" />

    <sequenceFlow id="flow7" sourceRef="resendRequestDecision" targetRef="handleRequest">

      <conditionExpression xsi:type="tFormalExpression">${resendRequest == 'true'}</conditionExpression>

    </sequenceFlow>

     <sequenceFlow id="flow8" sourceRef="resendRequestDecision" targetRef="theEnd2">

      <conditionExpression xsi:type="tFormalExpression">${resendRequest == 'false'}</conditionExpression>

    </sequenceFlow>

    <endEvent id="theEnd2" />

  </process>

</definitions>

为了让Activiti引擎知道这个流程，我们必须先进行“发布”。 发布意味着引擎会把BPMN 2.0 xml解析成可以执行的东西， “发布包”中的所有流程定义都会添加到数据库中。 这样，当引擎重启时，它依然可以获得“已发布”的流程：

ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine();

RepositoryService repositoryService = processEngine.getRepositoryService();

repositoryService.createDeployment()

  .addClasspathResource("org/activiti/test/VacationRequest.bpmn20.xml")

  .deploy();

Log.info("Number of process definitions: " + repositoryService.createProcessDefinitionQuery().count());

可以阅读[发布章](http://www.mossle.com/docs/activiti/#chDeployment)来了解更多关于发布的信息。

**启动一个流程实例**

把流程定义发布到Activiti引擎后，我们可以基于它发起新流程实例。 对每个流程定义，都可以有很多流程实例。 流程定义是“蓝图”，流程实例是它的一个运行的执行。

所有与流程运行状态相关的东西都可以通过**RuntimeService**获得。 有很多方法可以启动一个新流程实例。在下面的代码中，我们使用定义在流程定义xml 中的key来启动流程实例。 我们也可以在流程实例启动时添加一些流程变量，因为第一个用户任务的表达式需要这些变量。 流程变量经常会被用到，因为它们赋予来自同一个流程定义的不同流程实例的特别含义。 简单来说，流程变量是区分流程实例的关键。

Map<String, Object> variables = new HashMap<String, Object>();

variables.put("employeeName", "Kermit");

variables.put("numberOfDays", new Integer(4));

variables.put("vacationMotivation", "I'm really tired!");

RuntimeService runtimeService = processEngine.getRuntimeService();

ProcessInstance processInstance = runtimeService.startProcessInstanceByKey("vacationRequest", variables);

// Verify that we started a new process instance

Log.info("Number of process instances: " + runtimeService.createProcessInstanceQuery().count());

**完成任务**

流程启动后，第一步就是用户任务。这是必须由系统用户处理的一个环节。 通常，用户会有一个“任务列表”，展示了所有必须由整个用户处理的任务。 下面的代码展示了对应的查询可能是怎样的：

// Fetch all tasks for the management group

TaskService taskService = processEngine.getTaskService();

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

for (Task task : tasks) {

  Log.info("Task available: " + task.getName());

}

为了让流程实例继续运行，我们需要完成整个任务。对Activiti来说，就是需要complete任务。 下面的代码展示了如何做这件事：

Task task = tasks.get(0);

Map<String, Object> taskVariables = new HashMap<String, Object>();

taskVariables.put("vacationApproved", "false");

taskVariables.put("managerMotivation", "We have a tight deadline!");

taskService.complete(task.getId(), taskVariables);

流程实例会进入到下一个环节。在这里例子中， 流程又会进入第一个环节，因为任务没有被通过。

**挂起，激活一个流程**

我们可以挂起一个流程定义。当挂起流程定时时， 就不能创建新流程了（会抛出一个异常）。 可以通过RepositoryService挂起一个流程：

repositoryService.suspendProcessDefinitionByKey("vacationRequest");

try {

  runtimeService.startProcessInstanceByKey("vacationRequest");

} catch (ActivitiException e) {

  e.printStackTrace();

}

要想重新激活一个流程定义，可以调用repositoryService.activateProcessDefinitionXXX方法。

也可以挂起一个流程实例。挂起时，流程不能继续执行（比如，完成任务会抛出异常）， 异步操作（比如定时器）也不会执行。 骨气流程实例可以调用 runtimeService.suspendProcessInstance方法。 激活流程实例可以调用runtimeService.activateProcessInstanceXXX方法。

**更多知识**

上面章节中我们仅仅覆盖了Activiti功能的表层。 未来我们会继续扩展这些章节，以覆盖更多Activiti API。 当然，像其他开源项目一样，学习的最好方式 是研究代码，阅读javadoc。

**查询API**

有两种方法可以从引擎中查询数据：查询API和原生查询。查询API提供了完全类型安全的API。 你可以为自己的查询条件添加很多条件 （所以条件都以AND组合）和精确的排序条件。下面的代码展示了一个例子：

      List<Task> tasks = taskService.createTaskQuery()

         .taskAssignee("kermit")

         .processVariableValueEquals("orderId", "0815")

         .orderByDueDate().asc()

         .list();

有时，你需要更强大的查询，比如使用OR条件或不能使用查询API实现的条件。 这时，我们推荐原生查询，它让你可以编写自己的SQL查询。 返回类型由你使用的查询对象决定，数据会映射到正确的对象上。比如，任务，流程实例，，执行，等等。 因为查询会作用在数据库上，你必须使用数据库中定义的表名和列名；这要求了解内部数据结构， 因此使用原生查询时一定要注意。表名可以通过API获得，可以尽量减少对数据库的依赖。

      List<Task> tasks = taskService.createNativeTaskQuery()

        .sql("SELECT count(\*) FROM " + managementService.getTableName(Task.class) + " T WHERE T.NAME\_ = #{taskName}")

        .parameter("taskName", "gonzoTask")

        .list();

      long count = taskService.createNativeTaskQuery()

        .sql("SELECT count(\*) FROM " + managementService.getTableName(Task.class) + " T1, "

               + managementService.getTableName(VariableInstanceEntity.class) + " V1 WHERE V1.TASK\_ID\_ = T1.ID\_")

        .count();

**表达式**

Activiti使用UEL处理表达式。UEL即*统一表达式语言*，它时EE6规范的一部分（参考 [EE6规范](http://docs.oracle.com/javaee/6/tutorial/doc/gjddd.html)）。为了在所有运行环境都支持最新UEL的所有共嫩个，我们使用了一个JUEL的修改版本。

表达式可以用在很多场景下，比如[Java服务任务](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTaskXML)，[执行监听器](http://www.mossle.com/docs/activiti/#executionListeners)，[任务监听器](http://www.mossle.com/docs/activiti/#taskListeners)和[条件流](http://www.mossle.com/docs/activiti/#conditionalSequenceFlowXml)。 虽然有两重表达式，值表达式和方法表达式，Activiti进行了抽象，所以两者可以同样使用在需要表达式的场景中。

* **Value expression**：解析为值。默认，所有流程变量都可以使用。所有spring bean（spring环境中）也可以使用在表达式中。 一些实例：
* ${myVar}

${myBean.myProperty}

* **Method expression**：调用一个方法，使用或不使用参数。**当调用一个无参数的方法时，记得在方法名后添加空的括号（以区分值表达式）。** 传递的参数可以是字符串也可以是表达式，它们会被自动解析。例子：
* ${printer.print()}
* ${myBean.addNewOrder('orderName')}

${myBean.doSomething(myVar, execution)}

注意这些表达式支持解析原始类型（包括比较），bean，list，数组和map。

在所有流程实例中，表达式中还可以使用一些默认对象：

* execution：DelegateExecution提供外出执行的额外信息。
* task：DelegateTask提供当前任务的额外信息。**注意，只对任务监听器的表达式有效。**
* authenticatedUserId：当前登录的用户id。如果没有用户登录，这个变量就不可用。

想要更多具体的使用方式和例子，参考[spring中的表达式](http://www.mossle.com/docs/activiti/#springExpressions)，[Java服务任务](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTaskXML)，[执行监听器](http://www.mossle.com/docs/activiti/#executionListeners)，[任务监听器](http://www.mossle.com/docs/activiti/#taskListeners)或[条件流](http://www.mossle.com/docs/activiti/#conditionalSequenceFlowXml)。

**单元测试**

业务流程是软件项目的一部分，它也应该和普通的业务流程一样进行测试： 使用单元测试。 因为Activiti是一个嵌入式的java引擎， 为业务流程编写单元测试和写普通单元测试完全一样。

Activiti支持JUnit 3和4进行单元测试。使用JUnit 3时， 必须集成org.activiti.engine.test.ActivitiTestCase。 它通过保护的成员变量提供ProcessEngine和服务， 在测试的setup()中， 默认会使用classpath下的activiti.cfg.xml初始化流程引擎。 想使用不同的配置文件，可以重写*getConfigurationResource()*方法。 如果配置文件相同的话，对应的流程引擎会被静态缓存， 就可以用于多个单元测试。

继承了ActivitiTestCase你，可以在测试方法上使用 org.activiti.engine.test.Deployment注解。 测试执行前，与测试类在同一个包下的， 格式为testClassName.testMethod.bpmn20.xml的资源文件，会被部署。 测试结束后，发布包也会被删除，包括所有相关的流程实例，任务，等等。 Deployment注解也可以直接设置资源的位置。 参考[Javadocs](http://www.mossle.com/docs/javadocs/org/activiti/engine/test/Deployment.html)获得更多信息。

把这些放在一起，JUnit 3测试看起来像这样。

public class MyBusinessProcessTest extends ActivitiTestCase {

  @Deployment

  public void testSimpleProcess() {

    runtimeService.startProcessInstanceByKey("simpleProcess");

    Task task = taskService.createTaskQuery().singleResult();

    assertEquals("My Task", task.getName());

    taskService.complete(task.getId());

    assertEquals(0, runtimeService.createProcessInstanceQuery().count());

  }

}

要想在使用JUnit 4编写单元测试时获得同样的功能， 可以使用org.activiti.engine.test.ActivitiRule。 通过它，可以通过getter方法获得流程引擎和各种服务。 和 ActivitiTestCase一样（参考上面章节），使用这个Rule 也会启用org.activiti.engine.test.Deployment注解（参考上面章节使用和配置的介绍）， 它会在classpath下查找默认的配置文件。 如果配置文件相同的话，对应的流程引擎会被静态缓存， 就可以用于多个单元测试。

下面的代码演示了JUnit 4单元测试并使用了ActivitiRule的例子。

public class MyBusinessProcessTest {

  @Rule

  public ActivitiRule activitiRule = new ActivitiRule();

  @Test

  @Deployment

  public void ruleUsageExample() {

    RuntimeService runtimeService = activitiRule.getRuntimeService();

    runtimeService.startProcessInstanceByKey("ruleUsage");

    TaskService taskService = activitiRule.getTaskService();

    Task task = taskService.createTaskQuery().singleResult();

    assertEquals("My Task", task.getName());

    taskService.complete(task.getId());

    assertEquals(0, runtimeService.createProcessInstanceQuery().count());

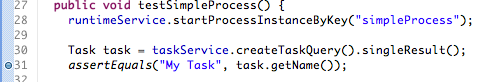
  }

}

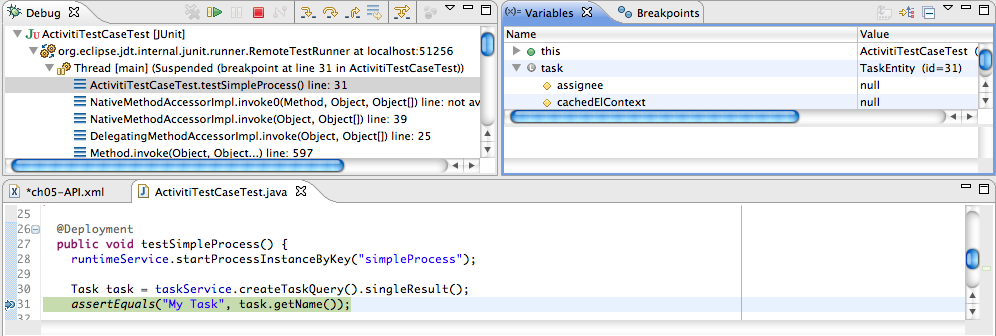
**调试单元测试**

当使用内存数据库H2进行单元测试时，下面的教程会告诉我们 如何在调试环境下更容易的监视Activiti的数据库。 这里的截图都是基于eclipse，这种机制很容易复用到其他IDE下。 IDEs.

假设我们已经在单元测试里设置了一个*断点*。 Ecilpse里，在代码左侧双击：



现在用*调试*模式运行单元测试（右击单元测试， 选择“运行为”和“单元测试”），测试会停在我们的断点上， 然后我们就可以监视测试的变量，它们显示在右侧面板里。



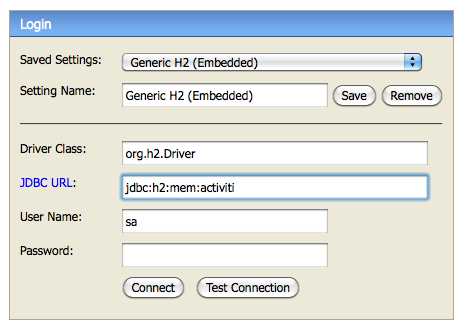
要监视Activiti的数据，打开*“显示”*窗口 （如果找不到，打开“窗口”->“显示视图”->“其他”，选择*显示*。） 并点击（代码已完成）org.h2.tools.Server.createWebServer("-web").start()



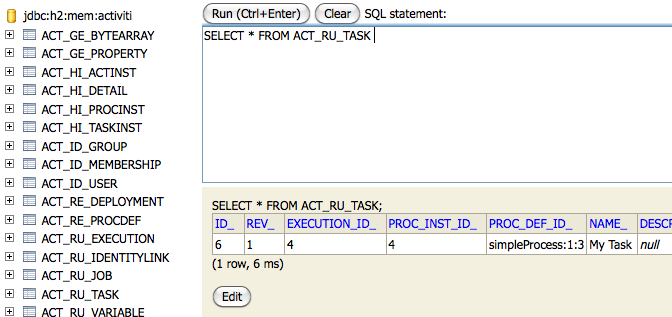
选择你点击的行，右击。然后选择“显示”（或者直接快捷方式就不用右击了）



现在打开一个浏览器，打开<http://localhost:8082>， 输入内存数据库的JDBC URL（默认为jdbc:h2:mem:activiti）， 点击连接按钮。



你仙子阿可以看到Activiti的数据，通过它们可以了解单元测试时如何以及为什么这样运行的。



**web应用中的流程引擎**

ProcessEngine是线程安全的， 可以在多线程下共享。在web应用中， 意味着可以在容器启动时创建流程引擎， 在容器关闭时关闭流程引擎。

下面代码演示了如何编写一个ServletContextListener 在普通的Servlet环境下初始化和销毁流程引擎：

public class ProcessEnginesServletContextListener implements ServletContextListener {

  public void contextInitialized(ServletContextEvent servletContextEvent) {

    ProcessEngines.init();

  }

  public void contextDestroyed(ServletContextEvent servletContextEvent) {

    ProcessEngines.destroy();

  }

}

contextInitialized方法会执行ProcessEngines.init()。 这会查找classpath下的activiti.cfg.xml文件， 根据配置文件创建一个ProcessEngine（比如，多个jar中都包含配置文件）。 如果classpath中包含多个配置文件，确认它们有不同的名字。 当需要使用流程引擎时，可以通过

ProcessEngines.getDefaultProcessEngine()

或

ProcessEngines.getProcessEngine("myName");

。 当然，也可以使用其他方式创建流程引擎， 可以参考[配置章节](http://www.mossle.com/docs/activiti/#configuration)中的描述。

ContextListener中的contextDestroyed方法会执行ProcessEngines.destroy(). 这会关闭所有初始化的流程引擎。

**Chapter 5. Spring集成**

**Table of Contents**

[ProcessEngineFactoryBean](http://www.mossle.com/docs/activiti/#N10680)

[事务](http://www.mossle.com/docs/activiti/#N10698)

[表达式](http://www.mossle.com/docs/activiti/#springExpressions)

[资源的自动部署](http://www.mossle.com/docs/activiti/#N106F6)

[单元测试](http://www.mossle.com/docs/activiti/#springUnitTest)

虽然没有Spring你也可以使用Activiti，但是我们提供了一些非常不错的集成特性。这一章我们将介绍这些特性。

**ProcessEngineFactoryBean**

可以把流程引擎（ProcessEngine）作为一个普通的Spring bean进行配置。 类 org.activiti.spring.ProcessEngineFactoryBean是集成的切入点。 这个bean需要一个流程引擎配置来创建流程引擎。这也意味着在文档的[配置这一章](http://www.mossle.com/docs/activiti/#configuration)的介绍属性的创建和配置对于Spring来说也是一样的。对于Spring集成的配置和流程引擎bean看起来像这样：

<bean id="processEngineConfiguration" class="org.activiti.spring.SpringProcessEngineConfiguration">

    ...

</bean>

<bean id="processEngine" class="org.activiti.spring.ProcessEngineFactoryBean">

  <property name="processEngineConfiguration" ref="processEngineConfiguration" />

</bean>

注意现在使用的 processEngineConfiguration bean 是 org.activiti.spring.SpringProcessEngineConfiguration 类。

**事务**

我们将会一步一步地解释在Spring examples中公布的 SpringTransactionIntegrationTest 下面是我们使用这个例子的Spring配置文件（你可以在SpringTransactionIntegrationTest-context.xml找到它）以下展示的部分包括数据源（dataSource）， 事务管理器（transactionManager），流程引擎（processEngine）和Activiti引擎服务。

当把数据源（DataSource）传递给 SpringProcessEngineConfiguration （使用"dataSource"属性）之后，Activiti内部使用了一个org.springframework.jdbc.datasource.TransactionAwareDataSourceProxy代理来封装传递进来的数据源（DataSource）。 这样做是为了确保从数据源（DataSource）获取的SQL连接能够与Spring的事物结合在一起发挥得更出色。这意味它不再需要在你的Spring配置中代理数据源（dataSource）了。 然而它仍然允许你传递一个TransactionAwareDataSourceProxy到SpringProcessEngineConfiguration中。在这个例子中并不会发生多余的包装。

**为了确保在你的Spring配置中申明的一个TransactionAwareDataSourceProxy，你不能把使用它的应用交给Spring事物控制的资源。（例如 DataSourceTransactionManager 和JPATransactionManager需要非代理的数据源 ）**

<beans xmlns="http://www.springframework.org/schema/beans"

       xmlns:context="http://www.springframework.org/schema/context"

       xmlns:tx="http://www.springframework.org/schema/tx"

       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

                           http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-2.5.xsd

                           http://www.springframework.org/schema/tx      http://www.springframework.org/schema/tx/spring-tx-3.0.xsd">

  <bean id="dataSource" class="org.springframework.jdbc.datasource.SimpleDriverDataSource">

    <property name="driverClass" value="org.h2.Driver" />

    <property name="url" value="jdbc:h2:mem:activiti;DB\_CLOSE\_DELAY=1000" />

    <property name="username" value="sa" />

    <property name="password" value="" />

  </bean>

  <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

    <property name="dataSource" ref="dataSource" />

  </bean>

  <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" />

  </bean>

  <bean id="processEngine" class="org.activiti.spring.ProcessEngineFactoryBean">

    <property name="processEngineConfiguration" ref="processEngineConfiguration" />

  </bean>

  <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="historyService" factory-bean="processEngine" factory-method="getHistoryService" />

  <bean id="managementService" factory-bean="processEngine" factory-method="getManagementService" />

...

Spring配置文件的其余部分包含beans和我们将要在这个特有的例子中的配置：

<beans>

  ...

  <tx:annotation-driven transaction-manager="transactionManager"/>

  <bean id="userBean" class="org.activiti.spring.test.UserBean">

    <property name="runtimeService" ref="runtimeService" />

  </bean>

  <bean id="printer" class="org.activiti.spring.test.Printer" />

</beans>

首先使用任意的一种Spring创建应用上下文的方式创建其Spring应用上下文。在这个例子中你可以使用类路径下面的XML资源来配置我们的Spring应用上下文：

ClassPathXmlApplicationContext applicationContext =

    new ClassPathXmlApplicationContext("org/activiti/examples/spring/SpringTransactionIntegrationTest-context.xml");

或者, 如果它是一个测试的话:

@ContextConfiguration("classpath:org/activiti/spring/test/transaction/SpringTransactionIntegrationTest-context.xml")

然后我们就可以得到Activiti的服务beans并且调用该服务上面的方法。ProcessEngineFactoryBean将会对该服务添加一些额外的拦截器，在Activiti服务上面的方法使用的是 Propagation.REQUIRED事物语义。所以，我们可以使用repositoryService去部署一个流程，如下所示：

RepositoryService repositoryService = (RepositoryService) applicationContext.getBean("repositoryService");

String deploymentId = repositoryService

  .createDeployment()

  .addClasspathResource("org/activiti/spring/test/hello.bpmn20.xml")

  .deploy()

  .getId();

其他相同的服务也是同样可以这么使用。在这个例子中，Spring的事物将会围绕在userBean.hello()上，并且调用Activiti服务的方法也会加入到这个事物中。

UserBean userBean = (UserBean) applicationContext.getBean("userBean");

userBean.hello();

这个UserBean看起来像这样。记得在上面Spring bean的配置中我们把repositoryService注入到userBean中。

public class UserBean {

  /\*\* 由Spring注入 \*/

  private RuntimeService runtimeService;

  @Transactional

  public void hello() {

        //这里，你可以在你们的领域模型中做一些事物处理。

        //当在调用Activiti RuntimeService的startProcessInstanceByKey方法时，

        //它将会结合到同一个事物中。

    runtimeService.startProcessInstanceByKey("helloProcess");

  }

  public void setRuntimeService(RuntimeService runtimeService) {

    this.runtimeService = runtimeService;

  }

}

**表达式**

当使用ProcessEngineFactoryBean时候，默认情况下，在BPMN流程中的所有[表达式](http://www.mossle.com/docs/activiti/#apiExpressions)都将会'看见'所有的Spring beans。 它可以限制你在表达式中暴露出的beans或者甚至可以在你的配置中使用一个Map不暴露任何beans。下面的例子暴露了一个单例bean（printer），可以把"printer"当作关键字使用. **想要不暴露任何beans，仅仅只需要在SpringProcessEngineConfiguration中传递一个空的list作为'beans'的属性。当不设置'beans'的属性时，在应用上下文中Spring beans都是可以使用的。**

<bean id="processEngineConfiguration" class="org.activiti.spring.SpringProcessEngineConfiguration">

  ...

  <property name="beans">

    <map>

      <entry key="printer" value-ref="printer" />

    </map>

  </property>

</bean>

  <bean id="printer" class="org.activiti.examples.spring.Printer" />

现在暴露出来的beans就可以在表达式中使用：例如，在SpringTransactionIntegrationTest中的 hello.bpmn20.xml展示的是如何使用UEL方法表达式去调用Spring bean的方法：

<definitions id="definitions" ...>

  <process id="helloProcess">

    <startEvent id="start" />

    <sequenceFlow id="flow1" sourceRef="start" targetRef="print" />

    <serviceTask id="print" **activiti:expression="#{printer.printMessage()}"** />

    <sequenceFlow id="flow2" sourceRef="print" targetRef="end" />

    <endEvent id="end" />

  </process>

</definitions>

这里的 Printer 看起来像这样：

public class Printer {

  public void printMessage() {

    System.out.println("hello world");

  }

}

并且Spring bean的配置（如上文所示）看起来像这样：

<beans ...>

  ...

  <bean id="printer" class="org.activiti.examples.spring.Printer" />

</beans>

**资源的自动部署**

Spring的集成也有一个专门用于对资源部署的特性。在流程引擎的配置中，你可以指定一组资源。当流程引擎被创建的时候， 所有在这里的资源都将会被自动扫描与部署。在这里有过滤以防止资源重新部署，只有当这个资源真正发生改变的时候，它才会向Activiti使用的数据库创建新的部署。 这对于很多用例来说，当Spring容器经常重启的情况下（例如 测试），使用它是非常不错的选择。

这里有一个例子：

<bean id="processEngineConfiguration" class="org.activiti.spring.SpringProcessEngineConfiguration">

  ...

**<property name="deploymentResources" value="classpath\*:/org/activiti/spring/test/autodeployment/autodeploy.\*.bpmn20.xml" />**

</bean>

<bean id="processEngine" class="org.activiti.spring.ProcessEngineFactoryBean">

  <property name="processEngineConfiguration" ref="processEngineConfiguration" />

</bean>

**单元测试**

当集成Spring时，使用标准的[Activiti测试工具类](http://www.mossle.com/docs/activiti/#apiUnitTesting)是非常容易的对业务流程进行测试。 下面的例子展示了如何在一个典型的基于Spring单元测试测试业务流程：

@RunWith(SpringJUnit4ClassRunner.class)

@ContextConfiguration("classpath:org/activiti/spring/test/junit4/springTypicalUsageTest-context.xml")

public class MyBusinessProcessTest {

  @Autowired

  private RuntimeService runtimeService;

  @Autowired

  private TaskService taskService;

  @Autowired

  @Rule

  public ActivitiRule activitiSpringRule;

  @Test

  @Deployment

  public void simpleProcessTest() {

    runtimeService.startProcessInstanceByKey("simpleProcess");

    Task task = taskService.createTaskQuery().singleResult();

    assertEquals("My Task", task.getName());

    taskService.complete(task.getId());

    assertEquals(0, runtimeService.createProcessInstanceQuery().count());

  }

}

注意对于这种方式，你需要在Spring配置中（在上文的例子中它是自动注入的）定义一个*org.activiti.engine.test.ActivitiRule*bean

<bean id="activitiRule" class="org.activiti.engine.test.ActivitiRule">

  <property name="processEngine" ref="processEngine" />

</bean>

**Chapter 6. 部署**

**Table of Contents**

[业务文档](http://www.mossle.com/docs/activiti/#N1071A)

[编程式部署](http://www.mossle.com/docs/activiti/#N1072E)

[通过Activiti Explorer控制台部署](http://www.mossle.com/docs/activiti/#deploymentWithExplorer)

[外部资源](http://www.mossle.com/docs/activiti/#N10752)

[Java类](http://www.mossle.com/docs/activiti/#N10757)

[在流程中使用Spring beans](http://www.mossle.com/docs/activiti/#N10763)

[创建独立应用](http://www.mossle.com/docs/activiti/#N10775)

[流程定义的版本](http://www.mossle.com/docs/activiti/#versioningOfProcessDefinitions)

[提供流程图片](http://www.mossle.com/docs/activiti/#providingProcessDiagram)

[自动生成流程图片](http://www.mossle.com/docs/activiti/#generatingProcessDiagram)

[类别](http://www.mossle.com/docs/activiti/#deploymentCategory)

**业务文档**

为了部署流程，它们不得不包装在一个业务文档中。一个业务文档是Activiti引擎部署的单元。一个业务文档相当与一个压缩文件，它包含BPMN2.0流程，任务表单，规则和其他任意类型的文件。 大体上，业务文档是包含命名资源的容器。

当一个业务文档被部署，它将会自动扫描以 .bpmn20.xml 或者.bpmn作为扩展名的BPMN文件。每个那样的文件都将会被解析并且可能会包含多个流程定义。

**Note**

业务归档中的Java类**将不能够添加到类路径下**。为了能够让流程运行，必须把存在于业务归档程中的流程定义使用的所有自定义的类（例如：Java服务任务或者实现事件的监听器）放在activiti引擎的类路径下：

**编程式部署**

通过一个压缩文件（支持Zip和Bar）部署业务归档，它看起来像这样：

String barFileName = "path/to/process-one.bar";

ZipInputStream inputStream = new ZipInputStream(new FileInputStream(barFileName));

repositoryService.createDeployment()

    .name("process-one.bar")

    .addZipInputStream(inputStream)

    .deploy();

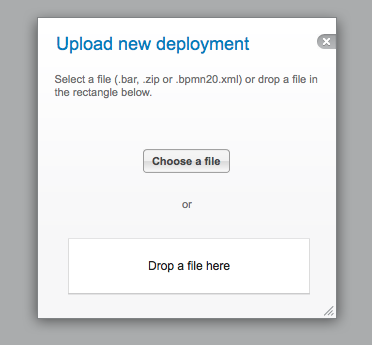
它也可以通过一个独立资源（例如bpmn，xml等）构建部署。 详细信息请查看javadocs。

**通过Activiti Explorer控制台部署**

Activiti web控制台允许你通过web界面的用户接口上传一个bar格式的压缩文件（或者一个bpmn20.xml格式的文件）。 选择*Management 标签* 和 点击 *Deployment*:



现在将会有一个弹出窗口允许你从电脑上面选择一个文件，或者你可以简单的拖拽到指定的区域（如果你的浏览器支持）。



**外部资源**

流程定义保存在Activiti所支持的数据库中。当使用服务任务、执行监听器或者从Activiti配置文件中配置的Spring beans时，流程定义能够引用这些委托类。 这些类或者Spring配置文件对于所有流程引擎中可能执行的流程定义必须是可用的。

**Java类**

当流程实例被启动的时候，在流程中被使用的所有自定义类（例如：服务任务中使用的JavaDelegates、事件监听器、任务监听器,...）应该存在与流程引擎的类路径下。

然后，在部署业务文档时，这些类不必都存在于类路径下。当使用Ant部署一个新的业务文档时，这意味着你的委托类不必存在与类路径下。

当你使用示例设置并添加你自定义的类，你应该添加包含自定义类的jar包到activiti-explorer控制台或者activiti-rest 的webapp lib文件夹中。以及不要忽略包含你自定义类的依赖关系（如果有）。 另外，你还可以包含你自己的依赖添加到你的Tomcat容器的安装目录中的${tomcat.home}/lib。

**在流程中使用Spring beans**

当表达式或者脚本使用Spring beans时，这些beans对于引擎执行流程定义时必须是可用的。如果你将要构建你自己的web应用并且按照[Spring集成这一章](http://www.mossle.com/docs/activiti/#springintegration)中描述那样在你的应用上下文配置流程引擎，这个看上去非常的简单。但是要记住，如果你也在使用 Activiti rest web应用，那么也应该更新 Activiti rest web应用的上下文。 你可以把在activiti-rest/lib/activiti-cfg.jar 文件中的activiti.cfg.xml替换成你的Spring上下文配置的activiti-context.xml文件。

**创建独立应用**

你可以考虑把Activiti rest web 应用加入到你的web应用之中，因此，就仅仅只需要配置一个 ProcessEngine，从而不用确保所有的流程引擎的所有委托类在类路径下面并且是否使用正确的spring配置。

**流程定义的版本**

BPMN中并没有版本的概念，没有版本也是不错的，因为可执行的BPMN流程作为你开发项目的一部分存在版本控制系统的知识库中（例如 SVN,Git 或者Mercurial）。 而在Activiti中，流程定义的版本是在部署时创建的。在部署的时候，流程定义被存储到Activiti使用的数据库之前，Activiti讲会自动给 流程定义 分配一个版本号。

对于业务文档中每一个的流程定义，都会通过下列部署执行初始化属性key, version, name 和 id:

* XML文件中流程定义（流程模型）的 id属性被当做是流程定义的 key属性。
* XML文件中的流程模型的name 属性被当做是流程定义的 name 属性。如果该name属性并没有指定，那么id属性被当做是name。
* 带有特定key的流程定义在第一次部署的时候，将会自动分配版本号为1，对于之后部署相同key的流程定义时候，这次部署的版本号将会设置为比当前最大的版本号大1的值。该key属性被用来区别不同的流程定义。
* 流程定义中的id属性被设置为 {processDefinitionKey}:{processDefinitionVersion}:{generated-id}, 这里的generated-id是一个唯一的数字被添加，用于确保在集群环境中缓存的流程定义的唯一性。

举个流程的例子

<definitions id="myDefinitions" >

  <process id="myProcess" name="My important process" >

    ...

当部署了这个流程定义之后，在数据库中的流程定义看起来像这样：

**Table 6.1.**

| id | key | name | version |
| --- | --- | --- | --- |
| myProcess:1:676 | myProcess | My important process | 1 |

假设我们现在部署用一个流程的最新版本号（例如 改变用户任务），但是流程定义的id保持不变。 流程定义表将包含以下列表信息：

**Table 6.2.**

| id | key | name | version |
| --- | --- | --- | --- |
| myProcess:1:676 | myProcess | My important process | 1 |
| myProcess:2:870 | myProcess | My important process | 2 |

当 runtimeService.startProcessInstanceByKey("myProcess")方法被调用时，它将会使用流程定义版本号为2的，因为这是最新版本的流程定义。可以说每次流程定义创建流程实例时，都会默认使用最新版本的流程定义。

我们应该创建第二个流程，在Activiti中，如下,定义并且部署它，该流程定义会添加到流程定义表中。

<definitions id="myNewDefinitions" >

  <process id="myNewProcess" name="My important process" >

    ...

这个表结构看起来像这样:

**Table 6.3.**

| id | key | name | version |
| --- | --- | --- | --- |
| myProcess:1:676 | myProcess | My important process | 1 |
| myProcess:2:870 | myProcess | My important process | 2 |
| myNewProcess:1:1033 | myNewProcess | My important process | 1 |

注意：为何新流程的key与我们的第一个流程是不同的？尽管流程定义的名称是相同的（当然，我们应该也是可以改变这一点的），Activiti仅仅只考虑id属性判断流程。因此，新的流程定义部署的版本号为1。

**提供流程图片**

流程定义的流程图可以被添加到部署中，该流程图将会持久化到Activiti所使用的数据库中并且可以通过Activiti的API进行访问。该流程图也可以被用来在Activiti Explorer控制台中的流程中进行显示。

如果在我们的类路径下面有一个流程，org/activiti/expenseProcess.bpmn20.xml ，该流程定义有一个流程key 'expense'。 以下遵循流程定义图片的命名规范（按照这个特地顺序）：

* 如果在部署时一个图片资源已经存在，它是BPMN2.0的XML文件名后面是流程定义的key并且是一个图片的后缀。那么该图片将被使用。在我们的例子中， 这应该是 org/activiti/expenseProcess.expense.png（或者 jpg/gif）。如果你在一个BPMN2.0 XML文件中定义多个流程定义图片，这种方式更有意义。每个流程定义图片的文件名中都将会有一个流程定义key。
* 如果并没有这样的图片存在，部署的时候寻找与匹配BPMN2.0 XML 文件的名称的图片资源。在我们的例子中，这应该是org/activiti/expenseProcess.png. 注意：这意味着在同一个BPMN2.0 XML文件夹中的**每个流程定义**都会有相同的流程定义图片。因此，在每一个BPMN 2.0 XML文件夹中仅仅只有一个流程定义，这绝对是不会有问题的。

当使用编程式的部署方式：

repositoryService.createDeployment()

  .name("expense-process.bar")

  .addClasspathResource("org/activiti/expenseProcess.bpmn20.xml")

  .addClasspathResource("org/activiti/expenseProcess.png")

  .deploy();

接下来，可以通过API来获取流程定义图片资源：

  ProcessDefinition processDefinition = repositoryService.createProcessDefinitionQuery()

                                                         .processDefinitionKey("expense")

                                                         .singleResult();

**String diagramResourceName = processDefinition.getDiagramResourceName();**

  InputStream imageStream = repositoryService.getResourceAsStream(processDefinition.getDeploymentId(), diagramResourceName);

**自动生成流程图片**

在部署的情况下没有提供图片，在 [上一节](http://www.mossle.com/docs/activiti/#providingProcessDiagram)中描述,如果流程定义中包含必要的'图像交换'信息时，Activiti流程引擎竟会自动生成一个图像。

该资源可以按照部署时 [提供流程图片](http://www.mossle.com/docs/activiti/#providingProcessDiagram)完全相同的方式获取。



如果，因为某种原因，在部署的时候，并不需要或者不必要生成流程定义图片，那么就需要在流程引擎配置的属性中使用isCreateDiagramOnDeploy：

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

现在就不会生成流程定义图片。

**类别**

部署和流程定义都是用户定义的类别。流程定义类别在BPMN文件中属性的初始化的值<definitions ... targetNamespace="yourCategory" ...

部署类别是可以直接使用API进行指定的看起来想这样：

repositoryService

    .createDeployment()

    .category("yourCategory")

    ...

    .deploy();

**Chapter 7. BPMN 2.0介绍**

**Table of Contents**

[啥是BPMN？](http://www.mossle.com/docs/activiti/#whatIsBpmn)

[定义一个流程](http://www.mossle.com/docs/activiti/#bpmnDefiningProcess)

[快速起步：10分钟教程](http://www.mossle.com/docs/activiti/#10minutetutorial)

[前提](http://www.mossle.com/docs/activiti/#bpmn10MinuteTutorialPrerequisites)

[目标](http://www.mossle.com/docs/activiti/#bpmn10MinuteTutorialGoal)

[用例](http://www.mossle.com/docs/activiti/#bpmnFirstExampleUseCase)

[流程图](http://www.mossle.com/docs/activiti/#bpmnFirstExampleDiagram)

[XML内容](http://www.mossle.com/docs/activiti/#bpmnFirstExampleXml)

[启动一个流程实例](http://www.mossle.com/docs/activiti/#bpmnFirstExamplStartProcess)

[任务列表](http://www.mossle.com/docs/activiti/#bpmnFirstExampleCandidateList)

[领取任务](http://www.mossle.com/docs/activiti/#bpmnFirstExampleClaimTask)

[完成任务](http://www.mossle.com/docs/activiti/#bpmnFirstExampleCompleteTask)

[结束流程](http://www.mossle.com/docs/activiti/#bpmnFirstExampleEndingProcess)

[代码总结](http://www.mossle.com/docs/activiti/#bpmnFirstExampleCode)

[更多思考](http://www.mossle.com/docs/activiti/#bpmnFirstExampleFutureEnhancements)

**啥是BPMN？**

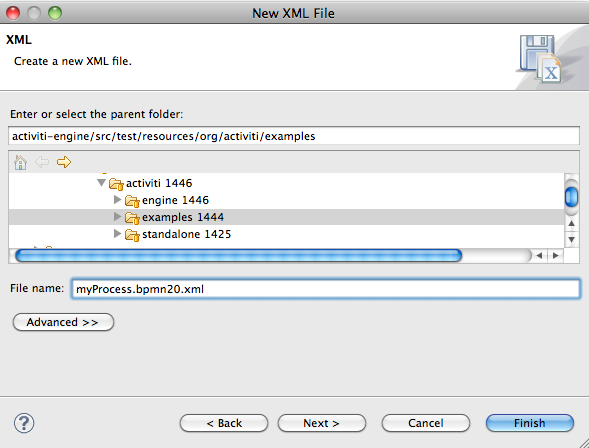
参考我们的[FAQ中的BPMN 2.0部分](http://activiti.org/faq.html#WhatIsBpmn20)。

**定义一个流程**

**Note**

文章假设你在使用[Eclipse IDE](http://eclipse.org/)来创建和编辑文件。 不过，其中只用到了Eclipse很少的特性。你可以使用喜欢的任何工具来创建包含BPMN 2.0的xml文件。

创建一个新的XML文件（*右击任何项目选择“新建”->“其他”->“XML-XML文件”*）并命名。 确认文件后缀为**ends with .bpmn20.xml 或 .bpmn**， 否则引擎无法发布。



BPMN 2.0根节点是definitions与阿奴。 这个元素中，可以定义多个流程定义（不过我们建议每个文件只包含一个流程定义， 可以简化开发过程中的维护难度）。 一个空的流程定义看起来像下面这样。注意，definitions元素 最少也要包含xmlns 和 targetNamespace的声明。 targetNamespace可以是任意值，它用来对流程实例进行分类。

<definitions

  xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"

  xmlns:activiti="http://activiti.org/bpmn"

  targetNamespace="Examples">

  <process id="myProcess" name="My First Process">

    ..

  </process>

</definitions>

你也可以选择添加线上的BPMN 2.0格式位置， 下面是ecilpse中的xml配置。

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL

                    http://www.omg.org/spec/BPMN/2.0/20100501/BPMN20.xsd

process元素有两个属性：

* **id**：这个属性是**必须的**， 它对应着Activiti ProcessDefinition对象的**key**属性。 ‘id可以用来启动流程定义的流程实例， 通过RuntimeService的startProcessInstanceByKey方法。 这个方法会一直使用**最新发布版本**的流程定义。

ProcessInstance processInstance = runtimeService.startProcessInstanceByKey("myProcess");

注意，它和startProcessInstanceById方法不同。 这个方法期望使用Activiti引擎在发布时自动生成的id。， 可以通过调用processDefinition.getId()方法获得这个值。 生成的id的格式为**'key:version'**， 最大长度限制为**64个字符**， 如果你在启动时抛出了一个ActivitiException，说明生成的id太长了， 需要限制流程的*key*的长度。

* **name**：这个属性是**可选的**， 对应ProcessDefinition的*name*属性。 引擎自己不会使用这个属性，它可以用来在用户接口显示便于阅读的名称。

**快速起步：10分钟教程**

这张我们会演示一个（非常简单）的业务流程，我们会通过它介绍 一些基本的Activiti概念和API。

**前提**

教程假设你已经[能安装并运行Activiti demo](http://www.mossle.com/docs/activiti/#demo.setup.one.minute.version)

**目标**

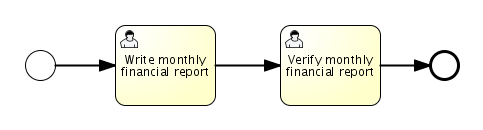
教程的目标是学习Activiti和一些基本的BPMN 2.0概念。 最终结果是一个简单的Java SE程序可以发布流程定义， 通过Activiti引擎API操作流程。 我们也会使用一些Activiti相关的工具。当然，我们在教程中所学的 也可以用于你构建自己的业务流程web应用。

**用例**

用例很直接：我们有一个公司，就叫BPMCorp。 在BPMCopr中，每个月都要给公司领导一个金融报表。 由会计部门负责。 当报表完成时，一个上级领导需要审批文档， 然后才能发给所有领导。

**流程图**

上面描述的业务流程可以用[Activiti Designer](http://www.mossle.com/docs/activiti/#activitiDesigner) 进行可视化设计。 然后，为了这个教程，我们会手工编写XML，这样可以学到更多知识细节。 我们流程的图形化BPMN 2.0标记看起来像这样：



我们看到有[空开始事件](http://www.mossle.com/docs/activiti/#bpmnNoneStartEvent)（左侧圆圈）, 后面是两个[用户任务](http://www.mossle.com/docs/activiti/#bpmnUserTask)： *“制作月度财报”*和 *“验证月度财报”*，最后是 [空结束事件](http://www.mossle.com/docs/activiti/#bpmnNoneEndEvent)（右侧粗线圆圈）。

**XML内容**

业务流程的XML内容（*FinancialReportProcess.bpmn20.xml*）如下所示： 很容易找到流程的主要元素（点击链接可以了解BPMN 2.0结构的详细信息）：

* [（空）开始事件](http://www.mossle.com/docs/activiti/#bpmnNoneStartEvent) 是我们流程的*入口*。
* [用户任务](http://www.mossle.com/docs/activiti/#bpmnUserTask)是流程中与操作者相关的任务声明。 注意第一个任务分配给*accountancy*组， 第二个任务分配给*management*组。 参考[用户任务分配章节](http://www.mossle.com/docs/activiti/#bpmnUserTaskAssignment) 了解更多关于用户任务分配人员和群组的问题。
* 当流程达到[空结束事件](http://www.mossle.com/docs/activiti/#bpmnNoneEndEvent)就会结束。
* 这些元素都使用[连线](http://www.mossle.com/docs/activiti/#bpmnSequenceFlow)连接。 这些连线拥有source 和 target属性， 定义了连线的*方向*。

<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>

**启动一个流程实例**

现在我们创建好了业务流程的**流程定义**。 有了这个流程定义，我们可以创建**流程实例**了。 这时，一个流程实例对应了特定月度财报的创建和审批。 所有流程实例都共享同一个流程定义。

为了使用流程定义创建流程实例， 首先要**发布**业务流程， 这意味着两方面：

* 流程定义会保存到持久化的数据存储里， 是为你的Activiti引擎特别配置。所以部署好你的业务流程， 我们就能确认引擎重启后还能找到流程定义。
* BPMN 2.0流程文件会解析成内存对象模型， 可以通过Activiti API操作。

可以通过[发布章节](http://www.mossle.com/docs/activiti/#chDeployment)获得关于发布的更多信息。

就像章节里描述的一样，有很多种方式可以进行发布。 一种方式是通过下面的API。注意所有与Activiti引擎的交互都是通过*services*。

Deployment deployment = repositoryService.createDeployment()

  .addClasspathResource("FinancialReportProcess.bpmn20.xml")

  .deploy();

现在我们可以启动一个新流程实例， 使用我们定义在流程定义里的id（对应XML文件中的process元素）。 注意这里的id对于Activiti来说， 应该叫做**key**。

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

这会创建一个流程实例，首先进入开始事件。 开始事件之后，它会沿着所有的外出连线（这里只有一条）执行， 到达第一个任务（“制作月度财报”）。 Activiti会把一个任务保存到数据库里。 这时，分配到这个任务的用户或群组会被解析，也会保存到数据库里。 需要注意，Activiti引擎会继续执行流程的环节，除非遇到一个 *等待状态*，比如用户任务。 在等待状态下，当前的流程实例的状态会保存到数据库中。 直到用户决定完成任务才能改变这个状态。这时，引擎会继续执行， 直到遇到下一个等待状态，或流程结束。 如果中间引擎重启或崩溃， 流程状态也会安全的保存在数据库里。

任务创建之后，startProcessInstanceByKey会在到达用户任务 这个*等待状态*之后才会返回。这时，任务分配给了一个组， 这意味着这个组是执行这个任务的**候选**组。

我们现在把所有东西都放在一起，来创建一个简单的java程序。 创建一个eclipse项目，把Activiti的jar和依赖放到classpath下。 （这些都可以在Activiti发布包的*libs*目录下找到）。 在调用Activiti服务之前，我们必须构造一个ProcessEngine， 它可以让我们访问服务。这里我们使用*'单独运行'*的配置， 这会使用demo安装时的数据库来构建ProcessEngine。

你可以在[这里](http://www.mossle.com/docs/activiti/images/FinancialReportProcess.bpmn20.xml)下载流程定义XML。 这个文件包含了上面介绍的XML，也包含了必须的BPMN[图像信息](http://www.mossle.com/docs/activiti/#generatingProcessDiagram) 以便在Activiti工具中能编辑流程。

public static void main(String[] args) {

  // Create Activiti process engine

  ProcessEngine processEngine = ProcessEngineConfiguration

    .createStandaloneProcessEngineConfiguration()

    .buildProcessEngine();

  // Get Activiti services

  RepositoryService repositoryService = processEngine.getRepositoryService();

  RuntimeService runtimeService = processEngine.getRuntimeService();

  // Deploy the process definition

  repositoryService.createDeployment()

    .addClasspathResource("FinancialReportProcess.bpmn20.xml")

    .deploy();

  // Start a process instance

  runtimeService.startProcessInstanceByKey("financialReport");

}

**任务列表**

我们现在可以通过TaskService来获得任务了，添加以下逻辑：

List<Task> tasks = taskService.createTaskQuery().taskCandidateUser("kermit").list();

注意我们传入的用户必须是*accountancy*组的一个成员， 要和流程定义中向对应：

<potentialOwner>

  <resourceAssignmentExpression>

    <formalExpression>**accountancy**</formalExpression>

  </resourceAssignmentExpression>

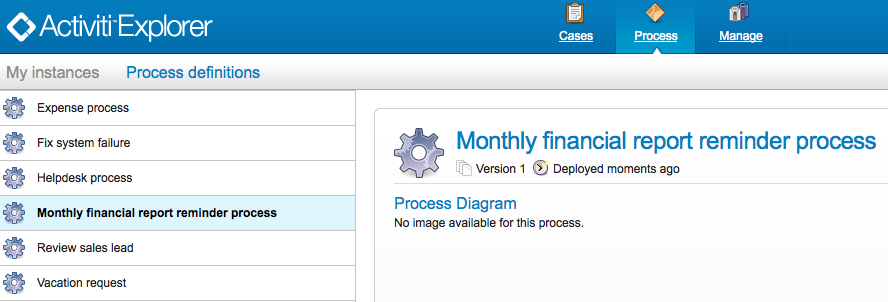
</potentialOwner>

我们也可以使用群组名称，通过任务查询API来获得相关的结果。 现在可以在代码中添加如下逻辑：

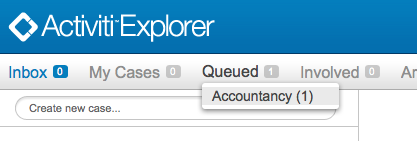
TaskService taskService = processEngine.getTaskService();

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

因为我们配置的ProcessEngine使用了与demo相同的数据， 我们可以登录到[Activiti Explorer](http://localhost:8080/activiti-explorer/) （使用fozzie//fozzie登录），然后可以启动业务流程， 选择*Processes*页，在*'月度财报'*的*'操作'*列 点击*'启动流程'*。



和上面介绍的那样，流程会执行到第一个用户任务。因为我们以kermit登录， 在启动流程实例之后，就可以看到有了一个新的待领任务。 选择*任务*页来查看这条新任务。 注意即使流程被其他人启动，任务还是会被会计组里的所有人作为一个候选任务看到。



**领取任务**

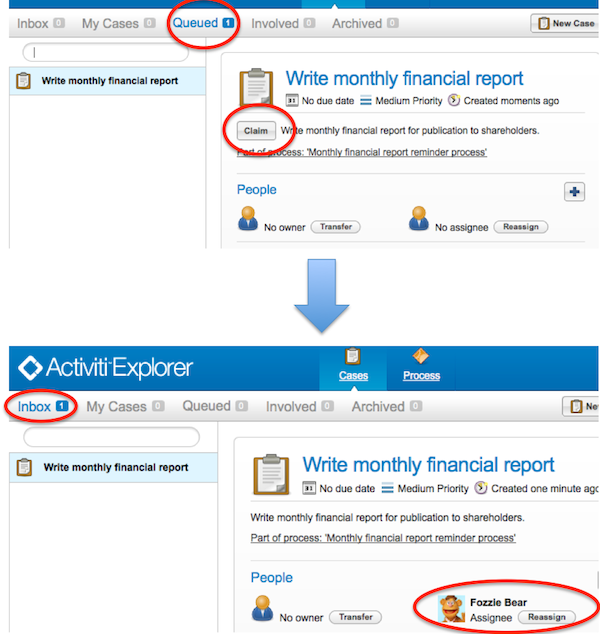
现在一个会计要**认领这个任务**。 认领以后，这个用户就会成为任务的**执行人** ， 任务会从会计组的其他成员的任务列表中消失。 认领任务的代码如下所示：

taskService.claim(task.getId(), "fozzie");

任务会进入**认领任务人的个人任务列表**中。

List<Task> tasks = taskService.createTaskQuery().taskAssignee("fozzie").list();

在Activiti Explorer UI中，点击*认领*按钮，会执行相同的操作。 任务会移动到登录用户的个人任务列表。 你也会看到任务的执行人已经变成当前登陆的用户。



**完成任务**

现在会计可以开始进行财报的工作了。报告完成后， 他可以**完成任务**， 意味着任务所需的所有工作都完成了。

taskService.complete(task.getId());

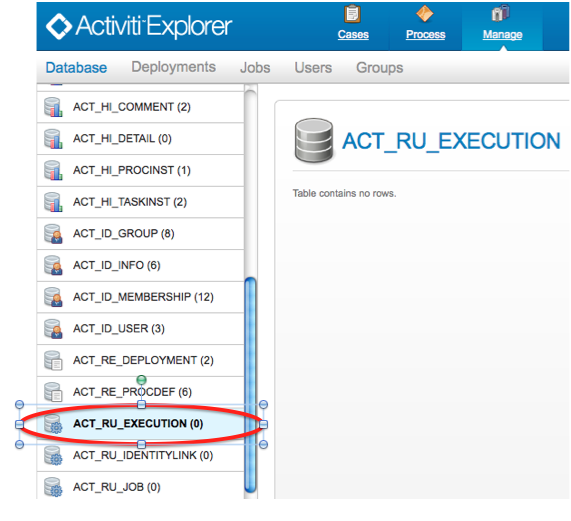
对于Activiti引擎，需要一个外部信息来让流程实例继续执行。 任务会把自己从运行库中删除。 流程会沿着单独一个外出连线执行，移动到第二个任务 （*'审批报告'*）。 与第一个任务相同的机制会使用到第二个任务上， 不同的是任务是分配给 *management*组。

在demo中，完成任务是通过点击任务列表中的*完成*按钮。 因为Fozzie不是会计，我们先从Activiti Explorer注销 然后使用*kermit*登陆（他是经理）。 第二个任务会进入未分配任务列表。

**结束流程**

审批任务可以像之前介绍的一样查询和领取。 完成第二个任务会让流程执行到结束事件，就会结束流程实例。 流程实例和所有相关的运行数据都会从数据库中删除。

登录Activiti Explorer就可以进行验证， 可以看到保存流程运行数据的表中已经没有数据了。



通过程序，你也可以使用historyService判断流程已经结束了。

HistoryService historyService = processEngine.getHistoryService();

HistoricProcessInstance historicProcessInstance =

historyService.createHistoricProcessInstanceQuery().processInstanceId(procId).singleResult();

System.out.println("Process instance end time: " + historicProcessInstance.getEndTime());

**代码总结**

把上述代码组合在一起，获得的代码如下所示 （这些代码考虑到你可能会在Activiti Explorer UI中启动一些流程实例。 这样，它会获得多个任务，而不是一个， 所以代码可以一直正常运行）：

public class TenMinuteTutorial {

  public static void main(String[] args) {

    // Create Activiti process engine

    ProcessEngine processEngine = ProcessEngineConfiguration

      .createStandaloneProcessEngineConfiguration()

      .buildProcessEngine();

    // Get Activiti services

    RepositoryService repositoryService = processEngine.getRepositoryService();

    RuntimeService runtimeService = processEngine.getRuntimeService();

    // Deploy the process definition

    repositoryService.createDeployment()

      .addClasspathResource("FinancialReportProcess.bpmn20.xml")

      .deploy();

    // Start a process instance

    String procId = runtimeService.startProcessInstanceByKey("financialReport").getId();

    // Get the first task

    TaskService taskService = processEngine.getTaskService();

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

    for (Task task : tasks) {

      System.out.println("Following task is available for accountancy group: " + task.getName());

      // claim it

      taskService.claim(task.getId(), "fozzie");

    }

    // Verify Fozzie can now retrieve the task

    tasks = taskService.createTaskQuery().taskAssignee("fozzie").list();

    for (Task task : tasks) {

      System.out.println("Task for fozzie: " + task.getName());

      // Complete the task

      taskService.complete(task.getId());

    }

    System.out.println("Number of tasks for fozzie: "

            + taskService.createTaskQuery().taskAssignee("fozzie").count());

    // Retrieve and claim the second task

    tasks = taskService.createTaskQuery().taskCandidateGroup("management").list();

    for (Task task : tasks) {

      System.out.println("Following task is available for accountancy group: " + task.getName());

      taskService.claim(task.getId(), "kermit");

    }

    // Completing the second task ends the process

    for (Task task : tasks) {

      taskService.complete(task.getId());

    }

    // verify that the process is actually finished

    HistoryService historyService = processEngine.getHistoryService();

    HistoricProcessInstance historicProcessInstance =

      historyService.createHistoricProcessInstanceQuery().processInstanceId(procId).singleResult();

    System.out.println("Process instance end time: " + historicProcessInstance.getEndTime());

  }

}

这段代码包含在实例中的一个单元测试中（似的，你可以运行单元测试来测试你的流程。 参考[单元测试章节](http://www.mossle.com/docs/activiti/#apiUnitTesting)来了解更多信息）。

**更多思考**

可以看到业务流程相对于现实来说太简单了。 然而，你可以了解Activiti中的BPMN 2.0结构， 你可以考虑对业务流程进行以下方面的加强：

* 定义**网关**来实现决策环节。 这样，经理可以驳回财报， 重新给会计创建一个任务。
* 考虑使用**变量**， 这样我们可以保存或引用报告， 把它显示到表单中。
* 在流程最后加入**服务任务**， 把报告发给每个领导。
* 其他

**Chapter 8. BPMN 2.0结构**

**Table of Contents**

[自定义扩展](http://www.mossle.com/docs/activiti/#bpmnCustomExtensions)

[事件（Event）](http://www.mossle.com/docs/activiti/#bpmnEvents)

[事件定义](http://www.mossle.com/docs/activiti/#eventDefinitions)

[定时器事件定义](http://www.mossle.com/docs/activiti/#timerEventDefinitions)

[错误事件定义](http://www.mossle.com/docs/activiti/#bpmnErrorEventDefinition)

[信号事件定义](http://www.mossle.com/docs/activiti/#bpmnSignalEventDefinition)

[消息事件定义](http://www.mossle.com/docs/activiti/#bpmnMessageEventDefinition)

[开始事件](http://www.mossle.com/docs/activiti/#bpmnStartEvents)

[空开始事件](http://www.mossle.com/docs/activiti/#bpmnNoneStartEvent)

[定时开始事件](http://www.mossle.com/docs/activiti/#bpmnTimerStartEvent)

[消息开始事件](http://www.mossle.com/docs/activiti/#bpmnMessageStartEvent)

[错误开始事件](http://www.mossle.com/docs/activiti/#bpmnErrorStartEvent)

[结束事件](http://www.mossle.com/docs/activiti/#bpmnEndEvent)

[空结束事件](http://www.mossle.com/docs/activiti/#bpmnNoneEndEvent)

[错误结束事件](http://www.mossle.com/docs/activiti/#bpmnErrorEndEvent)

[取消结束事件](http://www.mossle.com/docs/activiti/#bpmnCancelEndEvent)

[边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryEvent)

[定时边界事件](http://www.mossle.com/docs/activiti/#bpmnTimerBoundaryEvent)

[错误边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryErrorEvent)

[信号边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundarySignalEvent)

[消息边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryMessageEvent)

[取消边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryCancelEvent)

[补偿边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryCompensationEvent)

[中间捕获事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateCatchingEvent)

[定时中间捕获事件](http://www.mossle.com/docs/activiti/#bpmnTimerIntermediateEvent)

[信号中间捕获事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateSignalEvent)

[消息中间捕获事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateMessageEvent)

[内部触发事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateThrowEvent)

[中间触发空事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateThrowNoneEvent)

[信号中间触发事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateThrowSignalEvent)

[补偿中间触发事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateThrowCompensationEvent)

[顺序流](http://www.mossle.com/docs/activiti/#bpmnSequenceFlow)

[描述](http://www.mossle.com/docs/activiti/#sequenceFlowDescription)

[图形标记](http://www.mossle.com/docs/activiti/#sequenceFlowGraphicalNotation)

[XML内容](http://www.mossle.com/docs/activiti/#sequenceFlowXml)

[条件顺序流](http://www.mossle.com/docs/activiti/#bpmnConditionalSequenceFlow)

[默认顺序流](http://www.mossle.com/docs/activiti/#bpmnDefaultSequenceFlow)

[网关](http://www.mossle.com/docs/activiti/#bpmnGateways)

[排他网关](http://www.mossle.com/docs/activiti/#bpmnExclusiveGateway)

[并行网关](http://www.mossle.com/docs/activiti/#bpmnParallelGateway)

[包含网关](http://www.mossle.com/docs/activiti/#bpmnInclusiveGateway)

[基于事件网关](http://www.mossle.com/docs/activiti/#bpmnEventbasedGateway)

[任务](http://www.mossle.com/docs/activiti/#bpmnTask)

[用户任务](http://www.mossle.com/docs/activiti/#bpmnUserTask)

[脚本任务](http://www.mossle.com/docs/activiti/#bpmnScriptTask)

[Java服务任务](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTask)

[Web Service Task](http://www.mossle.com/docs/activiti/#bpmnWebserviceTask)

[Business Rule Task](http://www.mossle.com/docs/activiti/#bpmnBusinessRuleTask)

[Email Task](http://www.mossle.com/docs/activiti/#bpmnEmailTask)

[Mule Task](http://www.mossle.com/docs/activiti/#bpmnMuleTask)

[Camel Task](http://www.mossle.com/docs/activiti/#bpmnCamelTask)

[Manual Task](http://www.mossle.com/docs/activiti/#bpmnManualTask)

[Java Receive Task](http://www.mossle.com/docs/activiti/#bpmnReceiveTask)

[Shell Task](http://www.mossle.com/docs/activiti/#bpmnShellTask)

[Execution listener](http://www.mossle.com/docs/activiti/#executionListeners)

[Task listener](http://www.mossle.com/docs/activiti/#taskListeners)

[Multi-instance (for each)](http://www.mossle.com/docs/activiti/#bpmnMultiInstance)

[Compensation Handlers](http://www.mossle.com/docs/activiti/#bpmnCompensationHandlers)

[Sub-Processes and Call Activities](http://www.mossle.com/docs/activiti/#bpmnSubprocessAndCallActivity)

[Sub-Process](http://www.mossle.com/docs/activiti/#bpmnSubProcess)

[Event Sub-Process](http://www.mossle.com/docs/activiti/#bpmnEventSubprocess)

[Transaction subprocess](http://www.mossle.com/docs/activiti/#bpmnTransactionSubprocess)

[Call activity (subprocess)](http://www.mossle.com/docs/activiti/#bpmnCallActivity)

[Transactions and Concurrency](http://www.mossle.com/docs/activiti/#bpmnConcurrencyAndTransactions)

[Asynchronous Continuations](http://www.mossle.com/docs/activiti/#asyncContinuations)

[Exclusive Jobs](http://www.mossle.com/docs/activiti/#exclusiveJobs)

[Process Initiation Authorization](http://www.mossle.com/docs/activiti/#security)

本章介绍Activiti支持的BPMN 2.0结构， 以及对BPMN标准的扩展。

**自定义扩展**

BPMN 2.0标准对于各方都是一个好东西。自重用户不用担心会绑死在 供应商提供的专有解决方案上。 框架，特别是activiti这样的开源框架，可以提供相同功能 （甚至是更好的实现），足以和大的供应商媲美。 按照BPMN 2.0标准，从大供应商的解决方案迁移到activiti 只会经过一个简单而平滑的过程。

标准不好的一点是，它常常是不同公司之间大量讨论和妥协的结果。 （而且通常是愿景）。 作为开发者去阅读流程定义的BPMN 2.0 xml时，有时会感觉 用这种结构和方法去做事太麻烦了。 因此activiti把简化开发作为最优先的事情，我们会使用一些被称为 **'Activiti BPMN扩展'**的功能。 这些扩展是新的结构或方法来简化对应的结构， 它们并不属于BPMN 2.0规范。

虽然BPMN 2.0规范清楚的指明了如何开发自定义扩展， 但是我们还要确认一下几点：

* 自定义扩展的前提是 **总**有简单的方法 转换成**标准方法**。 所以当你决定使用自定义扩展时，不用担心没办法回头。
* 当使用自定义扩展时，总会清楚的指明使用了 新的XML元素，属性，等等。 比如会使用**activiti:**命名空间前缀。
* 这些扩展的目标是最终把它们加入到下一版本的BPMN规范中， 或者至少可以引起对特定BPMN结构的讨论。

因此无论是是否想要使用自定义扩展，这都取决于你。 很多因素会影响决定这个决定（图形编辑器，公司策略，等等）。 只是因为我们相信标准里的一些功能可以更简单或更高校， 所以才决定提供自定义扩展。请对扩展给予我们（正面或负面）的评价， 或者是对自定义扩展的心想法。 说不定有一天你的想法就会加入到规范中。

**事件（Event）**

事件用来表明流程的生命周期中发生了什么事。 事件总是画成一个圆圈。 在BPMN 2.0中， 事件有两大分类：*捕获（catching）* 或 *触发（throwing）* 事件。

* **捕获（Catching）：**当流程执行到事件， 它会等待被触发。触发的类型是由内部图表或XML中的类型声明来决定的。 捕获事件与触发事件在显示方面是根据内部图表是否被填充来区分的（白色的）。
* **触发（Throwing）：**当流程执行到事件， 会触发一个事件。触发的类型是由内部图表或XML中的类型声明来决定的。 触发事件与捕获事件在显示方面是根据内部图表是否被填充来区分的（被填充为黑色）。

**事件定义**

事件定义决定了事件的语义。如果没有事件定义，这个事件就不做什么特别的事情。 没有设置事件定义的开始事件不会在启动流程时做任何事情。如果给开始事件添加了一个事件定义 （比如定时器事件定义）我们就声明了开始流程的事件 "类型 " （这时定时器事件监听器会在某个时间被触发）。

**定时器事件定义**

定时器事件是根据指定的时间触发的事件。可以用于 [开始事件](http://www.mossle.com/docs/activiti/#bpmnTimerStartEvent)， [中间事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateCatchingEvent) 或 [边界事件](http://www.mossle.com/docs/activiti/#bpmnTimerBoundaryEvent)

定时器定义必须下面介绍的一个元素：

* **timeDate**。使用 [ISO 8601](http://en.wikipedia.org/wiki/ISO_8601#Dates) 格式指定一个确定的时间，触发事件的时间。示例：
* <timerEventDefinition>
* <timeDate>2011-03-11T12:13:14</timeDate>

</timerEventDefinition>

* **timeDuration**。指定定时器之前之前要等待多长时间， *timeDuration*可以设置为*timerEventDefinition*的子元素。 使用[ISO 8601](http://en.wikipedia.org/wiki/ISO_8601#Durations)规定的格式 （由BPMN 2.0规定）。示例（等待10天）。
* <timerEventDefinition>
* <timeDuration>P10D</timeDuration>

</timerEventDefinition>

* **timeCycle**。指定重复执行的间隔， 可以用来定期启动流程实例，或为超时时间发送多个提醒。 timeCycle元素可以使用两种格式。第一种是 [ISO 8601](http://en.wikipedia.org/wiki/ISO_8601#Repeating_intervals) 标准的格式。示例（重复3此，每次间隔10小时）：
* <timerEventDefinition>
* <timeCycle>R3/PT10H</timeCycle>

</timerEventDefinition>

另外，你可以使用cron表达式指定timeCycle，下面的例子是从整点开始，每5分钟执行一次：

0 0/5 \* \* \* ?

请参考[教程](http://www.quartz-scheduler.org/docs/tutorials/crontrigger.html) 来了解如何使用cron表达式。

**注意：** 第一个数字表示秒，而不是像通常Unix cron中那样表示分钟。

重复的时间周期能更好的处理相对时间，它可以计算一些特定的时间点 （比如，用户任务的开始时间），而cron表达式可以处理绝对时间 - 这对[定时启动事件](http://www.mossle.com/docs/activiti/#timerStartEventDescription)特别有用。

你可以在定时器事件定义中使用表达式，这样你就可以通过流程变量来英系那个定时器定义。 流程定义必须包含ISO 8601（或cron）格式的字符串，以匹配对应的时间类型。

  <boundaryEvent id="escalationTimer" cancelActivity="true" attachedToRef="firstLineSupport">

**<timerEventDefinition>**

**<timeDuration>${duration}</timeDuration>**

**</timerEventDefinition>**

  </boundaryEvent>

**注意：** 只有启用job执行器之后，定时器才会被触发。 （activiti.cfg.xml中的*jobExecutorActivate*需要设置为true， 不过，默认job执行器是关闭的）。

**错误事件定义**

错误事件是由指定错误触发的。

**重要提醒：**BPMN错误与Java异常完全不一样。 实际上，他俩一点儿共同点都没有。BPMN错误事件是为了对 *业务异常*建模。Java异常是要 [用特定方式](http://www.mossle.com/docs/activiti/#serviceTaskExceptionHandling)处理。

错误事件定义会引用一个error元素。下面是一个error元素的例子，引用了一个错误声明：

<endEvent id="myErrorEndEvent">

  <errorEventDefinition errorRef="myError" />

</endEvent>

引用相同error元素的错误事件处理器会捕获这个错误。

**信号事件定义**

信号事件会引用一个已命名的信号。信号全局范围的事件（广播语义）。 会发送给所有激活的处理器。

信号事件定义使用signalEventDefinition元素。 signalRef属性会引用definitions根节点里定义的signal子元素。 下面是一个流程的实例，其中会抛出一个信号，并被中间事件捕获。

<definitions... >

        <!-- declaration of the signal -->

        <signal id="alertSignal" name="alert" />

        <process id="catchSignal">

                <intermediateThrowEvent id="throwSignalEvent" name="Alert">

                        <!-- signal event definition -->

                        <signalEventDefinition signalRef="alertSignal" />

                </intermediateThrowEvent>

                ...

                <intermediateCatchEvent id="catchSignalEvent" name="On Alert">

                        <!-- signal event definition -->

                        <signalEventDefinition signalRef="alertSignal" />

                </intermediateCatchEvent>

                ...

        </process>

</definitions>

signalEventDefinition引用相同的signal元素。

**触发信号事件**

既可以通过bpmn节点由流程实例触发一个信号，也可以通过API触发。 下面的org.activiti.engine.RuntimeService中的方法 可以用来手工触发一个信号。

RuntimeService.signalEventReceived(String signalName);

RuntimeService.signalEventReceived(String signalName, String executionId);

signalEventReceived(String signalName);和 signalEventReceived(String signalName, String executionId);之间的区别是 第一个方法会把信号发送给全局所有订阅的处理器（广播语义）， 第二个方法只把信息发送给指定的执行。

**捕获信号事件**

信号事件可以被中间捕获信号事件或边界信息事件捕获。

**查询信号事件的订阅**

可以查询所有订阅了特定信号事件的执行：

 List<Execution> executions = runtimeService.createExecutionQuery()

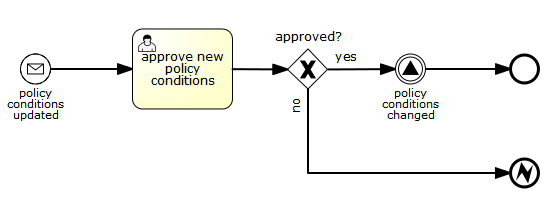
      .signalEventSubscriptionName("alert")

      .list();

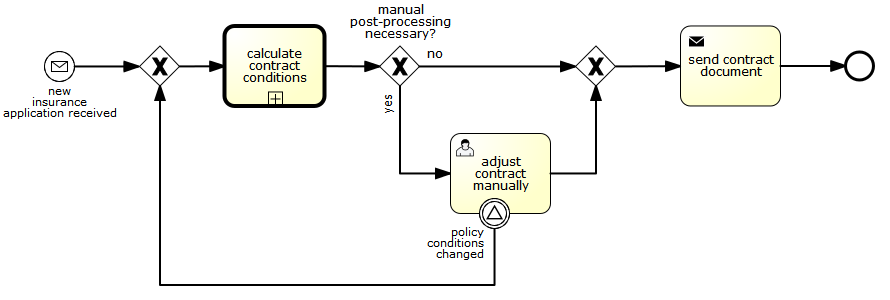
我们可以使用signalEventReceived(String signalName, String executionId)方法 吧信号发送给这些执行。

**信号事件实例**

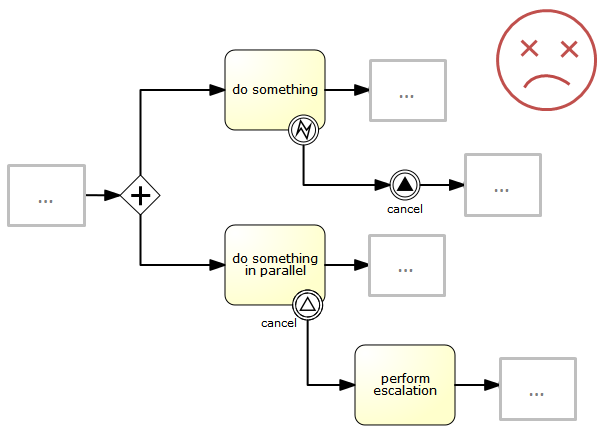
下面是两个不同流程使用信号交互的例子。第一个流程在保险规则更新或改变时启动。 在修改被参与者处理时，会触发一个信息，通知规则改变：



这个时间会被所有感兴趣的流程实例捕获。下面是一个订阅这个事件的流程实例。



**注意：**要了解信号事件是广播给**所有** 激活的处理器的。 这意味着在上面的例子中，所有流程实例都会接收到这个事件。 这就是我们想要的。然而，有的情况下并不想要这种广播行为。 考虑下面的流程：



上述流程描述的模式activiti并不支持。这种想法是执行“do something”任务时出现的错误，会被边界错误事件捕获， 然后使用信号传播给并发路径上的分支，进而中断"do something inparallel"任务。 目前，activiti实际运行的结果与期望一致。信号会传播给边界事件并中断任务。 **但是，根据信号的广播含义，它也会传播给所有其他订阅了信号事件的流程实例。** 所以，这就不是我们想要的结果。

**注意：** 信号事件不会执行任何与特定流程实例的联系。 如果你只想把一个信息发给指定的流程实例，需要手工关联，再使用 signalEventReceived(String signalName, String executionId)和对应的 [查询机制](http://www.mossle.com/docs/activiti/#bpmnSignalEventDefinitionQuery)。

**消息事件定义**

消息事件会引用一个命名的消息。每个消息都有名称和内容。和信号不同， 消息事件总会直接发送个一个接受者。

消息事件定义使用messageEventDefinition元素。 messaageRef属性引用了definitions根节点下的 一个message子元素。下面是一个使用两个消息事件的流程例子， 开始事件和中间捕获事件分别声明和引用了两个消息事件。

<definitions id="definitions"

  xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"

  xmlns:activiti="http://activiti.org/bpmn"

  targetNamespace="Examples"

  xmlns:tns="Examples">

  <message id="newInvoice" name="newInvoiceMessage" />

  <message id="payment" name="paymentMessage" />

  <process id="invoiceProcess">

    <startEvent id="messageStart" >

        <messageEventDefinition messageRef="newInvoice" />

    </startEvent>

    ...

    <intermediateCatchEvent id="paymentEvt" >

        <messageEventDefinition messageRef="payment" />

    </intermediateCatchEvent>

    ...

  </process>

</definitions>

**触发消息事件**

作为一个嵌入式的流程引擎，activiti不能真正接收一个消息。这些环境相关，与平台相关的活动 比如连接到JMS（Java消息服务）队列或主题或执行WebService或REST请求。 这个消息的接收是你要在应用或架构的一层实现的，流程引擎则内嵌其中。

在你的应用接收一个消息之后，你必须决定如何处理它。 如果消息应该触发启动一个新流程实例， 在下面的RuntimeService的两个方法中选择一个执行：

ProcessInstance startProcessInstanceByMessage(String messageName);

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

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

这些方法允许使用对应的消息系统流程实例。

如果消息需要被运行中的流程实例处理，首先要根据消息找到对应的流程实例 （参考下一节）然后触发这个等待中的流程。 RuntimeService提供了如下方法可以基于消息事件的订阅来触发流程继续执行：

void messageEventReceived(String messageName, String executionId);

void messageEventReceived(String messageName, String executionId, HashMap<String, Object> processVariables);

**查询消息事件的订阅**

Activiti支持消息开始事件和中间消息事件。

* 消息开始事件的情况，消息事件订阅分配给一个特定的 *process definition*。这个消息订阅可以使用ProcessDefinitionQuery查询到：
* ProcessDefinition processDefinition = repositoryService.createProcessDefinitionQuery()
* .messageEventSubscription("newCallCenterBooking")
* .singleResult();

因为同时只能有一个流程定义关联到消息的订阅点，查询总是返回0或一个结果。 如果流程定义更新了， 那么只有最新版本的流程定义会订阅到消息事件上。

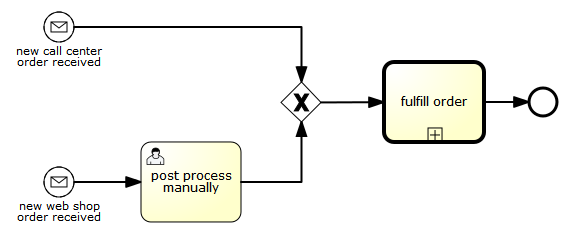
* 中间捕获消息事件的情况，消息事件订阅会分配给特定的*执行*。 这个消息事件订阅可以使用ExecutionQuery查询到：
* Execution execution = runtimeService.createExecutionQuery()
* .messageEventSubscriptionName("paymentReceived")
* .variableValueEquals("orderId", message.getOrderId())

      .singleResult();

这个查询可以调用对应的查询，通常是流程相关的信息 （这里，最多只能有一个流程实例对应着orderId）。

**消息事件实例**

下面是一个使用两个不同消息启动的流程实例：



可以用在，流程需要不同的方式来区分开始事件，而后最终会进入同样的路径。

**开始事件**

开始事件用来指明流程在哪里开始。开始事件的类型（流程在接收事件时启动， 还是在指定时间启动，等等），定义了流程*如何*启动， 这通过事件中不同的小图表来展示。 在XML中，这些类型是通过声明不同的子元素来区分的。

开始事件**都是捕获事件**： 最终这些事件都是（一直）等待着，直到对应的触发时机出现。

在开始事件中，可以设置下面的activiti特定属性：

* **initiator**：当流程启动时，把当前登录的用户保存到哪个变量名中。 示例如下：

<startEvent id="request" activiti:initiator="initiator" />

登录的用户必须使用IdentityService.setAuthenticatedUserId(String)方法设置， 并像这样包含在try-finally代码中：

try {

  identityService.setAuthenticatedUserId("bono");

  runtimeService.startProcessInstanceByKey("someProcessKey");

} finally {

  identityService.setAuthenticatedUserId(null);

}

这段代码来自Activiti Explorer，所以它可以和 [Chapter 9, *表单*](http://www.mossle.com/docs/activiti/#forms)一起结合使用。

**空开始事件**

**描述**

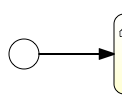
空开始事件技术上意味着没有指定启动流程实例的触发条件。 这就是说引擎不能预计什么时候流程实例会启动。 空开始事件用于，当流程实例要通过API启动的场景， 通过调用*startProcessInstanceByXXX*方法。

ProcessInstance processInstance = runtimeService.startProcessInstanceByXXX();

*注意：* 子流程都有一个空开始事件。

**图形标记**

空开始事件显示成一个圆圈，没有内部图表（没有触发类型）



**XML结构**

空开始事件的XML结构是普通的开始事件定义，没有任何子元素 （其他开始事件类型都有一个子元素来声明自己的类型）

<startEvent id="start" name="my start event" />

**空开始事件的自定义扩展**

**formKey**：引用用户在启动新流程实例时需要填写的表单模板， 更多信息可以参考[表单章节](http://www.mossle.com/docs/activiti/#forms)。 实例：

<startEvent id="request" activiti:formKey="org/activiti/examples/taskforms/request.form" />

**定时开始事件**

**描述**

定时开始事件用来在指定的时间创建流程实例。 它可以同时用于只启动一次的流程 和应该在特定时间间隔启动多次的流程。

*注意：*子流程不能使用定时开始事件。

*注意：*定时开始事件在流程发布后就会开始计算时间。 不需要调用startProcessInstanceByXXX，虽然也而已调用启动流程的方法， 但是那会导致调用startProcessInstanceByXXX时启动过多的流程。

*注意：*当包含定时开始事件的新版本流程部署时， 对应的上一个定时器就会被删除。这是因为通常不希望自动启动旧版本流程的流程实例。

**图形标记**

定时开始事件显示为了一个圆圈，内部是一个表。



**XML内容**

定时开始事件的XML内容是普通开始事件的声明，包含一个定时定义子元素。 请参考[定时定义](http://www.mossle.com/docs/activiti/#timerEventDefinitions) 查看配合细节。

示例：流程会启动4次，每次间隔5分钟，从2011年3月11日，12:13开始计时。

        <startEvent id="theStart">

            <timerEventDefinition>

                <timeCycle>R4/2011-03-11T12:13/PT5M</timeCycle>

            </timerEventDefinition>

        </startEvent>

示例：流程会根据选中的时间启动一次。

        <startEvent id="theStart">

            <timerEventDefinition>

                <timeDate>2011-03-11T12:13:14</timeDate>

            </timerEventDefinition>

        </startEvent>

**消息开始事件**

**描述**

[消息](http://www.mossle.com/docs/activiti/#bpmnMessageEventDefinition)开始事件可以用其使用一个命名的消息来启动流程实例。 这样可以帮助我们使用消息名称来*选择*正确的开始事件。

在**发布**包含一个或多个消息开始事件的流程定义时，需要考虑下面的条件：

* 消息开始事件的名称在给定流程定义中不能重复。流程定义不能包含多个名称相同的消息开始事件。 如果两个或以上消息开始事件应用了相同的事件，或两个或以上消息事件引用的消息名称相同，activiti会在发布流程定义时抛出异常。
* 消息开始事件的名称在所有已发布的流程定义中不能重复。 如果一个或多个消息开始事件引用了相同名称的消息，而这个消息开始事件已经部署到不同的流程定义中， activiti就会在发布时抛出一个异常。
* 流程版本：在发布新版本的流程定义时，之前订阅的消息订阅会被取消。 如果新版本中没有消息事件也会这样处理。

**启动**流程实例，消息开始事件可以使用 下列RuntimeService中的方法来触发：

ProcessInstance startProcessInstanceByMessage(String messageName);

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

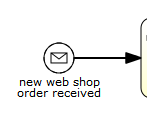
ProcessInstance startProcessInstanceByMessage(String messageName, String businessKey, Map<String, Object< processVariables);

这里的messageName是messageEventDefinition的messageRef属性引用的message元素的name属性。 **启动**流程实例时，要考虑一下因素：

* 消息开始事件只支持顶级流程。消息开始事件不支持内嵌子流程。
* 如果流程定义有多个消息开始事件，runtimeService.startProcessInstanceByMessage(...) 会选择对应的开始事件。
* 如果流程定义有多个消息开始事件和一个空开始事件。 runtimeService.startProcessInstanceByKey(...)和 runtimeService.startProcessInstanceById(...)会使用空开始事件启动流程实例。
* 如果流程定义有多个消息开始事件，而且没有空开始事件， runtimeService.startProcessInstanceByKey(...)和 runtimeService.startProcessInstanceById(...)会抛出异常。
* 如果流程定义只有一个消息开始事件， runtimeService.startProcessInstanceByKey(...)和 runtimeService.startProcessInstanceById(...)会使用这个消息开始事件启动流程实例。
* 如果流程被调用环节（callActivity）启动，消息开始事件只支持如下情况：
  + 在消息开始事件以外，还有一个单独的空开始事件
  + 流程只有一个消息开始事件，没有空开始事件。

**图形标记**

消息开始事件是一个圆圈，中间是一个消息事件图标。图标是白色未填充的，来表示捕获（接收）行为。



**XML内容**

消息开始事件的XML内容时在普通开始事件申请中包含一个 messageEventDefinition子元素：

<definitions id="definitions"

  xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"

  xmlns:activiti="http://activiti.org/bpmn"

  targetNamespace="Examples"

  xmlns:tns="Examples">

  <message id="newInvoice" name="newInvoiceMessage" />

  <process id="invoiceProcess">

    <startEvent id="messageStart" >

        <messageEventDefinition messageRef="tns:newInvoice" />

    </startEvent>

    ...

  </process>

</definitions>

**错误开始事件**

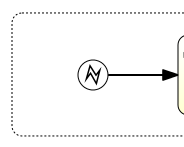
**描述**

[错误](http://www.mossle.com/docs/activiti/#bpmnErrorEventDefinition)开始事件可以用来触发一个事件子流程。 **错误开始事件不能用来启动流程实例**。

错误开始事件都是中断事件。

**图形标记**

错误开始事件是一个圆圈，包含一个错误事件标记。标记是白色未填充的，来表示捕获（接收）行为。



**XML内容**

错误开始事件的XML内容是普通开始事件定义中，包含一个 errorEventDefinition子元素。

<startEvent id="messageStart" >

        <errorEventDefinition errorRef="someError" />

</startEvent>

**结束事件**

结束事件表示（子）流程（分支）的结束。 结束事件**都是触发**事件。 这是说当流程达到结束事件，会触发一个*结果*。 结果的类型是通过事件的内部黑色图标表示的。 在XML内容中，是通过包含的子元素声明的。

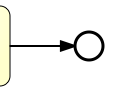
**空结束事件**

**描述**

空结束事件意味着到达事件时不会指定抛出的*结果*。 这样，引擎会直接结束当前执行的分支，不会做其他事情。

**图形标记**

空结束事件是一个粗边圆圈，内部没有小图表（无结果类型）



**XML内容**

空结束事件的XML内容是普通结束事件定义，不包含子元素 （其他结束事件类型都会包含声明类型的子元素）。

<endEvent id="end" name="my end event" />

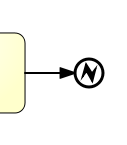
**错误结束事件**

**描述**

当流程执行到**错误结束事件**， 流程的当前分支就会结束，并抛出一个错误。 这个错误可以[被对应的中间边界错误事件捕获](http://www.mossle.com/docs/activiti/#bpmnBoundaryErrorEvent)。 如果找不到匹配的边界错误事件，就会抛出一个异常。

**图形标记**

错误结束事件是一个标准的结束事件（粗边圆圈），内部有错误图标。 错误图表是全黑的，表示触发语法。



**XML内容**

错误结束事件的内容是一个错误事件， 子元素为*errorEventDefinition*。

<endEvent id="myErrorEndEvent">

  <errorEventDefinition errorRef="myError" />

</endEvent>

*errorRef*属性引用定义在流程外部的*error*元素：

<error id="myError" errorCode="123" />

...

<process id="myProcess">

...

*error*的**errorCode**用来查找 匹配的捕获边界错误事件。 如果*errorRef*与任何*error*都不匹配， 就会使用*errorRef*来作为*errorCode*的缩写。 这是activiti特定的缩写。 更具体的说，见如下代码：

<error id="myError" errorCode="error123" />

...

<process id="myProcess">

...

  <endEvent id="myErrorEndEvent">

    <errorEventDefinition errorRef="myError" />

  </endEvent>

等同于

<endEvent id="myErrorEndEvent">

  <errorEventDefinition errorRef="error123" />

</endEvent>

注意errorRef必须与BPMN 2.0格式相符， 必须是一个合法的QName。

**取消结束事件**

[**[EXPERIMENTAL]**](http://www.mossle.com/docs/activiti/#experimental)

**描述**

取消结束事件只能与BPMN事务子流程结合使用。 当到达取消结束事件时，会抛出取消事件，它必须被取消边界事件捕获。 取消边界事件会取消事务，并触发补偿机制。

**图形标记**

取消结束事件显示为标准的结束事件（粗边圆圈），包含一个取消图标。 取消图标是全黑的，表示触发语法。

http://www.mossle.com/docs/activiti/images/bpmn.cancel.end.event.png

**XML内容**

取消结束事件内容是一个结束事件， 包含*cancelEventDefinition*子元素。

<endEvent id="myCancelEndEvent">

  <cancelEventDefinition />

</endEvent>

**边界事件**

边界事件都是*捕获*事件，它会附在一个环节上。 （边界事件不可能触发事件）。这意味着，当节点运行时， 事件会*监听*对应的触发类型。 当事件被*捕获*，节点就会中断， 同时执行事件的后续连线。

所以边界事件的定义方式都一样：

<boundaryEvent id="myBoundaryEvent" attachedToRef="theActivity">

      <XXXEventDefinition/>

</boundaryEvent>

边界事件使用如下方式进行定义：

* 唯一标识（流程范围）
* 使用*caught*属性 引用事件衣服的节点。 注意边界事件和它们附加的节点在同一级别上。 （比如，边界事件不是包含在节点内的）。
* 格式为*XXXEventDefinition*的XML子元素 （比如，*TimerEventDefinition*，*ErrorEventDefinition*，等等） 定义了边界事件的类型。参考对应的边界事件类型， 获得更多细节。

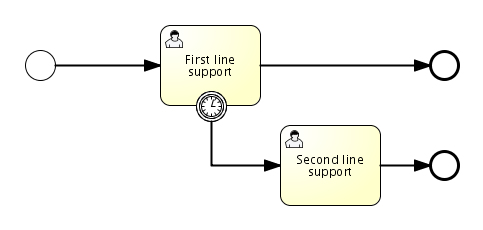
**定时边界事件**

**描述**

定时边界事件就是一个暂停等待警告的时钟。当流程执行到绑定了边界事件的环节， 会启动一个定时器。 当定时器触发时（比如，一定时间之后），环节就会中断， 并沿着定时边界事件的外出连线继续执行。

**图形标记**

定时边界事件是一个标准的边界事件（边界上的一个圆圈）， 内部是一个定时器小图标。



**XML内容**

定时器边界任务定义是一个[正规的边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryEvent)。 指定类型的子元素是**timerEventDefinition**元素。

<boundaryEvent id="escalationTimer" cancelActivity="true" attachedToRef="firstLineSupport">

**<timerEventDefinition>**

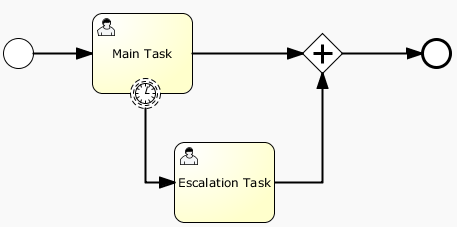
**<timeDuration>PT4H</timeDuration>**

**</timerEventDefinition>**

</boundaryEvent>

请参考[定时事件定义](http://www.mossle.com/docs/activiti/#timerEventDefinitions)获得更多定时其配置的细节。

在流程图中，可以看到上述例子中的圆圈边线是虚线：



经典场景是发送一个升级邮件，但是不打断正常流程的执行。

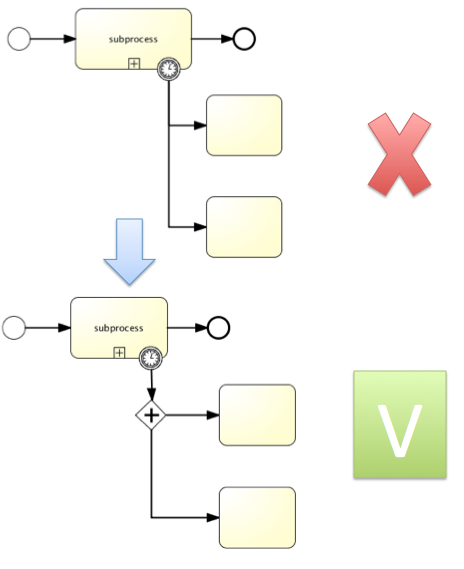
因为BPMN 2.0中，中断和非中断的事件还是有区别的。默认是中断事件。 非中断事件的情况，**不会**中断原始环节，那个环节还停留在原地。 对应的，会创建一个新分支，并沿着事件的流向继续执行。 在XML内容中，要把*cancelActivity*属性设置为false：

<boundaryEvent id="escalationTimer" cancelActivity="false" attachedToRef="firstLineSupport"/>

**注意：**边界定时事件只能在job执行器启用时使用。 （比如，把activiti.cfg.xml中的*jobExecutorActivate* 设置为true，因为默认job执行器默认是禁用的）。

**边界事件的已知问题**

使用边界事件有一个已知的同步问题。 目前，不能边界事件后面不能有多条外出连线 （参考[ACT-47](http://jira.codehaus.org/browse/ACT-47)）。 解决这个问题的方法是在一个连线后使用并发网关。



**错误边界事件**

**描述**

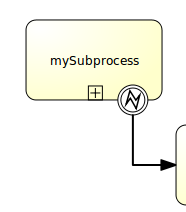
节点边界上的中间*捕获*错误事件， 或简写成**边界错误事件**， 它会捕获节点范围内抛出的错误。

定义一个边界错误事件，大多用于[内嵌子流程](http://www.mossle.com/docs/activiti/#bpmnSubProcess)， 或[调用节点](http://www.mossle.com/docs/activiti/#bpmnCallActivity)，对于子流程的情况，它会为所有内部的节点创建一个作用范围。 错误是由[错误结束事件](http://www.mossle.com/docs/activiti/#bpmnErrorEndEvent)抛出的。 这个错误会传递给上层作用域，直到找到一个错误事件定义向匹配的边界错误事件。

当捕获了错误事件时，边界任务绑定的节点就会销毁， 也会销毁内部所有的执行分支 （比如，同步节点，内嵌子流程，等等）。 流程执行会继续沿着边界事件的外出连线继续执行。

**图形标记**

边界错误事件显示成一个普通的中间事件（圆圈内部有一个小圆圈） 放在节点的标记上，内部有一个错误小图标。错误小图标是白色的， 表示它是一个*捕获*事件。



**Xml内容**

边界错误事件定义为普通的[边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryEvent)：

<boundaryEvent id="catchError" attachedToRef="mySubProcess">

  <errorEventDefinition errorRef="myError"/>

</boundaryEvent>

和[错误结束事件](http://www.mossle.com/docs/activiti/#bpmnErrorEndEvent)一样， *errorRef*引用了process元素外部的一个错误定义：

<error id="myError" errorCode="123" />

...

<process id="myProcess">

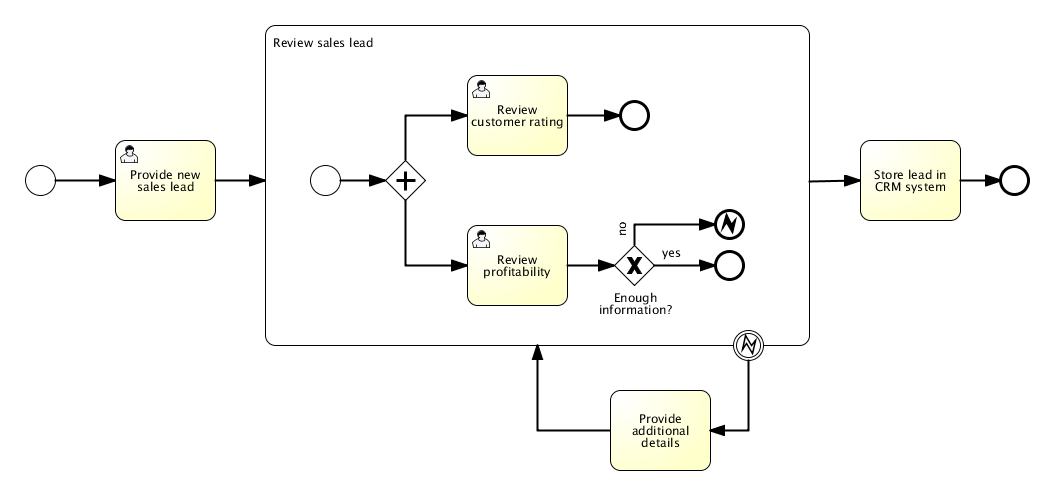
...

**errorCode**用来匹配捕获的错误：

* 如果没有设置*errorRef*，边界错误事件会捕获 **所有错误事件**，无论*错误*的errorCode是什么。
* 如果设置了*errorRef*，并引用了一个已存的*错误*， 边界事件就**只捕获错误代码与之相同的错误**。
* 如果设置了*errorRef*，但是BPMN 2.0中没有定义*错误*， **errorRef就会当做errorCode使用** （和错误结束事件的用法类似）。

**实例**

下面的流程实例演示了如何使用错误结束事件。 当完成*'审核盈利'*这个用户任务是，如果没有提供足够的信息， 就会抛出错误，错误会被子流程的边界任务捕获， 所有*'回顾销售'*子流程中的所有节点都会销毁。 （即使*'审核客户比率'*还没有完成）， 并创建一个*'提供更多信息'*的用户任务。



这个流程也放在demo中了。流程XML和单元测试可以在 *org.activiti.examples.bpmn.event.error*包下找到。

**信号边界事件**

**描述**

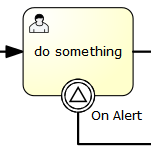
节点边界的中间*捕获*[信号](http://www.mossle.com/docs/activiti/#bpmnSignalEventDefinition)， 或简称为**边界信号事件**， 它会捕获信号定义引用的相同信号名的信号。

**注意：**与其他事件（比如边界错误事件）不同，边界信号事件不只捕获 它绑定方位的信号。信号事件是一个全局的范围（广播语义），就是说信号可以在任何地方触发， 即便是不同的流程实例。

**注意：**和其他事件（比如边界错误事件）不同，捕获信号后，不会停止信号的传播。 如果你有两个信号边界事件，它们捕获相同的信号事件，两个边界事件都会被触发， 即使它们在不同的流程实例中。

**图形标记**

边界信号事件显示为普通的中间事件（圆圈里有个小圆圈），位置在节点的边缘， 内部有一个信号小图标。信号图标是白色的（未填充）， 来表示*捕获*的意思。



**XML内容**

边界信号事件定义为普通的[边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryEvent)：

<boundaryEvent id="boundary" attachedToRef="task" cancelActivity="true">

          <signalEventDefinition signalRef="alertSignal"/>

</boundaryEvent>

**实例**

参考[信号事件定义](http://www.mossle.com/docs/activiti/#bpmnSignalEventDefinition)章节。

**消息边界事件**

**描述**

节点边界上的中间*捕获*[消息](http://www.mossle.com/docs/activiti/#bpmnMessageEventDefinition)， 或简称**边界消息事件**，根据引用的消息定义捕获相同消息名称的消息。

**图形标记**

边界消息事件显示成一个普通的中间事件（圆圈里有个小圆圈），位于节点边缘， 内部是一个消息小图标。消息图标是白色（无填充）， 表示*捕获*语义。



注意，边界消息事件可能是中断（右侧）或非中断（左侧）的。

**XML内容**

边界消息事件定义为标准的[边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryEvent)：

<boundaryEvent id="boundary" attachedToRef="task" cancelActivity="true">

          <messageEventDefinition messageRef="newCustomerMessage"/>

</boundaryEvent>

**实例**

参考[消息事件定义](http://www.mossle.com/docs/activiti/#bpmnMessageEventDefinition)章节。

**取消边界事件**

[**[EXPERIMENTAL]**](http://www.mossle.com/docs/activiti/#experimental)

**描述**

在事务性子流程的边界上的中间*捕获*取消， 或简称为**边界取消事件 cancel event**， 当事务取消时触发。当取消边界事件触发时，首先中断当前作用域的所有执行。 然后开始补偿事务内的所有激活的补偿边界事件。 补偿是同步执行的。例如，离开事务钱，边界事务会等待补偿执行完毕。 当补偿完成后，事务子流程会沿着取消边界事务的外出连线继续执行。

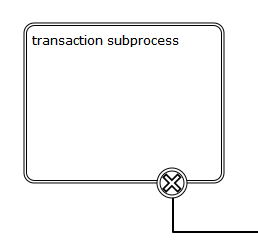
**注意：**每个事务子流程只能有一个取消边界事件。

**注意：**如果事务子流程包含内嵌子流程，补偿只会触发已经成功完成的子流程。

**注意：**如果取消边界子流程对应的事务子流程配置为多实例， 乳沟一个实例触发了取消，就会取消所有实例。 instances.

**图形标记**

取消边界事件显示为了一个普通的中间事件（圆圈里套小圆圈），在节点的边缘， 内部是一个取消小图标。取消图标是白色（无填充）， 表明是*捕获*语义。



**XML内容**

取消边界事件定义为普通[边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryEvent)：

<boundaryEvent id="boundary" attachedToRef="transaction" >

          <cancelEventDefinition />

</boundaryEvent>

因为取消边界事件都是中断的，所以不需要使用cancelActivity属性。

**补偿边界事件**

[**[EXPERIMENTAL]**](http://www.mossle.com/docs/activiti/#experimental)

**描述**

节点边界的中间*捕获*补偿， 或简称为**补偿边界事件**， 可以用来设置一个节点的补偿处理器。

补偿边界事件必须使用直接引用设置唯一的补偿处理器。

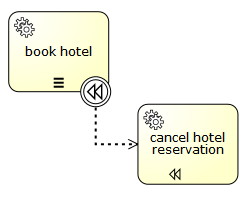
补偿边界事件与其他边界事件的策略不同。 其他边界事件（比如信号边界事件）当到达关联的节点就会被激活。 离开节点时，就会挂起，对应的事件订阅也会取消。 补偿边界事件则不同。补偿边界事件在关联的节点**成功完成**时激活。 当补偿事件触发或对应流程实例结束时，事件订阅才会删除。 它遵循如下规则：

* 补偿触发时，补偿边界事件对应的补偿处理器会调用相同次数，根据它对应的节点的成功次数。
* 如果补偿边界事件关联到多实例节点， 补偿事件会订阅每个实例。
* 如果补偿边界事件关联的节点中包含循环， 补偿事件会在每次节点执行时进行订阅。
* 如果流程实例结束，订阅的补偿事件都会结束。

**注意：**补偿边界事件不支持内嵌子流程。

**图形标记**

补偿边界事件显示为标准中间事件（圆圈里套圆圈），位于节点边缘， 内部有一个补偿小图标。补偿图标是白色的（无填充）， 表示*捕获*语义。另外，下面的图形演示了使用无方向的关联， 为边界事件设置补偿处理器。



**XML内容**

补偿边界事件定义为标准[边界事件](http://www.mossle.com/docs/activiti/#bpmnBoundaryEvent)：

<boundaryEvent id="compensateBookHotelEvt" attachedToRef="bookHotel" >

          <compensateEventDefinition />

</boundaryEvent>

<association associationDirection="One" id="a1"  sourceRef="compensateBookHotelEvt" targetRef="undoBookHotel" />

<serviceTask id="undoBookHotel" isForCompensation="true" activiti:class="..." />

因为补偿边界事件在节点成功完成后激活， 所以不支持cancelActivity属性。

**中间捕获事件**

所有中间捕获事件都使用同样的方式定义：

<intermediateCatchEvent id="myIntermediateCatchEvent" >

      <XXXEventDefinition/>

</intermediateCatchEvent>

中间捕获事件的定义包括

* 唯一标识（流程范围内）
* 一个结构为*XXXEventDefinition*的XML子元素 （比如*TimerEventDefinition*等） 定义了中间捕获事件的类型。参考特定的捕获事件类型， 获得更多详情。

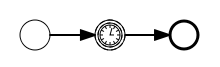
**定时中间捕获事件**

**描述**

定时中间事件作为一个监听器。当执行到达捕获事件节点， 就会启动一个定时器。 当定时器触发（比如，一段时间之后），流程就会沿着定时中间事件的外出节点继续执行。

**图形标记**

定时器中间事件显示成标准中间捕获事件，内部是一个定时器小图标。



**XML内容**

定时器中间事件定义为标准[中间捕获事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateCatchingEvent)。 指定类型的子元素为**timerEventDefinition**元素。

        <intermediateCatchEvent id="timer">

**<timerEventDefinition>**

**<timeDuration>PT5M</timeDuration>**

**</timerEventDefinition>**

        </intermediateCatchEvent>

参考[定时器事件定义](http://www.mossle.com/docs/activiti/#timerEventDefinitions)了解配置信息。

**信号中间捕获事件**

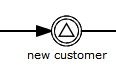
**描述**

中间*捕获*[信号](http://www.mossle.com/docs/activiti/#bpmnSignalEventDefinition)事件 通过引用信号定义来捕获相同信号名称的信号。

**注意：**与其他事件（比如错误事件）不同，信号不会在捕获之后被消费。 如果你有两个激活的信号边界事件捕获相同的信号事件，两个边界事件都会被触发， 即便它们在不同的流程实例中。

**图形标记**

中间信号捕获事件显示为一个普通的中间事件（圆圈套圆圈）， 内部有一个信号小图标。信号小图标是白色的（无填充）， 表示*捕获*语义。



**XML内容**

信号中间事件定义为普通的[中间捕获事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateCatchingEvent)。 对应类型的子元素是**signalEventDefinition**元素。

<intermediateCatchEvent id="signal">

**<signalEventDefinition signalRef="newCustomerSignal" />**

</intermediateCatchEvent>

**实例**

参考[信号事件定义](http://www.mossle.com/docs/activiti/#bpmnSignalEventDefinition)章节。

**消息中间捕获事件**

**描述**

一个中间*捕获*[消息](http://www.mossle.com/docs/activiti/#bpmnMessageEventDefinition)事件，捕获特定名称的消息。

**图形标记**

中间捕获消息事件显示为普通中间事件（圆圈套圆圈）， 内部是一个消息小图标。消息图标是白色的（无填充）， 表示*捕获*语义。

http://www.mossle.com/docs/activiti/images/bpmn.intermediate.message.catch.event.png

**XML内容**

消息中间事件定义为标准[中间捕获事件](http://www.mossle.com/docs/activiti/#bpmnIntermediateCatchingEvent)。 指定类型的子元素是**messageEventDefinition**元素。

<intermediateCatchEvent id="message">

**<messageEventDefinition signalRef="newCustomerMessage" />**

</intermediateCatchEvent>

**实例**

参考[消息事件定义](http://www.mossle.com/docs/activiti/#bpmnMessageEventDefinition)章节。

**内部触发事件**

所有内部触发事件的定义都是同样的：

<intermediateThrowEvent id="myIntermediateThrowEvent" >

      <XXXEventDefinition/>

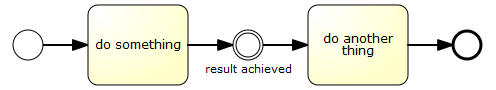
</intermediateThrowEvent>

内部触发事件定义包含

* 唯一标识（流程范围）
* 使用格式为*XXXEventDefinition*的XML子元素 （比如*signalEventDefinition*等） 定义中间触发事件的类型。 参考对应触发事件的类型，了解更多信息。

**中间触发空事件**

下面的流程图演示了一个空中间触发事件的例子， 它通常用于表示流程中的某个状态。



通过添加[执行监听器](http://www.mossle.com/docs/activiti/)，就可以很好地监控一些KPI。

<intermediateThrowEvent id="noneEvent">

  <extensionElements>

    <activiti:executionListener class="org.activiti.engine.test.bpmn.event.IntermediateNoneEventTest$MyExecutionListener" event="start" />

  </extensionElements>

</intermediateThrowEvent>

这里你可以添加自己的代码，把事件发送给BAM工具或DWH。引擎不会为这个事件做任何事情， 它直接径直通过。

**信号中间触发事件**

**描述**

中间*触发*[信号](http://www.mossle.com/docs/activiti/#bpmnSignalEventDefinition)事件为定义的信号抛出一个信号事件。

在activiti中，信号会广播到所有激活的处理器中（比如，所以捕获信号事件）。 信号可以通过同步和异步方式发布。

* 默认配置下，信号是**同步**发送的。就是说， 抛出事件的流程实例会等到信号发送给所有捕获流程实例才继续执行。 捕获流程实例也会在触发流程实例的同一个事务中执行， 意味着如果某个监听流程出现了技术问题（抛出异常），所有相关的实例都会失败。
* 信号也可以**异步**发送。这时它会在到达抛出信号事件后决定哪些处理器是激活的。 对这些激活的处理器，会保存一个异步提醒消息（任务），并发送给jobExecutor。

**图形标记**

中间信号触发事件显示为普通中间事件（圆圈套圆圈）， 内部又一个信号小图标。信号图标是黑色的（有填充）， 表示*触发*语义。



**XML内容**

消息中间事件定义为标准[中间触发事件](http://www.mossle.com/docs/activiti/)。 指定类型的子元素是**signalEventDefinition**元素。

<intermediateThrowEvent id="signal">

**<signalEventDefinition signalRef="newCustomerSignal" />**

</intermediateThrowEvent>

异步信号事件如下所示：

<intermediateThrowEvent id="signal">

**<signalEventDefinition signalRef="newCustomerSignal" activiti:async="true" />**

</intermediateThrowEvent>

**实例**

参考[信号事件定义](http://www.mossle.com/docs/activiti/#bpmnSignalEventDefinition)章节。

**补偿中间触发事件**

[**[EXPERIMENTAL]**](http://www.mossle.com/docs/activiti/#experimental)

**描述**

中间*触发*[补偿](http://www.mossle.com/docs/activiti/)事件 可以用来触发补偿。

**触发补偿：** 补偿可以由特定节点或包含补偿事件的作用域触发。 补偿是通过分配给节点的补偿处理器来完成的。

* 当补偿由节点触发，对应的补偿处理器会根据节点成功完成的次数执行相同次数。
* 如果补偿由当前作用域触发，当前作用域的所有节点都会执行补偿， 也包含并发分支。
* 补偿的触发是继承式的：如果执行补偿的节点是子流程，补偿会作用到子流程中包含的所有节点。 如果子流程是内嵌节点，补偿会递归触发。 然而，补偿不会传播到流程的上层： 如果补偿在子流程中触发，不会传播到子流程范围外。 bpmn规范定义，由节点触发的流程只会作用到“子流程同一级别”。
* activiti的补偿执行次序与流程执行顺序相反。 以为着最后完成的节点会最先执行补偿，诸如此类。
* 中间触发补偿事件可以用来补偿成功完成的事务性子流程。

**注意：** 如果补偿被一个包含子流程的作用域触发，子流程还包含了关联补偿处理器的节点， 补偿只会传播到子流程，如果它已经成功完成了。 如果子流程中的节点也完成了，并关联了补偿处理器， 如果子流程包含的这些节点还没有完成，就不会执行补偿处理器。 参考下面实例：



这个流程中，我们有两个并发分支，一些分支时内嵌子流程，一个是“使用信用卡”节点。 假设两个分支都启动了，第一个分支等待用户完成“审核预定”任务。第二个分支执行“使用信用卡”节点， 并发生了一个错误，这导致“取消预定”事件，并触发补偿。 这时，并发子流程还没有结束，意味着补偿事件不会传播给子流程， 所以“取消旅店预定”这个补偿处理器不会执行。 如果用户任务（就是内嵌子流程）在“取消预定”之前完成了， 补偿就会传播给内嵌子流程。

**流程变量：** 当补偿内嵌子流程时，用来执行补偿处理器的分支可以访问子流程的本地流程实例， 因为这时它是子流程完成的分支。 为了实现这个功能，流程变量的快照会分配给分支（为执行子流程而创建的分支）。 为此，有以下限制条件：

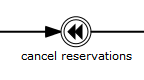
* 补偿处理器无法访问子流程内部创建的，添加到同步分支的变量。
* 分配给分支的流程变量在继承关系上层的（分配给流程实例的流程变量没有包含在快照中）： 补偿触发时，补偿处理器通过它们所在的地方访问这些流程变量。
* 变量快照只用于内嵌子流程，不适用其他节点。

**已知限制：**

* waitForCompletion="false"还不支持。当补偿使用中间触发补偿事件触发时， 事件没有等待，在补偿成功结束后。
* 补偿自己由并发分支执行。并发分支的执行顺序与被补偿的节点完成次序相反。 未来activiti可能支持选项来顺序执行补偿。
* 补偿不会传播给callActivity调用的子流程实例。

**图形标记**

中间补偿触发事件显示为标准中间事件（圆圈套圆圈）， 内部是一个补偿小图标。补偿图标是黑色的（有填充）， 表示*触发*语义。



**Xml内容**

补偿中间事件定义为普通的[中间触发事件](http://www.mossle.com/docs/activiti/)。 对应类型的子元素是**compensateEventDefinition**元素。

<intermediateThrowEvent id="throwCompensation">

        <compensateEventDefinition />

</intermediateThrowEvent>

另外，可选参数activityRef可以用来触发特定作用域/节点的补偿：

<intermediateThrowEvent id="throwCompensation">

        <compensateEventDefinition activityRef="bookHotel" />

</intermediateThrowEvent>

**顺序流**

**描述**

顺序流是连接两个流程节点的连线。 流程执行完一个节点后，会沿着节点的所有外出顺序流继续执行。 就是说，BPMN 2.0默认的行为就是并发的： 两个外出顺序流会创造两个单独的，并发流程分支。

**图形标记**

顺序流显示为从起点到终点的箭头。 箭头总是指向终点。

http://www.mossle.com/docs/activiti/images/bpmn.sequence.flow.png

**XML内容**

顺序流需要流程范围内唯一的**id**， 以及对**起点**与 **终点**元素的引用。

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

**条件顺序流**

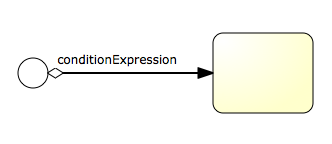
**描述**

可以为顺序流定义一个条件。离开一个BPMN 2.0节点时， 默认会计算外出顺序流的条件。 如果条件结果为*true*, 就会选择外出顺序流继续执行。当多条顺序流被选中时， 就会创建多条*分支*， 流程会继续以并行方式继续执行。

**注意：**上面的讨论仅涉及BPMN 2.0节点（和事件）， 不包括网关。网关会用特定的方式处理顺序流中的条件， 这与网关类型相关。

**图形标记**

条件顺序流显示为一个正常的顺序流，不过在起点有一个菱形。 条件表达式也会显示在顺序流上。



**XML内容**

条件顺序流定义为一个正常的顺序流， 包含**conditionExpression**子元素。 注意目前只支持*tFormalExpressions*， 如果没有设置*xsi:type=""*, 就会默认值支持目前支持的表达式类型。

<sequenceFlow id="flow" sourceRef="theStart" targetRef="theTask">

  <conditionExpression xsi:type="tFormalExpression">

    <![CDATA[${order.price > 100 && order.price < 250}]]>

  </conditionExpression>

</sequenceFlow>

当前条件表达式**只能使用UEL**， 可以参考[表达式章节](http://www.mossle.com/docs/activiti/#apiExpressions)获取更多信息。 使用的表达式需要返回boolean值，否则会在解析表达式时抛出异常。

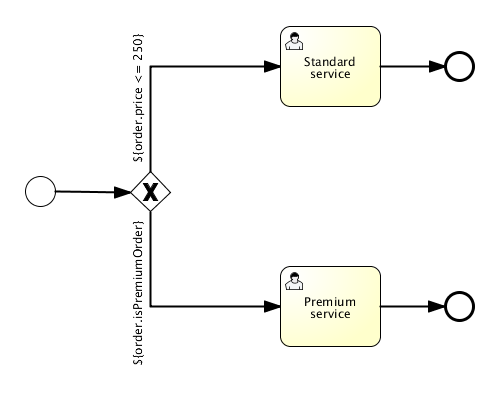
* 下面的例子引用了流程变量的数据， 通过getter调用JavaBean。
* <conditionExpression xsi:type="tFormalExpression">
* <![CDATA[${order.price > 100 && order.price < 250}]]>

</conditionExpression>

* 这个例子通过调用方法返回一个boolean值。
* <conditionExpression xsi:type="tFormalExpression">
* <![CDATA[${order.isStandardOrder()}]]>

</conditionExpression>

在activiti发布包中，包含以下流程实例，使用了值和方法表达式 （参考*org.activiti.examples.bpmn.expression)*包）：



**默认顺序流**

**描述**

所有的BPMN 2.0任务和网关都可以设置一个**默认顺序流**。 只有在节点的其他外出顺序流不能被选中是，才会使用它作为外出顺序流继续执行。 默认顺序流的条件设置不会生效。

**图形标记**

默认顺序流显示为了普通顺序流，起点有一个“斜线”标记。

http://www.mossle.com/docs/activiti/images/bpmn.default.sequence.flow.png

**XML内容**

默认顺序流通过对应节点的**default属性**定义。 下面的XML代码演示了排他网关设置了默认顺序流*flow 2*。 只有当*conditionA*和*conditionB*都返回false时， 才会选择它作为外出连线继续执行。

<exclusiveGateway id="exclusiveGw" name="Exclusive Gateway" **default="flow2"** />

<sequenceFlow id="flow1" sourceRef="exclusiveGw" targetRef="task1">

  <conditionExpression xsi:type="tFormalExpression">${conditionA}</conditionExpression>

</sequenceFlow>

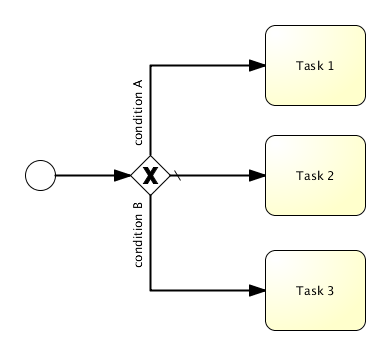
<sequenceFlow id="flow2" sourceRef="exclusiveGw" targetRef="task2"/>

<sequenceFlow id="flow3" sourceRef="exclusiveGw" targetRef="task3">

  <conditionExpression xsi:type="tFormalExpression">${conditionB}</conditionExpression>

</sequenceFlow>

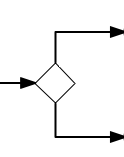
对应下面的图形显示：



**网关**

网关用来控制流程的流向（或像BPMN 2.0里描述的那样，流程的*tokens*。） 网关可以*消费*也可以*生成*token。

网关显示成菱形图形，内部有有一个小图标。 图标表示网关的类型。



**排他网关**

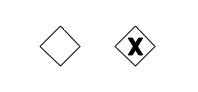
**描述**

排他网关（也叫*异或（XOR）网关*，或更技术性的叫法 *基于数据的排他网关*）， 用来在流程中实现**决策**。 当流程执行到这个网关，所有外出顺序流都会被处理一遍。 其中条件解析为true的顺序流（或者没有设置条件，概念上在顺序流上定义了一个*'true'*） 会被选中，让流程继续运行。

**注意这里的外出顺序流 与BPMN 2.0通常的概念是不同的。通常情况下，所有条件结果为true的顺序流 都会被选中，以并行方式执行，但排他网关只会选择一条顺序流执行。 就是说，虽然多个顺序流的条件结果为true， 那么XML中的第一个顺序流（也只有这一条）会被选中，并用来继续运行流程。 如果没有选中任何顺序流，会抛出一个异常。**

**图形标记**

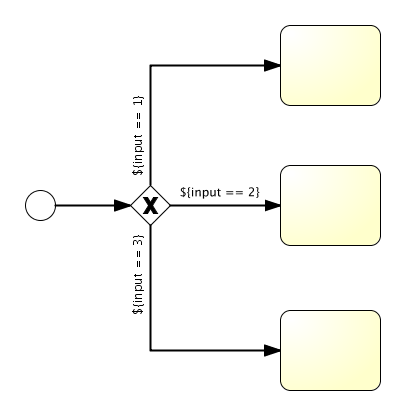
排他网关显示成一个普通网关（比如，菱形图形）， 内部是一个“X”图标，表示*异或（XOR）*语义。 注意，没有内部图标的网关，默认为排他网关。 BPMN 2.0规范不允许在同一个流程定义中同时使用没有X和有X的菱形图形。



**XML内容**

排他网关的XML内容是很直接的：用一行定义了网关， 条件表达式定义在外出顺序流中。 参考[条件顺序流](http://www.mossle.com/docs/activiti/#bpmnConditionalSequenceFlow) 获得这些表达式的可用配置。

参考下面模型实例：



它对应的XML内容如下：

<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>

**并行网关**

**描述**

网关也可以表示流程中的并行情况。最简单的并行网关是 **并行网关**，它允许将流程 *分*成多条分支，也可以把多条分支 *汇聚*到一起。 of execution.

并行网关的功能是基于进入和外出的顺序流的：

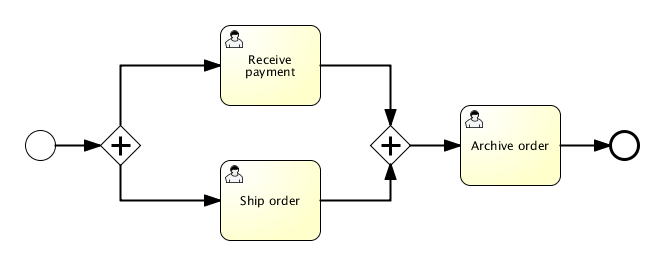
* **分支：** 并行后的所有外出顺序流，为每个顺序流都创建一个并发分支。
* **汇聚：** 所有到达并行网关，在此等待的进入分支， 直到所有进入顺序流的分支都到达以后， 流程就会通过汇聚网关。

注意，如果同一个并行网关有多个进入和多个外出顺序流， 它就同时具有**分支和汇聚功能**。 这时，网关会先汇聚所有进入的顺序流，然后再切分成多个并行分支。

**与其他网关的主要区别是，并行网关不会解析条件。 即使顺序流中定义了条件，也会被忽略。**

**图形标记**

并行网关显示成一个普通网关（菱形）内部是一个“加号”图标， 表示“与（AND）”语义。



**XML内容**

定义并行网关只需要一行XML：

<parallelGateway id="myParallelGateway" />

实际发生的行为（分支，聚合，同时分支聚合）， 要根据并行网关的顺序流来决定。

参考如下代码：

    <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" />

上面例子中，流程启动之后，会创建两个任务：

ProcessInstance pi = runtimeService.startProcessInstanceByKey("forkJoin");

TaskQuery query = taskService.createTaskQuery()

                         .processInstanceId(pi.getId())

                         .orderByTaskName()

                         .asc();

List<Task> tasks = query.list();

assertEquals(2, tasks.size());

Task task1 = tasks.get(0);

assertEquals("Receive Payment", task1.getName());

Task task2 = tasks.get(1);

assertEquals("Ship Order", task2.getName());

当两个任务都完成时，第二个并行网关会汇聚两个分支，因为它只有一条外出连线， 不会创建并行分支， 只会创建*归档订单*任务。

注意并行网关不需要是“平衡的”（比如， 对应并行网关的进入和外出节点数目相等）。 并行网关只是等待所有进入顺序流，并为每个外出顺序流创建并发分支， 不会受到其他流程节点的影响。 所以下面的流程在BPMN 2.0中是合法的：



**包含网关**

**描述**

**包含网关**可以看做是排他网关和并行网关的结合体。 和排他网关一样，你可以在外出顺序流上定义条件，包含网关会解析它们。 但是主要的区别是包含网关可以选择多于一条顺序流，这和并行网关一样。

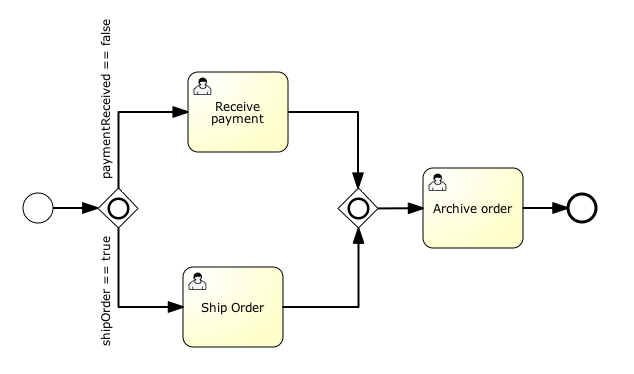
包含网关的功能是基于进入和外出顺序流的：

* **分支：** 所有外出顺序流的条件都会被解析，结果为true的顺序流会以并行方式继续执行， 会为每个顺序流创建一个分支。
* **汇聚：** 所有并行分支到达包含网关，会进入等待章台， 直到每个包含流程token的进入顺序流的分支都到达。 这是与并行网关的最大不同。换句话说，包含网关只会等待被选中执行了的进入顺序流。 在汇聚之后，流程会穿过包含网关继续执行。

注意，如果同一个包含节点拥有多个进入和外出顺序流， 它就会同时**含有分支和汇聚功能**。 这时，网关会先汇聚所有拥有流程token的进入顺序流， 再根据条件判断结果为true的外出顺序流，为它们生成多条并行分支。

**图形标记**

并行网关显示为一个普通网关（菱形），内部包含一个圆圈图标。



**XML内容**

定义一个包含网关需要一行XML：

<inclusiveGateway id="myInclusiveGateway" />

实际的行为（分支，汇聚或同时分支汇聚）， 是由连接在包含网关的顺序流决定的。

参考如下代码：

    <startEvent id="theStart" />

    <sequenceFlow id="flow1" sourceRef="theStart" targetRef="fork" />

**<inclusiveGateway id="fork" />**

    <sequenceFlow sourceRef="fork" targetRef="receivePayment" >

    <conditionExpression xsi:type="tFormalExpression">${paymentReceived == false}</conditionExpression>

    </sequenceFlow>

    <sequenceFlow sourceRef="fork" targetRef="shipOrder" >

    <conditionExpression xsi:type="tFormalExpression">${shipOrder == true}</conditionExpression>

    </sequenceFlow>

    <userTask id="receivePayment" name="Receive Payment" />

    <sequenceFlow sourceRef="receivePayment" targetRef="join" />

    <userTask id="shipOrder" name="Ship Order" />

    <sequenceFlow sourceRef="shipOrder" targetRef="join" />

**<inclusiveGateway id="join" />**

    <sequenceFlow sourceRef="join" targetRef="archiveOrder" />

    <userTask id="archiveOrder" name="Archive Order" />

    <sequenceFlow sourceRef="archiveOrder" targetRef="theEnd" />

    <endEvent id="theEnd" />

在上面的例子中，流程开始之后，如果流程变量为paymentReceived == false和shipOrder == true， 就会创建两个任务。如果，只有一个流程变量为true，就会只创建一个任务。 如果没有条件为true，就会抛出一个异常。 如果想避免异常，可以定义一个默认顺序流。下面的例子中，会创建一个任务，发货任务：

HashMap<String, Object> variableMap = new HashMap<String, Object>();

          variableMap.put("receivedPayment", true);

          variableMap.put("shipOrder", true);

          ProcessInstance pi = runtimeService.startProcessInstanceByKey("forkJoin");

TaskQuery query = taskService.createTaskQuery()

                         .processInstanceId(pi.getId())

                         .orderByTaskName()

                         .asc();

List<Task> tasks = query.list();

assertEquals(1, tasks.size());

Task task = tasks.get(0);

assertEquals("Ship Order", task.getName());

当任务完成后，第二个包含网关会汇聚两个分支， 因为只有一个外出顺序流，所以不会创建并行分支， 只有*归档订单*任务会被激活。

注意，包含网关不需要“平衡”（比如， 对应包含网关的进入和外出数目需要相等）。 包含网关会等待所有进入顺序流完成， 并为每个外出顺序流创建并行分支， 不会受到流程中其他元素的影响。

**基于事件网关**

**描述**

基于事件网关允许根据事件判断流向。网关的每个外出顺序流都要连接到一个中间捕获事件。 当流程到达一个基于事件网关，网关会进入等待状态：会暂停执行。 与此同时，会为每个外出顺序流创建相对的事件订阅。

注意基于事件网关的外出顺序流和普通顺序流不同。这些顺序流不会真的"执行"。 相反，它们让流程引擎去决定执行到基于事件网关的流程需要订阅哪些事件。 要考虑以下条件：

* 基于事件网关必须有两条或以上外出顺序流。
* 基于事件网关后，只能使用intermediateCatchEvent类型。 （activiti不支持基于事件网关后连接ReceiveTask。）
* 连接到基于事件网关的intermediateCatchEvent只能有一条进入顺序流。

**图形标记**

基于事件网关和其他BPMN网关一样显示成一个菱形， 内部包含指定图标。

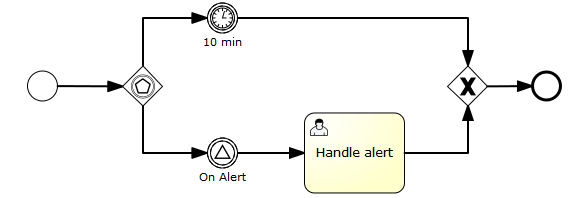


**XML内容**

用来定义基于事件网关的XML元素是eventBasedGateway。

**实例**

下面的流程是一个使用基于事件网关的例子。当流程执行到基于事件网关时， 流程会暂停执行。与此同时，流程实例会订阅警告信号事件，并创建一个10分钟后触发的定时器。 这会产生流程引擎为一个信号事件等待10分钟的效果。如果10分钟内发出信号，定时器就会取消，流程会沿着信号执行。 如果信号没有出现，流程会沿着定时器的方向前进，信号订阅会被取消。



<definitions id="definitions"

        xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"

        xmlns:activiti="http://activiti.org/bpmn"

        targetNamespace="Examples">

        <signal id="alertSignal" name="alert" />

        <process id="catchSignal">

                <startEvent id="start" />

                <sequenceFlow sourceRef="start" targetRef="gw1" />

                <eventBasedGateway id="gw1" />

                <sequenceFlow sourceRef="gw1" targetRef="signalEvent" />

                <sequenceFlow sourceRef="gw1" targetRef="timerEvent" />

                <intermediateCatchEvent id="signalEvent" name="Alert">

                        <signalEventDefinition signalRef="alertSignal" />

                </intermediateCatchEvent>

                <intermediateCatchEvent id="timerEvent" name="Alert">

                        <timerEventDefinition>

                                <timeDuration>PT10M</timeDuration>

                        </timerEventDefinition>

                </intermediateCatchEvent>

                <sequenceFlow sourceRef="timerEvent" targetRef="exGw1" />

                <sequenceFlow sourceRef="signalEvent" targetRef="task" />

                <userTask id="task" name="Handle alert"/>

                <exclusiveGateway id="exGw1" />

                <sequenceFlow sourceRef="task" targetRef="exGw1" />

                <sequenceFlow sourceRef="exGw1" targetRef="end" />

                <endEvent id="end" />

</process>

</definitions>

**任务**

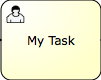
**用户任务**

**描述**

用户任务用来设置必须由人员完成的工作。 当流程执行到用户任务，会创建一个新任务， 并把这个新任务加入到分配人或群组的任务列表中。

**图形标记**

用户任务显示成一个普通任务（圆角矩形），左上角有一个小用户图标。



**XML内容**

XML中的用户任务定义如下。*id*属性是必须的。 *name*属性是可选的。

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

用户任务也可以设置描述。实际上所有BPMN 2.0元素都可以设置描述。 添加**documentation**元素可以定义描述。

<userTask id="theTask" name="Schedule meeting" >

  <documentation>

          Schedule an engineering meeting for next week with the new hire.

  </documentation>

描述文本可以通过标准的java方法来获得：

task.getDescription()

**持续时间**

任务可以用一个字段来描述任务的持续时间。可以使用查询API来对持续时间进行搜索， 根据在时间之前或之后进行搜索。

我们提供了一个节点扩展，在任务定义中设置一个表达式， 这样在任务创建时就可以为它设置初始持续时间。表达式**应该是java.util.Date， java.util.String (ISO8601格式)，ISO8601 持续时间 (比如PT50M)或null**。 例如：你可以在流程中使用上述格式输入日期，或在前一个服务任务中计算一个时间。 这里使用了持续时间，持续时间会基于当前时间进行计算，再通过给定的时间段累加。 比如，使用"PT30M"作为持续时间，任务就会从现在开始持续30分钟。

<userTask id="theTask" name="Important task" **activiti:dueDate="${dateVariable}"**/>

任务的持续时间也可以通过TaskService修改， 或在TaskListener中通过传入的DelegateTask参数修改。

**用户分配**

用户任务可以直接分配给一个用户。 这可以通过**humanPerformer**元素定义。 *humanPerformer*定义需要一个 **resourceAssignmentExpression**来实际定义用户。 当前，只支持**formalExpressions**。

<process ... >

  ...

  <userTask id='theTask' name='important task' >

    <humanPerformer>

      <resourceAssignmentExpression>

        <formalExpression>kermit</formalExpression>

      </resourceAssignmentExpression>

    </humanPerformer>

  </userTask>

**只有一个**用户可以坐拥任务的执行者分配给用户。 在activiti中，用户叫做**执行者**。 拥有执行者的用户不会出现在其他人的任务列表中， 只能出现执行者的**个人任务列表**中。

直接分配给用户的任务可以通过TaskService像下面这样获取：

List<Task> tasks = taskService.createTaskQuery().taskAssignee("kermit").list();

任务也可以加入到人员的**候选任务列表**中。 这时，需要使用**potentialOwner**元素。 用法和*humanPerformer*元素类似。注意它需要指定表达式中的每个项目是人员还是群组 （引擎猜不出来）。

<process ... >

  ...

  <userTask id='theTask' name='important task' >

    <potentialOwner>

      <resourceAssignmentExpression>

**<formalExpression>user(kermit), group(management)</formalExpression>**

      </resourceAssignmentExpression>

    </potentialOwner>

  </userTask>

使用*potentialOwner*元素定义的任务，可以像下面这样获取 （使用*TaskQuery*的发那个发与查询设置了执行者的任务类似）：

 List<Task> tasks = taskService.createTaskQuery().taskCandidateUser("kermit");

这会获取所有kermit为**候选人**的任务， 例如：表达式中包含*user(kermit)*。 这也会获得所有**分配包含kermit这个成员的群组** （比如，*group(management)*，前提是kermit是这个组的成员， 并且使用了activiti的账号组件）。 用户所在的群组是在运行阶段获取的，它们可以通过 [IdentityService](http://www.mossle.com/docs/activiti/#apiEngine)进行管理。

如果没有显示指定设置的是用户还是群组， 引擎会默认当做群组处理。所以下面的设置与*使用group(accountancy)*效果一样。

<formalExpression>accountancy</formalExpression>

**Activiti对任务分配的扩展**

当分配不复杂时，用户和组的设置非常麻烦。 为避免复杂性，可以使用用户任务的[自定义扩展](http://www.mossle.com/docs/activiti/#bpmnCustomExtensions)。

* **assignee属性**：这个自定义扩展可以直接把用户任务分配给指定用户。

<userTask id="theTask" name="my task" activiti:assignee="kermit" />

它和使用[上面](http://www.mossle.com/docs/activiti/#bpmnUserTaskAssignment)定义的**humanPerformer** 效果完全一样。

* **candidateUsers属性**：这个自定义扩展可以为任务设置候选人。

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

它和使用[上面](http://www.mossle.com/docs/activiti/#bpmnUserTaskAssignment)定义的**potentialOwner** 效果完全一样。 注意它不需要像使用*potentialOwner*通过*user(kermit)*声明， 因为这个属性只能用于人员。

* **candidateGroups属性**：这个自定义扩展可以为任务设置候选组。

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

它和使用[上面](http://www.mossle.com/docs/activiti/#bpmnUserTaskAssignment)定义的**potentialOwner** 效果完全一样。 注意它不需要像使用*potentialOwner*通过*group(management)*声明， 因为这个属性只能用于群组。

* *candidateUsers* 和 *candidateGroups* 可以同时设置在同一个用户任务中。

注意：虽然activiti提供了一个账号管理组件， 也提供了[IdentityService](http://www.mossle.com/docs/activiti/#apiEngine)， 但是账号组件不会检测设置的用户是否村爱。 它嵌入到应用中，也允许activiti与其他已存的账户管理方案集成。

如果上面的方式还不满足需求，还可以使用创建事件的[任务监听器](http://www.mossle.com/docs/activiti/#taskListeners) 来实现自定义的分配逻辑：

<userTask id="task1" name="My task" >

  <extensionElements>

    <activiti:taskListener event="create" class="org.activiti.MyAssignmentHandler" />

  </extensionElements>

</userTask>

DelegateTask会传递给TaskListener的实现， 通过它可以设置执行人，候选人和候选组：

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");

    ...

  }

}

使用spring时，可以使用向上面章节中介绍的自定义分配属性， 使用[表达式](http://www.mossle.com/docs/activiti/#springExpressions) 把[任务监听器](http://www.mossle.com/docs/activiti/#taskListeners)设置为spring代理的bean， 让这个监听器监听任务的*创建*事件。 下面的例子中，执行者会通过调用ldapService这个spring bean的findManagerOfEmployee方法获得。 流程变量*emp*会作为参数传递给bean。

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

也可以用来设置候选人和候选组：

<userTask id="task" name="My Task" activiti:candidateUsers="${ldapService.findAllSales()}"/>

注意方法返回类型只能为String或Collection<String> （对应候选人和候选组）：

public class FakeLdapService {

  public String findManagerForEmployee(String employee) {

    return "Kermit The Frog";

  }

  public List<String> findAllSales() {

    return Arrays.asList("kermit", "gonzo", "fozzie");

  }

}

**脚本任务**

**描述**

脚本任务时一个自动节点。当流程到达脚本任务， 会执行对应的脚本。

**图形标记**

脚本任务显示为标准BPMN 2.0任务（圆角矩形）， 左上角有一个脚本小图标。



**XML内容**

脚本任务定义需要指定**script** 和**scriptFormat**。

<scriptTask id="theScriptTask" name="Execute script" scriptFormat="groovy">

  <script>

    sum = 0

    for ( i in inputArray ) {

      sum += i

    }

  </script>

</scriptTask>

**scriptFormat**的值必须兼容 [JSR-223](http://jcp.org/en/jsr/detail?id=223)。 （java平台的脚本语言）。默认Javascript会包含在JDK中，不需要额外的依赖。 如果你想使用其他（JSR-223兼容）的脚本引擎， 需要把对应的jar添加到classpath下，并使用合适的名称。 比如，activiti单元测试经常使用groovy， 因为语法比java简单太多。

注意，groovy脚本引擎放在groovy-all.jar中。在2.0版本之前， 脚本引擎是groovy jar的一部分。这样，需要添加如下依赖：

<dependency>

      <groupId>org.codehaus.groovy</groupId>

      <artifactId>groovy-all</artifactId>

      <version>2.x.x<version>

</dependency>

**脚本中的变量**

到达脚本任务的流程可以访问的所有流程变量，都可以在脚本中使用。 实例中，脚本变量*'inputArray'*其实是流程变量 （整数数组）。

<script>

    sum = 0

    for ( i in **inputArray** ) {

      sum += i

    }

</script>

也可以在脚本中设置流程变量，直接调用 *execution.setVariable("variableName", variableValue)*。 默认，不会自动保存变量（**注意：activiti 5.12之前存在这个问题**）。 可以在脚本中自动保存任何变量。 （比如上例中的*sum*），只要把scriptTask 的autoStoreVariables属性设置为true。 然而，**最佳实践是不要用它，而是显示调用execution.setVariable()**， 因为一些当前版本的JDK对于一些脚本语言，无法实现自动保存变量。 参考[这里](http://www.jorambarrez.be/blog/2013/03/25/bug-on-jdk-1-7-0_17-when-using-scripttask-in-activiti/)获得更多信息。

<scriptTask id="script" scriptFormat="JavaScript" activiti:autoStoreVariables="false">

参数默认为false，意思是如果没有为脚本任务定义设置参数， 所有声明的变量将只存在于脚本执行的阶段。

如何在脚本中设置变量的例子：

<script>

    def scriptVar = "test123"

    execution.setVariable("myVar", scriptVar)

</script>

注意：下面这些命名已被占用，**不能用作变量名**： **out, out:print, lang:import, context, elcontext**。

**脚本结果**

脚本任务的返回值可以通过制定流程变量的名称，分配给已存或一个新流程变量， 使用脚本任务定义的*'activiti:resultVariable'*属性。 任何已存的流程变量都会被脚本执行的结果覆盖。 如果没有指定返回变量名，脚本的返回值会被忽略。

<scriptTask id="theScriptTask" name="Execute script" scriptFormat="juel" activiti:resultVariable="myVar">

  <script>#{echo}</script>

</scriptTask>

上例中，脚本的结果（表达式*'#{echo}'*的值） 在脚本完成后，会设置到*'myVar'*变量中。

**Java服务任务**

**描述**

java服务任务用来调用外部java类。

**图形标记**

服务任务显示为圆角矩形，左上角有一个齿轮小图标。

**XML内容**

有4钟方法来声明java调用逻辑：

* 实现JavaDelegate或ActivityBehavior
* 执行解析代理对象的表达式
* 调用一个方法表达式
* 调用一直值表达式

执行一个在流程执行中调用的类， 需要在**'activiti:class'**属性中设置全类名。

<serviceTask id="javaService"

             name="My Java Service Task"

             activiti:class="org.activiti.MyJavaDelegate" />

参考[*实现*章节](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTaskImplementation) 了解更多使用类的信息。

也可以使用表达式调用一个对象。对象必须遵循一些规则， 并使用activiti:class属性进行创建。 （了解[更多](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTaskImplementation)）。

 <serviceTask id="serviceTask" **activiti:delegateExpression="${delegateExpressionBean}"** />

这里，delegateExpressionBean是一个实现了JavaDelegate接口的bean， 它定义在实例的spring容器中。

要指定执行的UEL方法表达式， 需要使用**activiti:expression**。

<serviceTask id="javaService"

             name="My Java Service Task"

             activiti:expression="#{printer.printMessage()}" />

方法printMessage（无参数）会调用 名为printer对象的方法。

也可以为表达式中的方法传递参数。

<serviceTask id="javaService"

             name="My Java Service Task"

             activiti:expression="#{printer.printMessage(execution, myVar)}" />

这会调用名为printer对象上的方法printMessage。 第一个参数是DelegateExecution，在表达式环境中默认名称为execution。 第二个参数传递的是当前流程的名为myVar的变量。

要指定执行的UEL值表达式， 需要使用**activiti:expression**属性。

<serviceTask id="javaService"

             name="My Java Service Task"

             activiti:expression="#{split.ready}" />

ready属性的getter方法，getReady（无参数）， 会作用于名为split的bean上。 这个对象会被解析为流程对象和 （如果合适）spring环境中的对象。

**实现**

要在流程执行中实现一个调用的类，这个类需要实现*org.activiti.engine.delegate.JavaDelegate*接口， 并在*execute*方法中提供对应的业务逻辑。 当流程执行到特定阶段，它会指定方法中定义好的业务逻辑， 并按照默认BPMN 2.0中的方式离开节点。

让我们创建一个java类的例子，它可以流程变量中字符串转换为大写。 这个类需要实现*org.activiti.engine.delegate.JavaDelegate*接口， 这要求我们实现*execute(DelegateExecution)*方法。 它包含的业务逻辑会被引擎调用。流程实例信息，如流程变量和其他信息， 可以通过 [DelegateExecution](http://activiti.org/javadocs/org/activiti/engine/delegate/DelegateExecution.html) 接口访问和操作（点击对应操作的javadoc的链接，获得更多信息）。

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);

  }

}

注意：**serviceTask定义的class只会创建一个java类的实例**。 所有流程实例都会共享相同的类实例，并调用*execute(DelegateExecution)*。 这意味着，类不能使用任何成员变量，必须是线程安全的，它必须能模拟在不同线程中执行。 这也影响着[属性注入](http://www.mossle.com/docs/activiti/#serviceTaskFieldInjection)的处理方式。

流程定义中引用的类（比如，使用activiti:class）**不会 在部署时实例化**。只有当流程第一次执行到使用类的时候， 类的实例才会被创建。如果找不到类，会抛出一个ActivitiException。 这个原因是部署环境（更确切是的*classpath*）和真实环境往往是不同的。 比如当使用*ant*或业务归档上传到Activiti Explorer来发布流程 classpath没有包含引用的类。

[**[内部：非公共实现类]**](http://www.mossle.com/docs/activiti/#internal) 也可以提供实现 *org.activiti.engine.impl.pvm.delegate.ActivityBehavior*接口的类。 实现可以访问更强大的*ActivityExecution*, 它可以影响流程的流向。注意，这不是一个很好的实践， 应该尽量避免。所以，建议只有在高级情况下并且你确切知道你要做什么的情况下， 再使用*ActivityBehavior*接口。

**属性注入**

可以为代理类的属性注入数据。支持如下类型的注入：

* 固定的字符串
* 表达式

If available, the value is injected through a public setter method on your delegated class, following the Java Bean naming conventions (e.g. field fistName has setter setFirstName(...)). If no setter is available for that field, the value of private member will be set on the delegate. SecurityManagers in some environments don't allow modifying private fields, so it's safer to expose a public setter-method for the fields you want to have injected. **Regardless of the type of value declared in the process-definition, the type of the setter/private field on the injection target should always be org.activiti.engine.delegate.Expression.**

The following code snippet shows how to inject a constant value into a field. Field injection is supported when using the *'class'* attribute. Note that we need to **declare a 'extensionElements' XML element before the actual field injection declarations**, which is a requirement of the BPMN 2.0 XML Schema.

<serviceTask id="javaService"

    name="Java service invocation"

    activiti:class="org.activiti.examples.bpmn.servicetask.ToUpperCaseFieldInjected">

**<extensionElements>**

**<activiti:field name="text" stringValue="Hello World" />**

**</extensionElements>**

</serviceTask>

The class ToUpperCaseFieldInjected has a field text which is of type org.activiti.engine.delegate.Expression. When calling text.getValue(execution), the configured string value Hello World will be returned.

Alternatively, for longs texts (e.g. an inline e-mail) the *'activiti:string'* sub element can be used:

<serviceTask id="javaService"

    name="Java service invocation"

    activiti:class="org.activiti.examples.bpmn.servicetask.ToUpperCaseFieldInjected">

  <extensionElements>

    <activiti:field name="text">

**<activiti:string>**

**Hello World**

**</activiti:string>**

    </activiti:field>

  </extensionElements>

</serviceTask>

To inject values that are dynamically resolved at runtime, expressions can be used. Those expressions can use process variables, or Spring defined beans (if Spring is used). As noted in [Service Task Implementation](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTaskImplementation), an instance of the Java class is shared among all process-instances in a service task. To have dynamic injection of values in fields, you can inject value and method expressions in a org.activiti.engine.delegate.Expression which can be evaluated/invoked using the DelegateExecution passed in the execute method.

<serviceTask id="javaService" name="Java service invocation"

  activiti:class="org.activiti.examples.bpmn.servicetask.ReverseStringsFieldInjected">

  <extensionElements>

    <activiti:field name="text1">

**<activiti:expression>${genderBean.getGenderString(gender)}</activiti:expression>**

    </activiti:field>

    <activiti:field name="text2">

**<activiti:expression>Hello ${gender == 'male' ? 'Mr.' : 'Mrs.'} ${name}</activiti:expression>**

    </activiti:field>

  </ extensionElements>

</ serviceTask>

The example class below uses the injected expressions and resolves them using the current DelegateExecution. Full code and test can be found in org.activiti.examples.bpmn.servicetask.JavaServiceTaskTest.testExpressionFieldInjection

public class ReverseStringsFieldInjected implements JavaDelegate {

  private Expression text1;

  private Expression text2;

  public void execute(DelegateExecution execution) {

    String value1 = (String) text1.getValue(execution);

    execution.setVariable("var1", new StringBuffer(value1).reverse().toString());

    String value2 = (String) text2.getValue(execution);

    execution.setVariable("var2", new StringBuffer(value2).reverse().toString());

  }

}

Alternatively, you can also set the expressions as an attribute instead of a child-element, to make the XML less verbose.

<activiti:field name="text1" **expression="${genderBean.getGenderString(gender)}"** />

<activiti:field name="text1" **expression="Hello ${gender == 'male' ? 'Mr.' : 'Mrs.'} ${name}"** />

**Since the Java class instance is reused, the injection only happens once, when the serviceTask is called the first time. When the fields are altered by your code, the values won't be re-injected so you should treat them as immutable and don't make any changes to them.**

**Service task results**

The return value of a service execution (for service task using expression only) can be assigned to an already existing or to a new process variable by specifying the process variable name as a literal value for the *'activiti:resultVariable'* attribute of a service task definition. Any existing value for a specific process variable will be overwritten by the result value of the service execution. When not specifying a result variable name, the service execution result value gets ignored.

<serviceTask id="aMethodExpressionServiceTask"

    activiti:expression="#{myService.doSomething()}"

    activiti:resultVariable="myVar" />

In the example above, the result of the service execution (the return value of the *'doSomething()'* method invocation on an object that is made available under the name *'myService'* either in the process variables or as a Spring bean) is set to the process variable named *'myVar'* after the service execution completes.

**Handling exceptions**

When custom logic is executed, it is often required to catch certain business exceptions and handle them inside the surrounding process. Activiti provides different options to do that.

**Throwing BPMN Errors**

It is possible to throw BPMN Errors from user code inside Service Tasks or Script Tasks. In order to do this, a special ActivitiException called *BpmnError* can be thrown in JavaDelegates, scripts, expressions and delegate expressions. The engine will catch this exception and forward it to an appropriate error handler, e.g., a Boundary Error Event or an Error Event Sub-Process.

public class ThrowBpmnErrorDelegate implements JavaDelegate {

  public void execute(DelegateExecution execution) throws Exception {

    try {

      executeBusinessLogic();

    } catch (BusinessExeption e) {

      throw new BpmnError("BusinessExeptionOccured");

    }

  }

}

The constructor argument is an error code, which will be used to determine the error handler that is responsible for the error. See [Boundary Error Event](http://www.mossle.com/docs/activiti/#bpmnBoundaryErrorEvent) for information on how to catch a BPMN Error.

This mechanism should be used **only for business faults** that shall be handled by a Boundary Error Event or Error Event Sub-Process modeled in the process definition. Technical errors should be represented by other exception types and are usually not handled inside a process.

**Exception Sequence Flow**

[**[INTERNAL: non-public implementation classes]**](http://www.mossle.com/docs/activiti/#internal) Another option is to route process execution through another path in case some exception occurs. The following example shows how this is done.

<serviceTask id="javaService"

  name="Java service invocation"

  activiti:class="org.activiti.ThrowsExceptionBehavior">

</serviceTask>

<sequenceFlow id="no-exception" sourceRef="javaService" targetRef="theEnd" />

<sequenceFlow id="exception" sourceRef="javaService" targetRef="fixException" />

Here, the service task has two outgoing sequence flow, called exception and no-exception. This sequence flow id will be used to direct process flow in case of an exception:

public class ThrowsExceptionBehavior implements ActivityBehavior {

  public void execute(ActivityExecution execution) throws Exception {

    String var = (String) execution.getVariable("var");

    PvmTransition transition = null;

    try {

      executeLogic(var);

      transition = execution.getActivity().findOutgoingTransition("no-exception");

    } catch (Exception e) {

      transition = execution.getActivity().findOutgoingTransition("exception");

    }

    execution.take(transition);

  }

}

**Using an Activiti service from within a JavaDelegate**

For some use cases, it might be needed to use the Activiti services from within a Java service task (eg. starting a process instance through the RuntimeService, if the callActivity doesn't suit your needs). The *org.activiti.engine.delegate.DelegateExecution* allows to easily use these services through the *org.activiti.engine.EngineServices* interface:

public class StartProcessInstanceTestDelegate implements JavaDelegate {

  public void execute(DelegateExecution execution) throws Exception {

    RuntimeService runtimeService = execution.getEngineServices().getRuntimeService();

    runtimeService.startProcessInstanceByKey("myProcess");

  }

}

All of the Activiti service API's are available through this interface.

All data changes that occur as an effect of using these API calls, will be part of the current transaction. This also works in environments with dependency injection like Spring and CDI with or without a JTA enabled datasource. For example, the following snippet of code will do the same as the snippet above, but now the RuntimeService is injected rather than it is being fetched through the *org.activiti.engine.EngineServices* interface.

@Component("startProcessInstanceDelegate")

public class StartProcessInstanceTestDelegateWithInjection {

    @Autowired

    private RuntimeService runtimeService;

    public void startProcess() {

      runtimeService.startProcessInstanceByKey("oneTaskProcess");

    }

}

**Important technical note:** since the service call is being done as part of the current transaction any data that was produced or altered *before* the service task is executed, is not yet flushed to the database. All API calls work on the database data, which means that these uncommitted changes are not be 'visible' within the api call of the service task.

**Web Service Task**

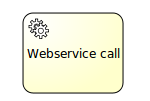
[**[EXPERIMENTAL]**](http://www.mossle.com/docs/activiti/#experimental)

**Description**

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

**Graphical Notation**

A Web Service task is visualized the same as a Java service task.



**XML representation**

To use a Web service we need to import its operations and complex types. This can be done automatically by using the import tag pointing to the WSDL of the Web service:

<import importType="http://schemas.xmlsoap.org/wsdl/"

        location="http://localhost:63081/counter?wsdl"

        namespace="http://webservice.activiti.org/" />

The previous declaration tells Activiti to import the definitions but it doesn't create the item definitions and messages for you. Let's suppose we want to invoke a specific method called 'prettyPrint', therefore we will need to create the corresponding message and item definitions for the request and response messages:

<message id="prettyPrintCountRequestMessage" itemRef="tns:prettyPrintCountRequestItem" />

<message id="prettyPrintCountResponseMessage" itemRef="tns:prettyPrintCountResponseItem" />

<itemDefinition id="prettyPrintCountRequestItem" structureRef="counter:prettyPrintCount" />

<itemDefinition id="prettyPrintCountResponseItem" structureRef="counter:prettyPrintCountResponse" />

Before declaring the service task, we have to define the BPMN interfaces and operations that actually reference the Web service ones. Basically, we define and 'interface' and the required 'operation's'. For each operation we reuse the previous defined message for in and out. For example, the following declaration defines the 'counter' interface and the 'prettyPrintCountOperation' operation:

<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 we can declare a Web Service Task by using the ##WebService implementation and a reference to the Web service operation.

<serviceTask id="webService"

        name="Web service invocation"

        implementation="##WebService"

        operationRef="tns:prettyPrintCountOperation">

**Web Service Task IO Specification**

Unless we are using the simplistic approach for data input and output associations (See below), each Web Service Task needs to declare an IO Specification which states which are the inputs and outputs of the task. The approach is pretty straightforward and BPMN 2.0 complaint, for our prettyPrint example we define the input and output sets according to the previously declared item definitions:

<ioSpecification>

        <dataInput itemSubjectRef="tns:prettyPrintCountRequestItem" id="dataInputOfServiceTask" />

        <dataOutput itemSubjectRef="tns:prettyPrintCountResponseItem" id="dataOutputOfServiceTask" />

        <inputSet>

                <dataInputRefs>dataInputOfServiceTask</dataInputRefs>

        </inputSet>

        <outputSet>

                <dataOutputRefs>dataOutputOfServiceTask</dataOutputRefs>

        </outputSet>

</ioSpecification>

**Web Service Task data input associations**

There are 2 ways of specifying data input associations:

* Using expressions
* Using the simplistic approach

To specify the data input association using expressions we need to define the source and target items and specify the corresponding assignments between the fields of each item. In the following example we assign prefix and suffix fields of the items:

<dataInputAssociation>

        <sourceRef>dataInputOfProcess</sourceRef>

        <targetRef>dataInputOfServiceTask</targetRef>

        <assignment>

                <from>${dataInputOfProcess.prefix}</from>

                <to>${dataInputOfServiceTask.prefix}</to>

        </assignment>

        <assignment>

                <from>${dataInputOfProcess.suffix}</from>

                <to>${dataInputOfServiceTask.suffix}</to>

        </assignment>

</dataInputAssociation>

On the other hand we can use the simplistic approach which is much more simple. The 'sourceRef' element is an Activiti variable name and the 'targetRef' element is a property of the item definition. In the following example we assign to the 'prefix' field the value of the variable 'PrefixVariable' and to the 'suffix' field the value of the variable 'SuffixVariable'.

<dataInputAssociation>

        <sourceRef>PrefixVariable</sourceRef>

        <targetRef>prefix</targetRef>

</dataInputAssociation>

<dataInputAssociation>

        <sourceRef>SuffixVariable</sourceRef>

        <targetRef>suffix</targetRef>

</dataInputAssociation>

**Web Service Task data output associations**

There are 2 ways of specifying data out associations:

* Using expressions
* Using the simplistic approach

To specify the data out association using expressions we need to define the target variable and the source expression. The approach is pretty straightforward and similar data input associations:

<dataOutputAssociation>

        <targetRef>dataOutputOfProcess</targetRef>

        <transformation>${dataOutputOfServiceTask.prettyPrint}</transformation>

</dataOutputAssociation>

On the other hand we can use the simplistic approach which is much more simple. The 'sourceRef' element is a property of the item definition and the 'targetRef' element is an Activiti variable name. The approach is pretty straightforward and similar data input associations:

<dataOutputAssociation>

        <sourceRef>prettyPrint</sourceRef>

        <targetRef>OutputVariable</targetRef>

</dataOutputAssociation>

**Business Rule Task**

[**[EXPERIMENTAL]**](http://www.mossle.com/docs/activiti/#experimental)

**Description**

A Business Rule task is used to synchronously execute one or more rules. Activiti uses Drools Expert, the Drools rule engine to execute business rules. Currently, the .drl files containing the business rules have to be deployed together with the process definition that defines a business rule task to execute those rules. This means that all .drl files that are used in a process have to be packaged in the process BAR file like for example the task forms. For more information about creating business rules for Drools Expert please refer to the Drools documentation at [JBoss Drools](http://www.jboss.org/drools/documentation)

if you want to plug in your implementation of the rule task, e.g. because you want to use Drools differently or you want to use a completly different rule engine, then you can use the class or expression attribute on the BusinessRuleTask and it will behave exactly like a [ServiceTask](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTask)

**Graphical Notation**

A Business Rule task is visualized the with a table icon.



**XML representation**

To execute one or more business rules that are deployed in the same BAR file as the process definition, we need to define the input and result variables. For the input variable definition a list of process variables can be defined separated by a comma. The output variable definition can only contain one variable name that will be used to store the output objects of the executed business rules in a process variable. Note that the result variable will contain a List of objects. If no result variable name is specified by default org.activiti.engine.rules.OUTPUT is used.

The following business rule task executes all business rules deployed with the process definition:

<process id="simpleBusinessRuleProcess">

  <startEvent id="theStart" />

  <sequenceFlow sourceRef="theStart" targetRef="businessRuleTask" />

**<businessRuleTask id="businessRuleTask" activiti:ruleVariablesInput="${order}"**

**activiti:resultVariable="rulesOutput" />**

  <sequenceFlow sourceRef="businessRuleTask" targetRef="theEnd" />

  <endEvent id="theEnd" />

</process>

The business rule task can also be configured to execute only a defined set of rules from the deployed .drl files. A list of rule names separated by a comma must be specified for this.

<businessRuleTask id="businessRuleTask" activiti:ruleVariablesInput="${order}"

      activiti:rules="rule1, rule2" />

In this case only rule1 and rule2 are executed.

You can also define a list of rules that should be excluded from execution.

<businessRuleTask id="businessRuleTask" activiti:ruleVariablesInput="${order}"

      activiti:rules="rule1, rule2" exclude="true" />

In this case all rules deployed in the same BAR file as the process definition will be executed, except for rule1 and rule2.

As mentioned earlier another option is to hook in the implementation of the BusinessRuleTask yourself:

<businessRuleTask id="businessRuleTask" activiti:class="${MyRuleServiceDelegate}" />

Now the BusinessRuleTask behaves exactly like a ServiceTask, but still keeps the BusinessRuleTask icon to visualize that we do business rule processing here.

**Email Task**

Activiti allows to enhance business processes with automatic mail service tasks that send e-mails to one or more recipients, including support for cc, bcc, HTML content, ... etc. Note that the mail task is **not** an 'official' task of the BPMN 2.0 spec (and it does not have a dedicated icon as a consequence). Hence, in Activiti the mail task is implemented as a dedicated service task.

**Mail server configuration**

The Activiti engine sends e-mails trough an external mail server with SMTP capabilities. To actually send e-mails, the engine needs to know how to reach the mail server. Following properties can be set in the *activiti.cfg.xml* configuration file:

**Table 8.1. Mail server configuration**

| Property | Required? | Description |
| --- | --- | --- |
| mailServerHost | no | The hostname of your mail server (e.g. mail.mycorp.com). Default is localhost |
| mailServerPort | yes, if not on the default port | The port for SMTP traffic on the mail server. The default is *25* |
| mailServerDefaultFrom | no | The default e-mail address of the sender of e-mails, when none is provided by the user. By default this is *activiti@activiti.org* |
| mailServerUsername | if applicable for your server | Some mail servers require credentials for sending e-mail. By default not set. |
| mailServerPassword | if applicable for your server | Some mail servers require credentials for sending e-mail. By default not set. |
| mailServerUseSSL | if applicable for your server | Some mail servers require ssl communication. By default set to false. |

**Defining an Email Task**

The Email task is implemented as a dedicated [Service Task](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTask) and is defined by setting *'mail'* for the *type* of the service task.

<serviceTask id="sendMail" **activiti:type="mail"**>

The Email task is configured by [field injection](http://www.mossle.com/docs/activiti/#serviceTaskFieldInjection). All the values for these properties can contain EL expression, which are resolved at runtime during process execution. Following properties can be set:

**Table 8.2. Mail server configuration**

| Property | Required? | Description |
| --- | --- | --- |
| to | yes | The recipients if the e-mail. Multiple recipients are defined in a comma-separated list |
| from | no | The sender e-mail address. If not provided, the [default configured](http://www.mossle.com/docs/activiti/#bpmnEmailTaskServerConfiguration) from address is used. |
| subject | no | The subject of the e-mail. |
| cc | no | The cc's of the e-mail. Multiple recipients are defined in a comma-separated list |
| bcc | no | The bcc's of the e-mail. Multiple recipients are defined in a comma-separated list |
| charset | no | Allows to change the charset of the email, which is necessary for many non-English languages. |
| html | no | A piece of HTML that is the content of the e-mail. |
| text | no | The content of the e-mail, in case one needs to send plain none-rich e-mails. Can be used in combination with *html*, for e-mail clients that don't support rich content. The client will then fall back to this text-only alternative. |

**Example usage**

The following XML snippet shows an example of using the Email Task.

<serviceTask id="sendMail" activiti:type="mail">

  <extensionElements>

    <activiti:field name="from" stringValue="order-shipping@thecompany.com" />

    <activiti:field name="to" expression="${recipient}" />

    <activiti:field name="subject" expression="Your order ${orderId} has been shipped" />

    <activiti:field name="html">

      <activiti:expression>

        <![CDATA[

          <html>

            <body>

              Hello ${male ? 'Mr.' : 'Mrs.' } ${recipientName},<br/><br/>

              As of ${now}, your order has been <b>processed and shipped</b>.<br/><br/>

              Kind regards,<br/>

              TheCompany.

            </body>

          </html>

        ]]>

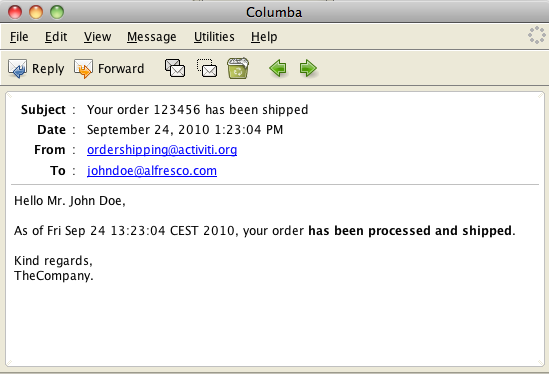
      </activiti:expression>

    </activiti:field>

  </extensionElements>

</serviceTask>

with the following result:



**Mule Task**

The mule task allows to send messages to Mule enhancing the integration features of Activiti. Note that the mule task is **not** an 'official' task of the BPMN 2.0 spec (and it does not have a dedicated icon as a consequence). Hence, in Activiti the mule task is implemented as a dedicated service task.

**Defining an Mule Task**

The Mule task is implemented as a dedicated [Service Task](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTask) and is defined by setting *'mule'* for the *type* of the service task.

<serviceTask id="sendMule" **activiti:type="mule"**>

The Mule task is configured by [field injection](http://www.mossle.com/docs/activiti/#serviceTaskFieldInjection). All the values for these properties can contain EL expression, which are resolved at runtime during process execution. Following properties can be set:

**Table 8.3. Mule server configuration**

| Property | Required? | Description |
| --- | --- | --- |
| endpointUrl | yes | The Mule endpoint you want to invoke. |
| language | yes | The language you want to use to evaluate the payloadExpression field. |
| payloadExpression | yes | An expression that will be the message's payload. |
| resultVariable | no | The name of the variable which will store the result of the invocation. |

**Example usage**

The following XML snippet shows an example of using the Mule Task.

      <extensionElements>

        <activiti:field name="endpointUrl">

          <activiti:string>vm://in</activiti:string>

        </activiti:field>

        <activiti:field name="language">

          <activiti:string>juel</activiti:string>

        </activiti:field>

        <activiti:field name="payloadExpression">

          <activiti:string>"hi"</activiti:string>

        </activiti:field>

        <activiti:field name="resultVariable">

          <activiti:string>theVariable</activiti:string>

        </activiti:field>

      </extensionElements>

**Camel Task**

The Camel task allows to send messages to and receive messages from Camel and thereby enhances the integration features of Activiti. Note that the Camel task is **not** an 'official' task of the BPMN 2.0 spec (and it does not have a dedicated icon as a consequence). Hence, in Activiti the Camel task is implemented as a dedicated service task. Also note to include the Activiti Camel module in your project to use the Camel task functionality.

**Defining a Camel Task**

The Camel task is implemented as a dedicated [Service Task](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTask) and is defined by setting *'camel'* for the *type* of the service task.

<serviceTask id="sendCamel" **activiti:type="camel"**>

The process definition itself needs nothing else then the camel type definition on a service task. The integration logic is all delegated to the Camel container. By default the Activiti Engine looks for a camelContext bean in the Spring container. The camelContext bean defines the Camel routes that will be loaded by the Camel container. In the following example the routes are loaded from a specific Java package, but you can also define routes directly in the Spring configuration itself.

<camelContext id="camelContext" xmlns="http://camel.apache.org/schema/spring">

  <packageScan>

    <package>org.activiti.camel.route</package>

  </packageScan>

</camelContext>

For more documentation about Camel routes you can look on the Camel website. This user guide only shows an example route definition related to the Activiti integration. When a Camel task is executed a route is searched for that corresponds to the key of process definition and the identifier of that task, like in the short example above "sendCamel". An example route defined in Java can look like this:

public class AsyncCamelRoute extends RouteBuilder {

  @Override

  public void configure() throws Exception {

    from("activiti:asyncCamelProcess:serviceTaskAsync1?copyVariablesToProperties=true").setHeader("destination", constant("activiti:asyncCamelProcess:receive1")).to("seda:asyncQueue");

    from("seda:asyncQueue").to("bean:sleepBean?method=sleep").to("seda:receiveQueue");

    from("activiti:asyncCamelProcess:serviceTaskAsync2?copyVariablesToProperties=true").setHeader("destination", constant("activiti:asyncCamelProcess:receive2")).to("seda:asyncQueue2");

    from("seda:asyncQueue2").to("bean:sleepBean?method=sleep").to("seda:receiveQueue");

    from("seda:receiveQueue").recipientList(header("destination"));

  }

}

Note the from endpoint definition "activiti:asyncCamelProcess:serviceTaskAsync1" where activiti relates to the Activiti Camel component. asyncCamelProcess refers to the process definition key of the process definition. And serviceTaskAsync1 relates to the identifier of the Camel task. The process variables can be copied to the body or the properties of the Camel payload. In this example the variables are copied as message properties (the default). copyVariablesToBodyAsMap=true will copy the variables to the body of the Camel message in a Map instance. copyCamelBodyToBody=true will copy the camelBody process variable to the body of the Camel message In this example you can also see that a Camel message can be sent to a receive task of an Activiti process instance. In this example a bit of additional Camel logic is used to send the Camel message to the receive task that's defined in the message header destination property. But eventually the message will be sent to the receive1 or receive2 receive tasks. By default the message body of the Camel message is expected as a Map (all Map entries will be copied to the Activiti execution) or as a String that will be copied to the camelBody process variable. When defining the copyVariablesToProperties=true option the message properties are copied to the Activiti execution.

In addition to sending messages from a process instance to Camel and back again you can also start a process instance from a Camel message.

from("direct:start").to("activiti:camelProcess");

Note that camelProcess is the process definition key of the process definition for which a process instance will be created when a message is read from the direct:start queue.

If you want to define multiple Camel context beans and/or want to use a different bean name, this can be overriden on the Camel task definition like this:

<serviceTask id="serviceTask1" activiti:type="camel">

  <extensionElements>

    <activiti:field name="camelContext" stringValue="customCamelContext" />

  </extensionElements>

</serviceTask>

You can also override the default Camel behavior class (org.activiti.camel.impl.CamelBehaviorDefaultImpl). This can be handy to customize the Camel logic or to change the default behavior of copying the process variables to something else. The Activiti Camel module provide two additional Camel behavior classes, CamelBehaviorBodyAsMapImpl and CamelBehaviorCamelBodyImpl. CamelBehaviorBodyAsMapImpl copies the process variables to a Map instance in the Camel message body by default (remember that you can override this on the endpoint using for example copyVariablesToProperties=true). CamelBehaviorCamelBodyImpl expects a camelBody process variable and copies it to the Camel message body. The default Camel behavior class can be overriden by using the following field definition:

<serviceTask id="serviceTask1" activiti:type="camel">

  <extensionElements>

    <activiti:field name="camelBehaviorClass" stringValue="org.activiti.camel.impl.CamelBehaviorCamelBodyImpl" />

  </extensionElements>

</serviceTask>

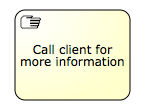
**Manual Task**

**Description**

A *Manual Task* defines a task that is external to the BPM engine. It is used to model work that is done by somebody, which the engine does not need to know of, nor is there a system or UI interface. For the engine, a manual task is handled as a **pass-through activity**, automatically continuing the process from the moment process execution arrives into it.

**Graphical Notation**

A manual task is visualized as a rounded rectangle, with a little 'hand' icon in the upper left corner



**XML representation**

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

**Java Receive Task**

**Description**

A Receive Task is a simple task that waits for the arrival of a certain message. Currently, we have only implemented Java semantics for this task. When process execution arrives at a Receive Task, the process state is committed to the persistence store. This means that the process will stay in this wait state, until a specific message is received by the engine, which triggers the continuation of the process past the Receive Task.

**Graphical notation**

A Receive Task is visualized as a task (rounded rectangle) with a message icon in the top left corner. The message is white (a black message icon would have send semantics)



**XML representation**

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

To continue a process instance that is currently waiting at such a Receive Task, the *runtimeService.signal(executionId)* must be called using the id of the execution that arrived in the Receive Task. The following code snippet shows how this works in practice:

ProcessInstance pi = runtimeService.startProcessInstanceByKey("receiveTask");

Execution execution = runtimeService.createExecutionQuery()

  .processInstanceId(pi.getId())

  .activityId("waitState")

  .singleResult();

assertNotNull(execution);

**runtimeService.signal(execution.getId());**

**Shell Task**

**Description**

The shell task allows to run shell scripts and commands. Note that the Shell task is **not** an 'official' task of BPMN 2.0 spec (and it does not have a dedicated icon as a consequence).

**defining a shell task**

The shell task is implemented as a dedicated [Service Task](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTask) and is defined by setting *'shell'* for the *type* of the service task.

<serviceTask id="shellEcho" **activiti:type="shell"**>

The Shell task is configured by [field injection](http://www.mossle.com/docs/activiti/#serviceTaskFieldInjection). All the values for these properties can contain EL expression, which are resolved at runtime during process execution. Following properties could be set:

**Table 8.4. Shell task parameter configuration**

| Property | Required? | Type | Description | Default |
| --- | --- | --- | --- | --- |
| command | yes | String | Shell command to execute. |  |
| arg0-5 | no | String | Parameter 0 to Parameter 5 |  |
| wait | no | true/false | wait if necessary, until the shell process has terminated. | true |
| redirectError | no | true/false | Merge standard error with the standard output. | false |
| cleanEnv | no | true/false | Shell process does not inherit current environment. | false |
| outputVariable | no | String | Name of variable to contain the output | Output is not recorded. |
| errorCodeVariable | no | String | Name of variable to contain result error code | Error level is not registered. |
| directory | no | String | Default directory of shell process | Current directory |

**Example usage**

The following XML snippet shows an example of using the shell Task. It runs shell script "cmd /c echo EchoTest", waits for it to be terminated and puts the result in resultVar

<serviceTask id="shellEcho" activiti:type="shell" >

  <extensionElements>

    <activiti:field name="command" stringValue="cmd" />

    <activiti:field name="arg1" stringValue="/c" />

    <activiti:field name="arg2" stringValue="echo" />

    <activiti:field name="arg3" stringValue="EchoTest" />

    <activiti:field name="wait" stringValue="true" />

    <activiti:field name="outputVariable" stringValue="resultVar" />

  </extensionElements>

</serviceTask>

**Execution listener**

**Compatibility note**: After releasing 5.3, we discovered that execution listeners and task listeners and expressions were still in non-public API. Those classes were in subpackages of org.activiti.engine.impl..., which has impl in it). org.activiti.engine.impl.pvm.delegate.ExecutionListener, org.activiti.engine.impl.pvm.delegate.TaskListener and org.activiti.engine.impl.pvm.el.Expression have been deprecated. From now on, you should use org.activiti.engine.delegate.ExecutionListener, org.activiti.engine.delegate.TaskListener and org.activiti.engine.delegate.Expression. In the new publicly available API, access to ExecutionListenerExecution.getEventSource() has been removed. Apart from the deprecation compiler warning, the existing code should run fine. But consider switching to the new public API interfaces (without .impl. in the package name).

Execution listeners allow you to execute external Java code or evaluate an expression when certain events occur during process execution. The events that can be captured are:

* Start and ending of a process instance.
* Taking a transition.
* Start and ending of an activity.
* Start and ending of a gateway.
* Start and ending of intermediate events.
* Ending an start event or starting an end event.

The following process definition contains 3 execution listeners:

  <process id="executionListenersProcess">

**<extensionElements>**

**<activiti:executionListener class="org.activiti.examples.bpmn.executionlistener.ExampleExecutionListenerOne" event="start" />**

**</extensionElements>**

    <startEvent id="theStart" />

    <sequenceFlow sourceRef="theStart" targetRef="firstTask" />

    <userTask id="firstTask" />

    <sequenceFlow sourceRef="firstTask" targetRef="secondTask">

**<extensionElements>**

**<activiti:executionListener class="org.activiti.examples.bpmn.executionListener.ExampleExecutionListenerTwo" />**

**</extensionElements>**

    </sequenceFlow>

    <userTask id="secondTask" >

**<extensionElements>**

**<activiti:executionListener expression="${myPojo.myMethod(execution.event)}" event="end" />**

**</extensionElements>**

    </userTask>

    <sequenceFlow sourceRef="secondTask" targetRef="thirdTask" />

    <userTask id="thirdTask" />

    <sequenceFlow sourceRef="thirdTask" targetRef="theEnd" />

    <endEvent id="theEnd" />

  </process>

The first execution listener is notified when the process starts. The listener is an external Java-class (like ExampleExecutionListenerOne) and should implement org.activiti.engine.delegate.ExecutionListener interface. When the event occurs (in this case end event) the method notify(ExecutionListenerExecution execution) is called.

public class ExampleExecutionListenerOne implements **ExecutionListener** {

**public void notify(ExecutionListenerExecution execution) throws Exception {**

    execution.setVariable("variableSetInExecutionListener", "firstValue");

    execution.setVariable("eventReceived", execution.getEventName());

  }

}

It is also possible to use a delegation class that implements the org.activiti.engine.delegate.JavaDelegate interface. These delegation classes can then be reused in other constructs, such as a delegation for a serviceTask.

The second execution listener is called when the transition is taken. Note that the listener element doesn't define an event, since only take events are fired on transitions. **Values in the event attribute are ignored when a listener is defined on a transition.**

The last execution listener is called when activity secondTask ends. Instead of using the class on the listener declaration, a expression is defined instead which is evaluated/invoked when the event is fired.

<activiti:executionListener expression="**${myPojo.myMethod(execution.eventName)}**" event="end" />

As with other expressions, execution variables are resolved and can be used. Because the execution implementation object has a property that exposes the event name, it's possible to pass the event-name to your methods using execution.eventName.

Execution listeners also support using a delegateExpression, [similar to a service task](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTaskXML).

<activiti:executionListener event="start" delegateExpression="${myExecutionListenerBean}" />

In Activiti 5.12 we also introduced a new type of execution listener, the org.activiti.engine.impl.bpmn.listener.ScriptExecutionListener. This script execution listener allows you to execute a piece of script logic for an execution listener event.

<activiti:executionListener event="start" class="org.activiti.engine.impl.bpmn.listener.ScriptExecutionListener" >

  <activiti:field name="script">

    <activiti:string>

      def bar = "BAR";  // local variable

      foo = "FOO"; // pushes variable to execution context

      execution.setVariable("var1", "test"); // test access to execution instance

      bar // implicit return value

    </activiti:string>

  </activiti:field>

  <activiti:field name="language" stringValue="groovy" />

  <activiti:field name="resultVariable" stringValue="myVar" />

<activiti:executionListener>

**Field injection on execution listeners**

When using an execution listener that is configured with the class attribute, field injection can be applied. This is exactly the same mechanism as used [Service task field injection](http://www.mossle.com/docs/activiti/#serviceTaskFieldInjection), which contains an overview of the possibilities provided by field injection.

The fragment below shows a simple example process with an execution listener with fields injected.

 <process id="executionListenersProcess">

**<extensionElements>**

**<activiti:executionListener class="org.activiti.examples.bpmn.executionListener.ExampleFieldInjectedExecutionListener" event="start">**

**<activiti:field name="fixedValue" stringValue="Yes, I am " />**

**<activiti:field name="dynamicValue" expression="${myVar}" />**

**</activiti:executionListener>**

**</extensionElements>**

    <startEvent id="theStart" />

    <sequenceFlow sourceRef="theStart" targetRef="firstTask" />

    <userTask id="firstTask" />

    <sequenceFlow sourceRef="firstTask" targetRef="theEnd" />

    <endEvent id="theEnd" />

  </process>

public class ExampleFieldInjectedExecutionListener implements ExecutionListener {

**private Expression fixedValue;**

**private Expression dynamicValue;**

  public void notify(ExecutionListenerExecution execution) throws Exception {

    execution.setVariable("var", **fixedValue.getValue(execution).toString()** + **dynamicValue.getValue(execution).toString()**);

  }

}

The class ExampleFieldInjectedExecutionListener concatenates the 2 injected fields (one fixed an the other dynamic) and stores this in the process variable 'var'.

@Deployment(resources = {"org/activiti/examples/bpmn/executionListener/ExecutionListenersFieldInjectionProcess.bpmn20.xml"})

public void testExecutionListenerFieldInjection() {

  Map<String, Object> variables = new HashMap<String, Object>();

**variables.put("myVar", "listening!");**

  ProcessInstance processInstance = runtimeService.startProcessInstanceByKey("executionListenersProcess", variables);

  Object varSetByListener = runtimeService.getVariable(processInstance.getId(), "var");

  assertNotNull(varSetByListener);

  assertTrue(varSetByListener instanceof String);

  // Result is a concatenation of fixed injected field and injected expression

**assertEquals("Yes, I am listening!", varSetByListener);**

}

**Task listener**

A *task listener* is used to execute custom Java logic or an expression upon the occurrence of a certain task-related event.

A task listener can only be added in the process definition as a child element of a [user task](http://www.mossle.com/docs/activiti/#bpmnUserTask). Note that this also must happen as a child of the *BPMN 2.0 extensionElements* and in the *activiti* namespace, since a task listener is an Activiti-specific construct.

<userTask id="myTask" name="My Task" >

  <extensionElements>

**<activiti:taskListener event="create" class="org.activiti.MyTaskCreateListener" />**

  </extensionElements>

</userTask>

A *task listener* supports following attributes:

* **event** (required): the type of task event on which the task listener will be invoked. Possible events are
  + **create**: occurs when the task has been created an **all task properties are set**.
  + **assignment**: occurs when the task is assigned to somebody. Note: when process execution arrives in a userTask, first an *assignment* event will be fired, **before** the *create* event is fired. This might seem an unnatural order, but the reason is pragmatic: when receiving the *create* event, we usually want to inspect all properties of the task including the assignee.
  + **complete**: occurs when the task is completed and just before the task is deleted from the runtime data.
* **class**: the delegation class that must be called. This class must implement the org.activiti.engine.impl.pvm.delegate.TaskListener interface.
* public class MyTaskCreateListener implements TaskListener {
* public void notify(DelegateTask delegateTask) {
* // Custom logic goes here
* }

}

It is also possible to use [field injection](http://www.mossle.com/docs/activiti/#serviceTaskFieldInjection) to pass process variables or the execution to the delegation class. Note that an instance of the delegation class is created upon process deployment (as is the case with any class delegation in Activiti), which means that the instance is shared between all process instance executions.

* **expression**: (cannot be used together with the *class* attribute): specifies an expression that will be executed when the event happens. It is possible to pass the DelegateTask object and the name of the event (using task.eventName) as parameter to the called object.

<activiti:taskListener event="create" expression="${myObject.callMethod(task, task.eventName)}" />

* **delegateExpression** allows to specify an expression that resolves to an object implementing the TaskListener interface, [similar to a service task](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTaskXML).

<activiti:taskListener event="create" delegateExpression="${myTaskListenerBean}" />

* In Activiti 5.12 we also introduced a new type of task listener, the org.activiti.engine.impl.bpmn.listener.ScriptTaskListener. This script task listener allows you to execute a piece of script logic for an task listener event.
* <activiti:taskListener event="complete" class="org.activiti.engine.impl.bpmn.listener.ScriptTaskListener" >
* <activiti:field name="script">
* <activiti:string>
* def bar = "BAR";  // local variable
* foo = "FOO"; // pushes variable to execution context
* task.setOwner("kermit"); // test access to task instance
* bar // implicit return value
* </activiti:string>
* </activiti:field>
* <activiti:field name="language" stringValue="groovy" />
* <activiti:field name="resultVariable" stringValue="myVar" />

<activiti:taskListener>

**Multi-instance (for each)**

**Description**

A *multi-instance activity* is a way of defining repetition for a certain step in a business process. In programming concepts, a multi-instance matches the **for each** construct: it allows to execute a certain step or even a complete subprocess for each item in a given collection, **sequentially or in parallel**.

A *multi-instance* is a regular activity that has extra properties defined (so-called *'multi-instance* characteristics'') which will cause the activity to be executed multiple times at runtime. Following activities can become a *multi-instance activity:*

* [User Task](http://www.mossle.com/docs/activiti/#bpmnUserTask)
* [Script Task](http://www.mossle.com/docs/activiti/#bpmnScriptTask)
* [Java Service Task](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTask)
* [Web Service Task](http://www.mossle.com/docs/activiti/#bpmnWebserviceTask)
* [Business Rule Task](http://www.mossle.com/docs/activiti/#bpmnBusinessRuleTask)
* [Email Task](http://www.mossle.com/docs/activiti/#bpmnEmailTask)
* [Manual Task](http://www.mossle.com/docs/activiti/#bpmnManualTask)
* [Receive Task](http://www.mossle.com/docs/activiti/#bpmnReceiveTask)
* [(Embedded) Sub-Process](http://www.mossle.com/docs/activiti/#bpmnSubProcess)
* [Call Activity](http://www.mossle.com/docs/activiti/#bpmnCallActivity)

A [Gateway](http://www.mossle.com/docs/activiti/#bpmnGateways) or [Event](http://www.mossle.com/docs/activiti/#bpmnEvents) can **not** become multi-instance.

As required by the spec, each parent execution of the created executions for each instance will have following variables:

* **nrOfInstances**: the total number of instances
* **nrOfActiveInstances**: the number of currently active, i.e. not yet finished, instances. For a sequential multi-instance, this will always be 1.
* **nrOfCompletedInstances**: the number of already completed instances.

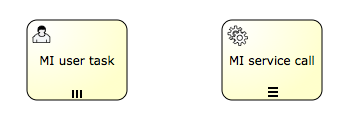
These values can be retrieved by calling the execution.getVariable(x) method.

Additionally, each of the created executions will have an execution-local variable (i.e. not visible for the other executions, and not stored on process instance level) :

* **loopCounter**: indicates the *index in the for-each loop* of that particular instance.

**Graphical notation**

If an activity is multi-instance, this is indicated by three short lines at the bottom of that activity. Three *vertical* lines indicates that the instances will be executed in parallel, while three *horizontal* lines indicate sequential execution.



**Xml representation**

To make an activity multi-instance, the activity xml element must have a multiInstanceLoopCharacteristics child element.

<multiInstanceLoopCharacteristics isSequential="false|true">

 ...

</multiInstanceLoopCharacteristics>

The **isSequential** attribute indicates if the instances of that activity are executed sequentially or parallel.

The number of instances are **calculated once, when entering the activity**. There are a few ways of configuring this. On way is directly specifying a number, by using the **loopCardinality** child element.

<multiInstanceLoopCharacteristics isSequential="false|true">

  <loopCardinality>5</loopCardinality>

</multiInstanceLoopCharacteristics>

Expressions that resolve to a positive number are also possible:

<multiInstanceLoopCharacteristics isSequential="false|true">

  <loopCardinality>${nrOfOrders-nrOfCancellations}</loopCardinality>

</multiInstanceLoopCharacteristics>

Another way to define the number of instances, is to specify the name of a process variable which is a collection using the loopDataInputRef child element. For each item in the collection, an instance will be created. Optionally, it is possible to set that specific item of the collection for the instance using the inputDataItem child element. This is shown in the following XML example:

<userTask id="miTasks" name="My Task ${loopCounter}" activiti:assignee="${assignee}">

  <multiInstanceLoopCharacteristics isSequential="false">

    <loopDataInputRef>assigneeList</loopDataInputRef>

    <inputDataItem name="assignee" />

  </multiInstanceLoopCharacteristics>

</userTask>

Suppose the variable assigneeList contains the values [kermit, gonzo, foziee]. In the snippet above, three user tasks will be created in parallel. Each of the executions will have a process variable named assignee containing one value of the collection, which is used to assign the user task in this example.

The downside of the loopDataInputRef and inputDataItem is that 1) the names are pretty hard to remember and 2) due to the BPMN 2.0 schema restrictions they can't contain expressions. Activiti solves this by offering the **collection** and **elementVariable** attributes on the multiInstanceCharacteristics:

<userTask id="miTasks" name="My Task" activiti:assignee="${assignee}">

  <multiInstanceLoopCharacteristics isSequential="true"

**activiti:collection="${myService.resolveUsersForTask()}" activiti:elementVariable="assignee"** >

  </multiInstanceLoopCharacteristics>

</userTask>

A multi-instance activity ends when all instances are finished. However, it is possible to specify an expression that is evaluated every time one instance ends. When this expression evaluates to true, all remaining instances are destroyed and the multi-instance activity ends, continuing the process. Such an expression must be defined in the **completionCondition** child element.

<userTask id="miTasks" name="My Task" activiti:assignee="${assignee}">

  <multiInstanceLoopCharacteristics isSequential="false"

     activiti:collection="assigneeList" activiti:elementVariable="assignee" >

**<completionCondition>${nrOfCompletedInstances/nrOfInstances >= 0.6 }</completionCondition>**

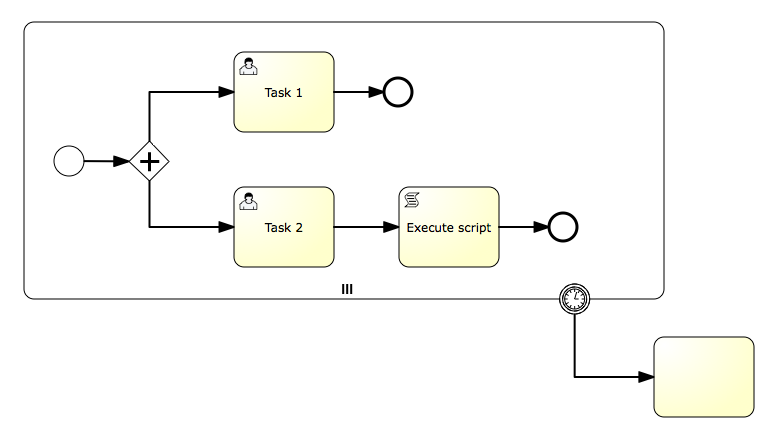
  </multiInstanceLoopCharacteristics>

</userTask>

In this example, there will be parallel instances created for each element of the assigneeList collection. However, when 60% of the tasks are completed, the other tasks are deleted and the process continues.

**Boundary events and multi-instance**

Since a multi-instance is a regular activity, it is possible to define a [boundary event](http://www.mossle.com/docs/activiti/#bpmnBoundaryEvent) on its boundary. In case of an interrupting boundary event, when the event is caught, **all instances** that are still active will be destroyed. Take for example following multi-instance subprocess:



Here, all instances of the subprocess will be destroyed when the timer fires, regardless of how many instances there are or which inner activities are currently not yet completed.

**Compensation Handlers**

**Description**

[**[EXPERIMENTAL]**](http://www.mossle.com/docs/activiti/#experimental)

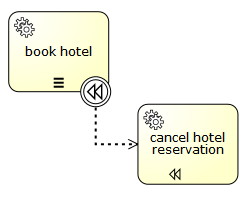
If an activity is used for compensating the effects of another activity, it can be declared to be a compensation handler. Compensation handlers are not contained in normal flow and are only executed when a compensation event is thrown.

Compensation handlers must not have incoming or outgoing sequence flows.

A compensation handler must be associated with a compensation boundary event using a directed association.

**Graphical notation**

If an activity is a compensation handler, the compensation event icon is displayed in the center bottom area. The following excerpt from a process diagram shows a service task with an attached compensation boundary event which is associated to a compensation handler. Notice the compensation handler icon in the bottom canter area of the "cancel hotel reservation" service task



**XML representation**

In order to declare an activity to be a compensation handler, we need to set the attribute isForCompensation to true:

<serviceTask id="undoBookHotel" isForCompensation="true" activiti:class="...">

</serviceTask>

**Sub-Processes and Call Activities**

**Sub-Process**

**Description**

A *Sub-Process* is an activity that contains other activities, gateways, events, etc. which on itself form a process that is part of the bigger process. A *Sub-Process* is completely defined inside a parent process (that's why it's often called an *embedded* Sub-Process).

Sub-Processes have two major use cases:

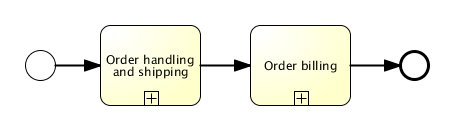
* Sub-Processes allow **hierarchical modeling**. Many modeling tools allow that Sub-Processes can be *collapsed*, hiding all the details of the Sub-Process and displaying a high-level end-to-end overview of the business process.
* A Sub-Process creates a new **scope for events**. Events that are thrown during execution of the Sub-Process, can be caught by [a boundary event](http://www.mossle.com/docs/activiti/#bpmnBoundaryEvent) on the boundary of the Sub-Process, thus creating a scope for that event limited to the Sub-Process.

Using a Sub-Process does impose some constraints:

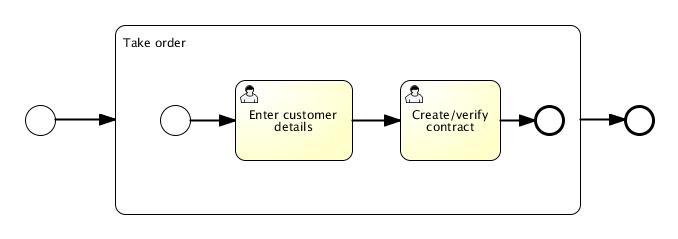
* A Sub-Process can only have **one none start event**, no other start event types are allowed. A Sub-Process must **at least have one end event**. Note that the BPMN 2.0 specification allows to omit the start and end events in a Sub-Process, but the current Activiti implementation does not support this.
* **Sequence flow can not cross Sub-Process boundaries.**

**Graphical Notation**

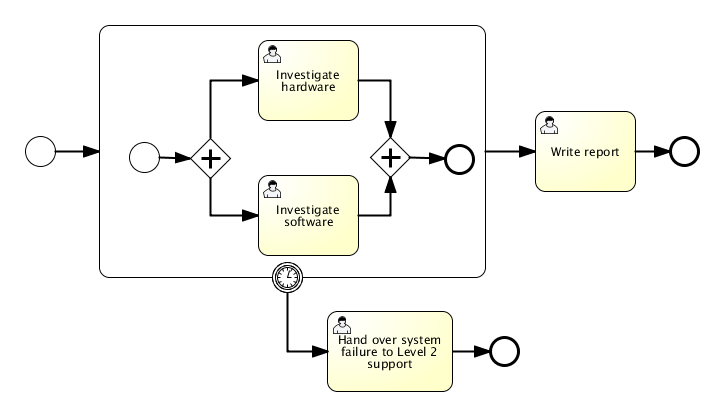
A Sub-Process is visualized as a typical activity, i.e. a rounded rectangle. In case the Sub-Process is *collapsed*, only the name and a plus-sign are displayed, giving a high-level overview of the process:



In case the Sub-Process is *expanded*, the steps of the Sub-Process are displayed within the Sub-Process boundaries:



One of the main reasons to use a Sub-Process, is to define a scope for a certain event. The following process model shows this: both the *investigate software/investigate hardware* tasks need to be done in parallel, but both tasks need to be done within a certain time, before *Level 2 support* is consulted. Here, the scope of the timer (i.e. which activities must be done in time) is constrained by the Sub-Process.



**XML representation**

A Sub-Process is defined by the *subprocess* element. All activities, gateways, events, etc. that are part of the Sub-Process, need to be enclosed within this element.

<subProcess id="subProcess">

  <startEvent id="subProcessStart" />

  ... other Sub-Process elements ...

  <endEvent id="subProcessEnd" />

 </subProcess>

**Event Sub-Process**

**Description**

The Event Sub-Process is new in BPMN 2.0. An Event Sub-Process is a subprocess that is triggered by an event. An Event Sub-Process can be added at the process level or at any subprocess level. The event used to trigger an event subprocess is configured using a start event. From this, it follows that none start events are not supported for Event Sub-Processes. An Event Sub-Process might be triggered using events like message events, error events, signal events, timer events, or compensation events. The subscription to the start event is created when the scope (process instance or subprocess) hosting the Event Sub-Process is created. The subscription is removed when the scope is destroyed.

An Event Sub-Process may be interrupting or non-interrupting. An interrupting subprocess cancels any executions in the current scope. A non-interrupting Event Sub-Process spawns a new concurrent execution. While an interrupting Event Sub-Process can only be triggered once for each activation of the scope hosting it, a non-interrupting Event Sub-Process can be triggered multiple times. The fact whether the subprocess is interrupting is configured using the start event triggering the Event Sub-Process.

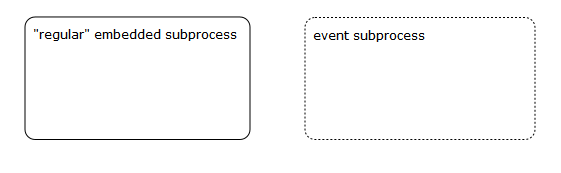
An Event Sub-Process must not have any incoming or outgoing sequence flows. Since an Event Sub-Process is triggered by an event, an incoming sequence flow makes no sense. When an Event Sub-Process is ended, either the current scope is ended (in case of an interrupting Event Sub-Process), or the concurrent execution spawned for the non-interrupting subprocess is ended.

**Current limitations:**

* Activiti only supports interrupting Event Sub-Processes.
* Activiti only supports Event Sub-Process triggered using an Error Start Event or Message Start Event.

**Graphical Notation**

An Event Sub-Process might be visualized as a an [embedded subprocess](http://www.mossle.com/docs/activiti/#bpmnSubProcessGraphicalNotation) with a dotted outline.



**XML representation**

An Event Sub-Process is represented using XML in the same way as a an [embedded subprocess](http://www.mossle.com/docs/activiti/). In addition the attribute triggeredByEvent must have the value true:

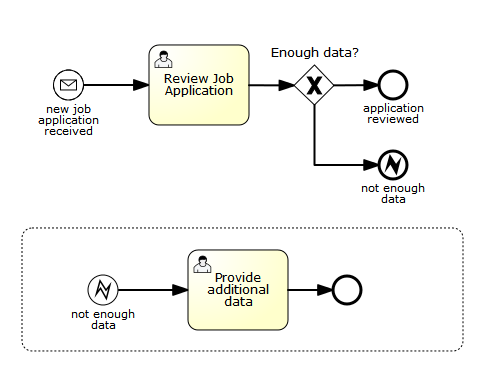
<subProcess id="eventSubProcess" triggeredByEvent="true">

        ...

</subProcess>

**Example**

The following is an example of an Event Sub-Process triggered using an Error Start Event. The Event Sub-Process is located at the "process level", i.e. is scoped to the process instance:



This is how the Event Sub-Process would look like in XML:

<subProcess id="eventSubProcess" triggeredByEvent="true">

        <startEvent id="catchError">

                <errorEventDefinition errorRef="error" />

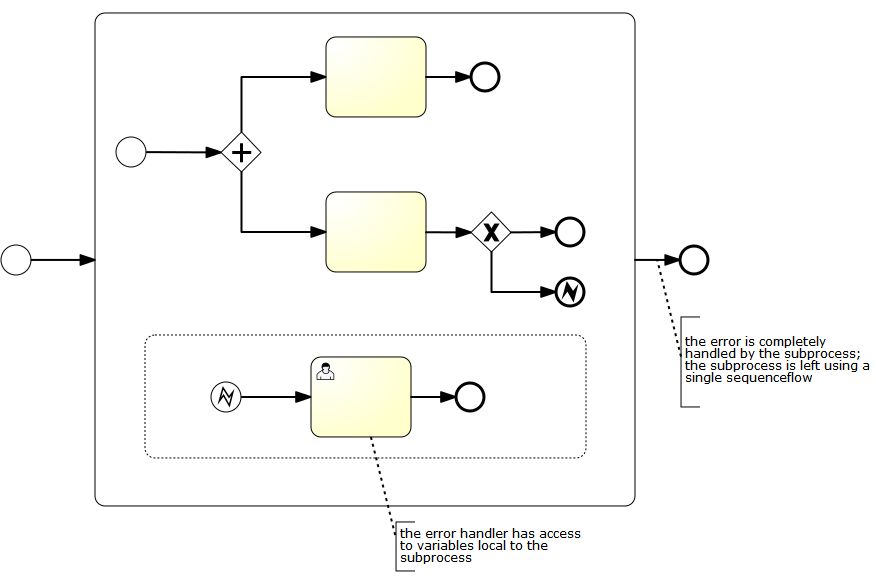
        </startEvent>

        <sequenceFlow id="flow2" sourceRef="catchError" targetRef="taskAfterErrorCatch" />

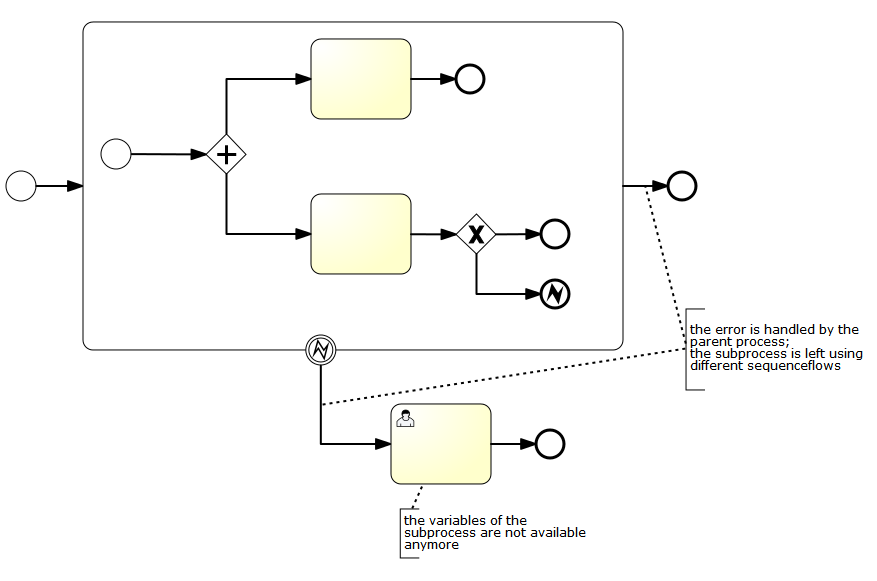
        <userTask id="taskAfterErrorCatch" name="Provide additional data" />

</subProcess>

As already stated, an Event Sub-Process can also be added to an embedded subprocess. If it is added to an embedded subprocess, it becomes an alternative to a boundary event. Consider the two following process diagrams. In both cases the embedded subprocess throws an error event. Both times the error is caught and handled using a user task.



as opposed to:



In both cases the same tasks are executed. However, there are differences between both modelling alternatives:

* The embedded subprocess is executed using the same execution which executed the scope it is hosted in. This means that an embedded subprocess has access to the variables local to it's scope. When using a boundary event, the execution created for executing the embedded subprocess is deleted by the sequence flow leaving the boundary event. This means that the variables created by the embedded subprocess are not available anymore.
* When using an Event Sub-Process, the event is completely handled by the subprocess it is added to. When using a boundary event, the event is handled by the parent process.

These two differences can help you decide whether a boundary event or an embedded subprocess is better suited for solving a particular process modeling / implementation problem.

**Transaction subprocess**

[**[EXPERIMENTAL]**](http://www.mossle.com/docs/activiti/#experimental)

**Description**

A transaction subprocess is an embedded subprocess, which can be used to group multiple activities to a transaction. A transaction is a logical unit of work which allows to group a set of individual activities, such that they either succeed or fail collectively.

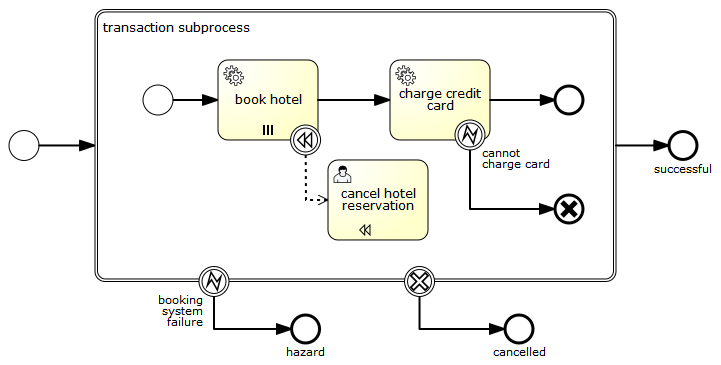
**Possible outcomes of a transaction:** A transaction can have three different outcomes:

* A transaction is *successful*, if it is neither cancelled not terminated by a hazard. If a transaction subprocess is successful, it is left using the outgoing sequenceflow(s). A successful transaction might be compensated if a compensation event is thrown later in the process.

Note: just as "ordinary" embedded subprocesses, a transaction may be compensated after successful completion using an intermediary throwing compensation event.

* A transaction is *cancelled*, if an execution reaches the cancel end event. In that case, all executions are terminated and removed. A single remaining execution is then set to the cancel boundary event, which triggers compensation. After compensation is completed, the transaction subprocess is left using the outgoing sequence flow(s) of the cancel boundary event.
* A transaction is ended by a *hazard*, if an error event is thrown, that is not caught within the scope of the transaction subprocess. (This also applies if the error is caught on the boundary of the transaction subprocess.) In this case, compensation is not performed.

The following diagram illustrates the three different outcomes:



**Relation to ACID transactions:** it is important not to confuse the bpmn transaction subprocess with technical (ACID) transactions. The bpmn transaction subprocess is not a way to scope technical transactions. In order to understand transaction management in Activiti, read the section on [concurrency and transactions](http://www.mossle.com/docs/activiti/#bpmnConcurrencyAndTransactions). A bpmn transaction is different from a technical transaction in the following ways:

* While an ACID transaction is typically short lived, a bpmn transaction may take hours, days or even months to complete. (Consider the case where one of the activities grouped by a transaction is a usertask, typically people have longer response times than applications. Or, in another situation, a bpmn transaction might wait for some business event to occur, like the fact that a particular order has been fulfilled.) Such operations usually take considerably longer to complete than updating a record in a database, or storing a message using a transactional queue.
* Because it is impossible to scope a technical transaction to the duration of a business activity, a bpmn transaction typically spans multiple ACID transactions.
* Since a bpmn transaction spans multiple ACID transactions, we loose ACID properties. For example, consider the example given above. Let's assume the "book hotel" and the "charge credit card" operations are performed in separate ACID transactions. Let's also assume that the "book hotel" activity is successful. Now we have an intermediary inconsistent state, because we have performed an hotel booking but have not yet charged the credit card. Now, in an ACID transaction, we would also perform different operations sequentially and thus also have an intermediary inconsistent state. What is different here, is that the inconsistent state is visible outside of the scope of the transaction. For example, if the reservations are made using an external booking service, other parties using the same booking service might already see that the hotel is booked. This means, that when implementing business transactions, we completely loose the isolation property (Granted: we usually also relax isolation when working with ACID transactions to allow for higher levels of concurrency, but there we have fine grained control and intermediary inconsistencies are only present for very short periods of times).
* A bpmn business transaction can also not be rolled back in the traditional sense. Since it spans multiple ACID transactions, some of these ACID transactions might already be committed at the time the bpmn transaction is cancelled. At this point, they cannot be rolled back anymore.

Since bpmn transactions are long-running in nature, the lack of isolation and a rollback mechanism need to be dealt with differently. In practice, there is usually no better solution than to deal with these problems in a domain specific way:

* The rollback is performed using compensation. If a cancel event is thrown in the scope of a transaction, the effects of all activities that executed successfully and have a compensation handler are compensated.
* The lack of isolation is also often dealt with using domain specific solutions. For instance, in the example above, an hotel room might appear to be booked to a second customer, before we have actually made sure that the first customer can pay for it. Since this might be undesirable from a business perspective, a booking service might choose to allow for a certain amount of overbooking.
* In addition, since the transaction can be aborted in case of a hazard, the booking service has to deal with the situation where a hotel room is booked but payment is never attempted (since the transaction was aborted). In that case the booking service might choose a strategy where a hotel room is reserved for a maximum period of time and if payment is not received until then, the booking is cancelled.

To sum it up: while ACID transactions offer a generic solution to such problems (rollback, isolation levels and heuristic outcomes), we need to find domain specific solutions to these problems when implementing business transactions.

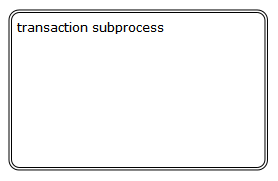
**Current limitations:**

* The bpmn specification requires that the process engine reacts to events issued by the underlying transaction protocol and for instance that a transaction is cancelled, if a cancel event occurs in the underlying protocol. As an embeddable engine, Activiti does currently not support this. (For some ramifications of this, see paragraph on consistency below.)

**Consistency on top of ACID transactions and optimistic concurrency:** A bpmn transaction guarantees consistency in the sense that either all activities compete successfully, or if some activity cannot be performed, the effects of all other successful activities are compensated. So either way we end up in a consistent state. However, it is important to recognize that in Activiti, the consistency model for bpmn transactions is superposed on top of the consistency model for process execution. Activiti executes processes in a transactional way. Concurrency is addressed using optimistic locking. In Activiti, bpmn error, cancel and compensation events are built on top of the same acid transactions and optimistic locking. For example, a cancel end event can only trigger compensation if it is actually reached. It is not reached if some undeclared exception is thrown by a service task before. Or, the effects of a compensation handler can not be committed if some other participant in the underlying ACID transaction sets the transaction to the state rollback-only. Or, when two concurrent executions reach a cancel end event, compensation might be triggered twice and fail with an optimistic locking exception. All of this is to say that when implementing bpmn transactions in Activiti, the same set of rules apply as when implementing "ordinary" processes and subprocesses. So to effectively guarantee consistency, it is important to implement processes in a way that does take the optimistic, transactional execution model into consideration.

**Graphical Notation**

An transaction subprocess might be visualized as a an [embedded subprocess](http://www.mossle.com/docs/activiti/#bpmnSubProcessGraphicalNotation) with a double outline.



**XML representation**

A transaction subprocess is represented using xml using the transaction tag:

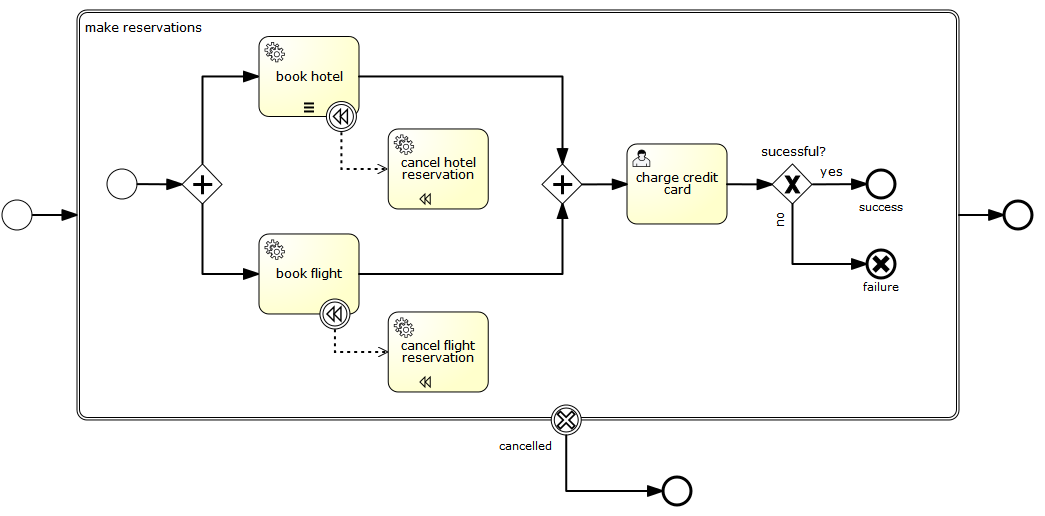
<transaction id="myTransaction" >

        ...

</transaction>

**Example**

The following is an example of a transaction subprocess:



**Call activity (subprocess)**

**Description**

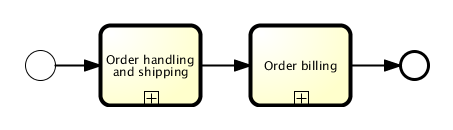
BPMN 2.0 makes a distinction between a regular [*subprocess*](http://www.mossle.com/docs/activiti/#bpmnSubProcess), often also called *embedded subprocess*, and the call activity, which looks very similar. From a conceptual point of view, both will call a subprocess when process execution arrives at the activity.

The difference is that the call activity references a process that is external to the process definition, whereas the [*subprocess*](http://www.mossle.com/docs/activiti/#bpmnSubProcess) is embedded within the original process definition. The main use case for the call activity is to have a reusable process definition that can be called from multiple other process definitions.

When process execution arrives in the *call activity*, a new execution is created that is a sub-execution of the execution that arrives in the call activity. This sub-execution is then used to execute the subprocess, potentially creating parallel child execution as within a regular process. The super-execution waits until the subprocess is completely ended, and continues the original process afterwards.

**Graphical Notation**

A call activity is visualized the same as a [subprocess](http://www.mossle.com/docs/activiti/#bpmnSubProcessGraphicalNotation), however with a thick border (collapsed and expanded). Depending on the modeling tool, a call activity can also be expanded, but the default visualization is the collapsed subprocess representation.



**XML representation**

A call activity is a regular activity, that requires a *calledElement* that references a process definition by its **key**. In practice, this means that the **id of the process** is used in the *calledElement*.

<callActivity id="callCheckCreditProcess" name="Check credit" calledElement="checkCreditProcess" />

Note that the process definition of the subprocess is **resolved at runtime**. This means that the subprocess can be deployed independently from the calling process, if needed.

**Passing variables**

You can pass process variables to the sub process and vice versa. The data is copied into the subprocess when it is started and copied back into the main process when it ends.

<callActivity id="callSubProcess" calledElement="checkCreditProcess" >

  <extensionElements>

          <activiti:in source="someVariableInMainProcess" target="nameOfVariableInSubProcess" />

          <activiti:out source="someVariableInSubProcss" target="nameOfVariableInMainProcess" />

  </extensionElements>

</callActivity>

We use an Activiti Extension as a shortcut for the BPMN standard elements called *dataInputAssociation* and *dataOutputAssociation*, which only work if you declare process variables in the BPMN 2.0 standard way.

It is possible to use expressions here as well:

<callActivity id="callSubProcess" calledElement="checkCreditProcess" >

        <extensionElements>

          <activiti:in sourceExpression="${x+5}"" target="y" />

          <activiti:out source="${y+5}" target="z" />

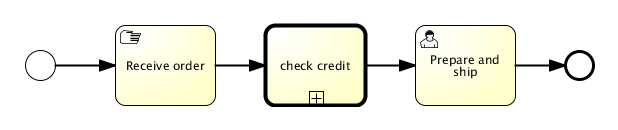
        </extensionElements>

</callActivity>

So in the end z = y+5 = x+5+5

**Example**

The following process diagram shows a simple handling of an order. Since the checking of the customer's credit could be common to many other processes, the *check credit step* is modeled here as a call activity.



The process looks as follows:

<startEvent id="theStart" />

<sequenceFlow id="flow1" sourceRef="theStart" targetRef="receiveOrder" />

<manualTask id="receiveOrder" name="Receive Order" />

<sequenceFlow id="flow2" sourceRef="receiveOrder" targetRef="callCheckCreditProcess" />

**<callActivity id="callCheckCreditProcess" name="Check credit" calledElement="checkCreditProcess" />**

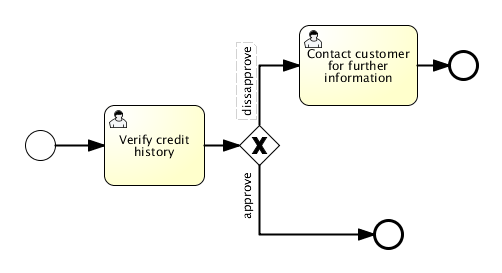
<sequenceFlow id="flow3" sourceRef="callCheckCreditProcess" targetRef="prepareAndShipTask" />

<userTask id="prepareAndShipTask" name="Prepare and Ship" />

<sequenceFlow id="flow4" sourceRef="prepareAndShipTask" targetRef="end" />

<endEvent id="end" />

The subprocess looks as follows:

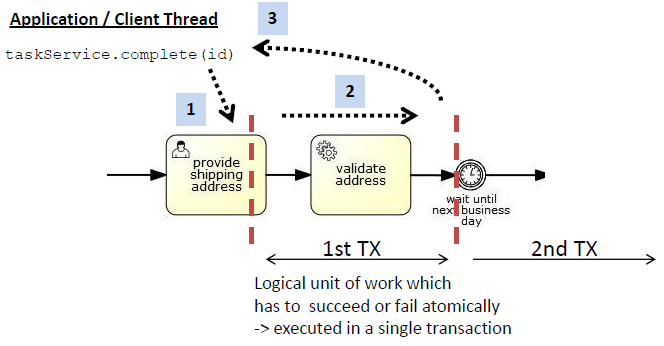


There is nothing special to the process definition of the subprocess. It could as well be used without being called from another process.

**Transactions and Concurrency**

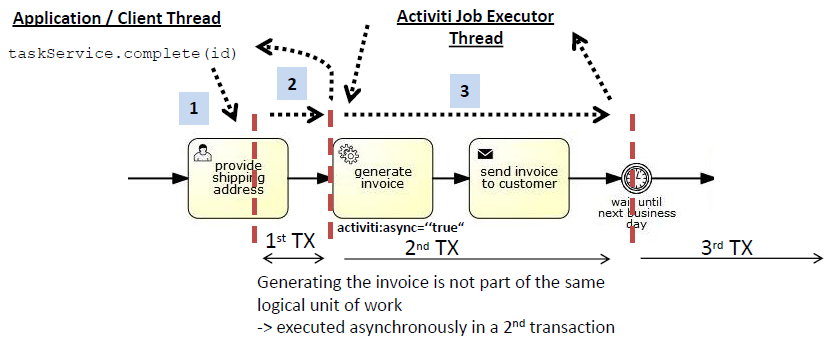
**Asynchronous Continuations**

Activiti executes processes in a transactional way which can be configured to suite your needs. Lets start by looking at how Activiti scopes transactions normally. If you trigger Activiti (i.e. start a process, complete a task, signal an execution), Activiti is going to advance in the process, until it reaches wait states on each active path of execution. More concretely speaking it performs a depth-first search through the process graph and returns if it has reached wait states on every branch of execution. A wait state is a task which is performed "later" which means that Activiti persists the current execution and waits to be triggered again. The trigger can either come from an external source for example if we have a user task or a receive message task, or from Activiti itself, if we have a timer event. This is illustrated in the following picture:



We see a segment of a BPMN processes with a usertask, a service task and a timer event. Completing the usertask and validating the address is part of the same unit of work, so it should succeed or fail atomically. That means that if the service task throws an exception we want to rollback the current transaction, such that the execution tracks back to the user task and the user task is still present in the database. This is also the default behavior of Activiti. In (1) an application or client thread completes the task. In that same thread Activiti is now executing the service and advances until it reaches a wait state, in this case the timer event (2). Then it returns the control to the caller (3) potentially committing the transaction (if it was started by Activiti).

In some cases this is not what we want. Sometimes we need custom control over transaction boundaries in a process, in order to be able to scope logical units of work. This is where asynchronous continuations come into play. Consider the following process (fragment):



This time we are completing the user task, generating an invoice and then send that invoice to the customer. This time the generation of the invoice is not part of the same unit of work so we do not want to rollback the completion of the usertask if generating an invoice fails. So what we want Activiti to do is complete the user task (1), commit the transaction and return the control to the calling application. Then we want to generate the invoice asynchronously, in a background thread. This background thread is the Activiti job executor (actually a thread pool) which periodically polls the database for jobs. So behind the scenes, when we reach the "generate invoice" task, we are creating a job "message" for Activiti to continue the process later and persisting it into the database. This job is then picked up by the job executor and executed. We are also giving the local job executor a little hint that there is a new job, to improve performance.

In order to use this feature, we can use the *activiti:async="true"* extension. So for example, the service task would look like this:

<serviceTask id="service1" name="Generate Invoice" activiti:class="my.custom.Delegate" activiti:async="true" />

activiti:async ca be specified on the following bpmn task types: task, serviceTask, scriptTask, businessRuleTask, sendTask, receiveTask, userTask, subProcess, callActivity

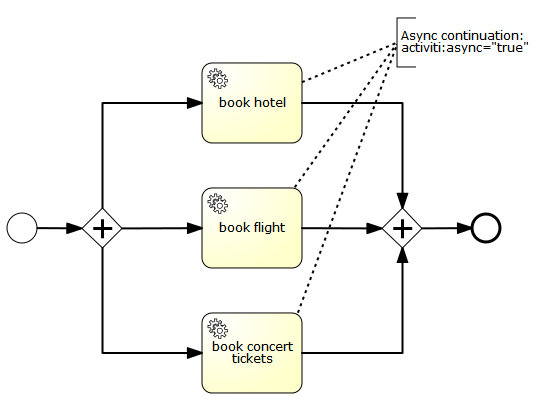
On a userTask, receiveTask or other wait states, the async continuation allows us to execute the start execution listeners in a separate thread/transaction.

**Exclusive Jobs**

Since Activiti 5.9, the JobExecutor makes sure that jobs from a single process instance are never executed concurrently. Why is this?

**Why exclusive Jobs?**

Consider the following process definition:



We have a parallel gateway followed by three service tasks which all perform an asynchronous continuation. As a result of this, three jobs are added to the database. Once such a job is present in the database it can be processes by the JobExecutor. The JobExecutor acquires the jobs and delegates them to a thread pool of worker threads which actually process the jobs. This means that using an asynchronous continuation, you can distribute the work to this thread pool (and in a clustered scenario even across multiple thread pools in the cluster). This is usually a good thing. However it also bears an inherent problem: consistency. Consider the parallel join after the service tasks. When execution of a service tasks is completed, we arrive at the parallel join and need to decide whether to wait for the other executions or whether we can move forward. That means, for each branch arriving at the parallel join, we need to take a decision whether we can continue or whether we need to wait for one or more other executions on the other branches.

Why is this a problem? Since the service tasks are configured using an asynchronous continuation, it is possible that the corresponding jobs are all acquired at the same time and delegated to different worker threads by the JobExecutor. The consequence is that the transactions in which the services are executed and in which the 3 individual executions arrive at the parallel join can overlap. And if they do so, each individual transaction will not "see", that another transaction is arriving at the same parallel join concurrently and thus assume that it has to wait for the others. However, if each transaction assumes that it has to wait for the other ones, none will continue the process after the parallel join and the process instance will remain in that state forever.

How does Activiti address this problem? Activiti performs optimistic locking. Whenever we take a decision based on data that might not be current (because another transaction might modify it before we commit, we make sure to increment the version of the same database row in both transactions). This way, whichever transaction commits first wins and the other ones fail with an optimistic locking exception. This solves the problem in the case of the process discussed above: if multiple executions arrive at the parallel join concurrently, they all assume that they have to wait, increment the version of their parent execution (the process instance) and then try to commit. Whichever execution is first will be able to commit and the other ones will fail with an optimistic locking exception. Since the executions are triggered by a job, Activiti will retry to perform the same job after waiting for a certain amount of time and hopefully this time pass the synchronizing gateway.

Is this a good solution? As we have seen, optimistic locking allows Activiti to prevent inconsistencies. It makes sure that we do not "keep stuck at the joining gateway", meaning: either all executions have passed the gateway or, there are jobs in the database making sure that we retry passing it. However, while this is a perfectly fine solution from the point of view of persistence and consistency, this might not always be desirable behavior at an higher level:

* Activiti will retry the same job for a fixed maximum number of times only ('3' in the default configuration). After that, the job will still be present in the database but not be retried actively anymore. That means that an operator would need to trigger the job manually.
* If a job has non-transactional side effects, those will not be rolled back by the failing transaction. For instance, if the "book concert tickets" service does not share the same transaction as Activiti, we might book multiple tickets if we retry the job.

In Activiti 5.9 we thus introduced a concept, which was already present in jBPM 4 and is called 'exclusive jobs'.

**What are exclusive jobs?**

An exclusive job cannot be performed at the same time as another exclusive job from the same process instance. Consider the process shown above: if we declare the service tasks to be exclusive, the JobExecutor will make sure that the corresponding jobs are not executed concurrently. Instead, it will make sure that whenever it acquires an exclusive job from a certain process instance, it acquires all other exclusive jobs from the same process instance and delegates them to the same worker thread. This ensures sequential execution execution of the jobs.

How can I enable this feature? Since Activiti 5.9, exclusive jobs are the default configuration. All asynchronous continuations and timer events are thus exclusive by default. In addition, if you want a job to be non-exclusive, you can configure it as such using activiti:exclusive="false". For example, the following servicetask would be asynchronous but non-exclusive.

<serviceTask id="service" activiti:expression="${myService.performBooking(hotel, dates)}" activiti:async="true" activiti:exclusive="false" />

Is this a good solution? We had some people asking whether this was a good solution. Their concern was that this would to prevent you from "doing things" in parallel and would thus be a performance problem. Again, two things have to be taken into consideration:

* It can be turned off if you are an expert and know what you are doing (and have understood the section named "Why exclusive Jobs?"). Other than that, it is more intuitive for most users if things like asynchronous continuations and timers just work.
* It is actually not a performance issue. Performance is an issue under heavy load. Heavy load means that all worker threads of the job executor are busy all the time. With exclusive jobs, Activiti will simply distribute the load differently. Exclusive jobs means that jobs from a single process instance are performed by the same thread sequentially. But consider: you have more than one single process instance. And jobs from other process instances are delegated to other threads and executed concurrently. This means that with exclusive jobs Activiti will not execute jobs from the same process instance concurrently, but it will still execute multiple instances concurrently. From an overall throughput perspective this is desirable in most scenarios as it usually leads to individual instances being done more quickly. Furthermore, data that is required for executing subsequent jobs of the same process instance will already be in the cache of the executing cluster node. If the jobs do not have this node affinity, that data might need to be fetched from the database again.

**Process Initiation Authorization**

By default everyone is allowed to start a new process instance of deployed process definitions. The process initiation authorization functionality allows to define users and groups so that web clients can optionally restrict users to start a new process instance. **NOTE** that the authorization definition is **NOT** validated by the Activiti Engine in any way. This functionality is only meant for developers to ease the implemention of authorization rules in a web client. The syntax is similar to the syntax of user assignment for a user task. A user or group can be assigned as potential initiator of a process using <activiti:potentialStarter> tag. Here is an example:

   <process id="potentialStarter">

     <extensionElements>

       <activiti:potentialStarter>

         <resourceAssignmentExpression>

           <formalExpression>group2, group(group3), user(user3)</formalExpression>

         </resourceAssignmentExpression>

       </activiti:potentialStarter>

     </extensionElements>

   <startEvent id="theStart"/>

   ...

In the above xml excerpt user(user3) refers directly to user user3 and group(group3) to group group3. No indicator will default to a group type. It is also possible to use attributes of the <process> tag, namely <activiti:candidateStarterUsers> and <activiti:candidateStarterGroups>. Here is an example:

      <process id="potentialStarter" activiti:candidateStarterUsers="user1, user2"

                                        activiti:candidateStarterGroups="group1">

      ...

It is possible to use both attributes simultaneously.

After the process initiation authorizations are defined, a developer can retrieve the authorization definition using the following methods. This code retrieves the list of process definitions which can be initiated by the given user:

      processDefinitions = repositoryService.createProcessDefinitionQuery().startableByUser("userxxx").list();

It's also possible to retrieve all identity links that are defined as potential starter for a specific process definition

      identityLinks = repositoryService.getIdentityLinksForProcessDefinition("processDefinitionId");

The following example shows how to get list of users who can initiate the given process:

      List<User> authorizedUsers =  identityService().createUserQuery().potentialStarter("processDefinitionId").list();

Exactly the same way, the list of groups that is configured as a potential starter to a given process definition can be retrieved:

      List<Group> authorizedGroups =  identityService().createGroupQuery().potentialStarter("processDefinitionId").list();

1

**Chapter 9. 表单**

**Table of Contents**

[表单属性](http://www.mossle.com/docs/activiti/#formProperties)

[外置表单的渲染](http://www.mossle.com/docs/activiti/#externalFormRendering)

Activiti提供了一种方便而且灵活的方式在业务流程中以手工方式添加表单。我们对表单的支持有2种方式： 通过表单属性对内置表单进行渲染和外置表单进行渲染。

**表单属性**

业务流程相关联的所有信息要么是包含自身的流程变量，要么是通过流程变量的引用。Activiti支持存储复杂的Java对象作为流程变量，如 序列化对象， Jpa实体对象或者整个XML文档作为字符串。

用户是在启动一个流程和完成用户任务时，与流程进行交互的。表单需要某个UI技术渲染之后才能够与用户进行交互。为了能够使用不同UI技术变得容易，流程定义包含一个对流程变量中复杂的Java类型对象到一个**'properties'**的Map<String,String>类型的转换逻辑。

使用Activiti API的方法查看公开的属性信息。然后，任意UI技术都能够在这些属性上面构建一个表单。该属性专门（并且更多局限性）为流程变量提供了一个视图。 表单所需要显示的属性可以从下面例子中的返回值**FormData**中获取。

StartFormData FormService.getStartFormData(String processDefinitionId)

or

TaskFormdata FormService.getTaskFormData(String taskId)

.

在默认情况下，内置的表单引擎，'sees'这些变量就像对待流程变量一样。如果任务表单属性和流程变量是一对一的关系，那么任务表单属性就不需要进行申明了，例如，下面的申明：

<startEvent id="start" />

当执行到开始事件时，所有的流程变量都是可用的，但

formService.getStartFormData(String processDefinitionId).getFormProperties()

会是一个空值，因为没有定义具体的映射。

在上面的实例中，所有被提交的属性都将会作为流程变量被存储在Activiti使用的数据库中。这意味着在一个表单中新添加一个简单的input输入字段，也会作为一个新的变量被存储。

属性来自于流程变量，但不一定非要作为流程变量存储起来，例如，一个流程变量可能是JPA实体如类Address。在某种UI技术中使用的表单属性StreetName可能会关联到一个表达式 #{address.street}。

类似的，用户提交的表单属性应该作为流程变量进行存储或者使用UEL值表达式将其作为流程变量的一个嵌套属性进行存储，例如#{address.street}。

同样的，提交的表单属性默认的行为是作为流程变量进行存储，除非一个 formProperty 申明了其他的规则。

类型转换同样也可以应用于表单数据和流程变量之间处理的一部分。

例如:

<userTask id="task">

  <extensionElements>

    <activiti:formProperty id="room" />

    <activiti:formProperty id="duration" type="long"/>

    <activiti:formProperty id="speaker" variable="SpeakerName" writable="false" />

    <activiti:formProperty id="street" expression="#{address.street}" required="true" />

  </extensionElements>

</userTask>

* 表单属性 room 将会被映射为String类型流程变量 room。
* 表单属性 duration 将会被映射为java.lang.Long类型流程变量 duration。
* 表单属性speaker将会被映射为流程变量 SpeakerName。它只能够在TaskFormData对象中使用。如果 属性speaker提交，将会抛出一个ActivitiException的异常。类似的，将其属性设置为readable="false",该属性就会在FormData进行排除，但是在提交后仍然会对其进行处理。
* 表单属性street将会映射为Java Bean address的属性street 作为String类型的流程变量。 当提交的表单属性并没有提供并且 required="true" 时，那么就会抛出一个异常。

表单数据也可以作为FormData的一部分提供类型元数据。该FormData可以从StartFormData FormService.getStartFormData(String processDefinitionId)和TaskFormdata FormService.getTaskFormData(String taskId)方法的返回值中获取。

我们支持以下的几种表单属性类型：

* string (org.activiti.engine.impl.form.StringFormType)
* long (org.activiti.engine.impl.form.LongFormType)
* enum (org.activiti.engine.impl.form.EnumFormType)
* date (org.activiti.engine.impl.form.DateFormType)
* boolean (org.activiti.engine.impl.form.BooleanFormType)

对于申明每一个表单属性，以下的FormProperty信息可以通过List<FormProperty> formService.getStartFormData(String processDefinitionId).getFormProperties() 和 List<FormProperty> formService.getTaskFormData(String taskId).getFormProperties()获取。

public interface FormProperty {

  /\*\*

  the key used to submit the property in {@link FormService#submitStartFormData(String, java.util.Map)}

   \* or {@link FormService#submitTaskFormData(String, java.util.Map)} \*/

  String getId();

  /\*\* the display label \*/

  String getName();

  /\*\* one of the types defined in this interface like e.g. {@link #TYPE\_STRING} \*/

  FormType getType();

  /\*\* optional value that should be used to display in this property \*/

  String getValue();

  /\*\* is this property read to be displayed in the form and made accessible with the methods

   \* {@link FormService#getStartFormData(String)} and {@link FormService#getTaskFormData(String)}. \*/

  boolean isReadable();

  /\*\* is this property expected when a user submits the form? \*/

  boolean isWritable();

  /\*\* is this property a required input field \*/

  boolean isRequired();

}

例子:

<startEvent id="start">

  <extensionElements>

    <activiti:formProperty id="speaker"

      name="Speaker"

      variable="SpeakerName"

      type="string" />

    <activiti:formProperty id="start"

      type="date"

      datePattern="dd-MMM-yyyy" />

    <activiti:formProperty id="direction" type="enum">

      <activiti:value id="left" name="Go Left" />

      <activiti:value id="right" name="Go Right" />

      <activiti:value id="up" name="Go Up" />

      <activiti:value id="down" name="Go Down" />

    </activiti:formProperty>

  </extensionElements>

</startEvent>

所有的表单属性的信息都是可以通过API进行访问的。可以通过 formProperty.getType().getName()获取类型的名称。 甚至可以通过 formProperty.getType().getInformation("datePattern")获取日期的匹配方式。通过 formProperty.getType().getInformation("values")可以获取到枚举值。

Activiti控制台支持表单属性并且可以根据表单定义对表单进行渲染。例如下面的XML片段

<startEvent ... >

  <extensionElements>

    <activiti:formProperty id="numberOfDays" name="Number of days" value="${numberOfDays}" type="long" required="true"/>

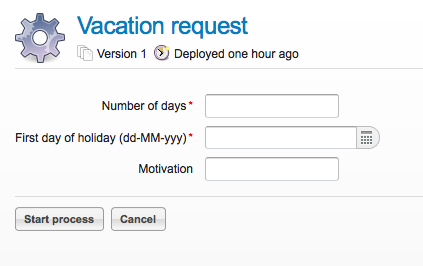
    <activiti:formProperty id="startDate" name="First day of holiday (dd-MM-yyy)" value="${startDate}" datePattern="dd-MM-yyyy hh:mm" type="date" required="true" />

    <activiti:formProperty id="vacationMotivation" name="Motivation" value="${vacationMotivation}" type="string" />

  </extensionElements>

</userTask>

当使用 Activiti控制台时，它将会被渲染成流程的启动表单。



**外置表单的渲染**

该API同样也允许你执行Activiti流程引擎之外的方式渲染你自己的任务表单。这些步骤说明你可以用你自己的方式对任务表单进行渲染。

本质上，所有需要渲染的表单属性都是通过2个服务方法中的一个进行装配的： StartFormData FormService.getStartFormData(String processDefinitionId) 和 TaskFormdata FormService.getTaskFormData(String taskId).

表单属性可以通过 ProcessInstance FormService.submitStartFormData(String processDefinitionId, Map<String,String> properties) and void FormService.submitStartFormData(String taskId, Map<String,String> properties)2种方式进行提交。

想要了解更多表单属性如何映射为流程变量，可以查看 [the section called “表单属性”](http://www.mossle.com/docs/activiti/#formProperties)

你可以将任何表单模版资源放进你要部署的业务文档之中（如果你想要将它们按照流程的版本进行存储）。它将会在部署中作为一种可用的资源。你可以使用String ProcessDefinition.getDeploymentId() 和 InputStream RepositoryService.getResourceAsStream(String deploymentId, String resourceName); 获取部署上去的表单模版。这样就可以获取你的表单模版定义文件 ，那么你就可以在你自己的应用中渲染/显示你的表单。

你也可以使用该功能获取任务表单之外的其他的部署资源用于其他的目的。

属性<userTask activiti:formKey="..."通过API String FormService.getStartFormData(String processDefinitionId).getFormKey() 和 String FormService.getTaskFormData(String taskId).getFormKey()暴露出来的。 你可以使用这个存储你部署的模版中的全名（例如org/activiti/example/form/my-custom-form.xml）,但是这并不是必须的。 例如，你可以在表单属性中存储一个通用的key，然后运用一种算法或者换转去得到你实际使用的模版。当你想要通过不同UI技术渲染不能的表单，这可能更加方便，例如，使用正常屏幕大小的web应用程序的表单， 移动手机小屏幕的表单和甚至可能是IM表单和email表单模版。

**Chapter 10. JPA**

**Table of Contents**

[要求](http://www.mossle.com/docs/activiti/#N12128)

[配置](http://www.mossle.com/docs/activiti/#jpaconfiguration)

[用法](http://www.mossle.com/docs/activiti/#N121B8)

[简单例子](http://www.mossle.com/docs/activiti/#N121BB)

[查询JPA流程变量](http://www.mossle.com/docs/activiti/#N12202)

[使用Spring beans和JPA结合的高级例子](http://www.mossle.com/docs/activiti/#N1222F)

你可以使用JPA实体作为流程变量，并且可以这样做：

* 基于流程变量更新已有的JPA实体，它可以在用户任务的表单中填写或者由服务任务生成。
* 重用已有的领域模型不需要编写显示的服务获取实体或者更新实体的值。
* 根据已有实体的属性做出判断（网关即分支聚合）。
* ...

**要求**

只支持符合以下要求的实体：

* 实体应该使用JPA注解进行配置，我们支持字段和属性访问两种方式。@MappedSuperclass也能够被使用。
* 实体中应该有一个使用@Id注解的主键，不支持复合主键 (@EmbeddedId 和 @IdClass)。Id 字段/属性能够使用JPA规范支持的任意类型： 原生态数据类型和他们的包装类型（boolean除外），String, BigInteger, BigDecimal, java.util.Date 和 java.sql.Date.

**配置**

为了能够使用JPA的实体，引擎必须有一个对EntityManagerFactory的引用。 这可以通过配置引用或者提供一个持久化单元名称。作为变量的JPA实体将会被自动检测并进行相应的处理

下面例子中的配置是使用jpaPersistenceUnitName：

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

    <!-- 数据库的配置 -->

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

    <property name="jdbcUrl" value="jdbc:h2:mem:JpaVariableTest;DB\_CLOSE\_DELAY=1000" />

**<property name="jpaPersistenceUnitName" value="activiti-jpa-pu" />**

**<property name="jpaHandleTransaction" value="true" />**

**<property name="jpaCloseEntityManager" value="true" />**

    <!-- job executor configurations -->

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

    <!-- mail server configurations -->

    <property name="mailServerPort" value="5025" />

</bean>

接下来例子中的配置提供了一个我们自定义的EntityManagerFactory(在这个例子中，使用了OpenJPA 实体管理器)。注意该代码片段仅仅包含与例子相关的beans，去掉了其他beans。OpenJPA实体管理的完整并可以使用的例子可以在activiti-spring-examples(/activiti-spring/src/test/java/org/activiti/spring/test/jpa/JPASpringTest.java)中找到。

<bean id="entityManagerFactory" class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">

  <property name="persistenceUnitManager" ref="pum"/>

  <property name="jpaVendorAdapter">

    <bean class="org.springframework.orm.jpa.vendor.OpenJpaVendorAdapter">

      <property name="databasePlatform" value="org.apache.openjpa.jdbc.sql.H2Dictionary" />

    </bean>

  </property>

</bean>

<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="jpaEntityManagerFactory" ref="entityManagerFactory" />**

**<property name="jpaHandleTransaction" value="true" />**

**<property name="jpaCloseEntityManager" value="true" />**

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

</bean>

同样的配置也可以在编程式创建一个引擎时完成，例如：

ProcessEngine processEngine = ProcessEngineConfiguration

  .createProcessEngineConfigurationFromResourceDefault()

**.setJpaPersistenceUnitName("activiti-pu")**

  .buildProcessEngine();

配置属性：

* **jpaPersistenceUnitName:** 使用持久化单元的名称（要确保该持久化单元在类路径下是可用的）。根据该规范，默认的路径是/META-INF/persistence.xml)。要么使用 jpaEntityManagerFactory 或者jpaPersistenceUnitName。
* **jpaEntityManagerFactory:** 一个实现了javax.persistence.EntityManagerFactory的bean的引用。它将被用来加载实体并且刷新更新。要么使用jpaEntityManagerFactory 或者jpaPersistenceUnitName。
* **jpaHandleTransaction:** 在被使用的EntityManager 实例上，该标记表示流程引擎是否需要开始和提交/回滚事物。当使用Java事物API（JTA）时，设置为false。
* **jpaCloseEntityManager:** 该标记表示流程引擎是否应该关闭从 EntityManagerFactory获取的 EntityManager的实例。当EntityManager 是由容器管理的时候需要设置为false（例如 当使用并不是单一事物作用域的扩展持久化上下文的时候）。

**用法**

**简单例子**

使用JPA变量的例子可以在 [JPAVariableTest](http://svn.codehaus.org/activiti/activiti/trunk/modules/activiti-engine/src/test/java/org/activiti/standalone/jpa/JPAVariableTest.java)中找到。我们将会一步一步的解释JPAVariableTest.testUpdateJPAEntityValues。

首先，我们需要创建一个基于META-INF/persistence.xml的EntityManagerFactory作为我们的持久化单元。它包含持久化单元中所有的类和一些供应商特定的配置。

我们将使用一个简单的实体作为测试，其中包含有一个id和String 类型的value属性，这也将会被持久化。在允许测试之前，我们创建一个实体并且保存它。

@Entity(name = "JPA\_ENTITY\_FIELD")

public class FieldAccessJPAEntity {

  @Id

  @Column(name = "ID\_")

  private Long id;

  private String value;

  public FieldAccessJPAEntity() {

    // Empty constructor needed for JPA

  }

  public Long getId() {

    return id;

  }

  public void setId(Long id) {

    this.id = id;

  }

  public String getValue() {

    return value;

  }

  public void setValue(String value) {

    this.value = value;

  }

}

我们启动一个新的流程实例，添加一个实体作为变量。至于其他的变量，它们将会被存储在流程引擎的持久化数据库中。下一次获取该变量的时候，它将会根据该类和存储Id从EntityManager中加载。

Map<String, Object> variables = new HashMap<String, Object>();

variables.put("entityToUpdate", entityToUpdate);

ProcessInstance processInstance = runtimeService.startProcessInstanceByKey("UpdateJPAValuesProcess", variables);

在我们的流程定义中的第一个节点是一个服务任务，它将会调用entityToUpdate上的setValue方法，它其实就是我们之前在启动流程实例时候设置的JPA变量并且它将会从当前流程引擎的上下文关联的EntityManager中加载。

<serviceTask id='theTask' name='updateJPAEntityTask' activiti:expression="${entityToUpdate.setValue('updatedValue')}" />

当完成服务任务时，流程实例将会停留在流程定义中定义的用户任务环节上。这时我们就可以查看该流程实例。与此同时，EntityManager已经被刷新了并且改变的实体已经被保存进数据库中。当我们获取entityToUpdate的变量value时，该实体将会被再次加载并且我们获取该实体属性的值 将会是 updatedValue。

// Servicetask in process 'UpdateJPAValuesProcess' should have set value on entityToUpdate.

Object updatedEntity = runtimeService.getVariable(processInstance.getId(), "entityToUpdate");

assertTrue(updatedEntity instanceof FieldAccessJPAEntity);

assertEquals("updatedValue", ((FieldAccessJPAEntity)updatedEntity).getValue());

**查询JPA流程变量**

你可以查询某一JPA实体作为变量的ProcessInstances 和 Executions 。 **注意，在ProcessInstanceQuery 和 ExecutionQuery查询中仅仅variableValueEquals(name, entity) 支持JPA实体变量** 。 方法 variableValueNotEquals, variableValueGreaterThan, variableValueGreaterThanOrEqual, variableValueLessThan 和 variableValueLessThanOrEqual并不被支持并且传递JPA实体值的时候会抛出一个ActivitiException。

 ProcessInstance result = runtimeService.createProcessInstanceQuery().variableValueEquals("entityToQuery", entityToQuery).singleResult();

**使用Spring beans和JPA结合的高级例子**

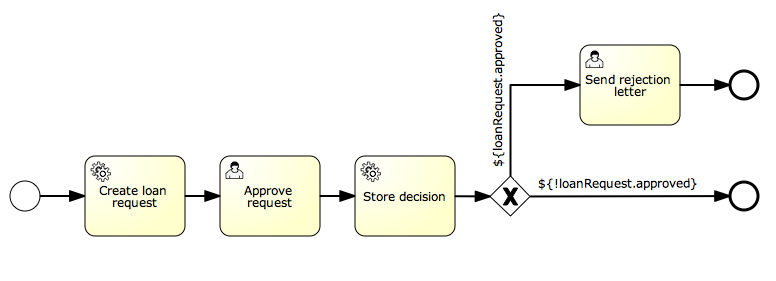
一个更加高级的例子，JPASpringTest，可以在activiti-spring-examples中找到。它描述了如下简单的使用情况：

* 已经存在了一个使用JPA实体的Spring-bean，它用来存储贷款申请。
* 使用Activiti，我们可以通过已经存在的bean获取已经使用的实体，并使用它作为变量用于我们的流程中。

按照下面的步骤定义流程：

* + 服务任务，创建一个新的贷款申请，使用已经存在的LoanRequestBean 接受启动流程时候的变量（例如 可以来自流程启动时候的表单）。 使用activiti:resultVariable（它作为一个变量对表达式返回的结果进行存储）将创建出来的实体作为变量进行存储。
  + 用户任务，允许经理查看贷款申请，并填入审批意见(同意/不同意)，审批意见将作为一个boolean变量approvedByManager进行存储
  + 服务任务，更新贷款申请实体因此该实体与流程保持同步
  + 根据贷款申请实体变量approved的值，将利用唯一网关（BPMN2规范）自动决定下一步该选择那一条路径：当申请批准，流程结束。否则，一个额外的任务将会使用 （发送拒绝信），所以这样就是可以让拒绝信手动通知客户。

请注意该流程并不包含任何表单，因为它仅仅被用于单元测试。



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

<definitions id="taskAssigneeExample"

  xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"

  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xmlns:activiti="http://activiti.org/bpmn"

  targetNamespace="org.activiti.examples">

  <process id="LoanRequestProcess" name="Process creating and handling loan request">

    <startEvent id='theStart' />

    <sequenceFlow id='flow1' sourceRef='theStart' targetRef='createLoanRequest' />

    <serviceTask id='createLoanRequest' name='Create loan request'

      activiti:expression="${loanRequestBean.newLoanRequest(customerName, amount)}"

      activiti:resultVariable="loanRequest"/>

    <sequenceFlow id='flow2' sourceRef='createLoanRequest' targetRef='approveTask' />

    <userTask id="approveTask" name="Approve request" />

    <sequenceFlow id='flow3' sourceRef='approveTask' targetRef='approveOrDissaprove' />

    <serviceTask id='approveOrDissaprove' name='Store decision'

      activiti:expression="${loanRequest.setApproved(approvedByManager)}" />

    <sequenceFlow id='flow4' sourceRef='approveOrDissaprove' targetRef='exclusiveGw' />

    <exclusiveGateway id="exclusiveGw" name="Exclusive Gateway approval" />

    <sequenceFlow id="endFlow1" sourceRef="exclusiveGw" targetRef="theEnd">

      <conditionExpression xsi:type="tFormalExpression">${loanRequest.approved}</conditionExpression>

    </sequenceFlow>

    <sequenceFlow id="endFlow2" sourceRef="exclusiveGw" targetRef="sendRejectionLetter">

      <conditionExpression xsi:type="tFormalExpression">${!loanRequest.approved}</conditionExpression>

    </sequenceFlow>

    <userTask id="sendRejectionLetter" name="Send rejection letter" />

    <sequenceFlow id='flow5' sourceRef='sendRejectionLetter' targetRef='theOtherEnd' />

    <endEvent id='theEnd' />

    <endEvent id='theOtherEnd' />

  </process>

</definitions>

虽然上面的例子非常的简单，但是它却展示了JPA结合Spring和参数化方法表达式的强大优势。这样所有的流程就不需要自定义java代码（当然，除了Spring bean之外） 并且大幅度的加快了流程部署。

**Chapter 11. 历史**

**Table of Contents**

[查询历史](http://www.mossle.com/docs/activiti/#historyQuery)

[HistoricProcessInstanceQuery](http://www.mossle.com/docs/activiti/#historyQueryProcessInstance)

[HistoricVariableInstanceQuery](http://www.mossle.com/docs/activiti/#historyQueryVariableInstance)

[HistoricActivityInstanceQuery](http://www.mossle.com/docs/activiti/#historyQueryActivityInstance)

[HistoricDetailQuery](http://www.mossle.com/docs/activiti/#historyQueryDetail)

[HistoricTaskInstanceQuery](http://www.mossle.com/docs/activiti/#historyQueryTaskInstance)

[历史配置](http://www.mossle.com/docs/activiti/#historyConfig)

[审计目的的历史](http://www.mossle.com/docs/activiti/#historyFormAuditPurposes)

历史是一个组件，它可以捕获发生在进程执行中的信息并永久的保存，与运行时数据不同的是，当流程实例运行完成之后它还会存在于数据库中。

有5个历史实体对象:

* HistoricProcessInstances 包含当前和已经结束的流程实例信息。
* HistoricVariableInstances 包含最新的流程变量或任务变量。
* HistoricActivityInstances 包含一个活动(流程上的节点)的执行信息 。
* HistoricTaskInstances 包含关于当前和过去的（已完成或已删除）任务实例信息。
* HistoricDetails 包含历史流程实例、活动实例、任务实例的各种信息。

由于数据库中保存着历史信息以及正在运行的流程实例信息，就要考虑怎样尽量少的对运行中的流程实例数据进行访问的方式查询这些表来保证执行的性能。

稍后, 这个信息体现在Activiti Explorer。同时它也是报告将生成的信息。

**查询历史**

在API中, 提供了对这5种实体的查询方法。类HistoryService 提供了以下几种方法 createHistoricProcessInstanceQuery(), createHistoricVariableInstanceQuery(), createHistoricActivityInstanceQuery(), createHistoricDetailQuery() 和 createHistoricTaskInstanceQuery()。

下面是一些API中查询历史信息的例子. 这些方法的详细描述可以在 [ZAjavadocs](http://www.mossle.com/docs/javadocs/index.html)中找到, 在包org.activiti.engine.history 中

**HistoricProcessInstanceQuery**

流程实例。

获取流程定义ID是'XXX'、已经结束、花费时间最长（持续时间最长）的10个HistoricProcessInstances

historyService.createHistoricProcessInstanceQuery()

  .finished()

  .processDefinitionId("XXX")

  .orderByProcessInstanceDuration().desc()

  .listPage(0, 10);

**HistoricVariableInstanceQuery**

在ID为'xxx'、已经结束的流程实例中查询所有HistoricVariableInstances，并按变量名排序

historyService.createHistoricVariableInstanceQuery()

  .processInstanceId("XXX")

  .orderByVariableName.desc()

  .list();

**HistoricActivityInstanceQuery**

获取所有已经结束的流程定义ID为’XXX'并且类型是'serviceTask'中的最后一个 HistoricActivityInstance

historyService.createHistoricActivityInstanceQuery()

  .activityType("serviceTask")

  .processDefinitionId("XXX")

  .finished()

  .orderByHistoricActivityInstanceEndTime().desc()

  .listPage(0, 1);

**HistoricDetailQuery**

下个例子, 获取所有id为123的流程实例中产量的可变更新信息。这个查询只会返回 HistoricVariableUpdates. 注意一些变量名可能包含多个 HistoricVariableUpdate 实体, 每次流程运行时会更新变量。 你可以用 orderByTime (变量被更新的时间) 或者 orderByVariableRevision (运行更新时变量的版本)来排序查询.

historyService.createHistoricDetailQuery()

  .variableUpdates()

  .processInstanceId("123")

  .orderByVariableName().asc()

  .list()

这个例子获取所有流程实例ID为123的流程中，提交任务或者启动流程时的[form-properties](http://www.mossle.com/docs/activiti/#formProperties) 。 这个查询只会返回 HistoricFormPropertiess 。

historyService.createHistoricDetailQuery()

  .formProperties()

  .processInstanceId("123")

  .orderByVariableName().asc()

  .list()

最后这个例子获取所有在执行ID为123的任务时的变量更新。 返回全部在任务中设置的变量（任务局部变量） HistoricVariableUpdates , 不是流程实例变量。

historyService.createHistoricDetailQuery()

  .variableUpdates()

  .taskId("123")

  .orderByVariableName().asc()

  .list()

任务局部变量可以用 TaskService 设置或者使用 DelegateTask, 在TaskListener里设置:

taskService.setVariableLocal("123", "myVariable", "Variable value");

public void notify(DelegateTask delegateTask) {

  delegateTask.setVariableLocal("myVariable", "Variable value");

}

**HistoricTaskInstanceQuery**

获取所有任务中10个花费时间最长（持续时间最长）并已经结束的 HistoricTaskInstances 。

historyService.createHistoricTaskInstanceQuery()

  .finished()

  .orderByHistoricTaskInstanceDuration().desc()

  .listPage(0, 10);

获取删除原因包含"无效"，最后分配给用户"kermit"的 HistoricTaskInstances。

historyService.createHistoricTaskInstanceQuery()

  .finished()

  .taskDeleteReasonLike("%invalid%")

  .taskAssignee("kermit")

  .listPage(0, 10);

**历史配置**

历史级别可以用编写代码的方法配置, 用枚举类型org.activiti.engine.impl.history.HistoryLevel (或者在5.11之前定义在ProcessEngineConfiguration中的常量 HISTORY\_\*):

ProcessEngine processEngine = ProcessEngineConfiguration

  .createProcessEngineConfigurationFromResourceDefault()

**.setHistory(HistoryLevel.AUDIT.getKey())**

  .buildProcessEngine();

级别可以在配置文件 activiti.cfg.xml 或者在 spring-context中配置:

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

  <property name="history" value="audit" />

  ...

</bean>

历史信息级别可以配置成以下几种:

* none: 忽略所有历史存档。这是流程执行时性能最好的状态，但没有任何历史信息可用。
* activity: 保存所有流程实例信息和活动实例信息。 在流程实例结束时, 最后一个流程实例中的最新的变量值将赋值给历史变量。 不会保存过程中的详细信息。
* audit: 这个是默认值. 它保存所有流程实例信息, 活动信息, 保证所有的变量和提交的表单属性保持同步 这样所有用户交互信息都是可追溯的，可以用来审计。
* full: 这个是最高级别的历史信息存档，同样也是最慢的。 这个级别存储发生在审核以及所有其它细节的信息, 主要是更新流程变量。

**在Activiti 5.11之前, 历史级别都存在数据库中 (表 ACT\_GE\_PROPERTY,属性名为 historyLevel). 从Activiti 5.11开始, 这个值不再用，并且从数据库中忽略或者删除掉。 这个配置可以在两次启动间修改, 因为是在启动之前修改的，所以不会抛出异常.**

**审计目的的历史**

当[配置](http://www.mossle.com/docs/activiti/#historyConfig) 在 audit级别之上 。 所有通过 FormService.submitStartFormData(String processDefinitionId, Map<String, String> properties) and FormService.submitTaskFormData(String taskId, Map<String, String> properties) 方法提交的属性都会被记录。

表单属性可以通过API查询，如下:

historyService

      .createHistoricDetailQuery()

      .formProperties()

      ...

      .list();

上面的例子只有类型为 HistoricFormProperty的详细信息会被查询出来。

如果你在调用IdentityService.setAuthenticatedUserId(String)提交之前设置了认证用户，那么提交表单的用户将被保存在历史信息中并可以在开始表单中 使用HistoricProcessInstance.getStartUserId()获取，在任务表单中用HistoricActivityInstance.getAssignee()获取。

**Chapter 12. Eclipse Designer**

**Table of Contents**

[Installation](http://www.mossle.com/docs/activiti/#eclipseDesignerInstallation)

[Activiti Designer editor features](http://www.mossle.com/docs/activiti/#eclipseDesignerEditorFeatures)

[Activiti Designer BPMN features](http://www.mossle.com/docs/activiti/#eclipseDesignerBPMNFeatures)

[Activiti Designer deployment features](http://www.mossle.com/docs/activiti/#eclipseDesignerDeployment)

[Extending Activiti Designer](http://www.mossle.com/docs/activiti/#eclipseDesignerExtending)

[Customizing the palette](http://www.mossle.com/docs/activiti/#eclipseDesignerCustomizingPalette)

[Validating diagrams and exporting to custom output formats](http://www.mossle.com/docs/activiti/#N127EE)

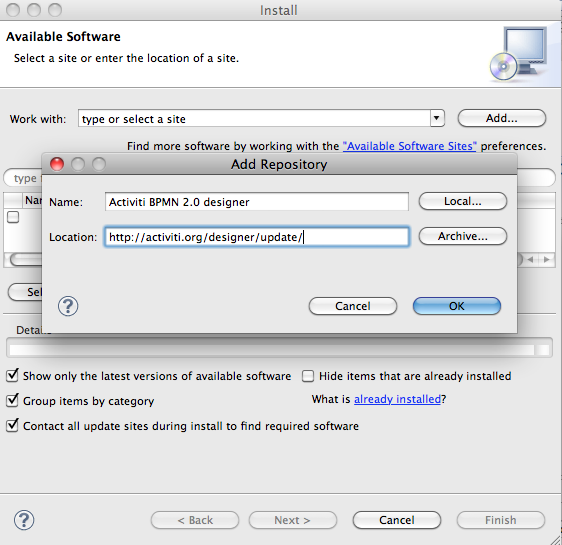
Activiti comes with an Eclipse plugin, the Activiti Eclipse Designer, that can be used to graphically model, test and deploy BPMN 2.0 processes.

**Installation**

The following installation instructions are verified on [Eclipse Indigo](http://www.eclipse.org/downloads/). Note that Eclipse Helio is **NOT** supported.

Go to **Help -> Install New Software**. In the following panel, click on *Add* button and fill in the following fields:

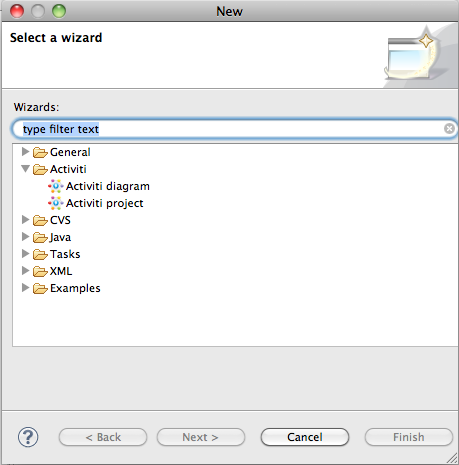
* **Name:** Activiti BPMN 2.0 designer
* **Location:** http://activiti.org/designer/update/



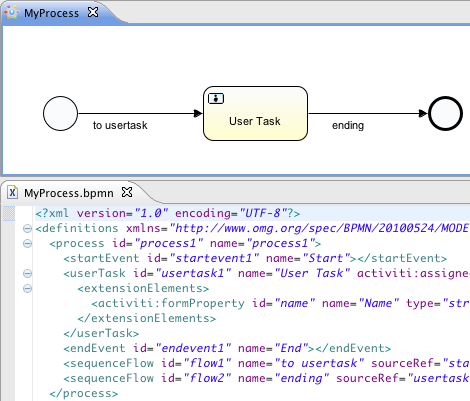
Make sure the **"Contact all updates sites.."** checkbox is **checked**, because all the necessary plugins will then be downloaded by Eclipse.

**Activiti Designer editor features**

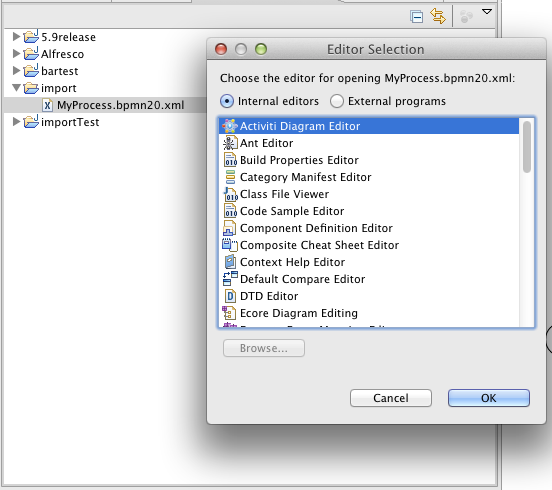
* Create Activiti projects and diagrams.



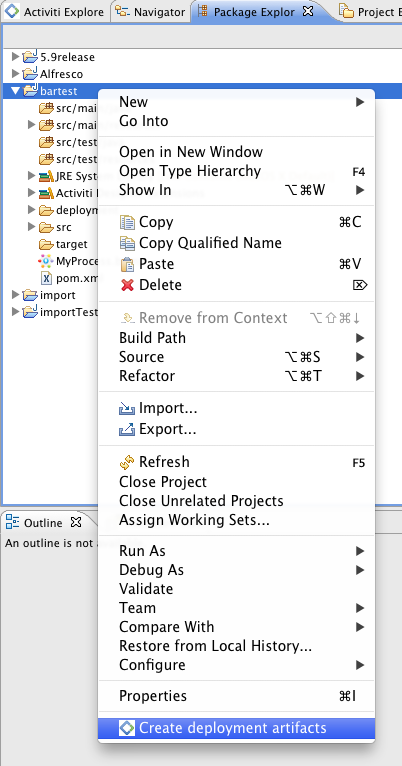
* The Activiti Designer creates a .bpmn file when creating a new Activiti diagram. When opened with the Activiti Diagram Editor view this will provide a graphical modeling canvas and palette. The same file can however be opened with an XML editor and it then shows the BPMN 2.0 XML elements of the process definition. So the Activiti Designer works with only one file for both the graphical diagram as well as the BPMN 2.0 XML. Note that in Activiti 5.9 the .bpmn extension is not yet supported as deployment artifact for a process definition. Therefore the "create deployment artifacts" feature of the Activiti Designer generates a BAR file with a .bpmn20.xml file that contains the content of the .bpmn file. You can also do a quick file rename yourself. Also note that you can open a .bpmn20.xml file with the Activiti Diagram Editor view as well.



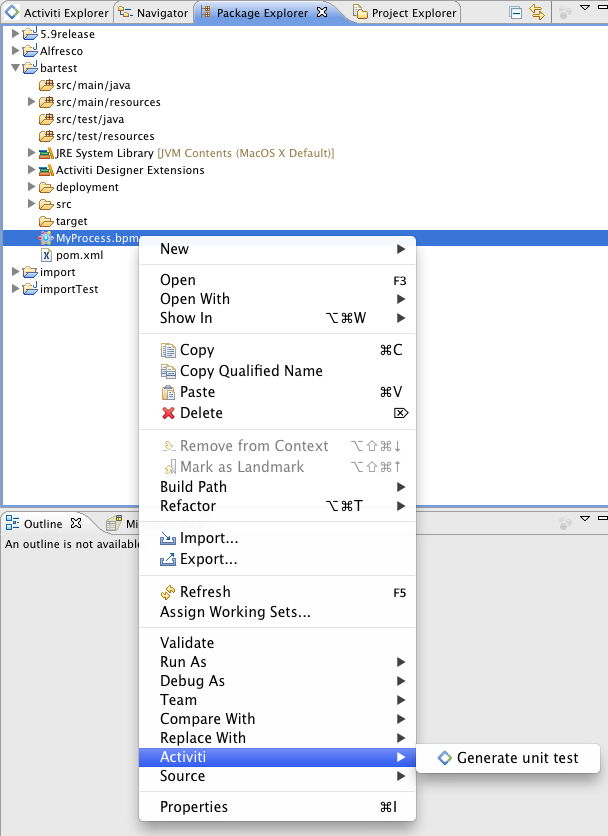
* BPMN 2.0 XML files can be imported into the Activiti Designer and a diagram will be created. Just copy the BPMN 2.0 XML file to your project and open the file with the Activiti Diagram Editor view. The Activiti Designer uses the BPMN DI information of the file to create the diagram. If you have a BPMN 2.0 XML file without BPMN DI information, no diagram can be created.



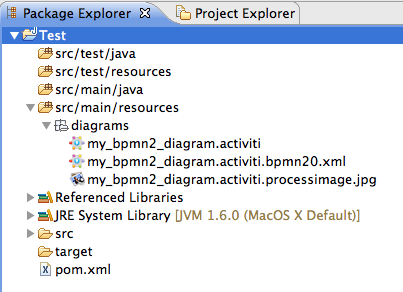
* For deployment a BAR file and optionally a JAR file is created by the Activiti Designer by right-clicking on an Activiti project in the package explorer and choosing the *Create deployment artifacts* option at the bottom of the popup menu. For more information about the deployment functionality of the Designer look a the [deployment](http://www.mossle.com/docs/activiti/#eclipseDesignerDeployment) section.



* Generate a unit test (right click on a BPMN 2.0 XML file in the package explorer and select *generate unit test*) A unit test is generated with an Activiti configuration that runs on an embedded H2 database. You can now run the unit test to test your process definition.

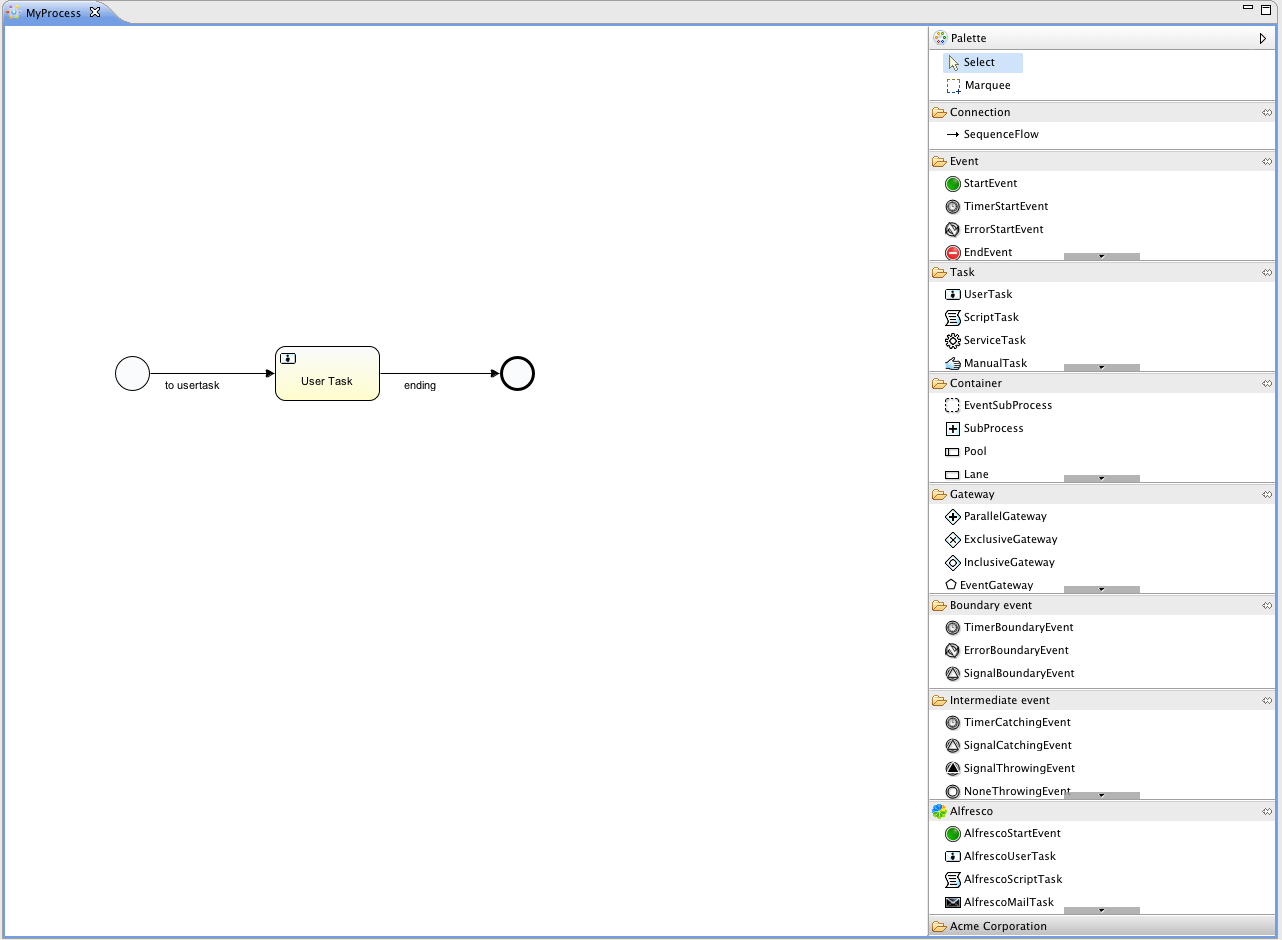


* The Activiti project is generated as a Maven project. To configure the dependencies you need to run *mvn eclipse:eclipse* and the Maven dependencies will be configured as expected. Note that for process design Maven dependencies are not needed. They are only needed to run unit tests.

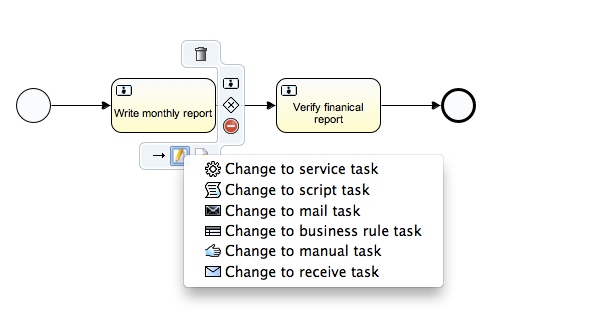


**Activiti Designer BPMN features**

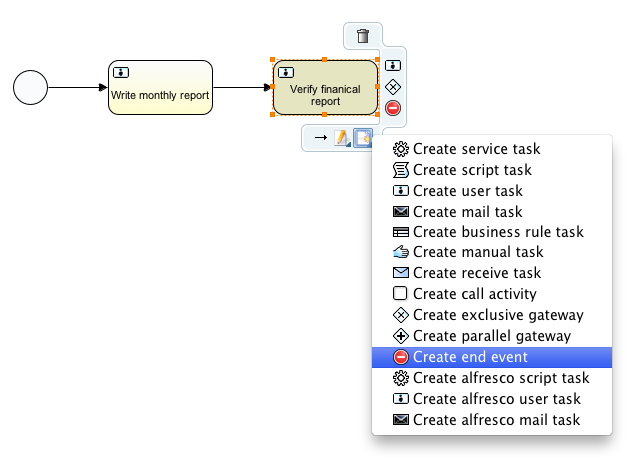
* Support for start none event, start error event, timer start event, end none event, end error event, sequence flow, parallel gateway, exclusive gateway, inclusive gateway, event gateway, embedded subprocess, event sub process, call activity, pool, lane, script task, user task, service task, mail task, manual task, business rule task, receive task, timer boundary event, error boundary event, signal boundary event, timer catching event, signal catching event, signal throwing event, none throwing event and four Alfresco specific elements (user, script, mail tasks and start event).



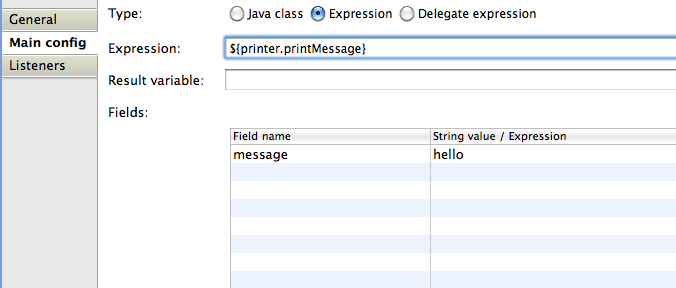
* You can quickly change the type of a task by hovering over the element and choosing the new task type.



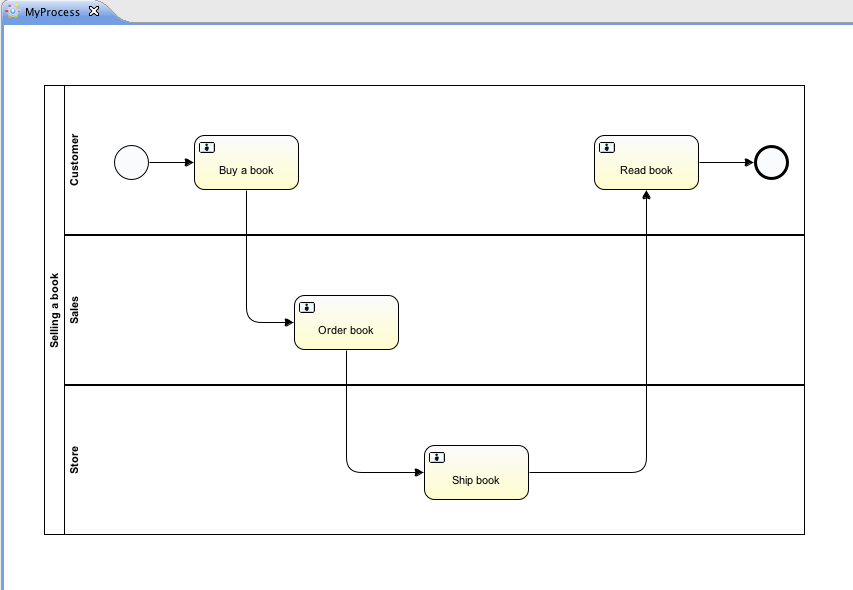
* You can quickly add new elements hovering over an element and choosing a new element type.



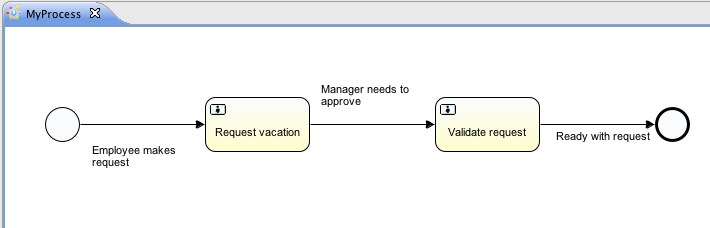
* Java class, expression or delegate expression configuration is supported for the Java service task. In addition field extensions can be configured.



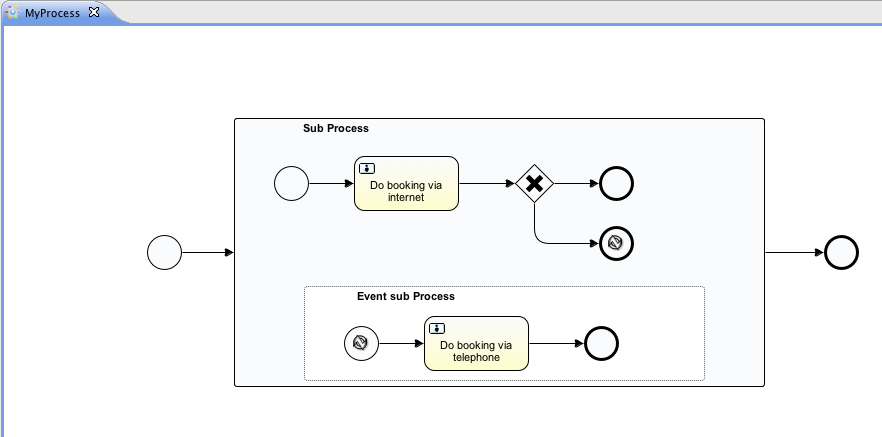
* Support for pools and lanes. Because Activiti reads different pools as different process definition, it makes the most sense to use only one pool. If you use multiple pools, be aware that drawing sequence flows between the pools will result in problems when deploying the process in the Activiti Engine. You can add as much lanes to a pool as you want.



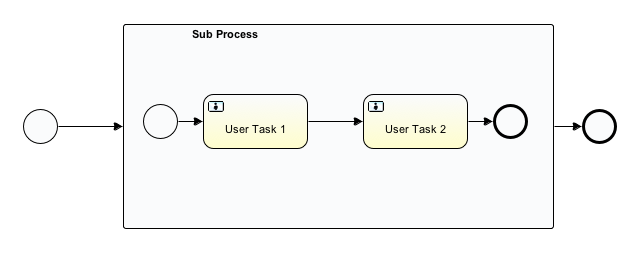
* You can add labels to sequence flows by filling the name property. You can position the labels yourself as the position is saved as part of the BPMN 2.0 XML DI information.



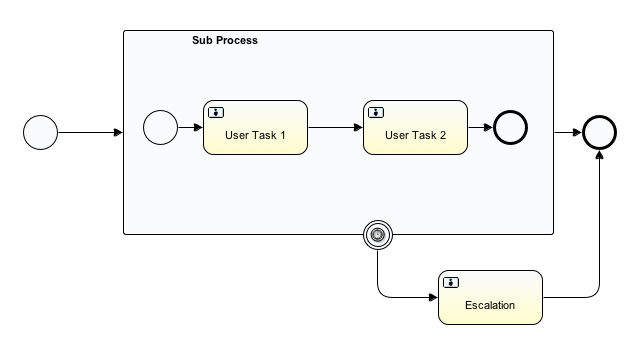
* Support for event sub processes.



* Support for expanded embedded sub processes. You can also add an embedded sub process in another embedded sub process.



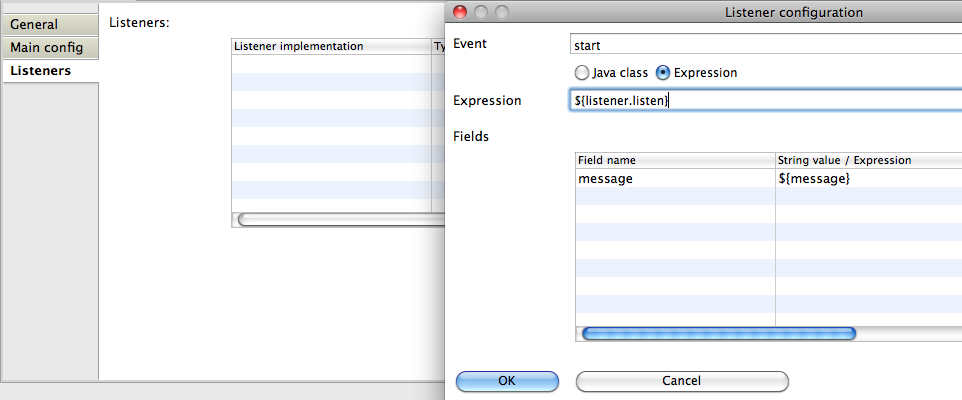
* Support for timer boundary events on tasks and embedded sub processes. Although, the timer boundary event makes the most sense when using it on a user task or an embedded sub process in the Activiti Designer.



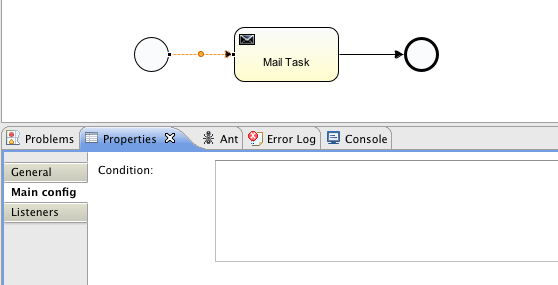
* Support for additional Activiti extensions like the Mail task, the candidate configuration of User tasks and Script task configuration.



* Support for the Activiti execution and task listeners. You can also add field extensions for execution listeners.

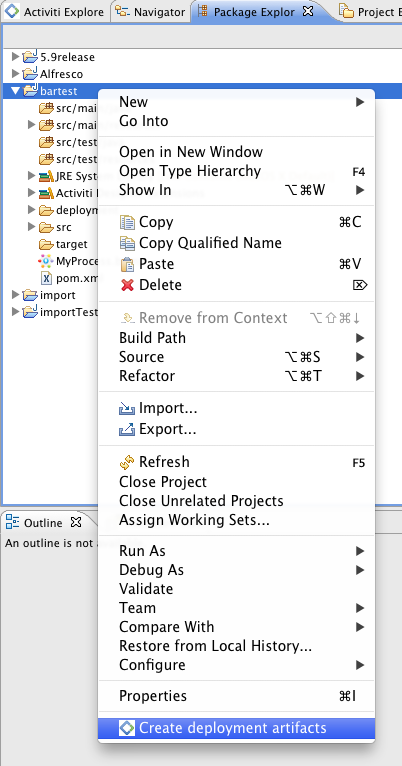


* Support for conditions on sequence flows.



**Activiti Designer deployment features**

Deploying process definitions and task forms on the Activiti Engine is not hard. You need a BAR file containing the process definition BPMN 2.0 XML file and optionally task forms and an image of the process that can be viewed in the Activiti Explorer. In the Activiti Designer it's made very easy to create a BAR file. When you've finished your process implementation just right-click on your Activiti project in the package explorer and choose for the **Create deployment artifacts** option at the bottom of the popup menu.



Then a deployment directory is created containing the BAR file and optionally a JAR file with the Java classes of your Activiti project.



This file can now be uploaded to the Activiti Engine using the deployments tab in Activiti Explorer, and you are ready to go.

When your project contains Java classes, the deployment is a bit more work. In that case the **Create deployment artifacts** step in the Activiti Designer will also generate a JAR file containing the compiled classes. This JAR file must be deployed to the activiti-XXX/WEB-INF/lib directory in your Activiti Tomcat installation directory. This makes the classes available on the classpath of the Activiti Engine.

**Extending Activiti Designer**

You can extend the default functionality offered by Activiti Designer. This section documents which extensions are available, how they can be used and provides some usage examples. Extending Activiti Designer is useful in cases where the default functionality doesn't suit your needs, you require additional capabilities or have domain specific requirements when modeling business processes. Extension of Activiti Designer falls into two distinct categories, extending the palette and extending output formats. Each of these extension ways requires a specific approach and different technical expertise.

**Note**

Extending Activiti Designer requires technical knowledge and more specifically, knowledge of programming in Java. Depending on the type of extension you want to create, you might also need to be familiar with Maven, Eclipse, OSGi, Eclipse extensions and SWT.

**Customizing the palette**

You can customize the palette that is offered to users when modeling processes. The palette is the collection of shapes that can be dragged onto the canvas in a process diagram and is displayed to the right hand side of the canvas. As you can see in the default palette, the default shapes are grouped into compartments (these are called "drawers") for Events, Gateways and so on. There are two options built-in to Activiti Designer to customize the drawers and shapes in the palette:

* Adding your own shapes / nodes to existing or new drawers
* Disabling any or all of the default BPMN 2.0 shapes offered by Activiti Designer, with the exception of the connection and selection tools

In order to customize the palette, you create a JAR file that is added to a specific installation of Activiti Designer (more on [how to do that](http://www.mossle.com/docs/activiti/#eclipseDesignerApplyingExtension) later). Such a JAR file is called an *extension*. By writing classes that are included in your extension, Activiti Designer understands which customizations you wish to make. In order for this to work, your classes should implement certain interfaces. There is an integration library available with those interfaces and base classes to extend which you should add to your project's classpath.

You can find the code examples listed below in source control with Activiti Designer. Take a look in the examples/money-tasks directory in the projects/designer directory of Activiti's source code.

**Note**

You can setup your project in whichever tool you prefer and build the JAR with your build tool of choice. For the instructions below, a setup is assumed with Eclipse Helios, using Maven (3.x) as build tool, but any setup should enable you to create the same results.

**Extension setup (Eclipse/Maven)**

Download and extract [Eclipse](http://www.eclipse.org/downloads) (most recent versions should work) and a recent version (3.x) of [Apache Maven](http://maven.apache.org/download.html). If you use a 2.x version of Maven, you will run into problems when building your project, so make sure your version is up to date. We assume you are familiar with using basic features and the Java editor in Eclipse. It's up to you whether your prefer to use Eclipse's features for Maven or run Maven commands from a command prompt.

Create a new project in Eclipse. This can be a general project type. Create a pom.xml file at the root of the project to contain the Maven project setup. Also create folders for the src/main/java and src/main/resources folders, which are Maven conventions for your Java source files and resources respectively. Open the pom.xml file and add the following lines:

<project

  xmlns="http://maven.apache.org/POM/4.0.0"

  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>org.acme</groupId>

  <artifactId>money-tasks</artifactId>

  <version>1.0.0</version>

  <packaging>jar</packaging>

  <name>Acme Corporation Money Tasks</name>

...

</project>

As you can see, this is just a basic pom.xml file that defines a groupId, artifactId and version for the project. We will create a customization that includes a single custom node for our money business.

Add the integration library to your project's dependencies by including this dependency in your pom.xml file:

<dependencies>

  <dependency>

    <groupId>org.activiti.designer</groupId>

    <artifactId>org.activiti.designer.integration</artifactId>

    <version>5.12.0</version> <!-- Use the current Activiti Designer version -->

    <scope>compile</scope>

  </dependency>

</dependencies>

...

<repositories>

  <repository>

      <id>Activiti</id>

      <url>https://maven.alfresco.com/nexus/content/groups/public/</url>

   </repository>

</repositories>

Finally, in the pom.xml file, add the configuration for the maven-compiler-plugin so the Java source level is at least 1.5 (see snippet below). You will need this in order to use annotations. You can also include instructions for Maven to generate the JAR's MANIFEST.MF file. This is not required, but you can use a specific property in the manifest to provide a name for your extension (this name may be shown at certain places in the designer and is primarily intended for future use if you have several extensions in the designer). If you wish to do so, include the following snippet in pom.xml:

<build>

  <plugins>

        <plugin>

      <artifactId>maven-compiler-plugin</artifactId>

      <configuration>

        <source>1.5</source>

        <target>1.5</target>

        <showDeprecation>true</showDeprecation>

        <showWarnings>true</showWarnings>

        <optimize>true</optimize>

      </configuration>

    </plugin>

    <plugin>

      <groupId>org.apache.maven.plugins</groupId>

      <artifactId>maven-jar-plugin</artifactId>

      <version>2.3.1</version>

      <configuration>

        <archive>

          <index>true</index>

          <manifest>

            <addClasspath>false</addClasspath>

            <addDefaultImplementationEntries>true</addDefaultImplementationEntries>

          </manifest>

          <manifestEntries>

            <ActivitiDesigner-Extension-Name>Acme Money</ActivitiDesigner-Extension-Name>

          </manifestEntries>

        </archive>

      </configuration>

    </plugin>

  </plugins>

</build>

The name for the extension is described by the ActivitiDesigner-Extension-Name property. The only thing left to do now is tell Eclipse to setup the project according to the instructions in pom.xml. So open up a command shell and go to the root folder of your project in the Eclipse workspace. Then execute the following Maven command:

mvn eclipse:eclipse

Wait until the build is successful. Refresh the project (use the project's context menu (right-click) and select Refresh). You should now have the src/main/java and src/main/resources folders as source folders in the Eclipse project.

**Note**

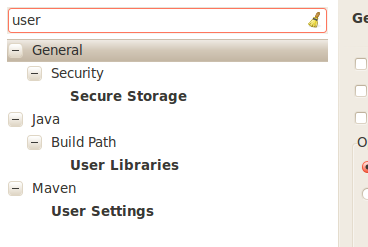
You can of course also use the [m2eclipse](http://www.eclipse.org/m2e) plugin and simply enable Maven dependency management from the context menu (right-click) of the project. Then choose Maven > Update project configuration from the project's context menu. That should setup the source folders as well.

That's it for the setup. Now you're ready to start creating customizations to Activiti Designer!

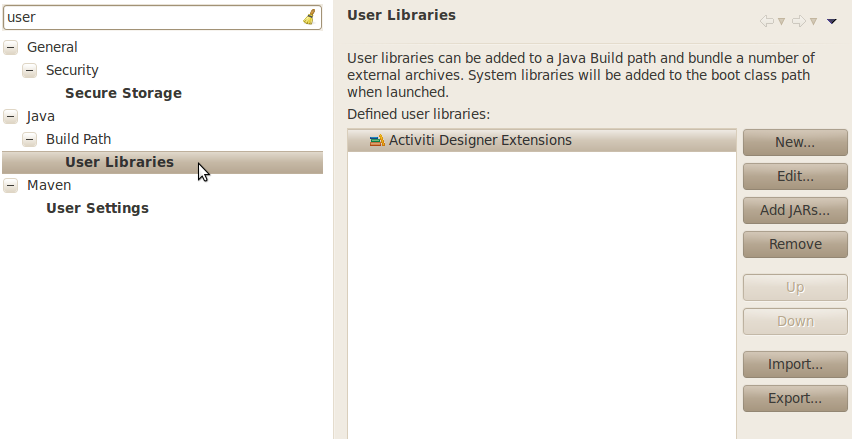
**Applying your extension to Activiti Designer**

You might be wondering how you can add your extension to Activiti Designer so your customizations are applied. These are the steps to do just that:

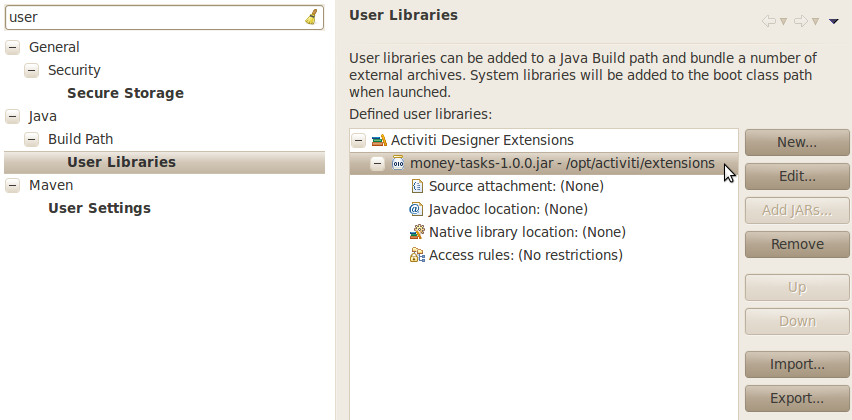
* Once you've created your extension JAR (for instance, by performing a mvn install in your project to build it with Maven), you need to transfer the extension to the computer where Activiti Designer is installed;
* Store the extension somewhere on the hard drive where it will be able to remain and remember the location. *Note:* the location must be outside the Eclipse workspace of Activiti Designer - storing the extension inside the workspace will lead to the user getting a popup error message and the extensions being unavailable;
* Start Activiti Designer and from the menu, select Window > Preferences
* In the preferences screen, type user as keyword. You should see an option to access the User Libraries in Eclipse in the Java section.



* Select the User Libraries item and a tree view shows up to the right where you can add libraries. You should see the default group where you can add extensions to Activiti Designer (depending on your Eclipse installation, you might see several others as well).



* Select the Activiti Designer Extensions group and click the Add JARs... button. Navigate to to folder where your extension is stored and select the extension file you want to add. After completing this, your preferences screen should show the extension as part of the Activiti Designer Extensions group, as shown below.



* Click the OK button to save and close the preferences dialog. The Activiti Designer Extensions group is automatically added to new Activiti projects you create. You can see the user library as entry in the project's tree in the Navigator or Package Explorer. If you already had Activiti projects in the workspace, you should also see the new extensions show up in the group. An example is shown below.



Diagrams you open will now have the shapes from the new extension in their palette (or shapes disabled, depending on the customizations in your extension). If you already had a diagram opened, close and reopen it to see the changes in the palette.

**Adding shapes to the palette**

With your project set up, you can now easily add shapes to the palette. Each shape you wish to add is represented by a class in your JAR. Take note that these classes are not the classes that will be used by the Activiti engine during runtime. In your extension you describe the properties that can be set in Activiti Designer for each shape. From these shapes, you can also define the runtime characteristics that should be used by the engine when a process instance reaches the node in the process. The runtime characteristics can use any of the options that Activiti supports for regular ServiceTasks. See [this section](http://www.mossle.com/docs/activiti/#eclipseDesignerConfiguringRuntime) for more details.

A shape's class is a simple Java class, to which a number of annotations are added. The class should implement the CustomServiceTask interface, but you shouldn't implement this interface yourself. Extend the AbstractCustomServiceTask base class instead (at the moment you MUST extend this class directly, so no abstract classes in between). In the Javadoc for that class you can find instructions on the defaults it provides and when you should override any of the methods it already implements. Overrides allow you to do things such as providing icons for the palette and in the shape on the canvas (these can be different) and specifying the base shape you want the node to have (activity, event, gateway).

/\*\*

 \* @author John Doe

 \* @version 1

 \* @since 1.0.0

 \*/

public class AcmeMoneyTask extends AbstractCustomServiceTask {

...

}

You will need to implement the getName() method to determine the name the node will have in the palette. You can also put the nodes in their own drawer and provide an icon. Override the appropriate methods from AbstractCustomServiceTask. If you want to provide an icon, make sure it's in the src/main/resources package in your JAR and is about 16x16 pixels and a JPEG or PNG format. The path you supply is relative to that folder.

You can add properties to the shape by adding members to the class and annotating them with the @Property annotation like this:

@Property(type = PropertyType.TEXT, displayName = "Account Number")

@Help(displayHelpShort = "Provide an account number", displayHelpLong = HELP\_ACCOUNT\_NUMBER\_LONG)

private String accountNumber;

There are several PropertyType values you can use, which are described in more detail in [this section](http://www.mossle.com/docs/activiti/#eclipseDesignerPropertyTypes). You can make a field required by setting the required attribute to true. A message and red background will appear if the user doesn't fill out the field.

If you want to ensure the order of the various properties in your class as they appear in the property screen, you should specify the order attribute of the @Property annotation.

As you can see, there's also an @Help annotation that's used to provide the user some guidance when filling out the field. You can also use the @Help annotation on the class itself - this information is shown at the top of the property sheet presented to the user.

Below is the listing for a further elaboration of the MoneyTask. A comment field has been added and you can see an icon is included for the node.

/\*\*

 \* @author John Doe

 \* @version 1

 \* @since 1.0.0

 \*/

@Runtime(javaDelegateClass = "org.acme.runtime.AcmeMoneyJavaDelegation")

@Help(displayHelpShort = "Creates a new account", displayHelpLong = "Creates a new account using the account number specified")

public class AcmeMoneyTask extends AbstractCustomServiceTask {

  private static final String HELP\_ACCOUNT\_NUMBER\_LONG = "Provide a number that is suitable as an account number.";

  @Property(type = PropertyType.TEXT, displayName = "Account Number", required = true)

  @Help(displayHelpShort = "Provide an account number", displayHelpLong = HELP\_ACCOUNT\_NUMBER\_LONG)

  private String accountNumber;

  @Property(type = PropertyType.MULTILINE\_TEXT, displayName = "Comments")

  @Help(displayHelpShort = "Provide comments", displayHelpLong = "You can add comments to the node to provide a brief description.")

  private String comments;

  /\*

   \* (non-Javadoc)

   \*

   \* @see org.activiti.designer.integration.servicetask.AbstractCustomServiceTask #contributeToPaletteDrawer()

   \*/

  @Override

  public String contributeToPaletteDrawer() {

    return "Acme Corporation";

  }

  @Override

  public String getName() {

    return "Money node";

  }

  /\*

   \* (non-Javadoc)

   \*

   \* @see org.activiti.designer.integration.servicetask.AbstractCustomServiceTask #getSmallIconPath()

   \*/

  @Override

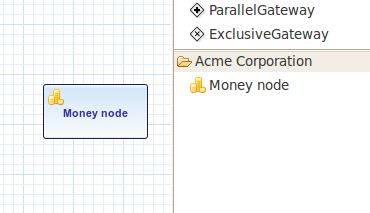
  public String getSmallIconPath() {

    return "icons/coins.png";

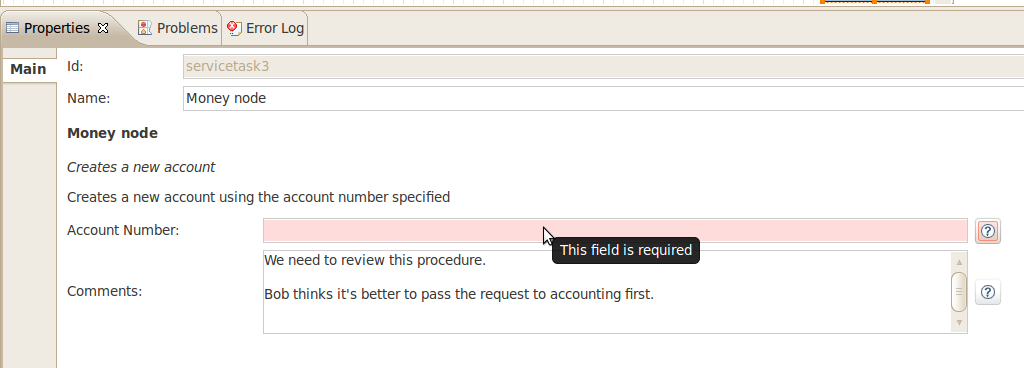
  }

}

If you extend Activiti Designer with this shape, The palette and corresponding node will look like this:

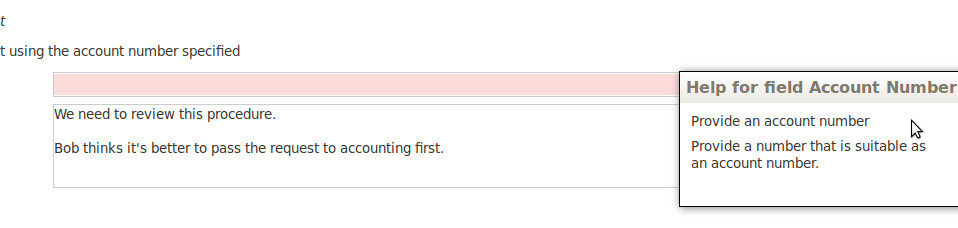


The properties screen for the money task is shown below. Note the required message for the accountNumber field.



Users can enter static text or use expressions that use process variables in the property fields when creating diagrams (e.g. "This little piggy went to ${piggyLocation}"). Generally, this applies to text fields where users are free to enter any text. If you expect users to want to use expressions and you apply runtime behavior to your CustomServiceTask (using @Runtime), make sure to use Expression fields in the delegate class so the expressions are correctly resolved at runtime. More information on runtime behavior can be found in [this section](http://www.mossle.com/docs/activiti/#eclipseDesignerConfiguringRuntime).

The help for fields is offered by the buttons to the right of each property. Clicking on the button shows a popup as displayed below.



**Configuring runtime execution of Custom Service Tasks**

With your fields setup and your extension applied to Designer, users can configure the properties of the service task when modelling a process. In most cases, you will want to use these user-configured properties when the process is executed by Activiti. To do this, you must instruct Activiti which class to instantiate when the process reaches your CustomServiceTask.

There is a special annotation for specifying the runtime characteristics of your CustomServiceTask, the @Runtime annotation. Here's an example of how to use it:

@Runtime(javaDelegateClass = "org.acme.runtime.AcmeMoneyJavaDelegation")

Your CustomServiceTask will result in a normal ServiceTask in the BPMN output of processes modelled with it. Activiti enables [several ways](http://www.mossle.com/docs/activiti/#bpmnJavaServiceTask) to define the runtime characteristics of ServiceTasks. Therefore, the @Runtime annotation can take one of three attributes, which match directly to the options Activiti provides, like this:

* javaDelegateClass maps to activiti:class in the BPMN output. Specify the fully qualified classname of a class that implements JavaDelegate.
* expression maps to activiti:expression in the BPMN output. Specify an expression to a method to be executed, such as a method in a Spring Bean. You should *not* specify any @Property annotations on fields when using this option. For more information, see below.
* javaDelegateExpression maps to activiti:delegateExpression in the BPMN output. Specify an expression to a class that implements JavaDelegate.

The user's property values will be injected into the runtime class if you provide members in the class for Activiti to inject into. The names should match the names of the members in your CustomServiceTask. For more information, consult [this part](http://www.mossle.com/docs/activiti/#serviceTaskFieldInjection) of the userguide. Note that since version 5.11.0 of the Designer you can use the Expression interface for dynamic field values. This means that the value of the property in the Activiti Designer must contain an expression and this expression will then be injected into an Expression property in the JavaDelegate implementation class.

**Note**

You can use @Property annotations on members of your CustomServiceTask, but this will not work if you use @Runtime's expression attribute. The reason for this is that the expression you specify will be attempted to be resolved to a *method* by Activiti, not to a class. Therefore, no injection into a class will be performed. Any members marked with @Property will be ignored by Designer if you use expression in your @Runtime annotation. Designer will not render them as editable fields in the node's property pane and will produce no output for the properties in the process' BPMN.

**Note**

Note that the runtime class shouldn't be in your extension JAR, as it's dependent on the Activiti libraries. Activiti needs to be able to find it at runtime, so it needs to be on the Activiti engine's classpath.

The examples project in Designer's source tree contains examples of the different options for configuring @Runtime. Take a look in the money-tasks project for some starting points. The examples refer to delegate class examples that are in the money-delegates project.

**Property types**

This section describes the property types you can use for a CustomServiceTask by setting its type to a PropertyType value.

**PropertyType.TEXT**

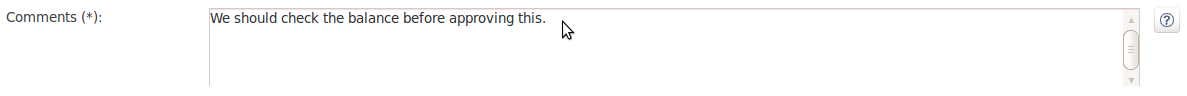
Creates a single line text field as shown below. Can be a required field and shows validation messages as a tooltip. Validation failures are displayed by changing the background of the field to a light red color.

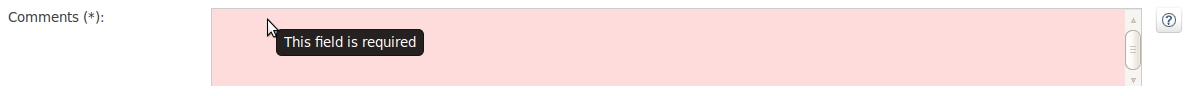
http://www.mossle.com/docs/activiti/images/designer.property.text.png

http://www.mossle.com/docs/activiti/images/designer.property.text.invalid.png

**PropertyType.MULTILINE\_TEXT**

Creates a multiline text field as shown below (height is fixed at 80 pixels). Can be a required field and shows validation messages as a tooltip. Validation failures are displayed by changing the background of the field to a light red color.





**PropertyType.PERIOD**

Creates a structured editor for specifying a period of time by editing amounts of each unit with a spinner control. The result is shown below. Can be a required field (which is interpreted such that not all values may be 0, so at least 1 part of the period must have a non-zero value) and shows validation messages as a tooltip. Validation failures are displayed by changing the background of the entire field to a light red color. The value of the field is stored as a string of the form 1y 2mo 3w 4d 5h 6m 7s, which represents 1 year, 2 months, 3 weeks, 4 days, 6 minutes and 7 seconds. The entire string is always stored, even if parts are 0.

http://www.mossle.com/docs/activiti/images/designer.property.period.png

http://www.mossle.com/docs/activiti/images/designer.property.period.invalid.png

**PropertyType.BOOLEAN\_CHOICE**

Creates a single checkbox control for boolean or toggle choices. Note that you can specify the required attribute on the Property annotation, but it will not be evaluated because that would leave the user without a choice whether to check the box or not. The value stored in the diagram is java.lang.Boolean.toString(boolean), which results in "true" or "false".

http://www.mossle.com/docs/activiti/images/designer.property.boolean.choice.png

**PropertyType.RADIO\_CHOICE**

Creates a group of radio buttons as shown below. Selection of any of the radio buttons is mutually exclusive with selection of any of the others (i.e., only one selection allowed). Can be a required field and shows validation messages as a tooltip. Validation failures are displayed by changing the background of the group to a light red color.

This property type expects the class member you have annotated to also have an accompanying @PropertyItems annotation (for an example, see below). Using this additional annotation, you can specify the list of items that should be offered in an array of Strings. Specify the items by adding two array entries for each item: first, the label to be shown; second, the value to be stored.

@Property(type = PropertyType.RADIO\_CHOICE, displayName = "Withdrawl limit", required = true)

@Help(displayHelpShort = "The maximum daily withdrawl amount ", displayHelpLong = "Choose the maximum daily amount that can be withdrawn from the account.")

@PropertyItems({ LIMIT\_LOW\_LABEL, LIMIT\_LOW\_VALUE, LIMIT\_MEDIUM\_LABEL, LIMIT\_MEDIUM\_VALUE, LIMIT\_HIGH\_LABEL, LIMIT\_HIGH\_VALUE })

private String withdrawlLimit;

http://www.mossle.com/docs/activiti/images/designer.property.radio.choice.png

http://www.mossle.com/docs/activiti/images/designer.property.radio.choice.invalid.png

**PropertyType.COMBOBOX\_CHOICE**

Creates a combobox with fixed options as shown below. Can be a required field and shows validation messages as a tooltip. Validation failures are displayed by changing the background of the combobox to a light red color.

This property type expects the class member you have annotated to also have an accompanying @PropertyItems annotation (for an example, see below). Using this additional annotation, you can specify the list of items that should be offered in an array of Strings. Specify the items by adding two array entries for each item: first, the label to be shown; second, the value to be stored.

@Property(type = PropertyType.COMBOBOX\_CHOICE, displayName = "Account type", required = true)

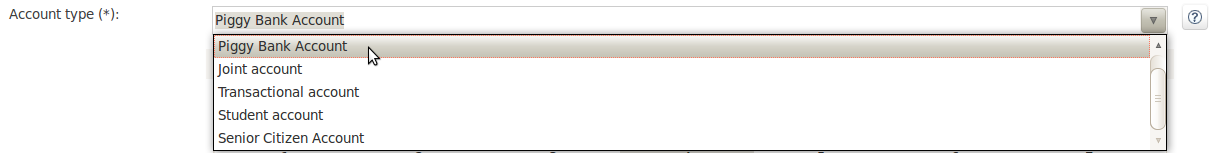
@Help(displayHelpShort = "The type of account", displayHelpLong = "Choose a type of account from the list of options")

@PropertyItems({ ACCOUNT\_TYPE\_SAVINGS\_LABEL, ACCOUNT\_TYPE\_SAVINGS\_VALUE, ACCOUNT\_TYPE\_JUNIOR\_LABEL, ACCOUNT\_TYPE\_JUNIOR\_VALUE, ACCOUNT\_TYPE\_JOINT\_LABEL,

  ACCOUNT\_TYPE\_JOINT\_VALUE, ACCOUNT\_TYPE\_TRANSACTIONAL\_LABEL, ACCOUNT\_TYPE\_TRANSACTIONAL\_VALUE, ACCOUNT\_TYPE\_STUDENT\_LABEL, ACCOUNT\_TYPE\_STUDENT\_VALUE,

  ACCOUNT\_TYPE\_SENIOR\_LABEL, ACCOUNT\_TYPE\_SENIOR\_VALUE })

private String accountType;



http://www.mossle.com/docs/activiti/images/designer.property.combobox.choice.invalid.png

**PropertyType.DATE\_PICKER**

Creates a date selection control as shown below. Can be a required field and shows validation messages as a tooltip (note, that the control used will auto-set the selection to the date on the system, so the value is seldom empty). Validation failures are displayed by changing the background of the control to a light red color.

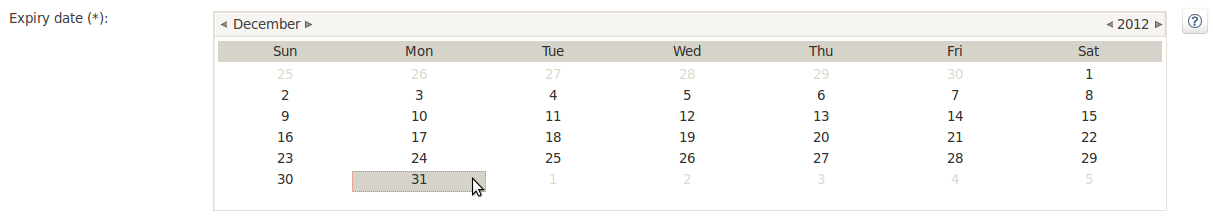
This property type expects the class member you have annotated to also have an accompanying @DatePickerProperty annotation (for an example, see below). Using this additional annotation, you can specify the date time pattern to be used to store dates in the diagram and the type of datepicker you would like to be shown. Both attributes are optional and have default values that will be used if you don't specify them (these are static variables in the DatePickerProperty annotation). The dateTimePattern attribute should be used to supply a pattern to the SimpleDateFormat class. When using the swtStyle attribute, you should specify an integer value that is supported by SWT's DateTime control, because this is the control that is used to render this type of property.

@Property(type = PropertyType.DATE\_PICKER, displayName = "Expiry date", required = true)

@Help(displayHelpShort = "The date the account expires ", displayHelpLong = "Choose the date when the account will expire if no extended before the date.")

@DatePickerProperty(dateTimePattern = "MM-dd-yyyy", swtStyle = 32)

private String expiryDate;



**PropertyType.DATA\_GRID**

Creates a data grid control as shown below. A data grid can be used to allow the user to enter an arbitrary amount of rows of data and enter values for a fixed set of columns in each of those rows (each individual combination of row and column is referred to as a cell). Rows can be added and removed as the user sees fit.

This property type expects the class member you have annotated to also have an accompanying @DataGridProperty annotation (for an example, see below). Using this additional annotation, you can specify some specific attributes of the data grid. You are required to reference a different class to determine which columns go into the grid with the itemClass attribute. Activiti Designer expects the member type to be a List. By convention, you can use the class of the itemClass attribute as its generic type. If, for example, you have a grocery list that you edit in the grid, you would define the columns of the grid in the GroceryListItem class. From your CustomServiceTask, you would refer to it like this:

@Property(type = PropertyType.DATA\_GRID, displayName = "Grocery List")

@DataGridProperty(itemClass = GroceryListItem.class)

private List<GroceryListItem> groceryList;

The "itemClass" class uses the same annotations you would otherwise use to specify fields of a CustomServiceTask, with the exception of using a data grid. Specifically, TEXT, MULTILINE\_TEXT and PERIOD are currently supported. You'll notice the grid will create single line text controls for each field, regardless of the PropertyType. This is done on purpose to keep the grid graphically appealing and readable. If you consider the regular display mode for a PERIOD PropertyType for instance, you can imagine it would never properly fit in a grid cell without cluttering the screen. For MULTILINE\_TEXT and PERIOD, a double-click mechanism is added to each field which pops up a larger editor for the PropertyType. The value is stored to the field after the user clicks OK and is therefore readable within the grid.

Required attributes are handled in a similar manner to regular fields of type TEXT and the entire grid is validated as soon as any field loses focus. The background color of the text control in a specific cell of the data grid is changed to light red if there are validation failures.

By default, the component allows the user to add rows, but not to determine the order of those rows. If you wish to allow this, you should set the orderable attribute to true, which enables buttons at the end of each row to move it up or down in the grid.

**Note**

At the moment, this property type is not correctly injected into your runtime class.



**Disabling default shapes in the palette**

This customization requires you to include a class in your extension that implements the DefaultPaletteCustomizer interface. You should not implement this interface directly, but subclass the AbstractDefaultPaletteCustomizer base class. Currently, this class provides no functionality, but future versions of the DefaultPaletteCustomizer interface will offer more capabilities for which this base class will provide some sensible defaults so it's best to subclass so your extension will be compatible with future releases.

Extending the AbstractDefaultPaletteCustomizer class requires you to implement one method, disablePaletteEntries(), from which you must return a list of PaletteEntry values. For each of the default shapes, you can disable it by adding its corresponding PaletteEntry value to your list. Note that if you remove shapes from the default set and there are no remaining shapes in a particular drawer, that drawer will be removed from the palette in its entirety. If you wish to disable all of the default shapes, you only need to add PaletteEntry.ALL to your result. As an example, the code below disables the Manual task and Script task shapes in the palette.

public class MyPaletteCustomizer extends AbstractDefaultPaletteCustomizer {

  /\*

   \* (non-Javadoc)

   \*

   \* @see org.activiti.designer.integration.palette.DefaultPaletteCustomizer#disablePaletteEntries()

   \*/

  @Override

  public List<PaletteEntry> disablePaletteEntries() {

    List<PaletteEntry> result = new ArrayList<PaletteEntry>();

    result.add(PaletteEntry.MANUAL\_TASK);

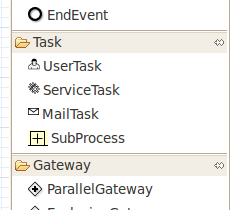
    result.add(PaletteEntry.SCRIPT\_TASK);

    return result;

  }

}

The result of applying this extension is shown in the picture below. As you can see, the manual task and script task shapes are no longer available in the Tasks drawer.



To disable all of the default shapes, you could use something similar to the code below.

public class MyPaletteCustomizer extends AbstractDefaultPaletteCustomizer {

  /\*

   \* (non-Javadoc)

   \*

   \* @see org.activiti.designer.integration.palette.DefaultPaletteCustomizer#disablePaletteEntries()

   \*/

  @Override

  public List<PaletteEntry> disablePaletteEntries() {

    List<PaletteEntry> result = new ArrayList<PaletteEntry>();

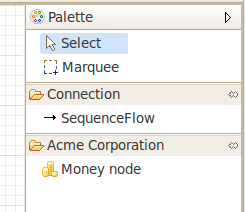
    result.add(PaletteEntry.ALL);

    return result;

  }

}

The result will look like this (notice that the drawers the default shapes were in are no longer in the palette):



**Validating diagrams and exporting to custom output formats**

Besides customizing the palette, you can also create extensions to Activiti Designer that can perform validations and save information from the diagram to custom resources in the Eclipse workspace. There are built-in extension points for doing this and this section explains how to use them.

Activiti Designer allows you to write extensions that validate diagrams. There are already validations of BPMN constructs in the tool by default, but you can add your own if you want to validate additional items such as modeling conventions or the values in properties of CustomServiceTasks. These extensions are known as Process Validators.

You can also Activiti Designer to publish to additional formats when saving diagrams. These extensions are called Export Marshallers and are invoked automatically by Activiti Designer on each save action by the user. This behavior can be enabled or disabled by setting a preference in Eclipse's preferences dialog for each format to be saved.

You can compare these extensions to the BPMN 2.0 validation, BPMN 2.0 export and process image saving that's performed during save actions by default in Activiti Designer. In fact, these functions use exactly the same extension features you can use to save to your own formats.

Often, you will want to combine a ProcessValidator and an ExportMarshaller. Let's say you have a number of CustomServiceTasks in use that have properties you would like to use in the process that gets generated. However, before the process is generated, you want to validate some of those values first. Combining a ProcessValidator and ExportMarshaller is the best way to accomplish this and Activiti Designer enables you to plug your extensions into the tool seamlessly.

To create a ProcessValidator or an ExportMarshaller, you need to create a different kind of extension than for extending the palette. The reason for this is simple: from your code you will need access to more APIs than are offered by the integration library. In particular, you will need classes that are available in Eclipse itself. So to get started, you should create an Eclipse plugin (which you can do by using Eclipse's PDE support) and package it in a custom Eclipse product or feature. It's beyond the scope of this user guide to explain all the details involved in developing Eclipse plugins, so the instructions below are limited to the functionality for extending Activiti Designer.

Your bundle should be dependent on the following libraries:



Both ProcessValidators and ExportMarshallers are created by extending a base class. These base classes inherit some useful methods from their superclass, the AbstractDiagramWorker class. Using these methods you can create information, warning and error markers that show up in Eclipse's problems view for the user to figure out what's wrong or important. You can also access the diagram through Resources and InputStreams for the diagram's content using these methods in the AbstractDiagramWorker class.

It's probably a good idea to invoke clearMarkers() as one of the first things you do in either a ProcessValidator or an ExportMarshaller; this will clear any previous markers for your worker (markers are automatically linked to the worker and clearing markers for one worker leaves other markers untouched). For example:

// Clear markers for this diagram first

clearMarkers(getResource(diagram.eResource().getURI()));

You should also use the progress monitor provided to report your progress back to the user because validations and/or marshalling actions can take up some time during which the user is forced to wait. Reporting progress requires some knowledge of how you should use Eclipse's features. Take a look at [this article](http://www.eclipse.org/articles/Article-Progress-Monitors/article.html) for a thorough explanation of the concepts and usage.

**Creating a ProcessValidator extension**

Create an extension to the org.activiti.designer.eclipse.extension.validation.ProcessValidator extension point in your plugin.xml file. For this extension point, you are required to subclass the AbstractProcessValidator class.

<?eclipse version="3.6"?>

<plugin>

  <extension

    point="org.activiti.designer.eclipse.extension.validation.ProcessValidator">

    <ProcessValidator

      class="org.acme.validation.AcmeProcessValidator">

    </ProcessValidator>

  </extension>

</plugin>

public class AcmeProcessValidator extends AbstractProcessValidator {

}

You have to implement a number of methods. Most importantly, implement getValidatorId() so you return a globally unique ID for your validator. This will enable you to invoke it from and ExportMarshaller, or event let someone *else* invoke your validator from their ExportMarshaller. Implement getValidatorName() and return a logical name for your validator. This name is shown to the user in dialogs. In getFormatName(), you can return the type of diagram the validator typically validates.

The validation work itself is done in the validateDiagram() method. From this point on, it's up to your specific functionality what you code here. Typically, however, you will want to start by getting hold of the nodes in the diagram's process, so you can iterate through them, collect, compare and validate data. This snippet shows you how to do this:

final EList<EObject> contents = getResourceForDiagram(diagram).getContents();

for (final EObject object : contents) {

  if (object instanceof StartEvent ) {

  // Perform some validations for StartEvents

  }

  // Other node types and validations

}

Don't forget to invoke addProblemToDiagram() and/or addWarningToDiagram(), etc as you go through your validations. Make sure you return a correct boolean result at the end to indicate whether you consider the validation as succeeded or failed. This can be used by and invoking ExportMarshaller to determine the next course of action.

**Creating an ExportMarshaller extension**

Create an extension to the org.activiti.designer.eclipse.extension.export.ExportMarshaller extension point in your plugin.xml file. For this extension point, you are required to subclass the AbstractExportMarshaller class. This abstract base class provides you with a number of useful methods when marshalling to your own format, but most importantly it allows you to save resources to the workspace and to invoke validators.

<?eclipse version="3.6"?>

<plugin>

  <extension

    point="org.activiti.designer.eclipse.extension.export.ExportMarshaller">

    <ExportMarshaller

      class="org.acme.export.AcmeExportMarshaller">

    </ExportMarshaller>

  </extension>

  </plugin>

public class AcmeExportMarshaller extends AbstractExportMarshaller {

}

You are required to implement some methods, such as getMarshallerName() and getFormatName(). These methods are used to display options to the user and to show information in progress dialogs, so make sure the descriptions you return reflect the functionality you are implementing.

The bulk of your work is performed in the marshallDiagram(Diagram diagram, IProgressMonitor monitor) method. You are provided with the diagram object, which contains all of the information about the objects in the diagram (BPMN constructs) and the graphical representation.

If you want to perform a certain validation first, you can invoke the validator directly from your marshaller. You receive a boolean result from the validator, so you know whether validation succeeded. In most cases you won't want to proceed with marshalling the diagram if it's not valid, but you might choose to go ahead anyway or even create a different resource if validation fails. For example:

final boolean validDiagram = invokeValidator(AcmeConstants.ACME\_VALIDATOR\_ID, diagram, monitor);

if (!validDiagram) {

  addProblemToDiagram(diagram, "Marshalling to " + getFormatName() + " format was skipped because validation of the diagram failed.", null);

} else {

  //proceed with marshalling

}

As you can see, here we have chosen to cancel the marshalling if the validator (identified by a constant here) returns false as result. We have also added an additional marker to the diagram so the user can see an explanation why the file wasn't created. This is not required, but seems helpful to the user and shows how you can use these utilities from both ProcessValidators and ExportMarshallers.

Once you have all the data you need, you should invoke the saveResource() method to create a file containing your data. You can invoke saveResource() as many times as you wish from a single ExportMarshaller; a marshaller can therefore be used to create more than one output file.

You can construct a filename for your output resource(s) by using some of the methods in the AbstractDiagramWorker class. There are a couple of useful variables you can have parsed, allowing you to create filenames such as <original-filename>\_<my-format-name>.xml. These variables are described in the Javadocs, but here's an example how to use one of them:

private static final String FILENAME\_PATTERN = ExportMarshaller.PLACEHOLDER\_ORIGINAL\_FILENAME + ".acme.axml";

...

saveResource(getRelativeURIForDiagram(diagram, FILENAME\_PATTERN), bais, this.monitor);

What happens here is that a static member is used to describe the filename pattern (this is just a best practice, you can specify the string any way you like of course) and the pattern uses the ExportMarshaller.PLACEHOLDER\_ORIGINAL\_FILENAME constant to insert a variable for the original filename. Later on in the marshallDiagram() method, getRelativeURIForDiagram() is invoked and it will parse the filename for any variables and substitute them. You provide saveResource() with an InputStream to your data and it will save the data to a resource with a relative path to the original diagram.

Again, you should also use the progress monitor provided to report your progress back to the user. How to do this is described in [this article](http://www.eclipse.org/articles/Article-Progress-Monitors/article.html).

**Chapter 13. Activiti Explorer**

**Table of Contents**

[流程图](http://www.mossle.com/docs/activiti/#N1291C)

[任务](http://www.mossle.com/docs/activiti/#N12938)

[启动流程实例](http://www.mossle.com/docs/activiti/#N12968)

[我的流程实例](http://www.mossle.com/docs/activiti/#N1297B)

[管理](http://www.mossle.com/docs/activiti/#N1298A)

[报表](http://www.mossle.com/docs/activiti/#explorer.reporting)

[报告数据JSON](http://www.mossle.com/docs/activiti/#explorer.reporting.json)

[实例流程](http://www.mossle.com/docs/activiti/#explorer.reporting.example)

[报告开始表单](http://www.mossle.com/docs/activiti/#explorer.reporting.start.form)

[流程例子](http://www.mossle.com/docs/activiti/#explorer.reporting.examples)

[修改数据库](http://www.mossle.com/docs/activiti/#explorer.change.database)

Activiti Explorer，我习惯称之为Activiti控制台，后面也这么翻译。Activiti控制台是一个web应用程序，当我们从Activiti的官方网站下载Activiti的压缩zip文件时候，Activiti控制台在${Activiti\_home}/wars文件夹下面。该控制台的目的并不是创建一个完善的web应用程序，仅仅是为客户端用户准备的应用程序，但是却能够练习和展示Activiti的功能。正如这样，控制台仅仅只是一个Demo， 可能有人会使用该控制台集成到他们自己的系统之中。另外，对于该控制台，我们使用了一个内存数据库，也很容易换成你自己的数据库（查看WEB-INF文件夹下面的applicationContext.xml文件）。

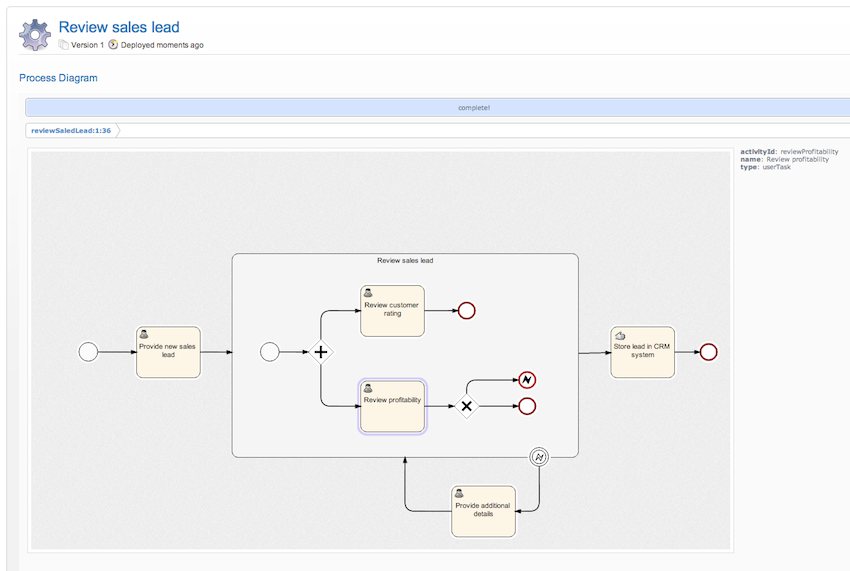
随后，登录进该控制台，你将会看见四个比较大的图标按钮用于显示主要功能。

http://www.mossle.com/docs/activiti/images/explorer.tabs.png

* **Tasks**: 任务管理功能。 这里，如果你是办理人，你可以看见运行中流程实例的自己的待办任务，或者你可以拾取组任务。 控制台涉及的功能，子任务的工作，不同角色的人，等等...控制台也可以允许创建一个独立的任务，该任务并没有关联任何流程实例。
* **Process**: 显示部署的流程定义列表，并且可以启动一个新的流程实例。
* **Reporting**: 生成报表和显示之前保存历史的结果数据。查看[报表这一节](http://www.mossle.com/docs/activiti/#explorer.reporting)可以获取更多的信息。
* **Manage**: 当登录的用户具有超级管理员权限才能够看见。用于管理Activiti的流程引擎：管理用于和组，执行和查看停止的jobs，查看数据库和部署新的流程定义。

**流程图**

控制台包含的功能，使用[Raphaël](http://raphaeljs.com/)Javascript框架自动生成一张流程图。 当流程定义XML包含的BPMN注入信息时。该流程图才能够生成。当流程定义XML中并没有BPMN注入信息但是部署的时候包含一张流程图，那么该图片也将会被显示。



当你并不想使用Javascript生成流程图，你可以在ui.properties文件中禁用它。

activiti.ui.jsdiagram = false

除了在控制台上面显示流程图，控制台也会很容易的包含你想要查看的流程图。 下面的URL将会显示流程定义图片，根据留存定义的ID：

http://localhost:8080/activiti-explorer/diagram-viewer/index.html?processDefinitionId=reviewSaledLead:1:36

它也可以显示当前流程实例的状态，通过添加一个processInstanceId的请求参数，如下:

http://localhost:8080/activiti-explorer/diagram-viewer/index.html?processDefinitionId=reviewSaledLead:1:36&processInstanceId=41

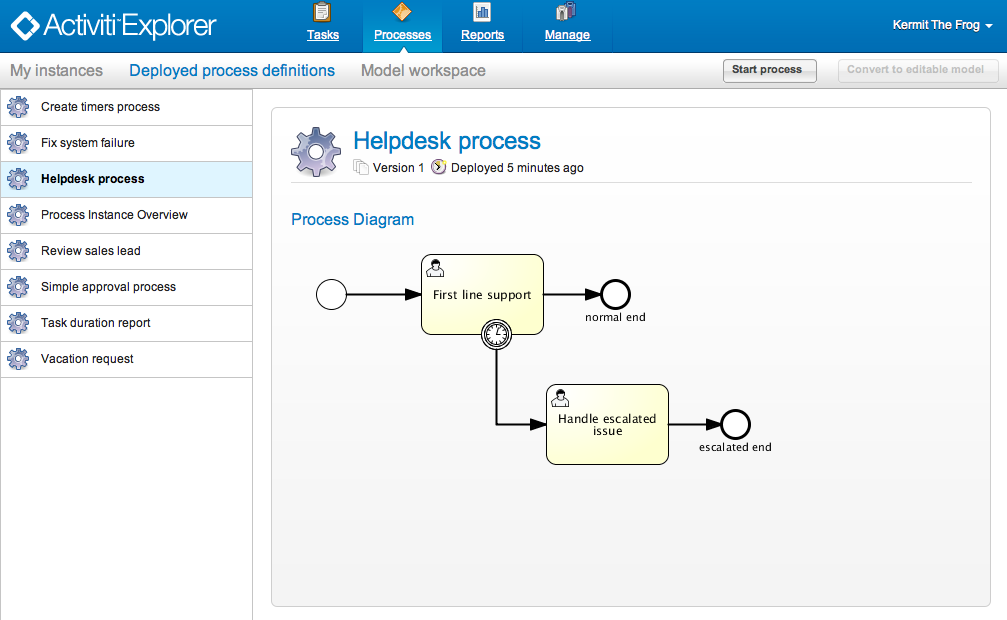
**任务**



* **Inbox:** 显示登录用户需要办理的所有任务列表。
* **My tasks:** 显示登录用户任务拥有者的任务列表。当你创建一个独立的任务，你可以自动化操作该任务。
* **Queued:** 显示不用的组任务列表，并且登录用户在该组中。这里的所有任务都必须先拾取然后才能够完成。
* **Involved:** 显示登录用户被参与的任务（即不是办理人和任务拥有者）。
* **归档**包含已经完成的（历史的）任务。

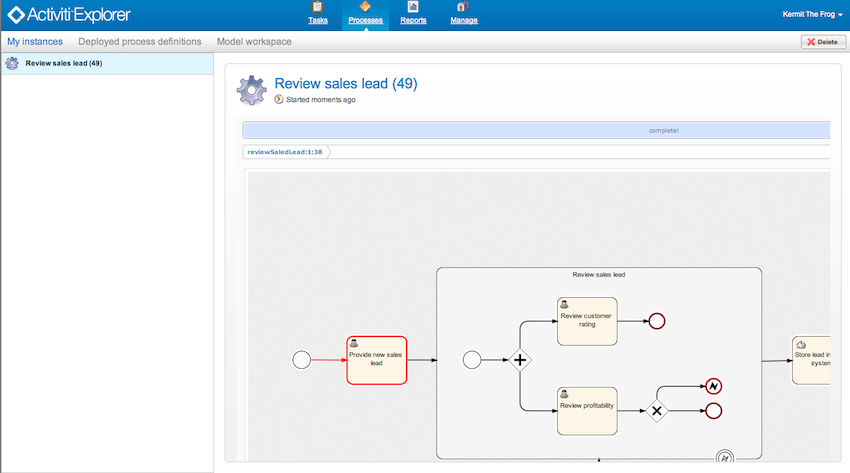
**启动流程实例**

在**流程定义**选项卡，允许你查看Activiti流程引擎部署的所有流程定义。你可以使用页面顶部右边的按钮启动一个新的流程实例。如果该流程定义有一个启动[表单](http://www.mossle.com/docs/activiti/#forms)， 那么在启动流程实例之前就先显示表单。



**我的流程实例**

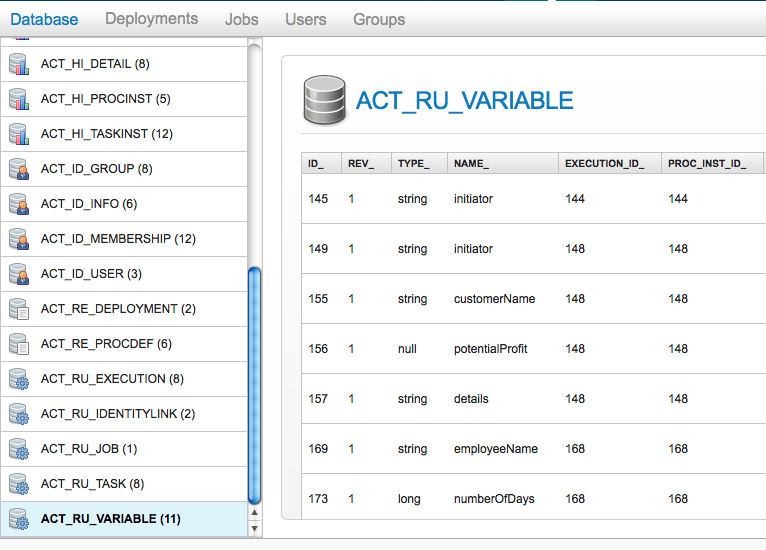
在**我的流程** 选项卡，显示当前登录用户未完成的用户任务的所有流程实例。这也很直观的显示了流程实例的当前活动和存储的流程变量。



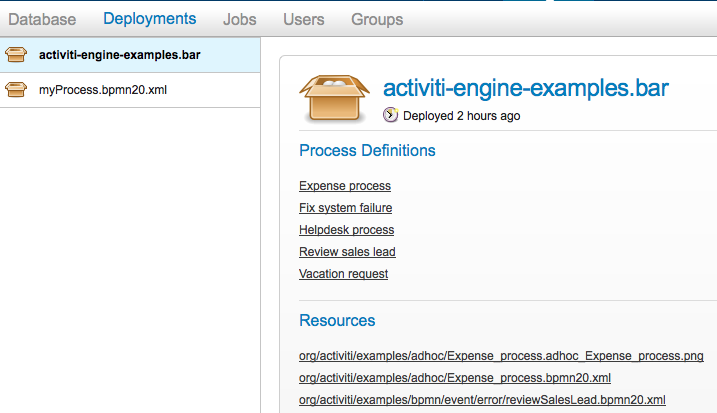
**管理**

在管理功能中，只有当登录用户只权限组*admin*中的成员时，该功能才会显示。当点击 *Manage* 图标按钮，提供以下选项列表

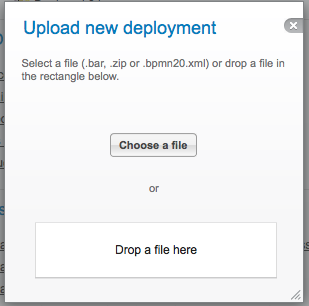
* **数据库:** 在数据库中显示Activiti有关内容.当开发流程或者排除故障等问题的时候是非常有用的。



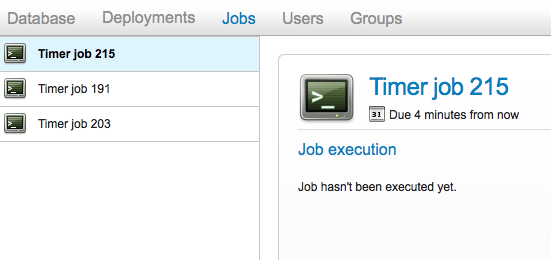
* **部署：** 显示当前流程引擎的部署，并且可以看见部署的内容（流程定义，流程图，业务规则，等等...）



当你点击 *部署* 按钮时，你也可以上传新的部署。从自己的计算机中选择一个业务文档或者一个BPMN20.XML文件，或者简单的拖拽到指定的区域就可以部署一个新的业务流程。



* **Jobs（作业）:** 在左边显示当前的作业（定时器，等等）并且运行手动执行他们（例如在截止时间之前触发定时器）。如果作业执行失败（例如邮件服务器不能正常工作），那么就会显示所有的异常。



* **用户和组** 管理用户和组：创建，修改和删除用户和组。关联用户和组，这样他们就会有更多的权限或者他们能够看见在任务分配给特地的组。



**报表**

控制台附带了一些报表例子并且有能力很轻松的在系统中添加新的报表。 报表功能是位于主功能中的*'报表'*按钮。

http://www.mossle.com/docs/activiti/images/explorer.reporting.png

**重要**: 如果要让报表工作，控制台需要配置历史的级别不能够没有。这默认的配置是满足这一要求的。

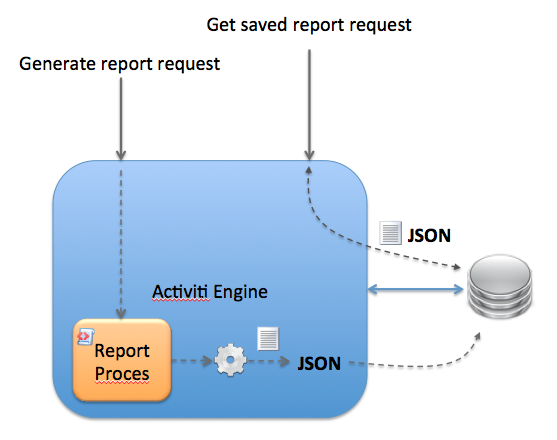
目前，该报表选项卡会显示2个子选项卡：

* **生成报表**: 显示系统中已知的报表列表。允许运行生成的报表。
* **保存报表**: 显示之前保存的所有报表列表。注意，这里仅仅显示的是个人保存的报表，并且不能看见其他人保存的报表。

流程的数据被用于生成报表中的列表和图标。第一次看上去可能会很奇怪，使用流程生成报表数据有几个优势。

* 该流程能够直接访问Activiti流程引擎的内部。他直接可以使用流程引擎访问数据库.
* 作业执行器能够用于任何其他的流程。这意味着你能够异步生存流程或者仅仅异步执行某些步骤。这也意味着你可以使用定时器，例如在某些时间点上面生成报表数据。
* 可以用已知的工具和已知的概念创建一个新的报表。同时，没有新的概念，服务或者应用被需要。部署或者上传一个新的报表与部署一个新的流程是一样的。
* 它可以使用BPMN2.0结构。这意味着所有的东西，比如并行网关， 可以实现基于数据或用户请求输入生成分支。

生成报表数据的流程定义需要**把'activiti-report'设置为分类**， 这样就能在Explorer的报表列表中显示出来。报表流程可繁可简。 能够看到报表的唯一要求是，流程会创建一个名为**reportData**的流程变量。 这个变量必须是json对象的二进制数组。 这个变量必须保存到Activiti的历史表中 （所以要求引擎必须启用历史功能） 所以可以在后面报表保存时获取。



**报告数据JSON**

报表流程必须生成一个变量*reportData*，这是一个要展现给用户的JSON数据。 这个json看起来像这样：

{

  "title": "My Report",

  "datasets": [

    {

      "type" : "lineChart",

      "description" : "My first chart",

      "xaxis" : "Category"

      "yaxis" : "Total sales"

      "data" :

      {

        "A" : 50,

        "B" : 33,

        "C" : 17,

        "D" : 87,

      }

    }

  ]

}

json数据会在Explorer中获取，并用来生成图表或列表。 json的元素为：

* **title**：这个报表的标题
* **datasets**：是数据集的数组，对应报表中不同的图表和列表。
* **type**每个数据集都有一个类型。 这个类型会用来决定如何渲染数据。当前支持的值有： **pieChart, lineChart, barChart 和 list.**
* **description**：每个图表可以 在报表中显示一个可选的描述。
* **x- 和 yaxis**：只对 *lineChart*类型起作用。 这个可选参数可以修改图表坐标系的名称。
* **data**：这是实际的数据。 数据是一个key-value格式的json对象。

**实例流程**

下面的例子演示了一个“流程实例总览”报表。流程本身非常简单，只包含一个脚本任务（除了开始和结束） 使用javascript生成json数据集。 虽然所有Explorer中的例子都使用javascript，它们也可以使用java服务任务。 执行流程最后的结果就是*reportData*变量，保存着数据。

**重要提示：**下面的例子只能运行在JDK 7+环境中。 因为使用了javascript引擎（*Rhino*），如果运行在老JDK版本中 会无法实现一些结果，来像下面一样编写脚本。 参考下面的一个JDk 6+兼容的例子。

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

<definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:activiti="http://activiti.org/bpmn"

    xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI" xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC"

    xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI" typeLanguage="http://www.w3.org/2001/XMLSchema"

    expressionLanguage="http://www.w3.org/1999/XPath"

    targetNamespace="activiti-report">

    <process id="process-instance-overview-report" name="Process Instance Overview" isExecutable="true">

        <startEvent id="startevent1" name="Start" />

        <sequenceFlow id="flow1" sourceRef="startevent1" targetRef="generateDataset" />

        <scriptTask id="generateDataset" name="Execute script" scriptFormat="JavaScript" activiti:autoStoreVariables="false">

          <script><![CDATA[

               importPackage(java.sql);

               importPackage(java.lang);

               importPackage(org.activiti.explorer.reporting);

               var result = ReportingUtil.executeSelectSqlQuery("SELECT PD.NAME\_, PD.VERSION\_ , count(\*) FROM ACT\_HI\_PROCINST PI inner join ACT\_RE\_PROCDEF PD on PI.PROC\_DEF\_ID\_ = PD.ID\_ group by PROC\_DEF\_ID\_");

               var reportData = {};

               reportData.datasets = [];

               var dataset = {};

               dataset.type = "pieChart";

               dataset.description = "Process instance overview (" + new java.util.Date() + ")";

               dataset.data = {};

               while (result.next()) { // process results one row at a time

                 var name = result.getString(1);

                 var version = result.getLong(2)

                 var count = result.getLong(3);

                 dataset.data[name + " (v" + version + ")"] = count;

               }

               reportData.datasets.push(dataset);

               execution.setVariable("reportData", new java.lang.String(JSON.stringify(reportData)).getBytes("UTF-8"));

          ]]></script>

        </scriptTask>

        <sequenceFlow id="flow3" sourceRef="generateDataset" targetRef="theEnd" />

        <endEvent id="theEnd" />

    </process>

</definitions>

除了流程xml顶部的标准xml，主要区别是*targetNamespace*设置为 **activiti-report**， 把分类设置为与部署的流程定义一样的名称。

脚本的第一行只是进行一些导入，避免每次使用时，都要写包名。 第一个有意义的代码是使用*ReportingUtil*读取activiti数据库。 返回结果是一个*JDBC 结果集*。 查询语句下面，javascript创建了使用的json。 json是符合[上面描述的需求](http://www.mossle.com/docs/activiti/#explorer.reporting.json)的。

最后一行脚本有一点儿奇怪。首先我们需要吧json对象转换成字符串， 使用javascript函数*JSON.stringify()*。 字符串需要保存为二进制数组类型的变量。这是一个技术问题： 二进制数组的大小是无限的，但是字符串的长度有限制。这就是为什么javascript字符串 必须转换成一个java字符串，以获得转换成二进制的功能。

兼容JDK 6（以及更高版本）的同一个流程有一些区别。 原生json功能无法使用，因此提供了一些帮助类（*ReportData* 和 *Dataset*）：

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

<definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:activiti="http://activiti.org/bpmn"

    xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI" xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC"

    xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI" typeLanguage="http://www.w3.org/2001/XMLSchema"

    expressionLanguage="http://www.w3.org/1999/XPath"

    targetNamespace="activiti-report">

    <process id="process-instance-overview-report" name="Process Instance Overview" isExecutable="true">

        <startEvent id="startevent1" name="Start" />

        <sequenceFlow id="flow1" sourceRef="startevent1" targetRef="generateDataset" />

        <scriptTask id="generateDataset" name="Execute script" scriptFormat="js" activiti:autoStoreVariables="false">

          <script><![CDATA[

               importPackage(java.sql);

               importPackage(java.lang);

               importPackage(org.activiti.explorer.reporting);

               var result = ReportingUtil.executeSelectSqlQuery("SELECT PD.NAME\_, PD.VERSION\_ , count(\*) FROM ACT\_HI\_PROCINST PI inner join ACT\_RE\_PROCDEF PD on PI.PROC\_DEF\_ID\_ = PD.ID\_ group by PROC\_DEF\_ID\_");

**var reportData = new ReportData;**

**var dataset = reportData.newDataset();**

**dataset.type = "pieChart";**

**dataset.description = "Process instance overview (" + new java.util.Date() + ")"**

               while (result.next()) { // process results one row at a time

                 var name = result.getString(1);

                 var version = result.getLong(2);

                 var count = result.getLong(3);

**dataset.add(name + " (v" + version + ")", count);**

               }

               execution.setVariable("reportData", reportData.toBytes());

          ]]></script>

        </scriptTask>

        <sequenceFlow id="flow3" sourceRef="generateDataset" targetRef="theEnd" />

        <endEvent id="theEnd" />

    </process>

</definitions>

**报告开始表单**

因为报表是使用普通流程来生成的，所以表单功能也可以使用。 直接在开始事件里加一个开始表单，Explorer就会在生成报表之前 把它展示给用户。

<startEvent id="startevent1" name="Start">

  <extensionElements>

    <activiti:formProperty id="processDefinition" name="Select process definition" type="processDefinition" required="true" />

      <activiti:formProperty id="chartType" name="Chart type" type="enum" required="true">

        <activiti:value id="pieChart" name="Pie chart" />

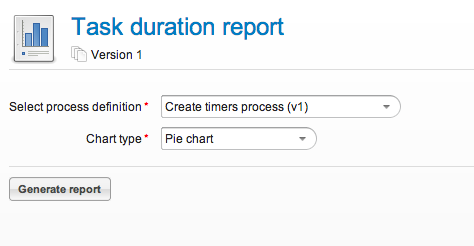
        <activiti:value id="barChart" name="Bar chart" />

      </activiti:formProperty>

  </extensionElements>

</startEvent>

这会为用户渲染一个普通的表单：



表单属性会在启动流程时提交，然后它们就可以像普通的流程变量一样使用， 脚本中可以使用它们来生成数据：

 var processDefinition = execution.getVariable("processDefinition");

**流程例子**

对于默认的，控制台包含4个报表例子：

* **Employee productivity(员工的工作效率)**： 报表演示使用折线图和开始表单。 报表的脚本也比其他例子要复杂， 因为数据会在脚本中先进行解释，再保存到报表数据中。
* **Helpdesk（一线与升级）**：使用饼图进行展示， 结合两个不同的数据库查询结果。
* **Process instance overview（流程实例总览）**：使用多个数据集的报表实例。 报表包含使用相同数据的饼图和列表视图，展示多种数据集可以用来在一个页面中生成不同图表。
* **Task duration（任务持续时间）**：另一个使用开始表单的例子， 会使用对应的变量来动态生成SQL查询语句。

**修改数据库**

如果修改控制台例子所用的数据库，改变属性文件apps/apache-tomcat-6.x/webapps/activiti-explorer/WEB-INF/classes/db.properties。 同样，在类路径下放上合适的数据库驱动（Tomcat 共享类库或者在 apps/apache-tomcat-6.x/webapps/activiti-explorer/WEB-INF/lib/中）

**Chapter 14. Activiti Modeler**

**Table of Contents**

[编辑模型](http://www.mossle.com/docs/activiti/#N12AEF)

[导入模型](http://www.mossle.com/docs/activiti/#N12AFA)

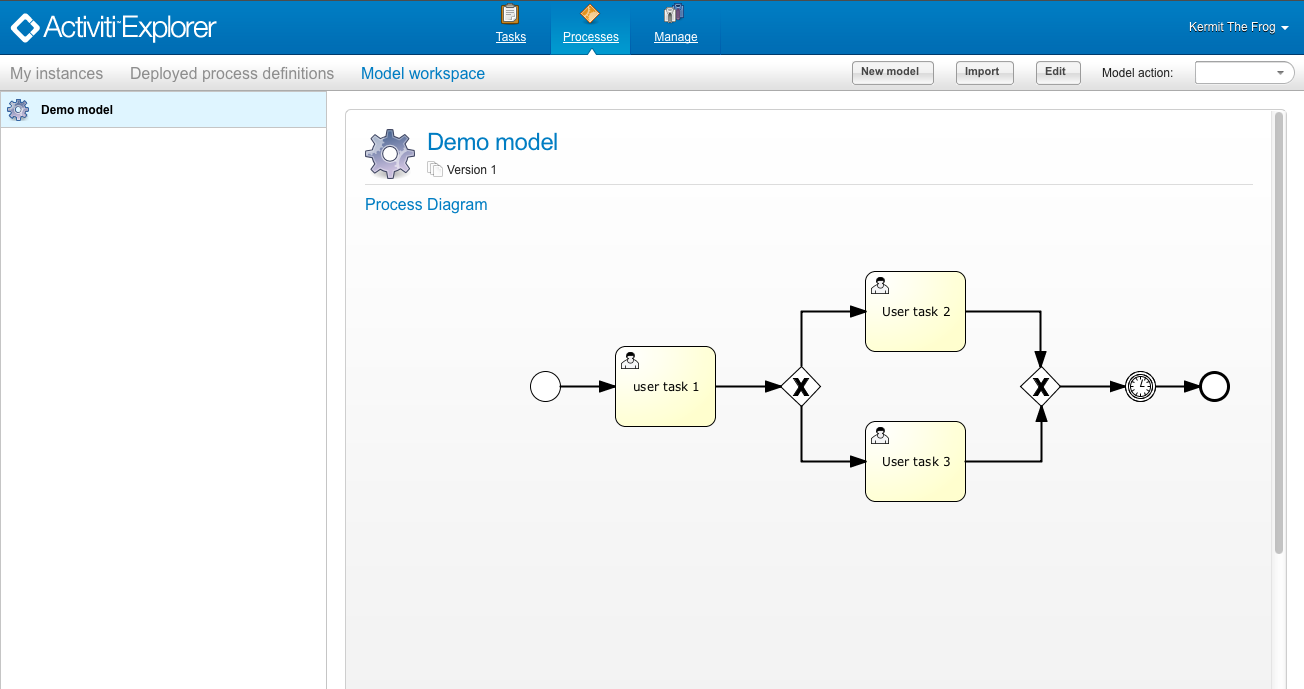
[把发布的流程定义转换成可编辑的模型](http://www.mossle.com/docs/activiti/#N12B05)

[把模型导出成BPMN XML](http://www.mossle.com/docs/activiti/#N12B10)

[把模型部署到Activiti引擎中](http://www.mossle.com/docs/activiti/#N12B1B)

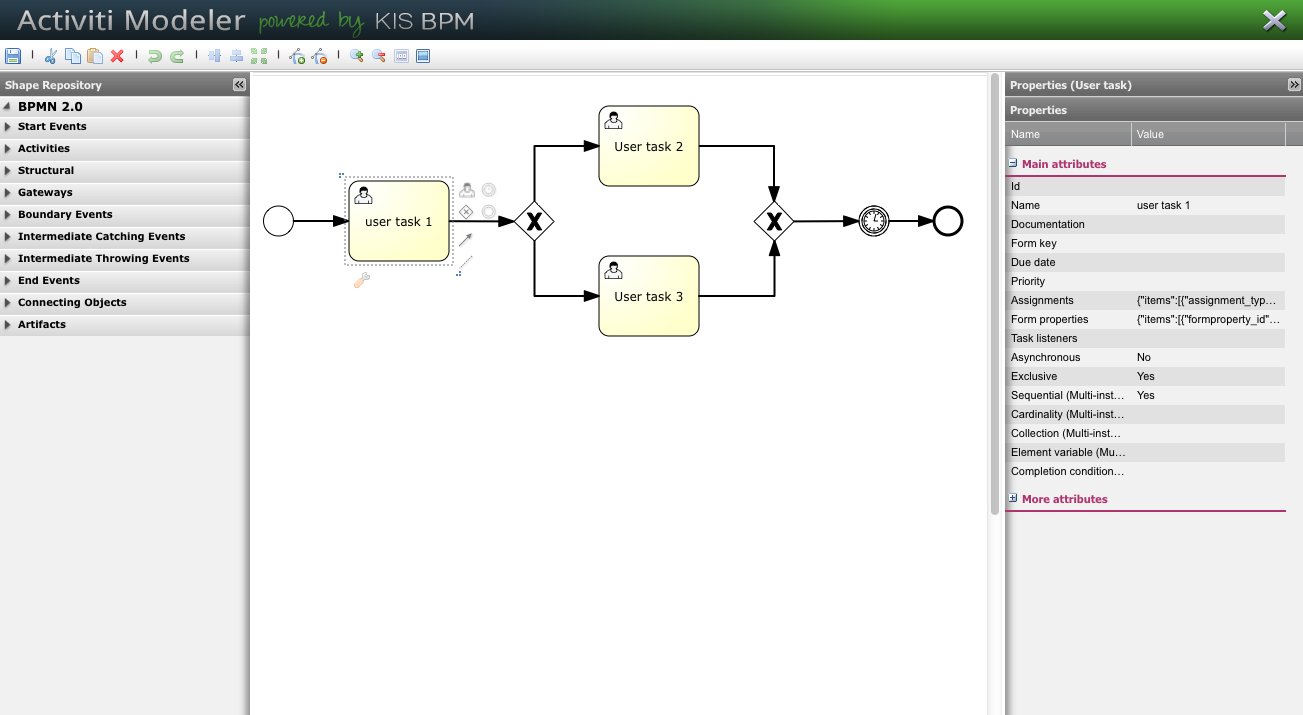
Activiti Modeler是一个BPMN web建模组件，它是[Activiti Explorer](http://www.mossle.com/docs/activiti/#activitiExplorer) web应用的一部分。 Modeler是[Signavio核心组件](http://code.google.com/p/signavio-core-components/)项目的一个分支。 新版Activiti Modeler的初始开发是由[KIS BPM](http://kisbpm.com)捐献给Activiti项目的。 和之前Activiti Modeler（Signavio核心组件）主要的区别是新Modeler是作为Activiti项目的一部分来维护和开发的。 Activiti Modeler的目标是支持所有BPMN元素和Activiti引擎支持的扩展。

当你运行Activiti Explorer使用默认配置时，模型工作台中会有一个示例流程。



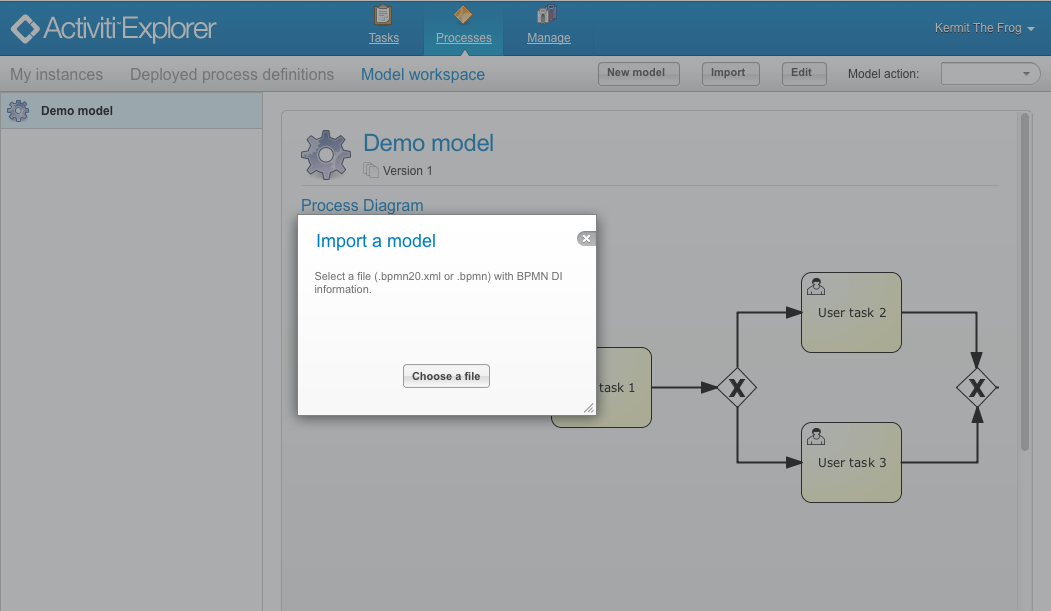
**编辑模型**

点击模型工作区的编辑按钮，会打开Modeler。 屏幕左侧是BPMN元素工具面板，也可以使用Activiti的扩展组件。 你可以在需要时把新元素拖拽到画布中。屏幕右侧是选中额元素的和苏醒。 例子截屏中选中了一个用户任务，你可以填写用户任务的属性，比如分配，表单属性和持续时间。 点击屏幕右上方的关闭按钮就可以返回Activiti Exporer。



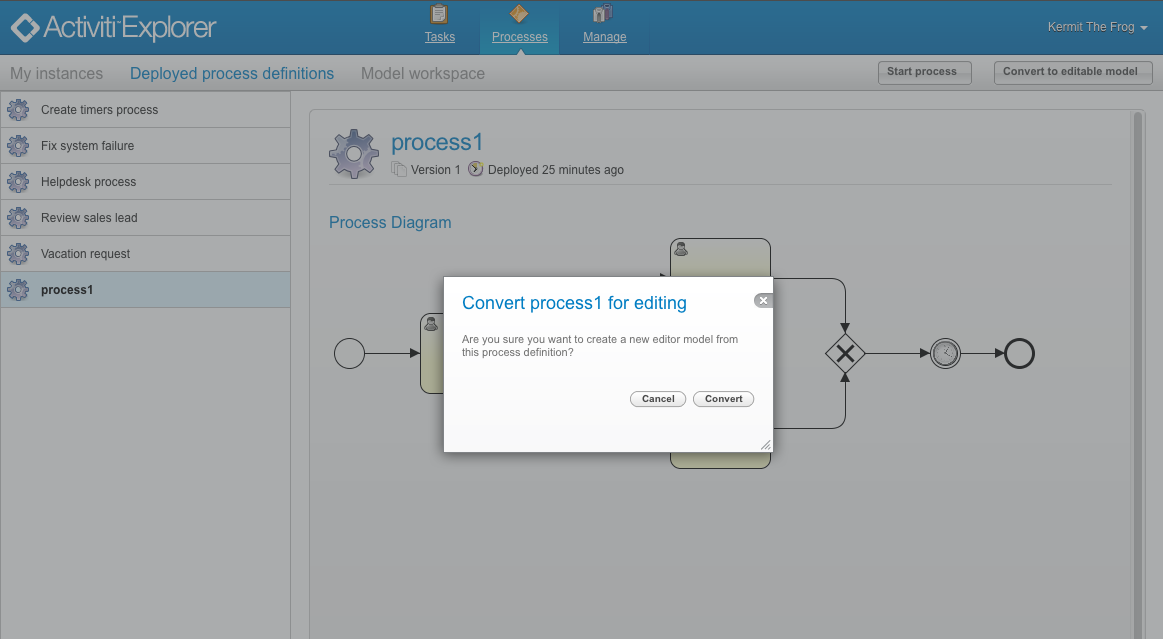
**导入模型**

你也可以把模型导入到模型工作台中，然后就可以在Activiti Modeler中进行编辑。 点击导入按钮，选择.bpmn或.bpmn20.xml文件。注意BPMN XML文件必须包含BPMN DI（坐标）信息。



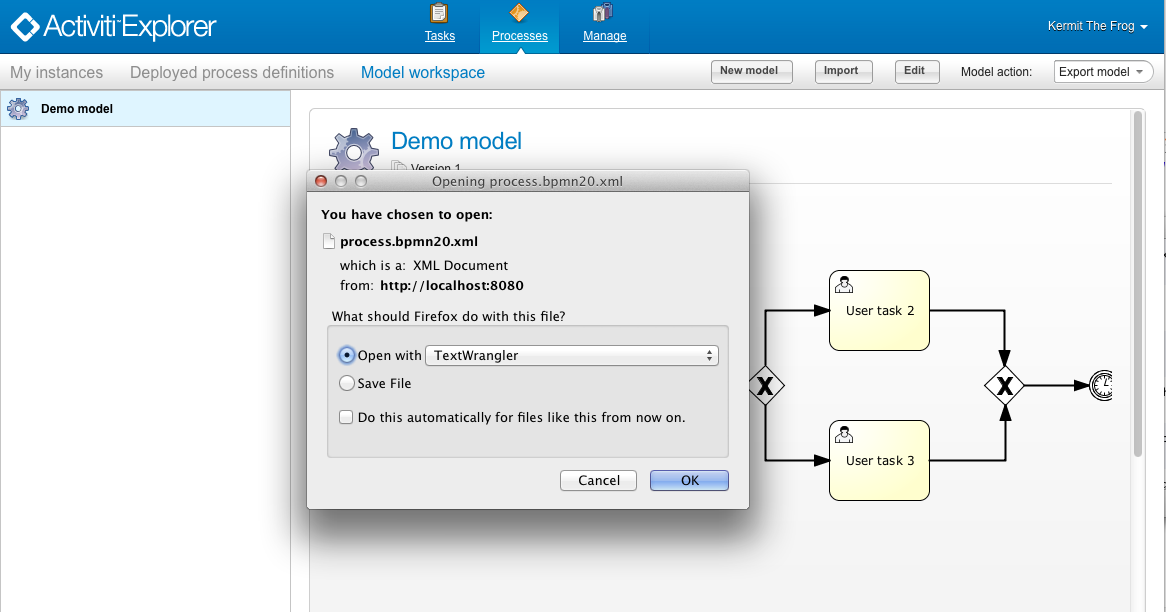
**把发布的流程定义转换成可编辑的模型**

发布的流程定义可以转换成模型，然后就可以在Activiti Modeler中编辑了。 注意流程定义必须包含BPMN DI（坐标）信息。



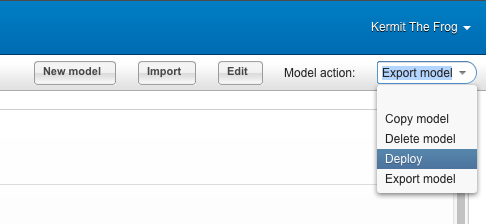
**把模型导出成BPMN XML**

模型工作区中的模型可以导出成BPMN XML文件。选择模型操作选项中的导出选项。



**把模型部署到Activiti引擎中**

在模型设置好所有运行所需的属性之后，它就可以发布到Activiti引擎里。 选择模型操作选项中的发布选项。



**Chapter 15. REST API**

**Table of Contents**

[General Activiti REST principles](http://www.mossle.com/docs/activiti/#N12B29)

[Installation and Authentication](http://www.mossle.com/docs/activiti/#N12B2C)

[Usage in Tomcat](http://www.mossle.com/docs/activiti/#restUsageInTomcat)

[Methods and return-codes](http://www.mossle.com/docs/activiti/#N12B78)

[Error response body](http://www.mossle.com/docs/activiti/#N12BFB)

[Request parameters](http://www.mossle.com/docs/activiti/#N12C02)

[Deployment](http://www.mossle.com/docs/activiti/#N12E29)

[List of Deployments](http://www.mossle.com/docs/activiti/#N12E34)

[Get a deployment](http://www.mossle.com/docs/activiti/#N12EAB)

[Create a new deployment](http://www.mossle.com/docs/activiti/#N12EF1)

[Delete a deployment](http://www.mossle.com/docs/activiti/#N12F2C)

[List resources in a deployment](http://www.mossle.com/docs/activiti/#N12F69)

[Get a deployment resource](http://www.mossle.com/docs/activiti/#N12FD3)

[Get a deployment resource content](http://www.mossle.com/docs/activiti/#N13044)

[Process Definitions](http://www.mossle.com/docs/activiti/#N13095)

[List of process definitions](http://www.mossle.com/docs/activiti/#N13098)

[Get a process definition](http://www.mossle.com/docs/activiti/#N1317D)

[Update category for a process definition](http://www.mossle.com/docs/activiti/#N131D4)

[Get a process definition resource content](http://www.mossle.com/docs/activiti/#N1320C)

[Get a process definition BPMN model](http://www.mossle.com/docs/activiti/#N1323B)

[Suspend a process definition](http://www.mossle.com/docs/activiti/#N13286)

[Activate a process definition](http://www.mossle.com/docs/activiti/#N132E9)

[Get all candidate starters for a process-definition](http://www.mossle.com/docs/activiti/#N13325)

[Add a candidate starter to a process definition](http://www.mossle.com/docs/activiti/#N1336B)

[Delete a candidate starter from a process definition](http://www.mossle.com/docs/activiti/#N133C3)

[Get a candidate starter from a process definition](http://www.mossle.com/docs/activiti/#N13421)

[Models](http://www.mossle.com/docs/activiti/#N1347F)

[Get a list of models](http://www.mossle.com/docs/activiti/#N13482)

[Get a model](http://www.mossle.com/docs/activiti/#N1354A)

[Update a model](http://www.mossle.com/docs/activiti/#N13590)

[Create a model](http://www.mossle.com/docs/activiti/#N135C6)

[Delete a model](http://www.mossle.com/docs/activiti/#N135F4)

[Get the editor source for a model](http://www.mossle.com/docs/activiti/#N13631)

[Set the editor source for a model](http://www.mossle.com/docs/activiti/#N13677)

[Get the extra editor source for a model](http://www.mossle.com/docs/activiti/#N136C6)

[Set the extra editor source for a model](http://www.mossle.com/docs/activiti/#N1370C)

[Process Instances](http://www.mossle.com/docs/activiti/#N1375B)

[Get a process instance](http://www.mossle.com/docs/activiti/#N1375E)

[Delete a process instance](http://www.mossle.com/docs/activiti/#N137A4)

[Activate or suspend a process instance](http://www.mossle.com/docs/activiti/#N137E1)

[Start a process instance](http://www.mossle.com/docs/activiti/#N1383A)

[List of process instances](http://www.mossle.com/docs/activiti/#restProcessInstancesGet)

[Query process instances](http://www.mossle.com/docs/activiti/#N13951)

[Get diagram for a process instance](http://www.mossle.com/docs/activiti/#N13992)

[Get involved people for process instance](http://www.mossle.com/docs/activiti/#N139DD)

[Add an involved user to a process instance](http://www.mossle.com/docs/activiti/#N13A26)

[Remove an involved user to from process instance](http://www.mossle.com/docs/activiti/#N13A83)

[List of variables for a process instance](http://www.mossle.com/docs/activiti/#N13ADE)

[Get a variable for a process instance](http://www.mossle.com/docs/activiti/#N13B2D)

[Create (or update) variables on a process instance](http://www.mossle.com/docs/activiti/#N13B8A)

[Update a single variable on a process instance](http://www.mossle.com/docs/activiti/#N13BF3)

[Create a new binary variable on a process-instance](http://www.mossle.com/docs/activiti/#N13C5B)

[Update an existing binary variable on a process-instance](http://www.mossle.com/docs/activiti/#N13CC6)

[Executions](http://www.mossle.com/docs/activiti/#N13D2C)

[Get an execution](http://www.mossle.com/docs/activiti/#N13D2F)

[Execute an action on an execution](http://www.mossle.com/docs/activiti/#N13D75)

[Get active activities in an execution](http://www.mossle.com/docs/activiti/#N13DEC)

[List of executions](http://www.mossle.com/docs/activiti/#restExecutionsGet)

[Query executions](http://www.mossle.com/docs/activiti/#N13ECD)

[List of variables for an execution](http://www.mossle.com/docs/activiti/#N13F14)

[Get a variable for an execution](http://www.mossle.com/docs/activiti/#N13F6C)

[Create (or update) variables on an execution](http://www.mossle.com/docs/activiti/#N13FD2)

[Update a variable on an execution](http://www.mossle.com/docs/activiti/#N1403F)

[Create a new binary variable on an execution](http://www.mossle.com/docs/activiti/#N1409E)

[Update an existing binary variable on a process-instance](http://www.mossle.com/docs/activiti/#N14111)

[Tasks](http://www.mossle.com/docs/activiti/#N14188)

[Get a task](http://www.mossle.com/docs/activiti/#N1418B)

[List of tasks](http://www.mossle.com/docs/activiti/#restTasksGet)

[Query for tasks](http://www.mossle.com/docs/activiti/#N14330)

[Update a task](http://www.mossle.com/docs/activiti/#N14373)

[Task actions](http://www.mossle.com/docs/activiti/#N143AE)

[Delete a task](http://www.mossle.com/docs/activiti/#N14413)

[Get all variables for a task](http://www.mossle.com/docs/activiti/#N1446A)

[Get a variable from a task](http://www.mossle.com/docs/activiti/#N144C3)

[Get the binary data for a variable](http://www.mossle.com/docs/activiti/#N14525)

[Create new variables on a task](http://www.mossle.com/docs/activiti/#N14595)

[Create a new binary variable on a task](http://www.mossle.com/docs/activiti/#N1460C)

[Update an existing variable on a task](http://www.mossle.com/docs/activiti/#N14682)

[Updating a binary variable on a task](http://www.mossle.com/docs/activiti/#N146FD)

[Delete a variable on a task](http://www.mossle.com/docs/activiti/#N14777)

[Delete all local variables on a task](http://www.mossle.com/docs/activiti/#N147CF)

[Get all identity links for a task](http://www.mossle.com/docs/activiti/#N1480C)

[Get all identitylinks for a task for either groups or users](http://www.mossle.com/docs/activiti/#N14852)

[Get a single identity link on a task](http://www.mossle.com/docs/activiti/#N1485C)

[Create an identity link on a task](http://www.mossle.com/docs/activiti/#N148C3)

[Delete an identity link on a task](http://www.mossle.com/docs/activiti/#N1491B)

[Create a new comment on a task](http://www.mossle.com/docs/activiti/#N14979)

[Get all comments on a task](http://www.mossle.com/docs/activiti/#N149CB)

[Get a comment on a task](http://www.mossle.com/docs/activiti/#N14A0F)

[Delete a comment on a task](http://www.mossle.com/docs/activiti/#N14A5C)

[Get all events for a task](http://www.mossle.com/docs/activiti/#N14A9F)

[Get an event on a task](http://www.mossle.com/docs/activiti/#N14AE3)

[Create a new attachment on a task, containing a link to an external resource](http://www.mossle.com/docs/activiti/#N14B30)

[Create a new attachment on a task, with an attached file](http://www.mossle.com/docs/activiti/#N14B82)

[Get all attachments on a task](http://www.mossle.com/docs/activiti/#N14BE5)

[Get an attachment on a task](http://www.mossle.com/docs/activiti/#N14C29)

[Get the content for an attachment](http://www.mossle.com/docs/activiti/#N14C8C)

[Delete an attachment on a task](http://www.mossle.com/docs/activiti/#N14CDE)

[History](http://www.mossle.com/docs/activiti/#N14D21)

[Get a historic process instance](http://www.mossle.com/docs/activiti/#N14D24)

[List of historic process instances](http://www.mossle.com/docs/activiti/#restHistoricProcessInstancesGet)

[Query for historic process instances](http://www.mossle.com/docs/activiti/#N14E14)

[Delete a historic process instance](http://www.mossle.com/docs/activiti/#N14E54)

[Get the identity links of a historic process instance](http://www.mossle.com/docs/activiti/#N14E75)

[Get the binary data for a historic process instance variable](http://www.mossle.com/docs/activiti/#N14E9F)

[Get a single historic task instance](http://www.mossle.com/docs/activiti/#N14ED2)

[Get historic task instances](http://www.mossle.com/docs/activiti/#restHistoricTaskInstancesGet)

[Query for historic task instances](http://www.mossle.com/docs/activiti/#N15049)

[Delete a historic task instance](http://www.mossle.com/docs/activiti/#N1508C)

[Get the identity links of a historic task instance](http://www.mossle.com/docs/activiti/#N150AD)

[Get the binary data for a historic task instance variable](http://www.mossle.com/docs/activiti/#N150D7)

[Get historic activity instances](http://www.mossle.com/docs/activiti/#restHistoricActivityInstancesGet)

[Query for historic activity instances](http://www.mossle.com/docs/activiti/#N151A3)

[List of historic variable instances](http://www.mossle.com/docs/activiti/#restHistoricVariableInstancesGet)

[Query for historic variable instances](http://www.mossle.com/docs/activiti/#N15251)

[Get the binary data for a historic task instance variable](http://www.mossle.com/docs/activiti/#N15291)

[Get historic detail](http://www.mossle.com/docs/activiti/#restHistoricDetailGet)

[Query for historic details](http://www.mossle.com/docs/activiti/#N1534B)

[Get the binary data for a historic detail variable](http://www.mossle.com/docs/activiti/#N15384)

[Forms](http://www.mossle.com/docs/activiti/#N153B7)

[Get form data](http://www.mossle.com/docs/activiti/#N153BA)

[Submit task form data](http://www.mossle.com/docs/activiti/#N15409)

[Database tables](http://www.mossle.com/docs/activiti/#N15445)

[List of tables](http://www.mossle.com/docs/activiti/#N15448)

[Get a single table](http://www.mossle.com/docs/activiti/#N1546D)

[Get column info for a single table](http://www.mossle.com/docs/activiti/#N154B1)

[Get row data for a single table](http://www.mossle.com/docs/activiti/#N154F5)

[Engine](http://www.mossle.com/docs/activiti/#N15573)

[Get engine properties](http://www.mossle.com/docs/activiti/#N15576)

[Get engine info](http://www.mossle.com/docs/activiti/#N1559B)

[Jobs](http://www.mossle.com/docs/activiti/#N155C0)

[Get a single job](http://www.mossle.com/docs/activiti/#N155C3)

[Delete a job](http://www.mossle.com/docs/activiti/#N15607)

[Execute a single job](http://www.mossle.com/docs/activiti/#N15644)

[Get the exception stacktrace for a job](http://www.mossle.com/docs/activiti/#N1568C)

[Get a list of jobs](http://www.mossle.com/docs/activiti/#N156C8)

[Users](http://www.mossle.com/docs/activiti/#N1578F)

[Get a single user](http://www.mossle.com/docs/activiti/#N15792)

[Get a list of users](http://www.mossle.com/docs/activiti/#N157D6)

[Update a user](http://www.mossle.com/docs/activiti/#N1586F)

[Create a user](http://www.mossle.com/docs/activiti/#N158AA)

[Delete a user](http://www.mossle.com/docs/activiti/#N158DD)

[Get a user's picture](http://www.mossle.com/docs/activiti/#N1591A)

[Updating a user's picture](http://www.mossle.com/docs/activiti/#N1595B)

[List a user's info](http://www.mossle.com/docs/activiti/#N159AB)

[Get a user's info](http://www.mossle.com/docs/activiti/#N159EF)

[Update a user's info](http://www.mossle.com/docs/activiti/#N15A3C)

[Create a new user's info entry](http://www.mossle.com/docs/activiti/#N15A97)

[Delete a user's info](http://www.mossle.com/docs/activiti/#N15AF1)

[Groups](http://www.mossle.com/docs/activiti/#N15B37)

[Get a single group](http://www.mossle.com/docs/activiti/#N15B3A)

[Get a list of groups](http://www.mossle.com/docs/activiti/#N15B7E)

[Update a group](http://www.mossle.com/docs/activiti/#N15BF9)

[Create a group](http://www.mossle.com/docs/activiti/#N15C31)

[Delete a group](http://www.mossle.com/docs/activiti/#N15C64)

[Get members in a group](http://www.mossle.com/docs/activiti/#N15CA1)

[Add a member to a group](http://www.mossle.com/docs/activiti/#N15CAC)

[Delete a member from a group](http://www.mossle.com/docs/activiti/#N15D05)

[Legacy REST - General Usage](http://www.mossle.com/docs/activiti/#N15D55)

[Repository](http://www.mossle.com/docs/activiti/#N15D6D)

[Upload Deployment](http://www.mossle.com/docs/activiti/#N15D71)

[Get Deployments](http://www.mossle.com/docs/activiti/#N15D95)

[Get Deployment Resources](http://www.mossle.com/docs/activiti/#N15DB9)

[Get Deployment Resource](http://www.mossle.com/docs/activiti/#N15DDD)

[Delete Deployment](http://www.mossle.com/docs/activiti/#N15E01)

[Delete Deployments](http://www.mossle.com/docs/activiti/#N15E25)

[Engine](http://www.mossle.com/docs/activiti/#N15E4D)

[Get Process Engine](http://www.mossle.com/docs/activiti/#N15E50)

[Processes](http://www.mossle.com/docs/activiti/#N15E75)

[List Process Definitions](http://www.mossle.com/docs/activiti/#N15E78)

[Get Process Definition Form Properties](http://www.mossle.com/docs/activiti/#N15E9C)

[Get Process Definition Form Resource](http://www.mossle.com/docs/activiti/#N15EC0)

[Get Process Definition Diagram](http://www.mossle.com/docs/activiti/#N15EE4)

[Start Process Instance](http://www.mossle.com/docs/activiti/#N15F08)

[List Process Instances](http://www.mossle.com/docs/activiti/#N15F2F)

[Get Process Instance Details](http://www.mossle.com/docs/activiti/#N15F53)

[Get Process Instance Diagram](http://www.mossle.com/docs/activiti/#N15F77)

[Get open tasks for a process instance](http://www.mossle.com/docs/activiti/#N15F9B)

[Signal an activity (receive task) for a specific process instance](http://www.mossle.com/docs/activiti/#N15FBF)

[Trigger a signal for a specific process instance](http://www.mossle.com/docs/activiti/#N15FE6)

[Tasks](http://www.mossle.com/docs/activiti/#N1600E)

[Get Task Summary](http://www.mossle.com/docs/activiti/#N16011)

[List Tasks](http://www.mossle.com/docs/activiti/#N16035)

[Get Task](http://www.mossle.com/docs/activiti/#N16063)

[Get Task Form](http://www.mossle.com/docs/activiti/#N16087)

[Perform Task Operation](http://www.mossle.com/docs/activiti/#N160AB)

[List Form Properties](http://www.mossle.com/docs/activiti/#N160D3)

[Add attachment to a task](http://www.mossle.com/docs/activiti/#N160F7)

[Get task attachment](http://www.mossle.com/docs/activiti/#N1611C)

[Add url to a task](http://www.mossle.com/docs/activiti/#N16140)

[Identity](http://www.mossle.com/docs/activiti/#N16166)

[Login](http://www.mossle.com/docs/activiti/#N16169)

[Get User](http://www.mossle.com/docs/activiti/#N16190)

[List User's Groups](http://www.mossle.com/docs/activiti/#N161B4)

[Search users](http://www.mossle.com/docs/activiti/#N161D6)

[Create user](http://www.mossle.com/docs/activiti/#N161FB)

[Add user to groups](http://www.mossle.com/docs/activiti/#N16225)

[Remove user from group](http://www.mossle.com/docs/activiti/#N1624C)

[Get user picture](http://www.mossle.com/docs/activiti/#N16270)

[Get Group](http://www.mossle.com/docs/activiti/#N16294)

[List Group Users](http://www.mossle.com/docs/activiti/#N162B8)

[Search groups](http://www.mossle.com/docs/activiti/#N162DA)

[Create group](http://www.mossle.com/docs/activiti/#N162FC)

[Add users to a group](http://www.mossle.com/docs/activiti/#N16326)

[Remove user from group](http://www.mossle.com/docs/activiti/#N1634D)

[Management](http://www.mossle.com/docs/activiti/#N16372)

[List Jobs](http://www.mossle.com/docs/activiti/#N16375)

[Get Job](http://www.mossle.com/docs/activiti/#N16399)

[Execute Job](http://www.mossle.com/docs/activiti/#N163BD)

[Execute Jobs](http://www.mossle.com/docs/activiti/#N163E1)

[List Database Tables](http://www.mossle.com/docs/activiti/#N16408)

[Get Table Meta Data](http://www.mossle.com/docs/activiti/#N1642C)

[Get Table Data](http://www.mossle.com/docs/activiti/#N1644E)

**General Activiti REST principles**

**Installation and Authentication**

Activiti includes a REST API to the Activiti Engine that can be installed by deploying the activiti-rest.war file to a servlet container like Apache Tomcat. However, it can also be used in another web-application by including the servlet and it's mapping in your application and add all activiti-rest dependencies to the classpath.

By default the Activiti Engine will connect to an in-memory H2 database. You can change the database settings in the db.properties file in the WEB-INF/classes folder. The REST API uses JSON format (http://www.json.org) and is built upon the Restlet (http://www.restlet.org).

All REST-resources require a valid Activiti-user to be authenticated by default. Basic HTTP access authentication is used, so you should always include a Authorization: Basic ...== HTTP-header when performing requests or include the username and password in the request-url (eg. http://username:password@localhost...).

**It's recommended to use Basic Authentication in combination with HTTPS.**

It's possible to remove authentication from certain resources or adding additional authorisation on top of being an authenticated user (eg. being part of group X allows a user to execute request with URL Y). This can be achieved by using an implementation of org.activiti.rest.filter.RestAuthenticator which has 2 methods:

* *boolean requestRequiresAuthentication(Request request)*: Called before a request is checked for authentication (valid username and password is passed in header). If this method returns true, the method needs authentication. If false is returned, the request will be done regardless of the request being authenticated. If false is returned, isRequestAuthorized will not be called either for this request.
* *boolean isRequestAuthorized(Request request)*: Called after a user is successfully authenticated against the Activiti identity-management but before the request is actually executed. Can be used to check if a certain request is valid for the authenticated user. If true is returned, the request is considered authorised and is executed. When false is returned, the request is not executed and the client gets an appropriate error.

The custom RestAuthenticator should be set on the org.activiti.rest.application.ActivitiRestServicesApplicationthat is used in the RestletServlet. The easiest way for this to create a subclass of the ActivitiRestServicesApplication and use the custom implementation classname in the servlet-mapping:

   <!-- Restlet adapter -->

  <servlet>

    <servlet-name>RestletServlet</servlet-name>

    <servlet-class>org.restlet.ext.servlet.ServerServlet</servlet-class>

    <init-param>

      <!-- Application class name -->

      <param-name>org.restlet.application</param-name>

      <param-value>com.my.company.CustomActivitiRestServicesApplication</param-value>

    </init-param>

  </servlet>

**Usage in Tomcat**

Due to [default security properties on Tomcat](http://tomcat.apache.org/tomcat-7.0-doc/security-howto.html), **escaped forward slashes (%2F and %5C) are not allowed by default (400-result is returned).** This may have an impact on the deployment resources and their data-URL, as the URL can potentially contain escaped forward slashes. **When issues are experienced with unexpected 400-results, set the following system-property: -Dorg.apache.tomcat.util.buf.UDecoder.ALLOW\_ENCODED\_SLASH=true.**

**Methods and return-codes**

**Table 15.1. HTTP-methods and corresponding operations**

| Method | Operations |
| --- | --- |
| GET | Get a single resource or get a collection of resources. |
| POST | Create a new resource. Also used for executing resource-queries which have a too complex request-structure to fit in the query-URL of a GET-request. |
| PUT | Update properties of an existing resource. Also used for invoking actions on an existing resource. |
| DELETE | Delete an existing resource. |

**Table 15.2. HTTP-methods response codes**

| Response | Description |
| --- | --- |
| 200 - Ok | The operation was successful and a response has been returned (GET and PUT requests). |
| 201 - Created | The operation was successful and the entity has been created and is returned in the response-body (POST request). |
| 204 - No content | The operation was successful and entity has been deleted and therefore there is no response-body returned (DELETE request). |
| 401 - Unauthorized | The operation failed. The operation requires an Authentication header to be set. If this was present in the request, the supplied credentials are not valid or the user is not authorized to perform this operation. |
| 403 - Forbidden | The operation is forbidden and should not be re-attempted. This does not imply an issue with authentication not authorization, it's an operation that is not allowed. Example: deleting a task that is part of a running process is not allowed and will never be allowed, regardless of the user or process/task state. |
| 404 - Not found | The operation failed.The requested resource was not found. |
| 405 - Method not allowed | The operation failed. The used method is not allowed for this resource. Eg. trying to update (PUT) a deployment-resource will result in a 405 status. |
| 409 - Conflict | The operation failed. The operation causes an update of a resource that has been updated by another operation, which makes the update no longer valid. Can also indicate a resource that is being created in a collection where a resource with that identifier already exists. |
| 415 - Unsupported Media Type | The operation failed. The request body contains an unsupported media type. Also occurs when the request-body JSON contains an unknown attribute or value that doesn't have the right format/type to be accepted. |
| 500 - Internal server error | The operation failed. An unexpected exception occured while executing the operation. The response-body contains details about the error. |

The media-type of the HTTP-responses is always application/json unless binary content is requested (eg. deployment resource data), the media-type of the content is used.

**Error response body**

When an error occurs (both client and server, 4XX and 5XX status-codes) the response body contains an object describing the error that occurred. An example for a 404-status when a task is not found:

{

  "statusCode" : 404,

  "errorMessage" : "Could not find a task with id '444'."

}

**Request parameters**

**URL fragments**

Parameters that are part of the url (eg. the deploymentId parameter in http://host/actviti-rest/service/repository/deployments/{deploymentId}) need to be properly escaped (see [URL-encoding or Percent-encoding](https://en.wikipedia.org/wiki/Percent-encoding)) in case the segment contains special characters. Most frameworks have this functionality built in, but it should be taken into account. Especially for segments that can contains forward-slashes (eg. deployment resource), this is required.

**Rest URL query parameters**

Parameters added as query-string in the URL (eg. the name parameter used in http://host/activiti-rest/service/deployments?name=Deployment) can have the folowing types and are mentioned in the corresponding REST-API documentation:

**Table 15.3. URL query parameter types**

| Type | Format |
| --- | --- |
| String | Plain text parameters. Can contain any valid characters that are allowed in URL's. In case of a XXXLike parameter, the string should contain the wildcard characted % (properly url-encoded). This allows to specify the intent of the like-search. Eg. 'Tas%' matches all values, starting with 'Tas'. |
| Integer | Parameter representing an integer value. Can only contain numeric non-decmimal values, between -2.147.483.648 and 2.147.483.647. |
| Long | Parameter representing a long value. Can only contain numeric non-decmimal values, between -9.223.372.036.854.775.808 and 9.223.372.036.854.775.807. |
| Boolean | Parameter representing a boolean value. Can be eiter true or false. All other values other than these two, will cause a '405 - Bad request' response. |
| Date | Parameter representing a date value. Use the ISO-8601 date-format (see [ISO-8601 on wikipedia](http://en.wikipedia.org/wiki/ISO_8601)) using both time and date-components (eg. 2013-04-03T23:45Z). |

**JSON body parameters**

**Table 15.4. JSON parameter types**

| Type | Format |
| --- | --- |
| String | Plain text parameters. In case of a XXXLike parameter, the string should contain the wildcard characted %. This allows to specify the intent of the like-search. Eg. 'Tas%' matches all values, starting with 'Tas'. |
| Integer | Parameter representing an integer value, using a JSON number. Can only contain numeric non-decmimal values, between -2.147.483.648 and 2.147.483.647. |
| Long | Parameter representing a long value, using a JSON number. Can only contain numeric non-decmimal values, between -9.223.372.036.854.775.808 and 9.223.372.036.854.775.807. |
| Boolean | Parameter representing a boolean value, using a JSON boolean. Can be eiter true or false. All other values other than these two, will cause a '405 - Bad request' response. |
| Date | Parameter representing a date value, using a JSON text. Use the ISO-8601 date-format (see [ISO-8601 on wikipedia](http://en.wikipedia.org/wiki/ISO_8601)) using both time and date-components (eg. 2013-04-03T23:45Z). |

**Paging and sorting**

Paging and order parameters can be added as query-string in the URL (eg. the name parameter used in http://host/activiti-rest/service/deployments?sort=name).

**Table 15.5. Variable query JSON parameters**

| Parameter | Default value | Description |
| --- | --- | --- |
| sort | different per query implementation | Name of the sort key, for which the default value and the allowed values are different per query implementation. |
| order | asc | Sorting order which can be 'asc' or 'desc'. |
| start | 0 | Parameter to allow for paging of the result. By default the result will start at 0. |
| size | 10 | Parameter to allow for paging of the result. By default the size will be 10. |

**JSON query variable format**

{

  "name" : "variableName",

  "value" : "variableValue",

  "operator" : "equals",

  "type" : "string"

}

**Table 15.6. Variable query JSON parameters**

| Parameter | Required | Description |
| --- | --- | --- |
| name | No | Name of the variable to include in a query. Can be empty in case 'equals' is used in some queries to query for resources that have **any variable name** with the given value. |
| value | Yes | Value of the variable included in the query, should include a correct format for the given type. |
| operator | Yes | Operator to use in query, can have the folowing values: equals, notEquals, equalsIgnoreCase, notEqualsIgnoreCase, lessThan, greaterThan, lessThanOrEquals, greaterThanOrEquals and like. |
| type | No | Type of variable to use. When omitted, the type will be deducted from the value parameter. Any JSON text-values will be considered of type string, JSON booleans of type boolean, JSON numbers of type long or integer depending on the size of the number. It's recommended to include an explicit type when in doubt. Types supported out of the box are listed below. |

**Table 15.7. Default query JSON types**

| Type name | Description |
| --- | --- |
| string | Value is threaded as and converted to a java.lang.String. |
| short | Value is threaded as and converted to a java.lang.Integer. |
| integer | Value is threaded as and converted to a java.lang.Integer. |
| long | Value is threaded as and converted to a java.lang.Long. |
| double | Value is threaded as and converted to a java.lang.Double. |
| boolean | Value is threaded as and converted to a java.lang.Boolean. |
| date | Value is treated as and converted to a java.util.Date. The JSON string will be converted using ISO-8601 date format. |

**Variable representation**

When working with variables (execution/process and task), the REST-api uses some common principles and JSON-format for both reading and writing. The JSON representation of a variable looks like this:

{

  "name" : "variableName",

  "value" : "variableValue",

  "valueUrl" : "http://...",

  "scope" : "local",

  "type" : "string"

}

**Table 15.8. Variable JSON attributes**

| Parameter | Required | Description |
| --- | --- | --- |
| name | Yes | Name of the variable. |
| value | No | Value of the variable. When writing a variable and value is omitted, null will be used as value. |
| valueUrl | No | When reading a variable of type binary or serializable, this attribute will point to the URL where the raw binary data can be fetched from. |
| scope | No | Scope of the variable. If 'local', the variable is explicitally defined on the resource it's requested from. When 'global', the variable is defined on the parent (or any parent in the parent-tree) of the resource it's requested from. When writing a variable and the scope is omitted, global is assumed. |
| type | No | Type of the variable. See table below for additional information on types. When writing a variable and this value is omitted, the type will be deducted from the raw JSON-attribute request type and is limited to either string, double, integer and boolean. It's advised to always include a type to make sure no wrong assumption about the type can be done. |

**Table 15.9. Variable Types**

| Type name | Description |
| --- | --- |
| string | Value is threaded as a java.lang.String. Raw JSON-text value is used when writing a variable. |
| integer | Value is threaded as a java.lang.Integer. When writing, JSON number value is used as base for conversion, falls back to JSON text. |
| short | Value is threaded as a java.lang.Short. When writing, JSON number value is used as base for conversion, falls back to JSON text. |
| long | Value is threaded as a java.lang.Long. When writing, JSON number value is used as base for conversion, falls back to JSON text. |
| double | Value is threaded as a java.lang.Double. When writing, JSON number value is used as base for conversion, falls back to JSON text. |
| boolean | Value is threaded as a java.lang.Boolean. When writing, JSON boolean value is used for conversion. |
| date | Value is treated as a java.util.Date. When writing, the JSON text will be converted using ISO-8601 date format. |
| binary | Binary variable, threated as an array of bytes. The value attribute is null, the valueUrl contains an URL pointing to the raw binary stream. |
| serializable | Serialized representation of a Serializable Java-object. As with the binary type, the value attribute is null, the valueUrl contains an URL pointing to the raw binary stream. All serializable variables (which are not of any of the above types) will be exposed as a variable of this type. |

It's possible to support additional variable-types with a custom JSON representation (either simple value or complex/nested JSON object). By extending the initializeVariableConverters() method on org.activiti.rest.api.RestResponseFactory, you can add additional org.activiti.rest.api.engine.variable.RestVariableConverter classes to support converting your POJO's to a format suitable for transerring through REST and converting the REST-value back to your POJO. The actual transformation to JSON is done by Jackson.

**Deployment**

**When using tomcat, please read** [**Usage in Tomcat**](http://www.mossle.com/docs/activiti/#restUsageInTomcat)**.**

**List of Deployments**

GET repository/deployments

**Table 15.10. URL query parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| name | No | String | Only return deployments with the given name. |
| nameLike | No | String | Only return deployments with a name like the given name. |
| category | No | String | Only return deployments with the given category. |
| categoryNotEquals | No | String | Only return deployments which don't have the given category. |
| sort | No | 'id' (default), 'name' or 'deploytime' | Property to sort on, to be used toghether with the 'order'. |
| The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL. | | | |

**Table 15.11. REST Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the request was successful. |

**Success response body:**

{

  "data": [

    {

      "id": "10",

      "name": "activiti-examples.bar",

      "deploymentTime": "2010-10-13T14:54:26.750+02:00",

      "category": "examples",

      "url": "http://localhost:8081/service/repository/deployments/10"

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "id",

  "order": "asc",

  "size": 1

}

**Get a deployment**

GET repository/deployments/{deploymentId}

**Table 15.12. Get a deployment - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| deploymentId | Yes | String | The id of the deployment to get. |

**Table 15.13. Get a deployment - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the deployment was found and returned. |
| 404 | Indicates the requested deployment was not found. |

**Success response body:**

{

  "id": "10",

  "name": "activiti-examples.bar",

  "deploymentTime": "2010-10-13T14:54:26.750+02:00",

  "category": "examples",

  "url": "http://localhost:8081/service/repository/deployments/10"

}

**Create a new deployment**

POST repository/deployments

**Request body:**

The request should body should contain data of type *multipart/form-data*. There should be only exactly file in the request, any additional files will be ignored. The deployment name is the name of the file-field passed in. If multiple resources need to be deployed in a single deployment, compress the resources in a zip and make sure the file-name ends with .bar or .zip.

**Table 15.14. Create a new deployment - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the deployment was created. |
| 400 | Indicates there was no content present in the request body or the content mime-type is not supported for deployment. The status-description contains additional information. |

**Success response body:**

{

  "id": "10",

  "name": "activiti-examples.bar",

  "deploymentTime": "2010-10-13T14:54:26.750+02:00",

  "category": null,

  "url": "http://localhost:8081/service/repository/deployments/10"

}

**Delete a deployment**

DELETE repository/deployments/{deploymentId}

**Table 15.15. Delete a deployment - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| deploymentId | Yes | String | The id of the deployment to delete. |

**Table 15.16. Delete a deployment - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the deployment was found and has been deleted. Response-body is intentionally empty. |
| 404 | Indicates the requested deployment was not found. |

**List resources in a deployment**

GET repository/deployments/{deploymentId}/resources

**Table 15.17. List resources in a deployment - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| deploymentId | Yes | String | The id of the deployment to get the resources for. |

**Table 15.18. List resources in a deployment - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the deployment was found and the resource list has been returned. |
| 404 | Indicates the requested deployment was not found. |

**Success response body:**

[

  {

    "id": "diagrams/my-process.bpmn20.xml",

    "url": "http://localhost:8081/activiti-rest/service/repository/deployments/10/resources/diagrams%2Fmy-process.bpmn20.xml",

    "dataUrl": "http://localhost:8081/activiti-rest/service/repository/deployments/10/resourcedata/diagrams%2Fmy-process.bpmn20.xml",

    "mediaType": "text/xml",

    "type": "processDefinition"

  },

  {

    "id": "image.png",

    "url": "http://localhost:8081/activiti-rest/service/repository/deployments/10/resources/image.png",

    "dataUrl": "http://localhost:8081/activiti-rest/service/repository/deployments/10/resourcedata/image.png",

    "mediaType": "image/png",

    "type": "resource"

  }

]

* mediaType: Contains the media-type the resource has. This is resolved using a (pluggable) MediaTypeResolver and contains, by default, a limited number of mime-type mappings.
* type: Type of resource, possible values:
  + resource: Plain old resource.
  + processDefinition: Resource that contains one or more process-definitions. This resource is picked up by the deployer.
  + processImage: Resource that represents a deployed process definition's graphical layout.

*The dataUrl property in the resulting json for a single resource contains the actual URL to use for retreiving the binary resource.*

**Get a deployment resource**

GET repository/deployments/{deploymentId}/resources/{resourceId}

**Table 15.19. Get a deployment resource - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| deploymentId | Yes | String | The id of the deployment the requested resource is part of. |
| resourceId | Yes | String | The id of the resource to get. **Make sure you URL-encode the resourceId in case it contains forward slashes. Eg: use 'diagrams%2Fmy-process.bpmn20.xml' instead of 'diagrams/Fmy-process.bpmn20.xml'.** |

**Table 15.20. Get a deployment resource - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates both deployment and resource have been found and the resource has been returned. |
| 404 | Indicates the requested deployment was not found or there is no resource with the given id present in the deployment. The status-description contains additional information. |

**Success response body:**

{

  "id": "diagrams/my-process.bpmn20.xml",

  "url": "http://localhost:8081/activiti-rest/service/repository/deployments/10/resources/diagrams%2Fmy-process.bpmn20.xml",

  "dataUrl": "http://localhost:8081/activiti-rest/service/repository/deployments/10/resourcedata/diagrams%2Fmy-process.bpmn20.xml",

  "mediaType": "text/xml",

  "type": "processDefinition"

}

* mediaType: Contains the media-type the resource has. This is resolved using a (pluggable) MediaTypeResolver and contains, by default, a limited number of mime-type mappings.
* type: Type of resource, possible values:
  + resource: Plain old resource.
  + processDefinition: Resource that contains one or more process-definitions. This resource is picked up by the deployer.
  + processImage: Resource that represents a deployed process definition's graphical layout.

**Get a deployment resource content**

GET repository/deployments/{deploymentId}/resourcedata/{resourceId}

**Table 15.21. Get a deployment resource content - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| deploymentId | Yes | String | The id of the deployment the requested resource is part of. |
| resourceId | Yes | String | The id of the resource to get the data for. **Make sure you URL-encode the resourceId in case it contains forward slashes. Eg: use 'diagrams%2Fmy-process.bpmn20.xml' instead of 'diagrams/Fmy-process.bpmn20.xml'.** |

**Table 15.22. Get a deployment resource content - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates both deployment and resource have been found and the resource data has been returned. |
| 404 | Indicates the requested deployment was not found or there is no resource with the given id present in the deployment. The status-description contains additional information. |

**Success response body:**

The response body will contain the binary resource-content for the requested resource. The response content-type will be the same as the type returned in the resources 'mimeType' property. Also, a content-disposition header is set, allowing browsers to dowload the file instead of displaying it.

**Process Definitions**

**List of process definitions**

GET repository/process-definitions

**Table 15.23. List of process definitions - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| version | No | integer | Only return process definitions with the given version. |
| name | No | String | Only return process definitions with the given name. |
| nameLike | No | String | Only return process definitions with a name like the given name. |
| key | No | String | Only return process definitions with the given key. |
| keyLike | No | String | Only return process definitions with a name like the given key. |
| resourceName | No | String | Only return process definitions with the given resource name. |
| resourceNameLike | No | String | Only return process definitions with a name like the given resource name. |
| category | No | String | Only return process definitions with the given category. |
| categoryLike | No | String | Only return process definitions with a category like the given name. |
| categoryNotEquals | No | String | Only return process definitions which don't have the given category. |
| deploymentId | No | String | Only return process definitions which are part of a deployment with the given id. |
| startableByUser | No | String | Only return process definitions which can be started by the given user. |
| latest | No | Boolean | Only return the latest process definition versions. Can only be used toghether with 'key' and 'keyLike' parameters, using any other parameter will result in a 400-response. |
| suspended | No | Boolean | If true, only returns process definitions which are suspended. If false, only active process definitions (which are not suspended) are returned. |
| sort | No | 'name' (default), 'id', 'key', 'category', 'deploymentId' and 'version' | Property to sort on, to be used toghether with the 'order'. |
| The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL. |  |  |  |

**Table 15.24. List of process definitions - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the process-definitions are returned |
| 400 | Indicates a parameter was passed in the wrong format or that 'latest' is used with other parameters other than 'key' and 'keyLike'. The status-message contains additional information. |

**Success response body:**

{

  "data": [

    {

      "id" : "oneTaskProcess:1:4",

      "url" : "http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4",

      "version" : 1,

      "key" : "oneTaskProcess",

      "category" : "Examples",

      "suspended" : false,

      "name" : "The One Task Process",

      "description" : "This is a process for testing purposes",

      "deploymentId" : "2",

      "deploymentUrl" : "http://localhost:8081/repository/deployments/2",

      "graphicalNotationDefined" : true,

      "resource" : "http://localhost:8182/repository/deployments/2/resources/testProcess.xml",

      "diagramResource" : "http://localhost:8182/repository/deployments/2/resources/testProcess.png",

      "startFormDefined" : false

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "name",

  "order": "asc",

  "size": 1

}

* graphicalNotationDefined: Indicates the process definition contains graphical information (BPMN DI).
* resource: Contains the actual deployed BPMN 2.0 xml.
* diagramResource: Contains a graphical representation of the process, null when no diagram is available.

**Get a process definition**

GET repository/process-definitions/{processDefinitionId}

**Table 15.25. Get a process definition - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processDefinitionId | Yes | String | The id of the process definition to get. |

**Table 15.26. Get a process definition - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the process definition was found and returned. |
| 404 | Indicates the requested process definition was not found. |

**Success response body:**

{

  "id" : "oneTaskProcess:1:4",

  "url" : "http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4",

  "version" : 1,

  "key" : "oneTaskProcess",

  "category" : "Examples",

  "suspended" : false,

  "name" : "The One Task Process",

  "description" : "This is a process for testing purposes",

  "deploymentId" : "2",

  "deploymentUrl" : "http://localhost:8081/repository/deployments/2",

  "graphicalNotationDefined" : true,

  "resource" : "http://localhost:8182/repository/deployments/2/resources/testProcess.xml",

  "diagramResource" : "http://localhost:8182/repository/deployments/2/resources/testProcess.png",

  "startFormDefined" : false

}

* graphicalNotationDefined: Indicates the process definition contains graphical information (BPMN DI).
* resource: Contains the actual deployed BPMN 2.0 xml.
* diagramResource: Contains a graphical representation of the process, null when no diagram is available.

**Update category for a process definition**

PUT repository/process-definitions/{processDefinitionId}

**Body JSON:**

{

  "category" : "updatedcategory"

}

**Table 15.27. Update category for a process definition - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the process was category was altered. |
| 400 | Indicates no category was defined in the request body. |
| 404 | Indicates the requested process definition was not found. |

**Success response body:** see response for repository/process-definitions/{processDefinitionId}.

**Get a process definition resource content**

GET repository/process-definitions/{processDefinitionId}/resourcedata

**Table 15.28. Get a process definition resource content - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processDefinitionId | Yes | String | The id of the process definition to get the resource data for. |

**Response:**

Exactly the same response codes/boy as GET repository/deployment/{deploymentId}/resourcedata/{resourceId}.

**Get a process definition BPMN model**

GET repository/process-definitions/{processDefinitionId}/model

**Table 15.29. Get a process definition BPMN model - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processDefinitionId | Yes | String | The id of the process definition to get the model for. |

**Table 15.30. Get a process definition BPMN model - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the process definition was found and the model is returned. |
| 404 | Indicates the requested process definition was not found. |

**Response body:** The response body is a JSON representation of the org.activiti.bpmn.model.BpmnModel and contains the full process definition model.

{

   "processes":[

      {

         "id":"oneTaskProcess",

         "xmlRowNumber":7,

         "xmlColumnNumber":60,

         "extensionElements":{

         },

         "name":"The One Task Process",

         "executable":true,

         "documentation":"One task process description",

         ...

    ],

    ...

}

**Suspend a process definition**

PUT repository/process-definitions/{processDefinitionId}

**Body JSON:**

{

  "action" : "suspend",

  "includeProcessInstances" : "false",

  "date" : "2013-04-15T00:42:12Z"

}

**Table 15.31. Suspend a process definition - JSON Body parameters**

| Parameter | Description | Required |
| --- | --- | --- |
| action | Action to perform. Either activate or suspend. | Yes |
| includeProcessInstances | Wether or not to suspend/activate running process-instances for this process-definition. If omitted, the process-instances are left in the state they are. | No |
| date | Date (ISO-8601) when the suspension/activation should be executed. If omitted, the suspend/activation is effective immediatly. | No |

**Table 15.32. Suspend a process definition - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the process was suspended. |
| 404 | Indicates the requested process definition was not found. |
| 409 | Indicates the requested process definition is already suspended. |

**Success response body:** see response for repository/process-definitions/{processDefinitionId}.

**Activate a process definition**

PUT repository/process-definitions/{processDefinitionId}

**Body JSON:**

{

  "action" : "activate",

  "includeProcessInstances" : "true",

  "date" : "2013-04-15T00:42:12Z"

}

See suspend process definition [JSON Body parameters](http://www.mossle.com/docs/activiti/#processDefinitionActionBodyParameters).

**Table 15.33. Activate a process definition - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the process was activated. |
| 404 | Indicates the requested process definition was not found. |
| 409 | Indicates the requested process definition is already active. |

**Success response body:** see response for repository/process-definitions/{processDefinitionId}.

**Get all candidate starters for a process-definition**

GET repository/process-definitions/{processDefinitionId}/identitylinks

**Table 15.34. Get all candidate starters for a process-definition - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processDefinitionId | Yes | String | The id of the process definition to get the identity links for. |

**Table 15.35. Get all candidate starters for a process-definition - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the process definition was found and the requested identity links are returned. |
| 404 | Indicates the requested process definition was not found. |

**Success response body:**

[

   {

      "url":"http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4/identitylinks/groups/admin",

      "user":null,

      "group":"admin",

      "type":"candidate"

   },

   {

      "url":"http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4/identitylinks/users/kermit",

      "user":"kermit",

      "group":null,

      "type":"candidate"

   }

]

**Add a candidate starter to a process definition**

POST repository/process-definitions/{processDefinitionId}/identitylinks

**Table 15.36. Add a candidate starter to a process definition - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processDefinitionId | Yes | String | The id of the process definition. |

**Request body (user):**

{

  "userId" : "kermit"

}

**Request body (group):**

{

  "groupId" : "sales"

}

**Table 15.37. Add a candidate starter to a process definition - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the process definition was found and the identity link was created. |
| 404 | Indicates the requested process definition was not found. |

**Success response body:**

{

  "url":"http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4/identitylinks/users/kermit",

  "user":"kermit",

  "group":null,

  "type":"candidate"

}

**Delete a candidate starter from a process definition**

DELETE repository/process-definitions/{processDefinitionId}/identitylinks/{family}/{identityId}

**Table 15.38. Delete a candidate starter from a process definition - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processDefinitionId | Yes | String | The id of the process definition. |
| family | Yes | String | Either users or groups, depending on the type of identity link. |
| identityId | Yes | String | Either the userId or groupId of the identity to remove as candidate starter. |

**Table 15.39. Delete a candidate starter from a process definition - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the process definition was found and the identity link was removed. The response body is intentionally empty. |
| 404 | Indicates the requested process definition was not found or the process definition doesn't have an identity-link that matches the url. |

**Success response body:**

{

  "url":"http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4/identitylinks/users/kermit",

  "user":"kermit",

  "group":null,

  "type":"candidate"

}

**Get a candidate starter from a process definition**

GET repository/process-definitions/{processDefinitionId}/identitylinks/{family}/{identityId}

**Table 15.40. Get a candidate starter from a process definition - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processDefinitionId | Yes | String | The id of the process definition. |
| family | Yes | String | Either users or groups, depending on the type of identity link. |
| identityId | Yes | String | Either the userId or groupId of the identity to get as candidate starter. |

**Table 15.41. Get a candidate starter from a process definition - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the process definition was found and the identity link was returned. |
| 404 | Indicates the requested process definition was not found or the process definition doesn't have an identity-link that matches the url. |

**Success response body:**

{

  "url":"http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4/identitylinks/users/kermit",

  "user":"kermit",

  "group":null,

  "type":"candidate"

}

**Models**

**Get a list of models**

GET repository/models

**Table 15.42. Get a list of models - URL query parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| id | No | String | Only return models with the given id. |
| category | No | String | Only return models with the given category. |
| categoryLike | No | String | Only return models with a category like the given value. Use the % character as wildcard. |
| categoryNotEquals | No | String | Only return models without the given category. |
| name | No | String | Only return models with the given name. |
| nameLike | No | String | Only return models with a name like the given value. Use the % character as wildcard. |
| key | No | String | Only return models with the given key. |
| deploymentId | No | String | Only return models which are deployed in the given deployment. |
| version | No | Integer | Only return models with the given version. |
| latestVersion | No | Boolean | If true, only return models which are the latest version. Best used in combination with key. If false is passed in as value, this is ignored and all versions are returned. |
| deployed | No | Boolean | If true, only deployed models are returned. If false, only undeployed models are returned (deploymentId is null). |
| sort | No | 'id' (default), 'category', 'createTime', 'key', 'lastUpdateTime', 'name' and 'version' | Property to sort on, to be used toghether with the 'order'. |
| The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL. |  |  |  |

**Table 15.43. Get a list of models - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the models are returned |
| 400 | Indicates a parameter was passed in the wrong format. The status-message contains additional information. |

**Success response body:**

{

   "data":[

      {

         "name":"Model name",

         "key":"Model key",

         "category":"Model category",

         "version":2,

         "metaInfo":"Model metainfo",

         "deploymentId":"7",

         "id":"10",

         "url":"http://localhost:8182/repository/models/10",

         "createTime":"2013-06-12T14:31:08.612+0000",

         "lastUpdateTime":"2013-06-12T14:31:08.612+0000",

         "deploymentUrl":"http://localhost:8182/repository/deployments/7"

      },

      ...

   ],

   "total":2,

   "start":0,

   "sort":"id",

   "order":"asc",

   "size":2

}

**Get a model**

GET repository/models/{modelId}

**Table 15.44. Get a model - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| modelId | Yes | String | The id of the model to get. |

**Table 15.45. Get a model - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the model was found and returned. |
| 404 | Indicates the requested model was not found. |

**Success response body:**

{

   "id":"5",

   "url":"http://localhost:8182/repository/models/5",

   "name":"Model name",

   "key":"Model key",

   "category":"Model category",

   "version":2,

   "metaInfo":"Model metainfo",

   "deploymentId":"2",

   "deploymentUrl":"http://localhost:8182/repository/deployments/2",

   "createTime":"2013-06-12T12:31:19.861+0000",

   "lastUpdateTime":"2013-06-12T12:31:19.861+0000"

}

**Update a model**

PUT repository/models/{modelId}

**Request body:**

{

   "name":"Model name",

   "key":"Model key",

   "category":"Model category",

   "version":2,

   "metaInfo":"Model metainfo",

   "deploymentId":"2"

}

All request values are optional. For example, you can only include the 'name' attribute in the request body JSON-object, only updating the name of the model, leaving all other fields unaffected. When an attribute is explicitly included and is set to null, the model-value will be updated to null. Example: {"metaInfo" : null} will clear the metaInfo of the model).

**Table 15.46. Update a model - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the model was found and updated. |
| 404 | Indicates the requested model was not found. |

**Success response body:**

{

   "id":"5",

   "url":"http://localhost:8182/repository/models/5",

   "name":"Model name",

   "key":"Model key",

   "category":"Model category",

   "version":2,

   "metaInfo":"Model metainfo",

   "deploymentId":"2",

   "deploymentUrl":"http://localhost:8182/repository/deployments/2",

   "createTime":"2013-06-12T12:31:19.861+0000",

   "lastUpdateTime":"2013-06-12T12:31:19.861+0000"

}

**Create a model**

POST repository/models

**Request body:**

{

   "name":"Model name",

   "key":"Model key",

   "category":"Model category",

   "version":1,

   "metaInfo":"Model metainfo",

   "deploymentId":"2"

}

All request values are optional. For example, you can only include the 'name' attribute in the request body JSON-object, only setting the name of the model, leaving all other fields null.

**Table 15.47. Create a model - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the model was created. |

**Success response body:**

{

   "id":"5",

   "url":"http://localhost:8182/repository/models/5",

   "name":"Model name",

   "key":"Model key",

   "category":"Model category",

   "version":1,

   "metaInfo":"Model metainfo",

   "deploymentId":"2",

   "deploymentUrl":"http://localhost:8182/repository/deployments/2",

   "createTime":"2013-06-12T12:31:19.861+0000",

   "lastUpdateTime":"2013-06-12T12:31:19.861+0000"

}

**Delete a model**

DELETE repository/models/{modelId}

**Table 15.48. Delete a model - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| modelId | Yes | String | The id of the model to delete. |

**Table 15.49. Delete a model - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the model was found and has been deleted. Response-body is intentionally empty. |
| 404 | Indicates the requested model was not found. |

**Get the editor source for a model**

GET repository/models/{modelId}/source

**Table 15.50. Get the editor source for a model - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| modelId | Yes | String | The id of the model. |

**Table 15.51. Get the editor source for a model - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the model was found and source is returned. |
| 404 | Indicates the requested model was not found. |

**Success response body:** Reponse body contains the model's raw editor source. The response's content-type is set to application/octet-stream, regardless of the content of the source.

**Set the editor source for a model**

PUT repository/models/{modelId}/source

**Table 15.52. Set the editor source for a model - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| modelId | Yes | String | The id of the model. |

**Request body:** The request should be of type multipart/form-data. There should be a single file-part included with the binary value of the source.

**Table 15.53. Set the editor source for a model - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the model was found and the source has been updated. |
| 404 | Indicates the requested model was not found. |

**Success response body:** Reponse body contains the model's raw editor source. The response's content-type is set to application/octet-stream, regardless of the content of the source.

**Get the extra editor source for a model**

GET repository/models/{modelId}/source-extra

**Table 15.54. Get the extra editor source for a model - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| modelId | Yes | String | The id of the model. |

**Table 15.55. Get the extra editor source for a model - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the model was found and source is returned. |
| 404 | Indicates the requested model was not found. |

**Success response body:** Reponse body contains the model's raw extra editor source. The response's content-type is set to application/octet-stream, regardless of the content of the extra source.

**Set the extra editor source for a model**

PUT repository/models/{modelId}/source-extra

**Table 15.56. Set the extra editor source for a model - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| modelId | Yes | String | The id of the model. |

**Request body:** The request should be of type multipart/form-data. There should be a single file-part included with the binary value of the extra source.

**Table 15.57. Set the extra editor source for a model - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the model was found and the extra source has been updated. |
| 404 | Indicates the requested model was not found. |

**Success response body:** Reponse body contains the model's raw editor source. The response's content-type is set to application/octet-stream, regardless of the content of the source.

**Process Instances**

**Get a process instance**

GET runtime/process-instances/{processInstanceId}

**Table 15.58. Get a process instance - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processInstanceId | Yes | String | The id of the process instance to get. |

**Table 15.59. Get a process instance - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the process instance was found and returned. |
| 404 | Indicates the requested process instance was not found. |

**Success response body:**

{

   "id":"7",

   "url":"http://localhost:8182/runtime/process-instances/7",

   "businessKey":"myBusinessKey",

   "suspended":false,

   "processDefinitionUrl":"http://localhost:8182/repository/process-definitions/processOne%3A1%3A4",

   "activityId":"processTask"

}

**Delete a process instance**

DELETE runtime/process-instances/{processInstanceId}

**Table 15.60. Delete a process instance - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processInstanceId | Yes | String | The id of the process instance to delete. |

**Table 15.61. Delete a process instance - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the process instance was found and deleted. Response body is left empty intentionally. |
| 404 | Indicates the requested process instance was not found. |

**Activate or suspend a process instance**

PUT runtime/process-instances/{processInstanceId}

**Table 15.62. Activate or suspend a process instance - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processInstanceId | Yes | String | The id of the process instance to activate/suspend. |

**Request response body (suspend):**

{

   "action":"suspend"

}

**Request response body (activate):**

{

   "action":"activate"

}

**Table 15.63. Activate or suspend a process instance - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the process instance was found and action was executed. |
| 400 | Indicates an invalid action was supplied. |
| 404 | Indicates the requested process instance was not found. |
| 409 | Indicates the requested process instance action cannot be executed since the process-instance is already activated/suspended. |

**Start a process instance**

POST runtime/process-instances

**Request body (start by process definition id):**

{

   "processDefinitionId":"oneTaskProcess:1:158",

   "businessKey":"myBusinessKey",

   "variables": [

      {

        "name":"myVar",

        "value":"This is a variable",

      },

      ...

   ]

}

**Request body (start by process definition key):**

{

   "processDefinitionKey":"oneTaskProcess",

   "businessKey":"myBusinessKey",

   "variables": [

      {

        "name":"myVar",

        "value":"This is a variable",

      },

      ...

   ]

}

**Request body (start by message):**

{

   "processDefinitionKey":"newOrderMessage",

   "businessKey":"myBusinessKey",

   "variables": [

      {

        "name":"myVar",

        "value":"This is a variable",

      },

      ...

   ]

}

Only one of processDefinitionId, processDefinitionKey or message can be used in the request body. Both businessKey and variables are optional. More information about the variable format can be found in [the REST variables section](http://www.mossle.com/docs/activiti/#restVariables). Note that the variable-scope that is supplied is ignored, process-variables are always local.

**Table 15.64. Start a process instance - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the process instance was created. |
| 400 | Indicates either the process-definition was not found (based on id or key), no process is started by sending the given message or an invalid variable has been passed. Status description contains additional information about the error. |

**Success response body:**

{

   "id":"7",

   "url":"http://localhost:8182/runtime/process-instances/7",

   "businessKey":"myBusinessKey",

   "suspended":false,

   "processDefinitionUrl":"http://localhost:8182/repository/process-definitions/processOne%3A1%3A4",

   "activityId":"processTask"

}

**List of process instances**

GET runtime/process-instances

**Table 15.65. List of process instances - URL query parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| id | No | String | Only return process instance with the given id. |
| processDefinitionKey | No | String | Only return process instances with the given process definition key. |
| processDefinitionId | No | String | Only return process instances with the given process definition id. |
| businessKey | No | String | Only return process instances with the given businessKey. |
| involvedUser | No | String | Only return process instances in which the given user is involved. |
| suspended | No | Boolean | If true, only return process instance which are suspended. If false, only return process instances which are not suspended (active). |
| superProcessInstanceId | No | String | Only return process instances which have the given super process-instance id (for processes that have a call-activities). |
| subProcessInstanceId | No | String | Only return process instances which have the given sub process-instance id (for processes started as a call-activity). |
| excludeSubprocesses | No | Boolean | Return only process instances which aren't sub processes. |
| includeProcessVariables | No | Boolean | Indication to include process variables in the result. |
| sort | No | String | Sort field, should be either one of id (default), processDefinitionId or processDefinitionKey. |
| The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL. |  |  |  |

**Table 15.66. List of process instances - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the process-instances are returned |
| 400 | Indicates a parameter was passed in the wrong format . The status-message contains additional information. |

**Success response body:**

{

   "data":[

      {

         "id":"7",

         "url":"http://localhost:8182/runtime/process-instances/7",

         "businessKey":"myBusinessKey",

         "suspended":false,

         "processDefinitionUrl":"http://localhost:8182/repository/process-definitions/processOne%3A1%3A4",

         "activityId":"processTask"

      },

      ...

   ],

   "total":2,

   "start":0,

   "sort":"id",

   "order":"asc",

   "size":2

}

**Query process instances**

POST query/process-instances

**Request body:**

{

  "processDefinitionKey":"oneTaskProcess",

  "variables":

  [

    {

        "name" : "myVariable",

        "value" : 1234,

        "operator" : "equals",

        "type" : "long"

    },

    ...

  ],

  ...

}

The request body can contain all possible filters that can be used in the [List process instances](http://www.mossle.com/docs/activiti/#restProcessInstancesGet) URL query. On top of these, it's possible to provide an array of variables to include in the query, with their format [described here](http://www.mossle.com/docs/activiti/#restQueryVariable).

The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL.

**Table 15.67. Query process instances - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the process-instances are returned |
| 400 | Indicates a parameter was passed in the wrong format . The status-message contains additional information. |

**Success response body:**

{

   "data":[

      {

         "id":"7",

         "url":"http://localhost:8182/runtime/process-instances/7",

         "businessKey":"myBusinessKey",

         "suspended":false,

         "processDefinitionUrl":"http://localhost:8182/repository/process-definitions/processOne%3A1%3A4",

         "activityId":"processTask"

      },

      ...

   ],

   "total":2,

   "start":0,

   "sort":"id",

   "order":"asc",

   "size":2

}

**Get diagram for a process instance**

GET runtime/process-instances/{processInstanceId}

**Table 15.68. Get diagram for a process instance - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processInstanceId | Yes | String | The id of the process instance to get the diagram for. |

**Table 15.69. Get diagram for a process instance - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the process instance was found and the diagram was returned. |
| 400 | Indicates the requested process instance was not found but the process doesn't contain any graphical information (BPMN:DI) and no diagram can be created. |
| 404 | Indicates the requested process instance was not found. |

**Success response body:**

{

   "id":"7",

   "url":"http://localhost:8182/runtime/process-instances/7",

   "businessKey":"myBusinessKey",

   "suspended":false,

   "processDefinitionUrl":"http://localhost:8182/repository/process-definitions/processOne%3A1%3A4",

   "activityId":"processTask"

}

**Get involved people for process instance**

GET runtime/process-instances/{processInstanceId}/identitylinks

**Table 15.70. Get involved people for process instance - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processInstanceId | Yes | String | The id of the process instance to the links for. |

**Table 15.71. Get involved people for process instance - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the process instance was found and links are returned. |
| 404 | Indicates the requested process instance was not found. |

**Success response body:**

[

   {

      "url":"http://localhost:8182/runtime/process-instances/5/identitylinks/users/john/customType",

      "user":"john",

      "group":null,

      "type":"customType"

   },

   {

      "url":"http://localhost:8182/runtime/process-instances/5/identitylinks/users/paul/candidate",

      "user":"paul",

      "group":null,

      "type":"candidate"

   }

]

Note that the groupId will always be null, as it's only possible to involve users with a process-instance.

**Add an involved user to a process instance**

POST runtime/process-instances/{processInstanceId}/identitylinks

**Table 15.72. Add an involved user to a process instance - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processInstanceId | Yes | String | The id of the process instance to the links for. |

**Request body:**

{

  "userId":"kermit",

  "type":"participant"

}

Both userId and type are required.

**Table 15.73. Add an involved user to a process instance - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the process instance was found and the link is created. |
| 400 | Indicates the requested body did not contain a userId or a type. |
| 404 | Indicates the requested process instance was not found. |

**Success response body:**

{

   "url":"http://localhost:8182/runtime/process-instances/5/identitylinks/users/john/customType",

   "user":"john",

   "group":null,

   "type":"customType"

}

Note that the groupId will always be null, as it's only possible to involve users with a process-instance.

**Remove an involved user to from process instance**

DELETE runtime/process-instances/{processInstanceId}/identitylinks/users/{userId}/{type}

**Table 15.74. Remove an involved user to from process instance - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processInstanceId | Yes | String | The id of the process instance. |
| userId | Yes | String | The id of the user to delete link for. |
| type | Yes | String | Type of link to delete. |

**Table 15.75. Remove an involved user to from process instance - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the process instance was found and the link has been deleted. Response body is left empty intentionally. |
| 404 | Indicates the requested process instance was not found or the link to delete doesn't exist. The response status contains additional information about the error. |

**Success response body:**

{

   "url":"http://localhost:8182/runtime/process-instances/5/identitylinks/users/john/customType",

   "user":"john",

   "group":null,

   "type":"customType"

}

Note that the groupId will always be null, as it's only possible to involve users with a process-instance.

**List of variables for a process instance**

GET runtime/process-instances/{processInstanceId}/variables

**Table 15.76. List of variables for a process instance - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processInstanceId | Yes | String | The id of the process instance to the variables for. |

**Table 15.77. List of variables for a process instance - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the process instance was found and variables are returned. |
| 404 | Indicates the requested process instance was not found. |

**Success response body:**

[

   {

      "name":"intProcVar",

      "type":"integer",

      "value":123,

      "scope":"local"

   },

   {

      "name":"byteArrayProcVar",

      "type":"binary",

      "value":null,

      "valueUrl":"http://localhost:8182/runtime/process-instances/5/variables/byteArrayProcVar/data",

      "scope":"local"

   },

   ...

]

In case the variable is a binary variable or serializable, the valueUrl points to an URL to fetch the raw value. If it's a plain variable, the value is present in the response. Note that only local scoped variables are returned, as there is no global scope for process-instance variables.

**Get a variable for a process instance**

GET runtime/process-instances/{processInstanceId}/variables/{variableName}

**Table 15.78. Get a variable for a process instance - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processInstanceId | Yes | String | The id of the process instance to the variables for. |
| variableName | Yes | String | Name of the variable to get. |

**Table 15.79. Get a variable for a process instance - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates both the process instance and variable were found and variable is returned. |
| 400 | Indicates the request body is incomplete or contains illegal values. The status description contains additional information about the error. |
| 404 | Indicates the requested process instance was not found or the process instance does not have a variable with the given name. Status description contains additional information about the error. |

**Success response body:**

   {

      "name":"intProcVar",

      "type":"integer",

      "value":123,

      "scope":"local"

   }

In case the variable is a binary variable or serializable, the valueUrl points to an URL to fetch the raw value. If it's a plain variable, the value is present in the response. Note that only local scoped variables are returned, as there is no global scope for process-instance variables.

**Create (or update) variables on a process instance**

POST runtime/process-instances/{processInstanceId}/variables

PUT runtime/process-instances/{processInstanceId}/variables

When using POST, all variables that are passed are created. In case one of the variables already exists on the process instance, the request results in an error (409 - CONFLICT). When PUT is used, unexisting variables are created on the process-instance and existing ones are overridden without any error.

**Table 15.80. Create (or update) variables on a process instance - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processInstanceId | Yes | String | The id of the process instance to the variables for. |

**Request body:**

[

   {

      "name":"intProcVar"

      "type":"integer"

      "value":123

   },

   ...

]

Any number of variables can be passed into the request body array. More information about the variable format can be found in [the REST variables section](http://www.mossle.com/docs/activiti/#restVariables). Note that scope is ignored, only local variables can be set in a process instance.

**Table 15.81. Create (or update) variables on a process instance - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the process instance was found and variable is created. |
| 400 | Indicates the request body is incomplete or contains illegal values. The status description contains additional information about the error. |
| 404 | Indicates the requested process instance was not found. |
| 409 | Indicates the process instance was found but already contains a variable with the given name (only thrown when POST method is used). Use the update-method instead. |

**Success response body:**

[

   {

      "name":"intProcVar",

      "type":"integer",

      "value":123,

      "scope":"local"

   },

   ...

]

**Update a single variable on a process instance**

PUT runtime/process-instances/{processInstanceId}/variables/{variableName}

**Table 15.82. Update a single variable on a process instance - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processInstanceId | Yes | String | The id of the process instance to the variables for. |
| variableName | Yes | String | Name of the variable to get. |

**Request body:**

 {

    "name":"intProcVar"

    "type":"integer"

    "value":123

 }

More information about the variable format can be found in [the REST variables section](http://www.mossle.com/docs/activiti/#restVariables). Note that scope is ignored, only local variables can be set in a process instance.

**Table 15.83. Update a single variable on a process instance - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates both the process instance and variable were found and variable is updated. |
| 404 | Indicates the requested process instance was not found or the process instance does not have a variable with the given name. Status description contains additional information about the error. |

**Success response body:**

   {

      "name":"intProcVar",

      "type":"integer",

      "value":123,

      "scope":"local"

   }

In case the variable is a binary variable or serializable, the valueUrl points to an URL to fetch the raw value. If it's a plain variable, the value is present in the response. Note that only local scoped variables are returned, as there is no global scope for process-instance variables.

**Create a new binary variable on a process-instance**

POST runtime/process-instances/{processInstanceId}/variables

**Table 15.84. Create a new binary variable on a process-instance - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processInstanceId | Yes | String | The id of the process instance to create the new variable for. |

**Request body:** The request should be of type multipart/form-data. There should be a single file-part included with the binary value of the variable. On top of that, the folowing additional form-fields can be present:

* name: Required name of the variable.
* type: Type of variable that is created. If omitted, binary is assumed and the binary data in the request will be stored as an array of bytes.

**Success response body:**

{

  "name" : "binaryVariable",

  "scope" : "local",

  "type" : "binary",

  "value" : null,

  "valueUrl" : "http://.../runtime/process-instances/123/variables/binaryVariable/data"

}

**Table 15.85. Create a new binary variable on a process-instance - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the variable was created and the result is returned. |
| 400 | Indicates the name of the variable to create was missing. Status message provides additional information. |
| 404 | Indicates the requested process instance was not found. |
| 409 | Indicates the process instance already has a variable with the given name. Use the PUT method to update the task variable instead. |
| 415 | Indicates the serializable data contains an object for which no class is present in the JVM running the Activiti engine and therefore cannot be deserialized. |

**Update an existing binary variable on a process-instance**

PUT runtime/process-instances/{processInstanceId}/variables

**Table 15.86. Update an existing binary variable on a process-instance - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processInstanceId | Yes | String | The id of the process instance to create the new variable for. |

**Request body:** The request should be of type multipart/form-data. There should be a single file-part included with the binary value of the variable. On top of that, the folowing additional form-fields can be present:

* name: Required name of the variable.
* type: Type of variable that is created. If omitted, binary is assumed and the binary data in the request will be stored as an array of bytes.

**Success response body:**

{

  "name" : "binaryVariable",

  "scope" : "local",

  "type" : "binary",

  "value" : null,

  "valueUrl" : "http://.../runtime/process-instances/123/variables/binaryVariable/data"

}

**Table 15.87. Update an existing binary variable on a process-instance - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the variable was udpated and the result is returned. |
| 400 | Indicates the name of the variable to update was missing. Status message provides additional information. |
| 404 | Indicates the requested process instance was not found or the process instance does not have a variable with the given name. |
| 415 | Indicates the serializable data contains an object for which no class is present in the JVM running the Activiti engine and therefore cannot be deserialized. |

**Executions**

**Get an execution**

GET runtime/executions/{executionId}

**Table 15.88. Get an execution - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| executionId | Yes | String | The id of the execution to get. |

**Table 15.89. Get an execution - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the execution was found and returned. |
| 404 | Indicates the execution was not found. |

**Success response body:**

{

   "id":"5",

   "url":"http://localhost:8182/runtime/executions/5",

   "parentId":null,

   "parentUrl":null,

   "processInstanceId":"5",

   "processInstanceUrl":"http://localhost:8182/runtime/process-instances/5",

   "suspended":false,

   "activityId":null

}

**Execute an action on an execution**

PUT runtime/executions/{executionId}

**Table 15.90. Execute an action on an execution - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| executionId | Yes | String | The id of the execution to execute action on. |

**Request body (signal an execution):**

{

  "action":"signal"

}

**Request body (signal event received for execution):**

{

  "action":"signalEventReceived",

  "signalName":"mySignal"

  "variables": [ ... ]

}

Notifies the execution that a signal event has been received, requires a signalName parameter. Optional variables can be passed that are set on the execution before the action is executed.

**Request body (signal event received for execution):**

{

  "action":"messageEventReceived",

  "messageName":"myMessage"

  "variables": [ ... ]

}

Notifies the execution that a message event has been received, requires a messageName parameter. Optional variables can be passed that are set on the execution before the action is executed.

**Table 15.91. Execute an action on an execution - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the execution was found and the action is performed. |
| 204 | Indicates the execution was found, the action was performed and the action caused the execution to end. |
| 400 | Indicates an illegal action was requested, required parameters are missing in the resuest body or illegal variables are passed in. Status description contains additional information about the error. |
| 404 | Indicates the execution was not found. |

**Success response body (in case execution is not ended due to action):**

{

   "id":"5",

   "url":"http://localhost:8182/runtime/executions/5",

   "parentId":null,

   "parentUrl":null,

   "processInstanceId":"5",

   "processInstanceUrl":"http://localhost:8182/runtime/process-instances/5",

   "suspended":false,

   "activityId":null

}

**Get active activities in an execution**

GET runtime/executions/{executionId}/activities

Returns all activities which are active in the execution and in all child-executions (and their children, recursively), if any.

**Table 15.92. Get active activities in an execution - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| executionId | Yes | String | The id of the execution to get activities for. |

**Table 15.93. Get active activities in an execution - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the execution was found and activities are returned. |
| 404 | Indicates the execution was not found. |

**Success response body:**

[

  "userTaskForManager",

  "receiveTask"

]

**List of executions**

GET repository/executions

**Table 15.94. List of executions - URL query parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| id | No | String | Only return executions with the given id. |
| processDefinitionKey | No | String | Only return executions with the given process definition key. |
| processDefinitionId | No | String | Only return executions with the given process definition id. |
| processInstanceId | No | String | Only return executions which are part of the process instance with the given id. |
| messageEventSubscriptionName | No | String | Only return executions which are subscribed to a message with the given name. |
| signalEventSubscriptionName | No | String | Only return executions which are subscribed to a signal with the given name. |
| parentId | No | String | Only return executions wich are a direct child of the given execution. |
| sort | No | String | Sort field, should be either one of processInstanceId (default), processDefinitionId or processDefinitionKey. |
| The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL. |  |  |  |

**Table 15.95. List of executions - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the executions are returned |
| 400 | Indicates a parameter was passed in the wrong format . The status-message contains additional information. |

**Success response body:**

{

   "data":[

      {

         "id":"5",

         "url":"http://localhost:8182/runtime/executions/5",

         "parentId":null,

         "parentUrl":null,

         "processInstanceId":"5",

         "processInstanceUrl":"http://localhost:8182/runtime/process-instances/5",

         "suspended":false,

         "activityId":null

      },

      {

         "id":"7",

         "url":"http://localhost:8182/runtime/executions/7",

         "parentId":"5",

         "parentUrl":"http://localhost:8182/runtime/executions/5",

         "processInstanceId":"5",

         "processInstanceUrl":"http://localhost:8182/runtime/process-instances/5",

         "suspended":false,

         "activityId":"processTask"

      }

   ],

   "total":2,

   "start":0,

   "sort":"processInstanceId",

   "order":"asc",

   "size":2

}

**Query executions**

POST query/executions

**Request body:**

{

  "processDefinitionKey":"oneTaskProcess",

  "variables":

  [

    {

        "name" : "myVariable",

        "value" : 1234,

        "operator" : "equals",

        "type" : "long"

    },

    ...

  ],

  "processInstanceVariables":

  [

    {

        "name" : "processVariable",

        "value" : "some string",

        "operator" : "equals",

        "type" : "string"

    },

    ...

  ],

  ...

}

The request body can contain all possible filters that can be used in the [List executions](http://www.mossle.com/docs/activiti/#restExecutionsGet) URL query. On top of these, it's possible to provide an array of variables and processInstanceVariables to include in the query, with their format [described here](http://www.mossle.com/docs/activiti/#restQueryVariable).

The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL.

**Table 15.96. Query executions - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the executions are returned |
| 400 | Indicates a parameter was passed in the wrong format . The status-message contains additional information. |

**Success response body:**

{

   "data":[

      {

         "id":"5",

         "url":"http://localhost:8182/runtime/executions/5",

         "parentId":null,

         "parentUrl":null,

         "processInstanceId":"5",

         "processInstanceUrl":"http://localhost:8182/runtime/process-instances/5",

         "suspended":false,

         "activityId":null

      },

      {

         "id":"7",

         "url":"http://localhost:8182/runtime/executions/7",

         "parentId":"5",

         "parentUrl":"http://localhost:8182/runtime/executions/5",

         "processInstanceId":"5",

         "processInstanceUrl":"http://localhost:8182/runtime/process-instances/5",

         "suspended":false,

         "activityId":"processTask"

      }

   ],

   "total":2,

   "start":0,

   "sort":"processInstanceId",

   "order":"asc",

   "size":2

}

**List of variables for an execution**

GET runtime/executions/{executionId}/variables?scope={scope}

**Table 15.97. List of variables for an execution - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| executionId | Yes | String | The id of the execution to the variables for. |
| scope | No | String | Either local or global. If ommitted, both local and global scoped variables are returned. |

**Table 15.98. List of variables for an execution - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the execution was found and variables are returned. |
| 404 | Indicates the requested execution was not found. |

**Success response body:**

[

   {

      "name":"intProcVar",

      "type":"integer",

      "value":123,

      "scope":"global"

   },

   {

      "name":"byteArrayProcVar",

      "type":"binary",

      "value":null,

      "valueUrl":"http://localhost:8182/runtime/process-instances/5/variables/byteArrayProcVar/data",

      "scope":"local"

   },

   ...

]

In case the variable is a binary variable or serializable, the valueUrl points to an URL to fetch the raw value. If it's a plain variable, the value is present in the response.

**Get a variable for an execution**

GET runtime/executions/{executionId}/variables/{variableName}?scope={scope}

**Table 15.99. Get a variable for an execution - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| executionId | Yes | String | The id of the execution to the variables for. |
| variableName | Yes | String | Name of the variable to get. |
| scope | No | String | Either local or global. If ommitted, local variable is returned (if exists). If not, a global variable is returned (if exists). |

**Table 15.100. Get a variable for an execution - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates both the execution and variable were found and variable is returned. |
| 400 | Indicates the request body is incomplete or contains illegal values. The status description contains additional information about the error. |
| 404 | Indicates the requested execution was not found or the execution does not have a variable with the given name in the requested scope (in case scope-query parameter was omitted, variable doesn't exist in local and global scope). Status description contains additional information about the error. |

**Success response body:**

   {

      "name":"intProcVar",

      "type":"integer",

      "value":123,

      "scope":"local"

   }

In case the variable is a binary variable or serializable, the valueUrl points to an URL to fetch the raw value. If it's a plain variable, the value is present in the response.

**Create (or update) variables on an execution**

POST runtime/executions/{executionId}/variables

PUT runtime/executions/{executionId}/variables

When using POST, all variables that are passed are created. In case one of the variables already exists on the execution in the requested scope, the request results in an error (409 - CONFLICT). When PUT is used, unexisting variables are created on the execution and existing ones are overridden without any error.

**Table 15.101. Create (or update) variables on an execution - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| executionId | Yes | String | The id of the execution to the variables for. |

**Request body:**

[

   {

      "name":"intProcVar"

      "type":"integer"

      "value":123,

      "scope":"local"

   },

   ...

]

**Note that you can only provide variables that have the same scope. If the request-body array contains variables from mixed scopes, the request results in an error (400 - BAD REQUEST).**Any number of variables can be passed into the request body array. More information about the variable format can be found in [the REST variables section](http://www.mossle.com/docs/activiti/#restVariables). Note that scope is ignored, only local variables can be set in a process instance.

**Table 15.102. Create (or update) variables on an execution - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the execution was found and variable is created. |
| 400 | Indicates the request body is incomplete or contains illegal values. The status description contains additional information about the error. |
| 404 | Indicates the requested execution was not found. |
| 409 | Indicates the execution was found but already contains a variable with the given name (only thrown when POST method is used). Use the update-method instead. |

**Success response body:**

[

   {

      "name":"intProcVar",

      "type":"integer",

      "value":123,

      "scope":"local"

   },

   ...

]

**Update a variable on an execution**

PUT runtime/executions/{executionId}/variables/{variableName}

**Table 15.103. Update a variable on an execution - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| executionId | Yes | String | The id of the execution to update the variables for. |
| variableName | Yes | String | Name of the variable to update. |

**Request body:**

 {

    "name":"intProcVar"

    "type":"integer"

    "value":123,

    "scope":"global"

 }

More information about the variable format can be found in [the REST variables section](http://www.mossle.com/docs/activiti/#restVariables).

**Table 15.104. Update a variable on an execution - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates both the process instance and variable were found and variable is updated. |
| 404 | Indicates the requested process instance was not found or the process instance does not have a variable with the given name. Status description contains additional information about the error. |

**Success response body:**

   {

      "name":"intProcVar",

      "type":"integer",

      "value":123,

      "scope":"global"

   }

In case the variable is a binary variable or serializable, the valueUrl points to an URL to fetch the raw value. If it's a plain variable, the value is present in the response.

**Create a new binary variable on an execution**

POST runtime/executions/{executionId}/variables

**Table 15.105. Create a new binary variable on an execution - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| executionId | Yes | String | The id of the execution to create the new variable for. |

**Request body:** The request should be of type multipart/form-data. There should be a single file-part included with the binary value of the variable. On top of that, the folowing additional form-fields can be present:

* name: Required name of the variable.
* type: Type of variable that is created. If omitted, binary is assumed and the binary data in the request will be stored as an array of bytes.
* scope: Scope of variable that is created. If omitted, local is assumed.

**Success response body:**

{

  "name" : "binaryVariable",

  "scope" : "local",

  "type" : "binary",

  "value" : null,

  "valueUrl" : "http://.../runtime/executions/123/variables/binaryVariable/data"

}

**Table 15.106. Create a new binary variable on an execution - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the variable was created and the result is returned. |
| 400 | Indicates the name of the variable to create was missing. Status message provides additional information. |
| 404 | Indicates the requested execution was not found. |
| 409 | Indicates the execution already has a variable with the given name. Use the PUT method to update the task variable instead. |
| 415 | Indicates the serializable data contains an object for which no class is present in the JVM running the Activiti engine and therefore cannot be deserialized. |

**Update an existing binary variable on a process-instance**

PUT runtime/executions/{executionId}/variables/{variableName}

**Table 15.107. Update an existing binary variable on a process-instance - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| executionId | Yes | String | The id of the execution to create the new variable for. |
| variableName | Yes | String | The name of the variable to update. |

**Request body:** The request should be of type multipart/form-data. There should be a single file-part included with the binary value of the variable. On top of that, the folowing additional form-fields can be present:

* name: Required name of the variable.
* type: Type of variable that is created. If omitted, binary is assumed and the binary data in the request will be stored as an array of bytes.
* scope: Scope of variable that is created. If omitted, local is assumed.

**Success response body:**

{

  "name" : "binaryVariable",

  "scope" : "local",

  "type" : "binary",

  "value" : null,

  "valueUrl" : "http://.../runtime/executions/123/variables/binaryVariable/data"

}

**Table 15.108. Update an existing binary variable on a process-instance - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the variable was udpated and the result is returned. |
| 400 | Indicates the name of the variable to update was missing. Status message provides additional information. |
| 404 | Indicates the requested execution was not found or the execution does not have a variable with the given name. |
| 415 | Indicates the serializable data contains an object for which no class is present in the JVM running the Activiti engine and therefore cannot be deserialized. |

**Tasks**

**Get a task**

GET runtime/tasks/{taskId}

**Table 15.109. Get a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to get. |

**Table 15.110. Get a task - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the task was found and returned. |
| 404 | Indicates the requested task was not found. |

**Success response body:**

{

  "assignee" : "kermit",

  "createTime" : "2013-04-17T10:17:43.902+0000",

  "delegationState" : "pending",

  "description" : "Task description",

  "dueDate" : "2013-04-17T10:17:43.902+0000",

  "execution" : "http://localhost:8182/runtime/executions/5",

  "id" : "8",

  "name" : "My task",

  "owner" : "owner",

  "parentTask" : "http://localhost:8182/runtime/tasks/9",

  "priority" : 50,

  "processDefinition" : "http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4",

  "processInstance" : "http://localhost:8182/runtime/process-instances/5",

  "suspended" : false,

  "taskDefinitionKey" : "theTask",

  "url" : "http://localhost:8182/runtime/tasks/8"

}

* delegationState: Delegation-state of the task, can be null, "pending" or "resolved".

**List of tasks**

GET runtime/tasks

**Table 15.111. List of tasks - URL query parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| name | No | String | Only return tasks with the given name. |
| nameLike | No | String | Only return tasks with a name like the given name. |
| description | No | String | Only return tasks with the given description. |
| priority | No | Integer | Only return tasks with the given priotiry. |
| minimumPriority | No | Integer | Only return tasks with a priority greater than the given value. |
| maximumPriority | No | Integer | Only return tasks with a priority lower than the given value. |
| assignee | No | String | Only return tasks assigned to the given user. |
| owner | No | String | Only return tasks owned by the given user. |
| unassigned | No | Boolean | Only return tasks that are not assigned to anyone. If false is passed, the value is ignored. |
| delegationState | No | String | Only return tasks that have the given delegation state. Possible values are pending and resolved. |
| candidateUser | No | String | Only return tasks that can be claimed by the given user. This includes both tasks where the user is an explicit candidate for and task that are claimable by a group that the user is a member of. |
| candidateGroup | No | String | Only return tasks that can be claimed by a user in the given group. |
| involvedUser | No | String | Only return tasks in which the given user is involved. |
| taskDefinitionKey | No | String | Only return tasks with the given task definition id. |
| taskDefinitionKeyLike | No | String | Only return tasks with a given task definition id like the given value. |
| processInstanceId | No | String | Only return tasks which are part of the process instance with the given id. |
| processInstanceBusinessKey | No | String | Only return tasks which are part of the process instance with the given business key. |
| executionId | No | String | Only return tasks which are part of the execution with the given id. |
| createdOn | No | ISO Date | Only return tasks which are created on the given date. |
| createdBefore | No | ISO Date | Only return tasks which are created before the given date. |
| createdAfter | No | ISO Date | Only return tasks which are created after the given date. |
| dueOn | No | ISO Date | Only return tasks which are due on the given date. |
| dueBefore | No | ISO Date | Only return tasks which are due before the given date. |
| dueAfter | No | ISO Date | Only return tasks which are due after the given date. |
| excludeSubTasks | No | Boolean | Only return tasks that are not a subtask of another task. |
| active | No | Boolean | If true, only return tasks that are not suspended (either part of a process that is not suspended or not part of a process at all). If false, only tasks that are part of suspended process instances are returned. |
| includeTaskLocalVariables | No | Boolean | Indication to include task local variables in the result. |
| includeProcessVariables | No | Boolean | Indication to include process variables in the result. |
| The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL. |  |  |  |

**Table 15.112. List of tasks - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the tasks are returned |
| 400 | Indicates a parameter was passed in the wrong format or that 'delegationState' has an invalid value (other than 'pending' and 'resolved'). The status-message contains additional information. |

**Success response body:**

{

  "data": [

    {

      "assignee" : "kermit",

      "createTime" : "2013-04-17T10:17:43.902+0000",

      "delegationState" : "pending",

      "description" : "Task description",

      "dueDate" : "2013-04-17T10:17:43.902+0000",

      "execution" : "http://localhost:8182/runtime/executions/5",

      "id" : "8",

      "name" : "My task",

      "owner" : "owner",

      "parentTask" : "http://localhost:8182/runtime/tasks/9",

      "priority" : 50,

      "processDefinition" : "http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4",

      "processInstance" : "http://localhost:8182/runtime/process-instances/5",

      "suspended" : false,

      "taskDefinitionKey" : "theTask",

      "url" : "http://localhost:8182/runtime/tasks/8"

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "name",

  "order": "asc",

  "size": 1

}

**Query for tasks**

POST repository/tasks

**Request body:**

{

  "name" : "My task",

  "description" : "The task description",

  ...

  "taskVariables" : [

    {

      "name" : "myVariable",

      "value" : 1234,

      "operator" : "equals",

      "type" : "long"

    }

  ],

    "processInstanceVariables" : [

      {

         ...

      }

    ]

  ]

}

All supported JSON parameter fields allowed are exactly the same as the parameters found for [getting a collection of tasks](http://www.mossle.com/docs/activiti/#restTasksGet), but passed in as JSON-body arguments rather than URL-parameters to allow for more advanced querying and preventing errors with request-uri's that are too long. On top of that, the query allows for filtering based on task and process variables. The taskVariables and processInstanceVariables are both json-arrays containing objects with the format [as described here.](http://www.mossle.com/docs/activiti/#restQueryVariable)

**Table 15.113. Query for tasks - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the tasks are returned |
| 400 | Indicates a parameter was passed in the wrong format or that 'delegationState' has an invalid value (other than 'pending' and 'resolved'). The status-message contains additional information. |

**Success response body:**

{

  "data": [

    {

      "assignee" : "kermit",

      "createTime" : "2013-04-17T10:17:43.902+0000",

      "delegationState" : "pending",

      "description" : "Task description",

      "dueDate" : "2013-04-17T10:17:43.902+0000",

      "execution" : "http://localhost:8182/runtime/executions/5",

      "id" : "8",

      "name" : "My task",

      "owner" : "owner",

      "parentTask" : "http://localhost:8182/runtime/tasks/9",

      "priority" : 50,

      "processDefinition" : "http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4",

      "processInstance" : "http://localhost:8182/runtime/process-instances/5",

      "suspended" : false,

      "taskDefinitionKey" : "theTask",

      "url" : "http://localhost:8182/runtime/tasks/8"

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "name",

  "order": "asc",

  "size": 1

}

**Update a task**

PUT runtime/tasks/{taskId}

**Body JSON:**

{

  "assignee" : "assignee",

  "delegationState" : "resolved",

  "description" : "New task description",

  "dueDate" : "2013-04-17T13:06:02.438+02:00",

  "name" : "New task name",

  "owner" : "owner",

  "parentTaskId" : "3",

  "priority" : 20

}

All request values are optional. For example, you can only include the 'assignee' attribute in the request body JSON-object, only updating the assignee of the task, leaving all other fields unaffected. When an attribute is explicitly included and is set to null, the task-value will be updated to null. Example: {"dueDate" : null} will clear the duedate of the task).

**Table 15.114. Update a task - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the task was updated. |
| 404 | Indicates the requested task was not found. |
| 409 | Indicates the requested task was updated simultaneously. |

**Success response body:** see response for runtime/tasks/{taskId}.

**Task actions**

POST runtime/tasks/{taskId}

**Complete a task - Body JSON:**

{

  "action" : "complete",

  "variables" : ...

}

Completes the task. Optional variable array can be passed in using the variables property. More information about the variable format can be found in [the REST variables section](http://www.mossle.com/docs/activiti/#restVariables). Note that the variable-scope that is supplied is ignored and the variables are set on the parent-scope unless a variable exists in a local scope, which is overridden in this case. This is the same behavior as the TaskService.completeTask(taskId, variables) invocation.

**Claim a task - Body JSON:**

{

  "action" : "claim",

  "assignee" : "userWhoClaims"

}

Claims the task by the given assignee. The assignee is required.

**Delegate a task - Body JSON:**

{

  "action" : "delegate",

  "assignee" : "userToDelegateTo"

}

Delegates the task to the given assignee. The assignee is required.

**Resolve a task - Body JSON:**

{

  "action" : "resolve"

}

Resolves the task delegation. The task is assigned back to the task owner (if any).

**Table 15.115. Task actions - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the action was executed. |
| 400 | When the body contains an invalid value or when the assignee is missing when the action requires it. |
| 404 | Indicates the requested task was not found. |
| 409 | Indicates the action cannot be performed due to a conflict. Either the task was updates simultaneously or the task was claimend by another user, in case of the 'claim' action. |

**Success response body:** see response for runtime/tasks/{taskId}.

**Delete a task**

DELETE runtime/tasks/{taskId}?cascadeHistory={cascadeHistory}&deleteReason={deleteReason}

**Table 15.116. >Delete a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to delete. |
| cascadeHistory | False | Boolean | Whether or not to delete the HistoricTask instance when deleting the task (if applicable). If not provided, this value defaults to false. |
| deleteReason | False | String | Reason why the task is deleted. This value is ignored when cascadeHistory is true. |

**Table 15.117. >Delete a task - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the task was found and has been deleted. Response-body is intentionally empty. |
| 403 | Indicates the requested task cannot be deleted because it's part of a workflow. |
| 404 | Indicates the requested task was not found. |

**Get all variables for a task**

GET runtime/tasks/{taskId}/variables?scope={scope}

**Table 15.118. Get all variables for a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to get variables for. |
| scope | False | String | Scope of variables to be returned. When 'local', only task-local variables are returned. When 'global', only variables from the task's parent execution-hierarchy are returned. When the parameter is omitted, both local and global variables are returned. |

**Table 15.119. Get all variables for a task - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the task was found and the requested variables are returned. |
| 404 | Indicates the requested task was not found. |

**Success response body:**

[

  {

    "name" : "doubleTaskVar",

    "scope" : "local",

    "type" : "double",

    "value" : 99.99

  },

  {

    "name" : "stringProcVar",

    "scope" : "global",

    "type" : "string",

    "value" : "This is a ProcVariable"

  },

  ...

]

The variables are returned as a JSON array. Full response description can be found in the general [REST-variables section](http://www.mossle.com/docs/activiti/#restVariables).

**Get a variable from a task**

GET runtime/tasks/{taskId}/variables/{variableName}?scope={scope}

**Table 15.120. Get a variable from a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to get a variable for. |
| variableName | Yes | String | The name of the variable to get. |
| scope | False | String | Scope of variable to be returned. When 'local', only task-local variable value is returned. When 'global', only variable value from the task's parent execution-hierarchy are returned. When the parameter is omitted, a local variable will be returned if it exists, otherwise a global variable. |

**Table 15.121. Get a variable from a task - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the task was found and the requested variables are returned. |
| 404 | Indicates the requested task was not found or the task doesn't have a variable with the given name (in the given scope). Status message provides additional information. |

**Success response body:**

{

  "name" : "myTaskVariable",

  "scope" : "local",

  "type" : "string",

  "value" : "Hello my friend"

}

Full response body description can be found in the general [REST-variables section](http://www.mossle.com/docs/activiti/#restVariables).

**Get the binary data for a variable**

GET runtime/tasks/{taskId}/variables/{variableName}/data?scope={scope}

**Table 15.122. Get the binary data for a variable - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to get a variable data for. |
| variableName | Yes | String | The name of the variable to get data for. Only variables of type binary and serializable can be used. If any other type of variable is used, a 404 is returned. |
| scope | False | String | Scope of variable to be returned. When 'local', only task-local variable value is returned. When 'global', only variable value from the task's parent execution-hierarchy are returned. When the parameter is omitted, a local variable will be returned if it exists, otherwise a global variable. |

**Table 15.123. Get the binary data for a variable - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the task was found and the requested variables are returned. |
| 404 | Indicates the requested task was not found or the task doesn't have a variable with the given name (in the given scope) or the variable doesn't have a binary stream available. Status message provides additional information. |

**Success response body:** The response body contains the binary value of the variable. When the variable is of type binary, the content-type of the response is set to application/octet-stream, regardless of the content of the variable or the request accept-type header. In case of serializable, application/x-java-serialized-object is used as content-type.

**Create new variables on a task**

POST runtime/tasks/{taskId}/variables

**Table 15.124. Create new variables on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to create the new variable for. |

**Request body for creating simple (non-binary) variables:**

[

  {

    "name" : "myTaskVariable",

    "scope" : "local",

    "type" : "string",

    "value" : "Hello my friend"

  },

  {

    ...

  }

]

The request body should be an array containing one or more JSON-objects representing the variables that should be created.

* name: Required name of the variable
* scope: Scope of variable that is created. If omitted, local is assumed.
* type: Type of variable that is created. If omitted, reverts to raw JSON-value type (string, boolean, integer or double).
* value: Variable value.

More information about the variable format can be found in [the REST variables section](http://www.mossle.com/docs/activiti/#restVariables).

**Success response body:**

[

  {

    "name" : "myTaskVariable",

    "scope" : "local",

    "type" : "string",

    "value" : "Hello my friend"

  },

  {

    ...

  }

]

**Table 15.125. Create new variables on a task - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the variables were created and the result is returned. |
| 400 | Indicates the name of a variable to create was missing or that an attempt is done to create a variable on a standalone task (without a process associated) with scope global or an empty array of variables was included in the request or request did not contain an array of variables. Status message provides additional information. |
| 404 | Indicates the requested task was not found. |
| 409 | Indicates the task already has a variable with the given name. Use the PUT method to update the task variable instead. |

**Create a new binary variable on a task**

POST runtime/tasks/{taskId}/variables

**Table 15.126. Create a new binary variable on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to create the new variable for. |

**Request body:** The request should be of type multipart/form-data. There should be a single file-part included with the binary value of the variable. On top of that, the folowing additionl form-fields can be present:

* name: Required name of the variable.
* scope: Scope of variable that is created. If omitted, local is assumed.
* type: Type of variable that is created. If omitted, binary is assumed and the binary data in the request will be stored as an array of bytes.

**Success response body:**

{

  "name" : "binaryVariable",

  "scope" : "local",

  "type" : "binary",

  "value" : null,

  "valueUrl" : "http://.../runtime/tasks/123/variables/binaryVariable/data"

}

**Table 15.127. Create a new binary variable on a task - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the variable was created and the result is returned. |
| 400 | Indicates the name of the variable to create was missing or that an attempt is done to create a variable on a standalone task (without a process associated) with scope global. Status message provides additional information. |
| 404 | Indicates the requested task was not found. |
| 409 | Indicates the task already has a variable with the given name. Use the PUT method to update the task variable instead. |
| 415 | Indicates the serializable data contains an object for which no class is present in the JVM running the Activiti engine and therefore cannot be deserialized. |

**Update an existing variable on a task**

PUT runtime/tasks/{taskId}/variables/{variableName}

**Table 15.128. Update an existing variable on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to update the variable for. |
| variableName | Yes | String | The name of the variable to update. |

**Request body for updating simple (non-binary) variables:**

{

  "name" : "myTaskVariable",

  "scope" : "local",

  "type" : "string",

  "value" : "Hello my friend"

}

* name: Required name of the variable
* scope: Scope of variable that is updated. If omitted, local is assumed.
* type: Type of variable that is updated. If omitted, reverts to raw JSON-value type (string, boolean, integer or double).
* value: Variable value.

More information about the variable format can be found in [the REST variables section](http://www.mossle.com/docs/activiti/#restVariables).

**Success response body:**

{

  "name" : "myTaskVariable",

  "scope" : "local",

  "type" : "string",

  "value" : "Hello my friend"

}

**Table 15.129. Update an existing variable on a task - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the variables was updated and the result is returned. |
| 400 | Indicates the name of a variable to update was missing or that an attempt is done to update a variable on a standalone task (without a process associated) with scope global. Status message provides additional information. |
| 404 | Indicates the requested task was not found or the task doesn't have a variable with the given name in the given scope. Status message contains additional information about the error. |

**Updating a binary variable on a task**

PUT runtime/tasks/{taskId}/variables/{variableName}

**Table 15.130. Updating a binary variable on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to update the variable for. |
| variableName | Yes | String | The name of the variable to update. |

**Request body:** The request should be of type multipart/form-data. There should be a single file-part included with the binary value of the variable. On top of that, the folowing additionl form-fields can be present:

* name: Required name of the variable.
* scope: Scope of variable that is updated. If omitted, local is assumed.
* type: Type of variable that is updated. If omitted, binary is assumed and the binary data in the request will be stored as an array of bytes.

**Success response body:**

{

  "name" : "binaryVariable",

  "scope" : "local",

  "type" : "binary",

  "value" : null,

  "valueUrl" : "http://.../runtime/tasks/123/variables/binaryVariable/data"

}

**Table 15.131. Updating a binary variable on a task - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the variable was updated and the result is returned. |
| 400 | Indicates the name of the variable to update was missing or that an attempt is done to update a variable on a standalone task (without a process associated) with scope global. Status message provides additional information. |
| 404 | Indicates the requested task was not found or the variable to update doesn't exist for the given task in the given scope. |
| 415 | Indicates the serializable data contains an object for which no class is present in the JVM running the Activiti engine and therefore cannot be deserialized. |

**Delete a variable on a task**

DELETE runtime/tasks/{taskId}/variables/{variableName}?scope={scope}

**Table 15.132. Delete a variable on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task the variable to delete belongs to. |
| variableName | Yes | String | The name of the variable to delete. |
| scope | No | String | Scope of variable to delete in. Can be either local or global. If omitted, local is assumed. |

**Table 15.133. Delete a variable on a task - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the task variable was found and has been deleted. Response-body is intentionally empty. |
| 404 | Indicates the requested task was not found or the task doesn't have a variable with the given name. Status message contains additional information about the error. |

**Delete all local variables on a task**

DELETE runtime/tasks/{taskId}/variables

**Table 15.134. Delete all local variables on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task the variable to delete belongs to. |

**Table 15.135. Delete all local variables on a task - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates all local task variables have been deleted. Response-body is intentionally empty. |
| 404 | Indicates the requested task was not found. |

**Get all identity links for a task**

GET runtime/tasks/{taskId}/identitylinks

**Table 15.136. Get all identity links for a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to get the identity links for. |

**Table 15.137. Get all identity links for a task - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the task was found and the requested idenity links are returned. |
| 404 | Indicates the requested task was not found. |

**Success response body:**

[

  {

    "userId" : "kermit",

    "groupId" : null,

    "type" : "candidate",

    "url" : "http://localhost:8081/activiti-rest/service/runtime/tasks/100/identitylinks/users/kermit/candidate"

  },

  {

    "userId" : null,

    "groupId" : "sales",

    "type" : "candidate",

    "url" : "http://localhost:8081/activiti-rest/service/runtime/tasks/100/identitylinks/groups/sales/candidate"

  },

  ...

]

**Get all identitylinks for a task for either groups or users**

GET runtime/tasks/{taskId}/identitylinks/users

GET runtime/tasks/{taskId}/identitylinks/groups

Returns only identity links targetting either users or groups. Response body and status-codes are exactly the same as when getting the full list of identity links for a task.

**Get a single identity link on a task**

GET runtime/tasks/{taskId}/identitylinks/{family}/{identityId}/{type}

**Table 15.138. Get all identitylinks for a task for either groups or users - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task . |
| family | Yes | String | Either groups or users, depending on what kind of identity is targetted. |
| identityId | Yes | String | The id of the identity. |
| type | Yes | String | The type of identity link. |

**Table 15.139. Get all identitylinks for a task for either groups or users - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the task and identity link was found and returned. |
| 404 | Indicates the requested task was not found or the task doesn't have the requested identityLink. The status contains additional information about this error. |

**Success response body:**

{

  "userId" : null,

  "groupId" : "sales",

  "type" : "candidate",

  "url" : "http://localhost:8081/activiti-rest/service/runtime/tasks/100/identitylinks/groups/sales/candidate"

}

**Create an identity link on a task**

POST runtime/tasks/{taskId}/identitylinks

**Table 15.140. Create an identity link on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task . |

**Request body (user):**

{

  "userId" : "kermit",

  "type" : "candidate",

}

**Request body (group):**

{

  "groupId" : "sales",

  "type" : "candidate",

}

**Table 15.141. Create an identity link on a task - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the task was found and the identity link was created. |
| 404 | Indicates the requested task was not found or the task doesn't have the requested identityLink. The status contains additional information about this error. |

**Success response body:**

{

  "userId" : null,

  "groupId" : "sales",

  "type" : "candidate",

  "url" : "http://localhost:8081/activiti-rest/service/runtime/tasks/100/identitylinks/groups/sales/candidate"

}

**Delete an identity link on a task**

DELETE runtime/tasks/{taskId}/identitylinks/{family}/{identityId}/{type}

**Table 15.142. Delete an identity link on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task. |
| family | Yes | String | Either groups or users, depending on what kind of identity is targetted. |
| identityId | Yes | String | The id of the identity. |
| type | Yes | String | The type of identity link. |

**Table 15.143. Delete an identity link on a task - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the task and identity link were found and the link has been deleted. Response-body is intentionally empty. |
| 404 | Indicates the requested task was not found or the task doesn't have the requested identityLink. The status contains additional information about this error. |

**Create a new comment on a task**

POST runtime/tasks/{taskId}/comments

**Table 15.144. Create a new comment on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to create the comment for. |

**Request body:**

{

  "message" : "This is a comment on the task."

}

**Success response body:**

{

  "id" : "123",

  "url" : "http://localhost:8081/activiti-rest/service/runtime/tasks/100/comments/123",

  "message" : "This is a comment on the task.",

  "author" : "kermit"

}

**Table 15.145. Create a new comment on a task - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the comment was created and the result is returned. |
| 400 | Indicates the comment is missing from the request. |
| 404 | Indicates the requested task was not found. |

**Get all comments on a task**

GET runtime/tasks/{taskId}/comments

**Table 15.146. Get all comments on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to get the comments for. |

**Success response body:**

[

  {

    "id" : "123",

    "url" : "http://localhost:8081/activiti-rest/service/runtime/tasks/100/comments/123",

    "message" : "This is a comment on the task.",

    "author" : "kermit"

  },

  {

    "id" : "456",

    "url" : "http://localhost:8081/activiti-rest/service/runtime/tasks/100/comments/456",

    "message" : "This is another comment on the task.",

    "author" : "gonzo"

  }

]

**Table 15.147. Get all comments on a task - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the task was found and the comments are returned. |
| 404 | Indicates the requested task was not found. |

**Get a comment on a task**

GET runtime/tasks/{taskId}/comments/{commentId}

**Table 15.148. Get a comment on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to get the comment for. |
| commentId | Yes | String | The id of the comment. |

**Success response body:**

{

  "id" : "123",

  "url" : "http://localhost:8081/activiti-rest/service/runtime/tasks/100/comments/123",

  "message" : "This is a comment on the task.",

  "author" : "kermit"

}

**Table 15.149. Get a comment on a task - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the task and comment were found and the comment is returned. |
| 404 | Indicates the requested task was not found or the tasks doesn't have a comment with the given ID. |

**Delete a comment on a task**

DELETE runtime/tasks/{taskId}/comments/{commentId}

**Table 15.150. Delete a comment on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to delete the comment for. |
| commentId | Yes | String | The id of the comment. |

**Table 15.151. Delete a comment on a task - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the task and comment were found and the comment is deleted. Response body is left empty intentionally. |
| 404 | Indicates the requested task was not found or the tasks doesn't have a comment with the given ID. |

**Get all events for a task**

GET runtime/tasks/{taskId}/events

**Table 15.152. Get all events for a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to get the events for. |

**Success response body:**

[

  {

    "action" : "AddUserLink",

    "id" : "4",

    "message" : [ "gonzo", "contributor" ],

    "taskUrl" : "http://localhost:8182/runtime/tasks/2",

    "time" : "2013-05-17T11:50:50.000+0000",

    "url" : "http://localhost:8182/runtime/tasks/2/events/4",

    "userId" : null

  },

  ...

]

**Table 15.153. Get all events for a task - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the task was found and the events are returned. |
| 404 | Indicates the requested task was not found. |

**Get an event on a task**

GET runtime/tasks/{taskId}/events/{eventId}

**Table 15.154. Get an event on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to get the event for. |
| eventId | Yes | String | The id of the event. |

**Success response body:**

{

  "action" : "AddUserLink",

  "id" : "4",

  "message" : [ "gonzo", "contributor" ],

  "taskUrl" : "http://localhost:8182/runtime/tasks/2",

  "time" : "2013-05-17T11:50:50.000+0000",

  "url" : "http://localhost:8182/runtime/tasks/2/events/4",

  "userId" : null

}

**Table 15.155. Get an event on a task - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the task and event were found and the event is returned. |
| 404 | Indicates the requested task was not found or the tasks doesn't have an event with the given ID. |

**Create a new attachment on a task, containing a link to an external resource**

POST runtime/tasks/{taskId}/attachments

**Table 15.156. Create a new attachment on a task, containing a link to an external resource - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to create the attachment for. |

**Request body:**

{

  "name":"Simple attachment",

  "description":"Simple attachment description",

  "type":"simpleType",

  "externalUrl":"http://activiti.org"

}

Only the attachment name is required to create a new attachment.

**Success response body:**

{

  "id":"3",

  "url":"http://localhost:8182/runtime/tasks/2/attachments/3",

  "name":"Simple attachment",

  "description":"Simple attachment description",

  "type":"simpleType",

  "taskUrl":"http://localhost:8182/runtime/tasks/2",

  "processInstanceUrl":null,

  "externalUrl":"http://activiti.org",

  "contentUrl":null

}

**Table 15.157. Create a new attachment on a task, containing a link to an external resource - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the attachment was created and the result is returned. |
| 400 | Indicates the attachment name is missing from the request. |
| 404 | Indicates the requested task was not found. |

**Create a new attachment on a task, with an attached file**

POST runtime/tasks/{taskId}/attachments

**Table 15.158. Create a new attachment on a task, with an attached file - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to create the attachment for. |

**Request body:** The request should be of type multipart/form-data. There should be a single file-part included with the binary value of the variable. On top of that, the folowing additionl form-fields can be present:

* name: Required name of the variable.
* description: Description of the attachment, optional.
* type: Type of attachment, optional. Supports any arbitrary string or a valid HTTP content-type.

**Success response body:**

{

      "id":"5",

      "url":"http://localhost:8182/runtime/tasks/2/attachments/5",

      "name":"Binary attachment",

      "description":"Binary attachment description",

      "type":"binaryType",

      "taskUrl":"http://localhost:8182/runtime/tasks/2",

      "processInstanceUrl":null,

      "externalUrl":null,

      "contentUrl":"http://localhost:8182/runtime/tasks/2/attachments/5/content"

   }

**Table 15.159. Create a new attachment on a task, with an attached file - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the attachment was created and the result is returned. |
| 400 | Indicates the attachment name is missing from the request or no file was present in the request. The error-message contains additional information. |
| 404 | Indicates the requested task was not found. |

**Get all attachments on a task**

GET runtime/tasks/{taskId}/attachments

**Table 15.160. Get all attachments on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to get the attachments for. |

**Success response body:**

[

  {

    "id":"3",

    "url":"http://localhost:8182/runtime/tasks/2/attachments/3",

    "name":"Simple attachment",

    "description":"Simple attachment description",

    "type":"simpleType",

    "taskUrl":"http://localhost:8182/runtime/tasks/2",

    "processInstanceUrl":null,

    "externalUrl":"http://activiti.org",

    "contentUrl":null

  },

  {

    "id":"5",

    "url":"http://localhost:8182/runtime/tasks/2/attachments/5",

    "name":"Binary attachment",

    "description":"Binary attachment description",

    "type":"binaryType",

    "taskUrl":"http://localhost:8182/runtime/tasks/2",

    "processInstanceUrl":null,

    "externalUrl":null,

    "contentUrl":"http://localhost:8182/runtime/tasks/2/attachments/5/content"

  }

]

**Table 15.161. Get all attachments on a task - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the task was found and the attachments are returned. |
| 404 | Indicates the requested task was not found. |

**Get an attachment on a task**

GET runtime/tasks/{taskId}/attachments/{attachmentId}

**Table 15.162. Get an attachment on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to get the attachment for. |
| attachmentId | Yes | String | The id of the attachment. |

**Success response body:**

  {

    "id":"5",

    "url":"http://localhost:8182/runtime/tasks/2/attachments/5",

    "name":"Binary attachment",

    "description":"Binary attachment description",

    "type":"binaryType",

    "taskUrl":"http://localhost:8182/runtime/tasks/2",

    "processInstanceUrl":null,

    "externalUrl":null,

    "contentUrl":"http://localhost:8182/runtime/tasks/2/attachments/5/content"

  }

* externalUrl - contentUrl:In case the attachment is a link to an external resource, the externalUrl contains the URL to the external content. If the attachment content is present in the Activiti engine, the contentUrl will contain an URL where the binary content can be streamed from.
* type:Can be any arbitrary value. When a valid formatted media-type (eg. application/xml, text/plain) is included, the binary content HTTP response content-type will be set the the given value.

**Table 15.163. Get an attachment on a task - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the task and attachment were found and the attachment is returned. |
| 404 | Indicates the requested task was not found or the tasks doesn't have a attachment with the given ID. |

**Get the content for an attachment**

GET runtime/tasks/{taskId}/attachment/{attachmentId}/content

**Table 15.164. Get the content for an attachment - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to get a variable data for. |
| attachmentId | Yes | String | The id of the attachment, a 404 is returned when the attachment points to an external URL rather than content attached in Activiti. |

**Table 15.165. Get the content for an attachment - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the task and attachment was found and the requested content is returned. |
| 404 | Indicates the requested task was not found or the task doesn't have an attachment with the given id or the attachment doesn't have a binary stream available. Status message provides additional information. |

**Success response body:** The response body contains the binary content. By default, the content-type of the response is set to application/octet-stream unless the attachment type contains a valid Content-type.

**Delete an attachment on a task**

DELETE runtime/tasks/{taskId}/attachments/{attachmentId}

**Table 15.166. Delete an attachment on a task - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes | String | The id of the task to delete the attachment for. |
| attachmentId | Yes | String | The id of the attachment. |

**Table 15.167. Delete an attachment on a task - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the task and attachment were found and the attachment is deleted. Response body is left empty intentionally. |
| 404 | Indicates the requested task was not found or the tasks doesn't have a attachment with the given ID. |

**History**

**Get a historic process instance**

GET history/historic-process-instances/{processInstanceId}

**Table 15.168. Get a historic process instance - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates that the historic process instances could be found. |
| 404 | Indicates that the historic process instances could not be found. |

**Success response body:**

{

  "data": [

    {

      "id" : "5",

      "businessKey" : "myKey",

      "processDefinitionId" : "oneTaskProcess%3A1%3A4",

      "processDefinitionUrl" : "http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4",

      "startTime" : "2013-04-17T10:17:43.902+0000",

      "endTime" : "2013-04-18T14:06:32.715+0000",

      "durationInMillis" : 86400056,

      "startUserId" : "kermit",

      "startActivityId" : "startEvent",

      "endActivityId" : "endEvent",

      "deleteReason" : null,

      "superProcessInstanceId" : "3",

      "url" : "http://localhost:8182/history/historic-process-instances/5",

      "variables": null

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "name",

  "order": "asc",

  "size": 1

}

**List of historic process instances**

GET history/historic-process-instances

**Table 15.169. List of historic process instances - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processInstanceId | No | String | An id of the historic process instance. |
| processDefinitionKey | No | String | The process definition key of the historic process instance. |
| processDefinitionId | No | String | The process definition id of the historic process instance. |
| businessKey | No | String | The business key of the historic process instance. |
| involvedUser | No | String | An involved user of the historic process instance. |
| finished | No | Boolean | Indication if the historic process instance is finished. |
| superProcessInstanceId | No | String | An optional parent process id of the historic process instance. |
| excludeSubprocesses | No | Boolean | Return only historic process instances which aren't sub processes. |
| finishedAfter | No | Date | Return only historic process instances that were finished after this date. |
| finishedBefore | No | Date | Return only historic process instances that were finished before this date. |
| startedAfter | No | Date | Return only historic process instances that were started after this date. |
| startedBefore | No | Date | Return only historic process instances that were started before this date. |
| startedBy | No | String | Return only historic process instances that were started by this user. |
| includeProcessVariables | No | Boolean | An indication if the historic process instance variables should be returned as well. |
| The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL. |  |  |  |

**Table 15.170. List of historic process instances - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates that historic process instances could be queried. |
| 400 | Indicates an parameter was passed in the wrong format. The status-message contains additional information. |

**Success response body:**

{

  "data": [

    {

      "id" : "5",

      "businessKey" : "myKey",

      "processDefinitionId" : "oneTaskProcess%3A1%3A4",

      "processDefinitionUrl" : "http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4",

      "startTime" : "2013-04-17T10:17:43.902+0000",

      "endTime" : "2013-04-18T14:06:32.715+0000",

      "durationInMillis" : 86400056,

      "startUserId" : "kermit",

      "startActivityId" : "startEvent",

      "endActivityId" : "endEvent",

      "deleteReason" : null,

      "superProcessInstanceId" : "3",

      "url" : "http://localhost:8182/history/historic-process-instances/5",

      "variables": [

        {

          "name": "test",

          "variableScope": "local",

          "value": "myTest"

        }

      ]

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "name",

  "order": "asc",

  "size": 1

}

**Query for historic process instances**

POST query/historic-process-instances

**Request body:**

{

  "processDefinitionId" : "oneTaskProcess%3A1%3A4",

  ...

  "variables" : [

    {

      "name" : "myVariable",

      "value" : 1234,

      "operator" : "equals",

      "type" : "long"

    }

  ]

}

All supported JSON parameter fields allowed are exactly the same as the parameters found for [getting a collection of historic process instances](http://www.mossle.com/docs/activiti/#restHistoricProcessInstancesGet), but passed in as JSON-body arguments rather than URL-parameters to allow for more advanced querying and preventing errors with request-uri's that are too long. On top of that, the query allows for filtering based on process variables. The variables property is a json-array containing objects with the format [as described here.](http://www.mossle.com/docs/activiti/#restQueryVariable)

**Table 15.171. Query for historic process instances - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the tasks are returned |
| 400 | Indicates an parameter was passed in the wrong format. The status-message contains additional information. |

**Success response body:**

{

  "data": [

    {

      "id" : "5",

      "businessKey" : "myKey",

      "processDefinitionId" : "oneTaskProcess%3A1%3A4",

      "processDefinitionUrl" : "http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4",

      "startTime" : "2013-04-17T10:17:43.902+0000",

      "endTime" : "2013-04-18T14:06:32.715+0000",

      "durationInMillis" : 86400056,

      "startUserId" : "kermit",

      "startActivityId" : "startEvent",

      "endActivityId" : "endEvent",

      "deleteReason" : null,

      "superProcessInstanceId" : "3",

      "url" : "http://localhost:8182/history/historic-process-instances/5",

      "variables": [

        {

          "name": "test",

          "variableScope": "local",

          "value": "myTest"

        }

      ]

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "name",

  "order": "asc",

  "size": 1

}

**Delete a historic process instance**

DELETE history/historic-process-instances/{processInstanceId}

**Table 15.172. Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates that the historic process instance was deleted. |
| 404 | Indicates that the historic process instance could not be found. |

**Get the identity links of a historic process instance**

GET history/historic-process-instance/{processInstanceId}/identitylinks

**Table 15.173. Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the identity links are returned |
| 404 | Indicates the process instance could not be found. |

**Success response body:**

[

 {

  "type" : "participant",

  "userId" : "kermit",

  "groupId" : null,

  "taskId" : null,

  "taskUrl" : null,

  "processInstanceId" : "5",

  "processInstanceUrl" : "http://localhost:8182/history/historic-process-instances/5"

 }

]

**Get the binary data for a historic process instance variable**

GET history/historic-process-instances/{processInstanceId}/variables/{variableName}/data

**Table 15.174. Get the binary data for a historic process instance variable - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the process instance was found and the requested variable data is returned. |
| 404 | Indicates the requested process instance was not found or the process instance doesn't have a variable with the given name or the variable doesn't have a binary stream available. Status message provides additional information. |

**Success response body:** The response body contains the binary value of the variable. When the variable is of type binary, the content-type of the response is set to application/octet-stream, regardless of the content of the variable or the request accept-type header. In case of serializable, application/x-java-serialized-object is used as content-type.

**Get a single historic task instance**

GET history/historic-task-instances/{taskId}

**Table 15.175. Get a single historic task instance - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates that the historic task instances could be found. |
| 404 | Indicates that the historic task instances could not be found. |

**Success response body:**

{

  "id" : "5",

  "processDefinitionId" : "oneTaskProcess%3A1%3A4",

  "processDefinitionUrl" : "http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4",

  "processInstanceId" : "3",

  "processInstanceUrl" : "http://localhost:8182/history/historic-process-instances/3",

  "executionId" : "4",

  "name" : "My task name",

  "description" : "My task description",

  "deleteReason" : null,

  "owner" : "kermit",

  "assignee" : "fozzie",

  "startTime" : "2013-04-17T10:17:43.902+0000",

  "endTime" : "2013-04-18T14:06:32.715+0000",

  "durationInMillis" : 86400056,

  "workTimeInMillis" : 234890,

  "claimTime" : "2013-04-18T11:01:54.715+0000",

  "taskDefinitionKey" : "taskKey",

  "formKey" : null,

  "priority" : 50,

  "dueDate" : "2013-04-20T12:11:13.134+0000",

  "parentTaskId" : null,

  "url" : "http://localhost:8182/history/historic-task-instances/5",

  "variables": null

}

**Get historic task instances**

GET history/historic-task-instances

**Table 15.176. Get historic task instances - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | No | String | An id of the historic task instance. |
| processInstanceId | No | String | The process instance id of the historic task instance. |
| processDefinitionKey | No | String | The process definition key of the historic task instance. |
| processDefinitionId | No | String | The process definition id of the historic task instance. |
| processDefinitionName | No | String | The process definition name of the historic task instance. |
| processBusinessKey | No | String | The process instance business key of the historic task instance. |
| executionId | No | String | The execution id of the historic task instance. |
| taskName | No | String | The task name of the historic task instance. |
| taskNameLike | No | String | The task name with 'like' operator for the historic task instance. |
| taskDescription | No | String | The task description of the historic task instance. |
| taskDescriptionLike | No | String | The task description with 'like' operator for the historic task instance. |
| taskDefinitionKey | No | String | The task identifier from the process definition for the historic task instance. |
| taskDeleteReason | No | String | The task delete reason of the historic task instance. |
| taskDeleteReasonLike | No | String | The task delete reason with 'like' operator for the historic task instance. |
| taskAssignee | No | String | The assignee of the historic task instance. |
| taskAssigneeLike | No | String | The assignee with 'like' operator for the historic task instance. |
| taskOwner | No | String | The owner of the historic task instance. |
| taskOwnerLike | No | String | The owner with 'like' operator for the historic task instance. |
| taskInvolvedUser | No | String | An involved user of the historic task instance. |
| taskPriority | No | String | The priority of the historic task instance. |
| finished | No | Boolean | Indication if the historic task instance is finished. |
| processFinished | No | Boolean | Indication if the process instance of the historic task instance is finished. |
| parentTaskId | No | String | An optional parent task id of the historic task instance. |
| dueDate | No | Date | Return only historic task instances that have a due date equal this date. |
| dueDateAfter | No | Date | Return only historic task instances that have a due date after this date. |
| dueDateBefore | No | Date | Return only historic task instances that have a due date before this date. |
| taskCreatedOn | No | Date | Return only historic task instances that were created on this date. |
| includeTaskLocalVariables | No | Boolean | An indication if the historic task instance local variables should be returned as well. |
| includeProcessVariables | No | Boolean | An indication if the historic task instance global variables should be returned as well. |
| The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL. |  |  |  |

**Table 15.177. Get historic task instances - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates that historic process instances could be queried. |
| 400 | Indicates an parameter was passed in the wrong format. The status-message contains additional information. |

**Success response body:**

{

  "data": [

    {

      "id" : "5",

      "processDefinitionId" : "oneTaskProcess%3A1%3A4",

      "processDefinitionUrl" : "http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4",

      "processInstanceId" : "3",

      "processInstanceUrl" : "http://localhost:8182/history/historic-process-instances/3",

      "executionId" : "4",

      "name" : "My task name",

      "description" : "My task description",

      "deleteReason" : null,

      "owner" : "kermit",

      "assignee" : "fozzie",

      "startTime" : "2013-04-17T10:17:43.902+0000",

      "endTime" : "2013-04-18T14:06:32.715+0000",

      "durationInMillis" : 86400056,

      "workTimeInMillis" : 234890,

      "claimTime" : "2013-04-18T11:01:54.715+0000",

      "taskDefinitionKey" : "taskKey",

      "formKey" : null,

      "priority" : 50,

      "dueDate" : "2013-04-20T12:11:13.134+0000",

      "parentTaskId" : null,

      "url" : "http://localhost:8182/history/historic-task-instances/5",

      "taskVariables": [

        {

          "name": "test",

          "variableScope": "local",

          "value": "myTest"

        }

      ],

      "processVariables": [

        {

          "name": "processTest",

          "variableScope": "global",

          "value": "myProcessTest"

        }

      ]

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "name",

  "order": "asc",

  "size": 1

}

**Query for historic task instances**

POST query/historic-task-instances

**Query for historic task instances - Request body:**

{

  "processDefinitionId" : "oneTaskProcess%3A1%3A4",

  ...

  "variables" : [

    {

      "name" : "myVariable",

      "value" : 1234,

      "operator" : "equals",

      "type" : "long"

    }

  ]

}

All supported JSON parameter fields allowed are exactly the same as the parameters found for [getting a collection of historic task instances](http://www.mossle.com/docs/activiti/#restHistoricTaskInstancesGet), but passed in as JSON-body arguments rather than URL-parameters to allow for more advanced querying and preventing errors with request-uri's that are too long. On top of that, the query allows for filtering based on process variables. The taskVariables and processVariables properties are json-arrays containing objects with the format [as described here.](http://www.mossle.com/docs/activiti/#restQueryVariable)

**Table 15.178. Query for historic task instances - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the tasks are returned |
| 400 | Indicates an parameter was passed in the wrong format. The status-message contains additional information. |

**Success response body:**

{

  "data": [

    {

      "id" : "5",

      "processDefinitionId" : "oneTaskProcess%3A1%3A4",

      "processDefinitionUrl" : "http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4",

      "processInstanceId" : "3",

      "processInstanceUrl" : "http://localhost:8182/history/historic-process-instances/3",

      "executionId" : "4",

      "name" : "My task name",

      "description" : "My task description",

      "deleteReason" : null,

      "owner" : "kermit",

      "assignee" : "fozzie",

      "startTime" : "2013-04-17T10:17:43.902+0000",

      "endTime" : "2013-04-18T14:06:32.715+0000",

      "durationInMillis" : 86400056,

      "workTimeInMillis" : 234890,

      "claimTime" : "2013-04-18T11:01:54.715+0000",

      "taskDefinitionKey" : "taskKey",

      "formKey" : null,

      "priority" : 50,

      "dueDate" : "2013-04-20T12:11:13.134+0000",

      "parentTaskId" : null,

      "url" : "http://localhost:8182/history/historic-task-instances/5",

      "taskVariables": [

        {

          "name": "test",

          "variableScope": "local",

          "value": "myTest"

        }

      ],

      "processVariables": [

        {

          "name": "processTest",

          "variableScope": "global",

          "value": "myProcessTest"

        }

      ]

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "name",

  "order": "asc",

  "size": 1

}

**Delete a historic task instance**

DELETE history/historic-task-instances/{taskId}

**Table 15.179. Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates that the historic task instance was deleted. |
| 404 | Indicates that the historic task instance could not be found. |

**Get the identity links of a historic task instance**

GET history/historic-task-instance/{taskId}/identitylinks

**Table 15.180. Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the identity links are returned |
| 404 | Indicates the task instance could not be found. |

**Success response body:**

[

 {

  "type" : "assignee",

  "userId" : "kermit",

  "groupId" : null,

  "taskId" : "6",

  "taskUrl" : "http://localhost:8182/history/historic-task-instances/5",

  "processInstanceId" : null,

  "processInstanceUrl" : null

 }

]

**Get the binary data for a historic task instance variable**

GET history/historic-task-instances/{taskId}/variables/{variableName}/data

**Table 15.181. Get the binary data for a historic task instance variable - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the task instance was found and the requested variable data is returned. |
| 404 | Indicates the requested task instance was not found or the process instance doesn't have a variable with the given name or the variable doesn't have a binary stream available. Status message provides additional information. |

**Success response body:** The response body contains the binary value of the variable. When the variable is of type binary, the content-type of the response is set to application/octet-stream, regardless of the content of the variable or the request accept-type header. In case of serializable, application/x-java-serialized-object is used as content-type.

**Get historic activity instances**

GET history/historic-activity-instances

**Table 15.182. Get historic activity instances - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| activityId | No | String | An id of the activity instance. |
| activityInstanceId | No | String | An id of the historic activity instance. |
| activityName | No | String | The name of the historic activity instance. |
| activityType | No | String | The element type of the historic activity instance. |
| executionId | No | String | The execution id of the historic activity instance. |
| finished | No | Boolean | Indication if the historic activity instance is finished. |
| taskAssignee | No | String | The assignee of the historic activity instance. |
| processInstanceId | No | String | The process instance id of the historic activity instance. |
| processDefinitionId | No | String | The process definition id of the historic activity instance. |
| The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL. |  |  |  |

**Table 15.183. Get historic activity instances - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates that historic activity instances could be queried. |
| 400 | Indicates an parameter was passed in the wrong format. The status-message contains additional information. |

**Success response body:**

{

  "data": [

    {

      "id" : "5",

      "activityId" : "4",

      "activityName" : "My user task",

      "activityType" : "userTask",

      "processDefinitionId" : "oneTaskProcess%3A1%3A4",

      "processDefinitionUrl" : "http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4",

      "processInstanceId" : "3",

      "processInstanceUrl" : "http://localhost:8182/history/historic-process-instances/3",

      "executionId" : "4",

      "taskId" : "4",

      "calledProcessInstanceId" : null,

      "assignee" : "fozzie",

      "startTime" : "2013-04-17T10:17:43.902+0000",

      "endTime" : "2013-04-18T14:06:32.715+0000",

      "durationInMillis" : 86400056

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "name",

  "order": "asc",

  "size": 1

}

**Query for historic activity instances**

POST query/historic-activity-instances

**Request body:**

{

  "processDefinitionId" : "oneTaskProcess%3A1%3A4"

}

All supported JSON parameter fields allowed are exactly the same as the parameters found for [getting a collection of historic task instances](http://www.mossle.com/docs/activiti/#restHistoricTaskInstancesGet), but passed in as JSON-body arguments rather than URL-parameters to allow for more advanced querying and preventing errors with request-uri's that are too long.

**Table 15.184. Query for historic activity instances - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the activities are returned |
| 400 | Indicates an parameter was passed in the wrong format. The status-message contains additional information. |

**Success response body:**

{

  "data": [

    {

      "id" : "5",

      "activityId" : "4",

      "activityName" : "My user task",

      "activityType" : "userTask",

      "processDefinitionId" : "oneTaskProcess%3A1%3A4",

      "processDefinitionUrl" : "http://localhost:8182/repository/process-definitions/oneTaskProcess%3A1%3A4",

      "processInstanceId" : "3",

      "processInstanceUrl" : "http://localhost:8182/history/historic-process-instances/3",

      "executionId" : "4",

      "taskId" : "4",

      "calledProcessInstanceId" : null,

      "assignee" : "fozzie",

      "startTime" : "2013-04-17T10:17:43.902+0000",

      "endTime" : "2013-04-18T14:06:32.715+0000",

      "durationInMillis" : 86400056

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "name",

  "order": "asc",

  "size": 1

}

**List of historic variable instances**

GET history/historic-variable-instances

**Table 15.185. List of historic variable instances - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| processInstanceId | No | String | The process instance id of the historic variable instance. |
| taskId | No | String | The task id of the historic variable instance. |
| excludeTaskVariables | No | Boolean | Indication to exclude the task variables from the result. |
| variableName | No | String | The variable name of the historic variable instance. |
| variableNameLike | No | String | The variable name using the 'like' operator for the historic variable instance. |
| The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL. |  |  |  |

**Table 15.186. List of historic variable instances - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates that historic variable instances could be queried. |
| 400 | Indicates an parameter was passed in the wrong format. The status-message contains additional information. |

**Success response body:**

{

  "data": [

    {

      "id" : "14",

      "processInstanceId" : "5",

      "processInstanceUrl" : "http://localhost:8182/history/historic-process-instances/5",

      "taskId" : "6",

      "variable" : {

        "name" : "myVariable",

        "variableScope", "global",

        "value" : "test"

      }

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "name",

  "order": "asc",

  "size": 1

}

**Query for historic variable instances**

POST query/historic-variable-instances

**Request body:**

{

  "processDefinitionId" : "oneTaskProcess%3A1%3A4",

  ...

  "variables" : [

    {

      "name" : "myVariable",

      "value" : 1234,

      "operator" : "equals",

      "type" : "long"

    }

  ]

}

All supported JSON parameter fields allowed are exactly the same as the parameters found for [getting a collection of historic process instances](http://www.mossle.com/docs/activiti/#restHistoricVariableInstancesGet), but passed in as JSON-body arguments rather than URL-parameters to allow for more advanced querying and preventing errors with request-uri's that are too long. On top of that, the query allows for filtering based on process variables. The variables property is a json-array containing objects with the format [as described here.](http://www.mossle.com/docs/activiti/#restQueryVariable)

**Table 15.187. Query for historic variable instances - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the tasks are returned |
| 400 | Indicates an parameter was passed in the wrong format. The status-message contains additional information. |

**Success response body:**

{

  "data": [

    {

      "id" : "14",

      "processInstanceId" : "5",

      "processInstanceUrl" : "http://localhost:8182/history/historic-process-instances/5",

      "taskId" : "6",

      "variable" : {

        "name" : "myVariable",

        "variableScope", "global",

        "value" : "test"

      }

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "name",

  "order": "asc",

  "size": 1

}

**Get the binary data for a historic task instance variable**

GET history/historic-variable-instances/{varInstanceId}/data

**Table 15.188. Get the binary data for a historic task instance variable - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the variable instance was found and the requested variable data is returned. |
| 404 | Indicates the requested variable instance was not found or the variable instance doesn't have a variable with the given name or the variable doesn't have a binary stream available. Status message provides additional information. |

**Success response body:** The response body contains the binary value of the variable. When the variable is of type binary, the content-type of the response is set to application/octet-stream, regardless of the content of the variable or the request accept-type header. In case of serializable, application/x-java-serialized-object is used as content-type.

**Get historic detail**

GET history/historic-detail

**Table 15.189. Get historic detail - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| id | No | String | The id of the historic detail. |
| processInstanceId | No | String | The process instance id of the historic detail. |
| executionId | No | String | The execution id of the historic detail. |
| activityInstanceId | No | String | The activity instance id of the historic detail. |
| taskId | No | String | The task id of the historic detail. |
| selectOnlyFormProperties | No | Boolean | Indication to only return form properties in the result. |
| selectOnlyVariableUpdates | No | Boolean | Indication to only return variable updates in the result. |
| The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL. |  |  |  |

**Table 15.190. Get historic detail - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates that historic detail could be queried. |
| 400 | Indicates an parameter was passed in the wrong format. The status-message contains additional information. |

**Success response body:**

{

  "data": [

    {

      "id" : "26",

      "processInstanceId" : "5",

      "processInstanceUrl" : "http://localhost:8182/history/historic-process-instances/5",

      "executionId" : "6",

      "activityInstanceId", "10",

      "taskId" : "6",

      "taskUrl" : "http://localhost:8182/history/historic-task-instances/6",

      "time" : "2013-04-17T10:17:43.902+0000",

      "detailType" : "variableUpdate",

      "revision" : 2,

      "variable" : {

        "name" : "myVariable",

        "variableScope", "global",

        "value" : "test"

      },

      "propertyId", null,

      "propertyValue", null

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "name",

  "order": "asc",

  "size": 1

}

**Query for historic details**

POST query/historic-detail

**Request body:**

{

  "processInstanceId" : "5",

}

All supported JSON parameter fields allowed are exactly the same as the parameters found for [getting a collection of historic process instances](http://www.mossle.com/docs/activiti/#restHistoricDetailGet), but passed in as JSON-body arguments rather than URL-parameters to allow for more advanced querying and preventing errors with request-uri's that are too long.

**Table 15.191. Query for historic details - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the historic details are returned |
| 400 | Indicates an parameter was passed in the wrong format. The status-message contains additional information. |

**Success response body:**

{

  "data": [

    {

      "id" : "26",

      "processInstanceId" : "5",

      "processInstanceUrl" : "http://localhost:8182/history/historic-process-instances/5",

      "executionId" : "6",

      "activityInstanceId", "10",

      "taskId" : "6",

      "taskUrl" : "http://localhost:8182/history/historic-task-instances/6",

      "time" : "2013-04-17T10:17:43.902+0000",

      "detailType" : "variableUpdate",

      "revision" : 2,

      "variable" : {

        "name" : "myVariable",

        "variableScope", "global",

        "value" : "test"

      },

      "propertyId", null,

      "propertyValue", null

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "name",

  "order": "asc",

  "size": 1

}

**Get the binary data for a historic detail variable**

GET history/historic-detail/{detailId}/data

**Table 15.192. Get the binary data for a historic detail variable - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the historic detail instance was found and the requested variable data is returned. |
| 404 | Indicates the requested historic detail instance was not found or the historic detail instance doesn't have a variable with the given name or the variable doesn't have a binary stream available. Status message provides additional information. |

**Success response body:** The response body contains the binary value of the variable. When the variable is of type binary, the content-type of the response is set to application/octet-stream, regardless of the content of the variable or the request accept-type header. In case of serializable, application/x-java-serialized-object is used as content-type.

**Forms**

**Get form data**

GET form/form-data

**Table 15.193. Get form data - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| taskId | Yes (if no processDefinitionId) | String | The task id corresponding to the form data that needs to be retrieved. |
| processDefinitionId | Yes (if no taskId) | String | The process definition id corresponding to the start event form data that needs to be retrieved. |

**Table 15.194. Get form data - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates that form data could be queried. |
| 404 | Indicates that form data could not be found. |

**Success response body:**

{

  "data": [

    {

      "formKey" : null,

      "deploymentId" : "2",

      "processDefinitionId" : "3",

      "processDefinitionUrl" : "http://localhost:8182/repository/process-definition/3",

      "taskId" : "6",

      "taskUrl" : "http://localhost:8182/runtime/task/6",

      "formProperties" : [

        {

          "id" : "room",

          "name" : "Room",

          "type" : "string",

          "value" : null,

          "readable" : true,

          "writable" : true,

          "required" : true,

          "datePattern" : null,

          "enumValues" : [

            {

              "id" : "normal",

              "name" : "Normal bed"

            },

            {

              "id" : "kingsize",

              "name" : "Kingsize bed"

            },

          ]

        }

      ]

    }

  ],

  "total": 1,

  "start": 0,

  "sort": "name",

  "order": "asc",

  "size": 1

}

**Submit task form data**

POST form/form-data

**Request body for task form:**

{

  "taskId" : "5",

  "properties" : [

    {

      "id" : "room",

      "value" : "normal"

    }

  ]

}

**Request body for start event form:**

{

  "processDefinitionId" : "5",

  "businessKey" : "myKey", (optional)

  "properties" : [

    {

      "id" : "room",

      "value" : "normal"

    }

  ]

}

**Table 15.195. Submit task form data - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates request was successful and the form data was submitted |
| 400 | Indicates an parameter was passed in the wrong format. The status-message contains additional information. |

**Success response body for start event form data (no response for task form data):**

{

  "id" : "5",

  "url" : "http://localhost:8182/history/historic-process-instances/5",

  "businessKey" : "myKey",

  "suspended", false,

  "processDefinitionId" : "3",

  "processDefinitionUrl" : "http://localhost:8182/repository/process-definition/3",

  "activityId" : "myTask"

}

**Database tables**

**List of tables**

GET management/tables

**Table 15.196. List of tables - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the request was successful. |

**Success response body:**

[

   {

      "name":"ACT\_RU\_VARIABLE",

      "url":"http://localhost:8182/management/tables/ACT\_RU\_VARIABLE",

      "count":4528

   },

   {

      "name":"ACT\_RU\_EVENT\_SUBSCR",

      "url":"http://localhost:8182/management/tables/ACT\_RU\_EVENT\_SUBSCR",

      "count":3

   },

   ...

]

**Get a single table**

GET management/tables/{tableName}

**Table 15.197. Get a single table - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| tableName | Yes | String | The name of the table to get. |

**Success response body:**

{

      "name":"ACT\_RE\_PROCDEF",

      "url":"http://localhost:8182/management/tables/ACT\_RE\_PROCDEF",

      "count":60

}

**Table 15.198. Get a single table - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the table exists and the table count is returned. |
| 404 | Indicates the requested table does not exist. |

**Get column info for a single table**

GET management/tables/{tableName}/columns

**Table 15.199. Get column info for a single table - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| tableName | Yes | String | The name of the table to get. |

**Success response body:**

{

   "tableName":"ACT\_RU\_VARIABLE",

   "columnNames":[

      "ID\_",

      "REV\_",

      "TYPE\_",

      "NAME\_",

      ...

   ],

   "columnTypes":[

      "VARCHAR",

      "INTEGER",

      "VARCHAR",

      "VARCHAR",

      ...

   ]

}

**Table 15.200. Get column info for a single table - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the table exists and the table column info is returned. |
| 404 | Indicates the requested table does not exist. |

**Get row data for a single table**

GET management/tables/{tableName}/data

**Table 15.201. Get row data for a single table - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| tableName | Yes | String | The name of the table to get. |

**Table 15.202. Get row data for a single table - URL query parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| start | No | Integer | Index of the first row to fetch. Defaults to 0. |
| size | No | Integer | Number of rows to fetch, starting from start. Defaults to 10. |
| orderAscendingColumn | No | String | Name of the column to sort the resulting rows on, ascending. |
| orderDescendingColumn | No | String | Name of the column to sort the resulting rows on, descending. |

**Success response body:**

{

  "total":3,

   "start":0,

   "sort":null,

   "order":null,

   "size":3,

   "data":[

      {

         "TASK\_ID\_":"2",

         "NAME\_":"var1",

         "REV\_":1,

         "TEXT\_":"123",

         "LONG\_":123,

         "ID\_":"3",

         "TYPE\_":"integer"

      },

      ...

   ]

}

**Table 15.203. Get row data for a single table - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the table exists and the table row data is returned. |
| 404 | Indicates the requested table does not exist. |

**Engine**

**Get engine properties**

GET management/properties

Returns a read-only view of the properties used internally in the engine.

**Success response body:**

{

   "next.dbid":"101",

   "schema.history":"create(5.13)",

   "schema.version":"5.13"

}

**Table 15.204. Get engine properties - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the properties are returned. |

**Get engine info**

GET management/engine

Returns a read-only view of the engine that is used in this REST-service.

**Success response body:**

{

   "name":"default",

   "version":"5.13",

   "resourceUrl":"file://activiti/activiti.cfg.xml",

   "exception":null

}

**Table 15.205. Get engine info - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the engine info is returned. |

**Jobs**

**Get a single job**

GET management/jobs/{jobId}

**Table 15.206. Get a single job - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| jobId | Yes | String | The id of the job to get. |

**Success response body:**

{

   "id":"8",

   "url":"http://localhost:8182/management/jobs/8",

   "processInstanceId":"5",

   "processInstanceUrl":"http://localhost:8182/runtime/process-instances/5",

   "processDefinitionId":"timerProcess:1:4",

   "processDefinitionUrl":"http://localhost:8182/repository/process-definitions/timerProcess%3A1%3A4",

   "executionId":"7",

   "executionUrl":"http://localhost:8182/runtime/executions/7",

   "retries":3,

   "exceptionMessage":null,

   "dueDate":"2013-06-04T22:05:05.474+0000"

}

**Table 15.207. Get a single job - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the job exists and is returned. |
| 404 | Indicates the requested job does not exist. |

**Delete a job**

DELETE management/jobs/{jobId}

**Table 15.208. Delete a job - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| jobId | Yes | String | The id of the job to delete. |

**Table 15.209. Delete a job - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the job was found and has been deleted. Response-body is intentionally empty. |
| 404 | Indicates the requested job was not found. |

**Execute a single job**

PUT management/jobs/{jobId}

**Body JSON:**

{

  "action" : "execute"

}

**Table 15.210. Execute a single job - JSON Body parameters**

| Parameter | Description | Required |
| --- | --- | --- |
| action | Action to perform. Only execute is supported. | Yes |

**Table 15.211. Execute a single job - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the job was executed. Response-body is intentionally empty. |
| 404 | Indicates the requested job was not found. |
| 500 | Indicates the an exception occured while executing the job. The status-description contains additional detail about the error. The full error-stacktrace can be fetched later on if needed. |

**Get the exception stacktrace for a job**

GET management/jobs/{jobId}/exception-stracktrace

**Table 15.212. Get the exception stacktrace for a job - URL parameters**

| Parameter | Description | Required |
| --- | --- | --- |
| jobId | Id of the job to get the stacktrace for. | Yes |

**Table 15.213. Get the exception stacktrace for a job - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the requested job was not found and the stacktrace has been returned. The response contains the raw stacktrace and always has a Content-type of text/plain. |
| 404 | Indicates the requested job was not found or the job doesn't have an exception stacktrace. Status-description contains additional information about the error. |

**Get a list of jobs**

GET management/jobs

**Table 15.214. Get a list of jobs - URL query parameters**

| Parameter | Description | Type |
| --- | --- | --- |
| id | Only return job with the given id | String |
| processInstanceId | Only return jobs part of a process with the given id | String |
| executionId | Only return jobs part of an execution with the given id | String |
| processDefinitionId | Only return jobs with the given process definition id | String |
| withRetriesLeft | If true, only return jobs with retries left. If false, this parameter is ignored. | Boolean |
| executable | If true, only return jobs which are executable. If false, this parameter is ignored. | Boolean |
| timersOnly | If true, only return jobs which are timers. If false, this parameter is ignored. Cannot be used toghether with 'messagesOnly'. | Boolean |
| messagesOnly | If true, only return jobs which are messages. If false, this parameter is ignored. Cannot be used toghether with 'timersOnly' | Boolean |
| withException | If true, only return jobs for which an exception occured while executing it. If false, this parameter is ignored. | Boolean |
| dueBefore | Only return jobs which are due to be executed before the given date. Jobs without duedate are never returned using this parameter. | Date |
| dueAfter | Only return jobs which are due to be executed after the given date. Jobs without duedate are never returned using this parameter. | Date |
| exceptionMessage | Only return jobs with the given exception message | String |
| sort | Field to sort results on, should be one of id, dueDate, executionId, processInstanceId or retries. | String |
| The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL. |  |  |

**Success response body:**

{

   "data":[

      {

         "id":"13",

         "url":"http://localhost:8182/management/jobs/13",

         "processInstanceId":"5",

         "processInstanceUrl":"http://localhost:8182/runtime/process-instances/5",

         "processDefinitionId":"timerProcess:1:4",

         "processDefinitionUrl":"http://localhost:8182/repository/process-definitions/timerProcess%3A1%3A4",

         "executionId":"12",

         "executionUrl":"http://localhost:8182/runtime/executions/12",

         "retries":0,

         "exceptionMessage":"Can't find scripting engine for 'unexistinglanguage'",

         "dueDate":"2013-06-07T10:00:24.653+0000"

      },

      ...

   ],

   "total":2,

   "start":0,

   "sort":"id",

   "order":"asc",

   "size":2

}

**Table 15.215. Get a list of jobs - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the requested jobs were returned. |
| 400 | Indicates an illegal value has been used in a url query parameter or the both 'messagesOnly' and 'timersOnly' are used as parameters. Status description contains additional details about the error. |

**Users**

**Get a single user**

GET identity/users/{userId}

**Table 15.216. Get a single user - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| userId | Yes | String | The id of the user to get. |

**Success response body:**

{

   "id":"testuser",

   "firstName":"Fred",

   "lastName":"McDonald",

   "url":"http://localhost:8182/identity/users/testuser",

   "email":"no-reply@activiti.org"

}

**Table 15.217. Get a single user - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the user exists and is returned. |
| 404 | Indicates the requested user does not exist. |

**Get a list of users**

GET identity/users

**Table 15.218. Get a list of users - URL query parameters**

| Parameter | Description | Type |
| --- | --- | --- |
| id | Only return user with the given id | String |
| firstName | Only return users with the given firstname | String |
| lastName | Only return users with the given lastname | String |
| email | Only return users with the given email | String |
| firstNameLike | Only return userswith a firstname like the given value. Use % as wildcard-character. | String |
| lastNameLike | Only return users with a lastname like the given value. Use % as wildcard-character. | String |
| emailLike | Only return users with an email like the given value. Use % as wildcard-character. | String |
| memberOfGroup | Only return users which are a member of the given group. | String |
| potentialStarter | Only return users which are potential starters for a process-definition with the given id. | String |
| sort | Field to sort results on, should be one of id, firstName, lastname or email. | String |
| The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL. |  |  |

**Success response body:**

{

   "data":[

      {

         "id":"anotherUser",

         "firstName":"Tijs",

         "lastName":"Barrez",

         "url":"http://localhost:8182/identity/users/anotherUser",

         "email":"no-reply@alfresco.org"

      },

      {

         "id":"kermit",

         "firstName":"Kermit",

         "lastName":"the Frog",

         "url":"http://localhost:8182/identity/users/kermit",

         "email":null

      },

      {

         "id":"testuser",

         "firstName":"Fred",

         "lastName":"McDonald",

         "url":"http://localhost:8182/identity/users/testuser",

         "email":"no-reply@activiti.org"

      }

   ],

   "total":3,

   "start":0,

   "sort":"id",

   "order":"asc",

   "size":3

}

**Table 15.219. Get a list of users - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the requested users were returned. |

**Update a user**

PUT identity/users/{userId}

**Body JSON:**

{

  "firstName":"Tijs",

  "lastName":"Barrez",

  "email":"no-reply@alfresco.org"

}

All request values are optional. For example, you can only include the 'firstName' attribute in the request body JSON-object, only updating the firstName of the user, leaving all other fields unaffected. When an attribute is explicitly included and is set to null, the user-value will be updated to null. Example: {"firstName" : null} will clear the firstName of the user).

**Table 15.220. Update a user - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the user was updated. |
| 404 | Indicates the requested user was not found. |
| 409 | Indicates the requested user was updated simultaneously. |

**Success response body:** see response for identity/users/{userId}.

**Create a user**

POST identity/users

**Body JSON:**

{

  "id":"tijs",

  "firstName":"Tijs",

  "lastName":"Barrez",

  "email":"no-reply@alfresco.org"

}

**Table 15.221. Create a user - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the user was created. |
| 400 | Indicates the id of the user was missing. |

**Success response body:** see response for identity/users/{userId}.

**Delete a user**

DELETE identity/users/{userId}

**Table 15.222. Delete a user - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| userId | Yes | String | The id of the user to delete. |

**Table 15.223. Delete a user - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the user was found and has been deleted. Response-body is intentionally empty. |
| 404 | Indicates the requested user was not found. |

**Get a user's picture**

GET identity/users/{userId}/picture

**Table 15.224. Get a user's picture - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| userId | Yes | String | The id of the user to get the pîcture for. |

**Response Body:** The response body contains the raw picture data, representing the user's picture. The Content-type of the response corresponds to the mimeType that was set when creating the picture.

**Table 15.225. Get a user's picture - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the user was found and has a picture, which is returned in the body. |
| 404 | Indicates the requested user was not found or the user does not have a profile picture. Status-description contains additional information about the error. |

**Updating a user's picture**

GET identity/users/{userId}/picture

**Table 15.226. Updating a user's picture - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| userId | Yes | String | The id of the user to get the pîcture for. |

**Request body:** The request should be of type multipart/form-data. There should be a single file-part included with the binary value of the picture. On top of that, the folowing additionl form-fields can be present:

* mimeType: Optional mime-type for the uploaded picture. If omitted, the default of image/jpeg is used as a mime-type for the picture.

**Table 15.227. Updating a user's picture - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the user was found and the picture has been updated. The response-body is left empty intentionally. |
| 404 | Indicates the requested user was not found. |

**List a user's info**

PUT identity/users/{userId}/info

**Table 15.228. List a user's info - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| userId | Yes | String | The id of the user to get the info for. |

**Response Body:**

[

   {

      "key":"key1",

      "url":"http://localhost:8182/identity/users/testuser/info/key1"

   },

   {

      "key":"key2",

      "url":"http://localhost:8182/identity/users/testuser/info/key2"

   }

]

**Table 15.229. List a user's info - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the user was found and list of info (key and url) is returned. |
| 404 | Indicates the requested user was not found. |

**Get a user's info**

GET identity/users/{userId}/info/{key}

**Table 15.230. Get a user's info - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| userId | Yes | String | The id of the user to get the info for. |
| key | Yes | String | The key of the user info to get. |

**Response Body:**

{

   "key":"key1",

   "value":"Value 1",

   "url":"http://localhost:8182/identity/users/testuser/info/key1"

}

**Table 15.231. Get a user's info - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the user was found and the user has info for the given key.. |
| 404 | Indicates the requested user was not found or the user doesn't have info for the given key. Status description contains additional information about the error. |

**Update a user's info**

PUT identity/users/{userId}/info/{key}

**Table 15.232. Update a user's info - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| userId | Yes | String | The id of the user to update the info for. |
| key | Yes | String | The key of the user info to update. |

**Request Body:**

{

   "value":"The updated value"

}

**Response Body:**

{

   "key":"key1",

   "value":"The updated value",

   "url":"http://localhost:8182/identity/users/testuser/info/key1"

}

**Table 15.233. Update a user's info - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the user was found and the info has been updated. |
| 400 | Indicates the value was missing from the request body. |
| 404 | Indicates the requested user was not found or the user doesn't have info for the given key. Status description contains additional information about the error. |

**Create a new user's info entry**

POST identity/users/{userId}/info

**Table 15.234. Create a new user's info entry - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| userId | Yes | String | The id of the user to create the info for. |

**Request Body:**

{

   "key":"key1",

   "value":"The value"

}

**Response Body:**

{

   "key":"key1",

   "value":"The value",

   "url":"http://localhost:8182/identity/users/testuser/info/key1"

}

**Table 15.235. Create a new user's info entry - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the user was found and the info has been created. |
| 400 | Indicates the key or value was missing from the request body. Status description contains additional information about the error. |
| 404 | Indicates the requested user was not found. |
| 409 | Indicates there is already an info-entry with the given key for the user, update the resource instance (PUT). |

**Delete a user's info**

DELETE identity/users/{userId}/info/{key}

**Table 15.236. Delete a user's info - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| userId | Yes | String | The id of the user to delete the info for. |
| key | Yes | String | The key of the user info to delete. |

**Table 15.237. Delete a user's info - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the user was found and the info for the given key has been deleted. Response body is left empty intentionally. |
| 404 | Indicates the requested user was not found or the user doesn't have info for the given key. Status description contains additional information about the error. |

**Groups**

**Get a single group**

GET identity/groups/{groupId}

**Table 15.238. Get a single group - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| groupId | Yes | String | The id of the group to get. |

**Success response body:**

{

   "id":"testgroup",

   "url":"http://localhost:8182/identity/groups/testgroup",

   "name":"Test group",

   "type":"Test type"

}

**Table 15.239. Get a single group - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the group exists and is returned. |
| 404 | Indicates the requested group does not exist. |

**Get a list of groups**

GET identity/groups

**Table 15.240. Get a list of groups - URL query parameters**

| Parameter | Description | Type |
| --- | --- | --- |
| id | Only return group with the given id | String |
| name | Only return groups with the given name | String |
| type | Only return groups with the given type | String |
| nameLike | Only return groups with a name like the given value. Use % as wildcard-character. | String |
| member | Only return groups which have a member with the given username. | String |
| potentialStarter | Only return groups which members are potential starters for a process-definition with the given id. | String |
| sort | Field to sort results on, should be one of id, name or type. | String |
| The general [paging and sorting query-parameters](http://www.mossle.com/docs/activiti/#restPagingAndSort) can be used for this URL. |  |  |

**Success response body:**

{

   "data":[

     {

        "id":"testgroup",

        "url":"http://localhost:8182/identity/groups/testgroup",

        "name":"Test group",

        "type":"Test type"

     },

      ...

   ],

   "total":3,

   "start":0,

   "sort":"id",

   "order":"asc",

   "size":3

}

**Table 15.241. Get a list of groups - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the requested groups were returned. |

**Update a group**

PUT identity/groups/{groupId}

**Body JSON:**

{

   "name":"Test group",

   "type":"Test type"

}

All request values are optional. For example, you can only include the 'name' attribute in the request body JSON-object, only updating the name of the group, leaving all other fields unaffected. When an attribute is explicitly included and is set to null, the group-value will be updated to null.

**Table 15.242. Update a group - Response codes**

| Response code | Description |
| --- | --- |
| 200 | Indicates the group was updated. |
| 404 | Indicates the requested group was not found. |
| 409 | Indicates the requested group was updated simultaneously. |

**Success response body:** see response for identity/groups/{groupId}.

**Create a group**

POST identity/groups

**Body JSON:**

{

   "id":"testgroup",

   "name":"Test group",

   "type":"Test type"

}

**Table 15.243. Create a group - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the group was created. |
| 400 | Indicates the id of the group was missing. |

**Success response body:** see response for identity/groups/{groupId}.

**Delete a group**

DELETE identity/groups/{groupId}

**Table 15.244. Delete a group - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| groupId | Yes | String | The id of the group to delete. |

**Table 15.245. Delete a group - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the group was found and has been deleted. Response-body is intentionally empty. |
| 404 | Indicates the requested group was not found. |

**Get members in a group**

There is no GET allowed on identity/groups/members. Use the identity/users?memberOfGroup=sales URL to get all users that are part of a particular group.

**Add a member to a group**

POST identity/groups/{groupId}/members

**Table 15.246. Add a member to a group - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| groupId | Yes | String | The id of the group to add a member to. |

**Body JSON:**

{

   "userId":"kermit"

}

**Table 15.247. Add a member to a group - Response codes**

| Response code | Description |
| --- | --- |
| 201 | Indicates the group was found and the member has been added. |
| 404 | Indicates the userId was not included in the request body. |
| 404 | Indicates the requested group was not found. |
| 409 | Indicates the requested user is already a member of the group. |

**Response Body:**

{

   "userId":"kermit",

   "groupId":"sales",

    "url":"http://localhost:8182/identity/groups/sales/members/kermit"

}

**Delete a member from a group**

DELETE identity/groups/{groupId}/members/{userId}

**Table 15.248. Delete a member from a group - URL parameters**

| Parameter | Required | Value | Description |
| --- | --- | --- | --- |
| groupId | Yes | String | The id of the group to remove a member from. |
| userId | Yes | String | The id of the user to remove. |

**Table 15.249. Delete a member from a group - Response codes**

| Response code | Description |
| --- | --- |
| 204 | Indicates the group was found and the member has been deleted. The response body is left empty intentionally. |
| 404 | Indicates the requested group was not found or that the user is not a member of the group. The status description contains additional information about the error. |

**Response Body:**

{

   "userId":"kermit",

   "groupId":"sales",

    "url":"http://localhost:8182/identity/groups/sales/members/kermit"

}

**Legacy REST - General Usage**

**The folowing section contains documentation of the Legacy REST-api, which has been deprecated since the 5.13 release. The REST-urls will not be removed in the future but will not be maintained. Any future additions and improvements will be done to the new REST API.**

Activiti includes a REST API to the Activiti Engine that can be installed by deploying the activiti-rest.war file to a servlet container like Apache Tomcat. By default the Activiti Engine will connect to a standalone H2 database. You can change the database settings in the db.properties file in the WEB-INF/classes folder. The REST API uses JSON format (http://www.json.org) and is built upon the Restlet (http://www.restlet.org).

Each REST API call has its individual authorization level and you must be logged in as a user to invoke a REST API call (except for the /login service). Authentication is done using Basic HTTP Authentication, so if you logged in as an admin (i.e. kermit) you should be able to perform all calls as described below.

The API follows normal REST API conventions using GET for read operations, POST for creating objects, PUT for updating and performing operations on already created objects and finally DELETE for deleting objects. When performing a call that affects multiple objects POST is used on all such operations for consistency and making sure that an unlimited number of objects may be used. The reason for using POST is that the HTTP DELETE method doesn't implicitly allow request bodies and therefore, a call using DELETE, in theory, could get it's request body stripped out by a proxy. So to be certain this doesn't happen we use POST, even when PUT could have been used to update multiple objects, for consistency.

All rest calls use a content type of "application/json" (except for upload requests which uses "multipart/form-data").

The base URL for invoking a REST call is http://localhost:8080/activiti-rest/service/. So for example to list the process definitions in the engine point your browser to: http://localhost:8080/activiti-rest/service/process-definitions

You may also use curl to run queries against the REST API, e.g.:

curl --basic --user kermit:kermit http://localhost:8080/activiti-rest/service/tasks?assignee=kermit

Please look below to see what REST API calls that currently are available. Please consider the "API" sections as a "one line hint" to what functionality of the core API that is used to implement the REST API call.

**Repository**

**Upload Deployment**

Uploads and installs a deployment of format .bpmn20.xml, .bpmn, .bar or .zip using normal "html form upload" (enctype=multipart/form-data) in other words not a json-request. The deployment information is returned as a response if the deploy action succeeds.

* **Request:** POST /deployment
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getRepositoryService().createDeployment().name(fileName).deploymentBuilder.deploy()
* **Response:**
* {
* "id": "10",
* "name": "activiti-examples.bar",
* "deploymentTime": "2010-10-13T14:54:26.750+02:00",
* "category": "examples"

}

**Get Deployments**

Returns a paginated list deployments that can be sorted by "id", "name" or "deploymentTime".

* **Request:** GET /deployments?start={start=0}&size={size=10}&sort={sort=id}&order={order=asc}
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getRepositoryService().createDeploymentQuery().listPage()
* **Response:**
* {
* "data": [
* {
* "id": "10",
* "name": "activiti-examples.bar",
* "deploymentTime": "2010-10-13T14:54:26.750+02:00",
* "category": "examples"
* }
* ],
* "total": 1,
* "start": 0,
* "sort": "id",
* "order": "asc",
* "size": 1

}

**Get Deployment Resources**

Returns all resources from the deployment. Example: /deployment/10/resources

* **Request:** GET /deployment/{deploymentId}/resources
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getRepositoryService().getDeploymentResourceNames(deploymentId)
* **Response:**
* {
* "id": "10",
* "name": "activiti-examples.bar",
* "deploymentTime": "2010-10-13T14:54:26.750+02:00",
* "category": "examples",
* "resources": ["resource1", "resource2"]

}

**Get Deployment Resource**

Returns a resource from the deployment. Example: /deployment/10/resource/org/activiti/examples/bpmn/usertask/FinancialReportProcess.bpmn20.xml

* **Request:** GET /deployment/{deploymentId}/resource/{resourceName}
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getRepositoryService().getResourceAsStream(deploymentId, resourceName)
* **Response:**

I.e a .bpmn20.xml file, an image or whatever type of file the deployment resource contained.

**Delete Deployment**

Deletes a deployment.

* **Request:** DELETE /deployment/{deploymentId}?cascade={cascade?}
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getRepositoryService().deleteDeployment(deploymentId)
* **Response:**
* {
* "success": true

}

**Delete Deployments**

Deletes multiple deployment.

* **Request:** POST /deployments/delete?cascade={cascade?}
* {
* "deploymentIds": [ "10", "11" ]

}

* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getRepositoryService().deleteDeployment(deploymentId)
* **Response:**
* {
* "success": true

}

**Engine**

**Get Process Engine**

Returns the process engine initialization details. If something went wrong during startup, details about the error will be given in the "exception" attribute in the response.

* **Request:** GET /process-engine
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName)
* **Response:**
* {
* "name": "default",
* "resourceUrl": "jar:file:\//<path-to-deployment>\/activiti-cfg.jar!\/activiti.properties",
* "exception": null,
* "version": "5.11"

}

**Processes**

**List Process Definitions**

Returns details about the deployed process definitions that can be sorted by "id", "name", "version" or "deploymentId". The name of the BPMN2.0 XML process diagram is given in the "resourceName" attribute and can, in combination with the "deploymentId" attribute, be retrieved from the GET Deployment Resource REST API call above.

* **Paginated Request:** GET /process-definitions?start={start=0}&size={size=10}&sort={sort=id}&order={order=asc}
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getRepositoryService().createProcessDefinitionQuery().listPage()
* **Paginated Response:**
* {
* "data": [
* {
* "id": "financialReport:1",
* "key": "financialReport",
* "version": 1,
* "name": "Monthly financial report",
* "resourceName": "org/activiti/examples/bpmn/usertask/FinancialReportProcess.bpmn20.xml",
* "diagramResourceName": "org/activiti/examples/bpmn/usertask/FinancialReportProcess.png",
* "deploymentId": "10",
* "startFormResourceKey": null,
* "isGraphicNotationDefined": true
* }
* ],
* "total": 1,
* "start": 0,
* "sort": "id",
* "order": "asc",
* "size": 1

}

**Get Process Definition Form Properties**

Returns a process definition's form properties.

* **Request:** GET /process-definition/{processDefinitionId}/properties
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getFormService().getStartFormData(processDefinitionId)
* **Response:**
* "data": [
* {
* "id": "fullName",
* "name": "Full name",
* "value": "${name}",
* "type": "String",
* "required": false,
* "readable": true,
* "writeable": true
* }
* ]

**Get Process Definition Form Resource**

Returns a process definition's form.

* **Request:** GET /process-definition/{processDefinitionId}/form
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getTaskService().getRenderedStartFormById(processDefinitionId)
* **Response:**

<user-defined-response>

**Get Process Definition Diagram**

Returns a PNG diagram of the process definition if it's available.

* **Request:** GET /process-definition/{processDefinitionId}/diagram
* **API:** repositoryService.getResourceAsStream(processDefinition.getDeploymentId(), processDefinition.getDiagramResourceName());
* **Response:**

Png diagram of the process definition.

**Start Process Instance**

Creates a process instance based on a process definition and returns details about the newly created process instance. Additional variables (from a form) may be passed using the body object. In other words placing attributes next to the "processDefinitionId" attribute.

Note that if a value is submitted as true (instead of "true") it will be treated as a Boolean even if no descriptor is used. The same is also valid for number, i.e., 123 will become an Integer but "123" will become a String (unless a descriptor is defined).

* **Request:** POST /process-instance
* {
* "processDefinitionId":"financialReport:1:1700",
* "businessKey":"order-4711"

}

* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getRuntimeService().startProcessInstanceById(processDefinitionId[, businessKey][, variables])
* **Response:**
* {
* "id": "217",
* "processDefinitionId": "financialReport:1:1700",
* "businessKey": "order-4711",
* "processInstanceId": "217"

}

**List Process Instances**

Returns details about the active process instances that can be sorted by "id", "startTime", "businessKey" or "processDefinitionId". You can filter instances by "processDefinitionId" and "businessKey".

* **Paginated Request:** GET /process-instances?start={start=0}&size={size=10}&sort={sort=id}&order={order=asc}&businessKey={businessKey}&processDefinitionId={processDefinitionId}
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getHistoryService().createHistoricProcessInstanceQuery().xxx.listPages()
* **Paginated Response:**
* {
* "data": [
* {
* "id": "2",
* "processDefinitionId": "financialReport:1",
* "businessKey": "55",
* "startTime": "2010-10-13T14:54:26.750+02:00",
* "startUserId": "kermit"
* }
* ],
* "total": 1,
* "start": 0,
* "sort": "id",
* "order": "asc",
* "size": 1

  }

**Get Process Instance Details**

Returns all details about a specific process instance. This can be a running or completed process instance.

* **Request:** GET /process-instance/{processInstanceId}
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getHistoryService().createHistoricProcessInstanceQuery().processInstanceId(..).singleResult()
* **Response:**
* {
* "id": "2",
* "processDefinitionId": "financialReport:1",
* "businessKey": "55",
* "startTime": "2010-10-13T14:54:26.750+02:00",
* "startActivityId": "startFinancialAnalysis",
* "startUserId": "kermit",
* "completed": false,
* "tasks": [
* {
* "taskId": "3",
* "taskName": "Analyze report",
* "owner": null,
* "assignee": "Kermit",
* "startTime": "2010-10-13T14:53:26.750+02:00",
* "completed": false
* }
* ],
* "activities": [
* {
* "activityId": "4",
* "activityName": "Get report",
* "activityType": "ServiceTask",
* "startTime": "2010-10-13T14:53:25.750+02:00",
* "completed": true,
* "startTime": "2010-10-13T14:53:25.950+02:00",
* "duration": 200
* }
* ],
* "variables": [
* {
* "variableName": "reportName",
* "variableValue": "classified.pdf",
* }
* ]
* "historyVariables": [
* {
* "variableName": "reportName",
* "variableValue": "classified.pdf",
* "variableType": "String",
* "revision": 1,
* "time": "2010-10-13T14:53:26.750+02:00"
* }
* ]

  }

**Get Process Instance Diagram**

Returns a PNG diagram of process with highlighted active executions. Returns 404 if process definition does not contain DI information.

* **Request:** GET /process-instance/{processInstanceId}/diagram
* **API:** ProcessDiagramGenerator.generateDiagram(pde, "png", getRuntimeService().getActiveActivityIds(processInstanceId));
* **Response:**

Png diagram of process with highlighted active executions.

**Get open tasks for a process instance**

Returns a list of open tasks for a running process instance

* **Request:** GET /process-instance/{processInstanceId}/tasks
* **API:** taskService.createTaskQuery().processInstanceId(processInstanceId);
* **Paginated Response:**
* {
* "data": [
* {
* "id": "127",
* "name": "Handle vacation request",
* "description": "Vacation request by Kermit",
* "delegationState": "pending",
* "dueDate": "2010-10-13T14:54:26.750+02:00",
* "priority": 50,
* "assignee": null,
* "executionId": "118",
* "formResourceKey": "org/activiti/examples/taskforms/approve.form",
* "owner": "Kermit",
* "parentTaskId": "120",
* "processDefinitionId": "financialReport:1",
* "processInstanceId": "123",
* "taskDefinitionKey": "125"
* }
* ],
* "total": 1,
* "start": 0,
* "sort": "id",
* "order": "asc",
* "size": 1

}

**Signal an activity (receive task) for a specific process instance**

Signals an activity execution (receive task).

* **Request:** POST /process-instance/{processInstanceId}/signal
* {
* "activityId":"receiveTask",
* "variable1":"value",
* "variable2":"value"

}

* **API:** runtimeService.signal(execution.getId(), variables);
* **Response:**
* {
* "success": true

}

**Trigger a signal for a specific process instance**

Sends a signal to a specific process instance, causing any subscribing signal event to fire. Additional variables may be passed in the body. If you don't want to set any variables, you may leave the body empty.

* **Request:** POST /process-instance/{processInstanceId}/event/{signalName}
* {
* "variable1":"value",
* "variable2":"value"

}

* **API:** runtimeService.signalEventReceived(signalName, execution.getId()[, variables]);
* **Response:**
* {
* "success": true

}

**Tasks**

**Get Task Summary**

Returns a task summary for a specific user: The number of tasks assigned the user, how many unassigned tasks that the user may claim and how many unassigned tasks there are per group that the user is a member of.

* **Request:** GET /tasks-summary?user={userId}
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getTaskService().createTaskQuery().xxx().count()
* **Response:**
* {
* "assigned": {
* "total": 0
* },
* "unassigned": {
* "total": 1,
* "groups":
* {
* "accountancy": 1,
* "sales": 0,
* "engineering": 0,
* "management": 0
* }
* }

}

**List Tasks**

Returns a paginated list of tasks that can be sorted by: "id", "name", "description", "priority", "assignee", "executionId" or "processInstanceId". The list must be based on a user of a specific role: assignee (lists the tasks assigned to the user) or candidate (lists the tasks that the user may claim) or a candidate group (lists tasks that the members of the group may claim). If the task has a form it is given in the "formResourceKey" attribute. The form for a task can be retrieved from the GET Task Form REST API call.

* **Paginated Request:** GET /tasks?[assignee={userId}|candidate={userId}|candidate-group={groupId}]&start={start=0}&size={size=10}&sort={sort=id}&order={order=asc}
* **Example:**

curl --basic --user kermit:kermit http://localhost:8080/activiti-rest/service/tasks?assignee=kermit

* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getTaskService().createTaskQuery().xxx().listPage()
* **Paginated Response:**
* {
* "data": [
* {
* "id": "127",
* "name": "Handle vacation request",
* "description": "Vacation request by Kermit",
* "delegationState": "pending",
* "dueDate": "2010-10-13T14:54:26.750+02:00",
* "priority": 50,
* "assignee": null,
* "executionId": "118",
* "formResourceKey": "org/activiti/examples/taskforms/approve.form",
* "owner": "Kermit",
* "parentTaskId": "120",
* "processDefinitionId": "financialReport:1",
* "processInstanceId": "123",
* "taskDefinitionKey": "125"
* }
* ],
* "total": 1,
* "start": 0,
* "sort": "id",
* "order": "asc",
* "size": 1

}

**Get Task**

Returns details about the task with the task id.

* **Request:** GET /task/{taskId}
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getTaskService().createTaskQuery().taskId(taskId).singleResult()
* **Response:**
* {
* "id": "127",
* "name": "Handle vacation request",
* "description": "Vacation request by Kermit",
* "delegationState": "pending",
* "dueDate": "2010-10-13T14:54:26.750+02:00",
* "priority": 50,
* "assignee": null,
* "executionId": "118",
* "formResourceKey": "org/activiti/examples/taskforms/approve.form",
* "owner": "Kermit",
* "parentTaskId": "120",
* "processDefinitionId": "financialReport:1",
* "processInstanceId": "123",
* "taskDefinitionKey": "125",
* "subTaskList": [
* {
* "id": "129",
* "name": "Analyze request",
* "description": "Analyze request",
* "delegationState": "pending",
* "dueDate": "2010-10-13T14:54:26.750+02:00",
* "priority": 50,
* "assignee": null,
* "executionId": "118",
* "owner": "Kermit",
* "parentTaskId": "127"
* }
* ],
* "identityLinkList" : [
* {
* "type": "candidate",
* "userId": "Fozzie",
* "groupId": null
* }
* ],
* "attachmentList" : [
* {
* "id": "130",
* "name": "vacation\_request.xls",
* "description": "Vacation request",
* "type": "application/pdf",
* "url": null
* }
* ]

}

**Get Task Form**

Returns a task's form.

* **Request:** GET /task/{taskId}/form
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getTaskService().getRenderedTaskForm(taskId)
* **Response:**

<user-defined-response>

**Perform Task Operation**

Performs an operation (claim, unclaim, assign or complete) on a task. For the "complete" operation additional variables (from a form) may be passed in the body. To read more about additional variables from forms, visit the Start Process Instance section

* **Request:** PUT /task/{taskId}/[claim|unclaim|complete|assign] Claim and unclaim don't need a JSON body, but for complete you can provide a number of variables and for assign a userId field is required. An example body for the complete operation:
* {
* "variableName1": "variableValue1",
* "variableName2": "variableValue2"

}

An example body for the assign operation:

{

      "userId": "newAssignee"

}

* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getTaskService().xxx(taskId ...)
* **Response:**
* {
* "success": true

}

**List Form Properties**

Returns a list of properties of a form of a running task, which is defined by the process.

* **Request:** GET /form/{taskId}/properties
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getFormService().getTaskFormData(taskId).getFormProperties()
* **Response:**
* {
* "data": [
* {
* "id": "userName",
* "name": "User",
* "value": "foobar",
* "type": "string",
* "required": "true",
* "readable": "true",
* "writable": "true"
* }
* ]

}

**Add attachment to a task**

Add an attachment to a task instance

* **Request:** PUT /task/{taskId}/attachment

{}

* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getTaskService().createAttachment(...)
* **Response:**
* {
* "id": "130",
* "name": "vacation\_request.xls",
* "description": "Vacation request",
* "type": "application/pdf",
* "url": null

        }

**Get task attachment**

Returns the task attachment

* **Request:** GET /attachment/{attachmentId}
* **API:** taskService.getAttachment(attachmentId);
* **Response:**

The attachment.

**Add url to a task**

Add an url to a task instance

* **Request:** PUT /task/{taskId}/url

{}

* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getTaskService().createAttachment(...)
* **Response:**
* {
* "id": "130",
* "name": "google.com",
* "description": "Good search sitet",
* "type": null,
* "url": "http://www.google.com"

        }

**Identity**

**Login**

Authenticates a user. If user and password doesn't match a response with status 403 is returned. If authentication is successful, a response with status 200 is returned.

* **Request:** POST /login
* {
* "userId": "kermit",
* "password": "kermit"

}

* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getIdentityService().checkPassword(userId, password)
* **Response:**
* {
* "success": true

}

**Get User**

Returns details about a user.

* **Request:** GET /user/{userId}
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getIdentityService().createUserQuery().userId(userId).singleResult();
* **Response:**
* {
* "id": "kermit",
* "firstName": "Kermit",
* "lastName": "the Frog",
* "email": "kermit@server.com"

}

**List User's Groups**

Returns a paginated list groups belonging to a user that can be sort by "id", "name" or "type".

* **Paginated Request:** GET /user/{userId}/groups?start={start=0}&size={size=10}&sort={sort=id}&order={order=asc}
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getIdentityService().xxx(userId)
* **Paginated Response:**
* {
* data: [
* {
* "id": "admin",
* "name": "System administrator",
* "type": "security-role"
* }
* ],
* "total": 1,
* "start": 0,
* "sort": "id",
* "order": "asc",
* "size": 1

}

**Search users**

Returns a list users from whom the first or last name corresponds to the search text.

* **Paginated Request:** GET /users?searchText=ker
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getIdentityService().createUserQuery().userFirstNameLike(searchText).list() ProcessEngines.getProcessEngine(configuredProcessEngineName).getIdentityService().createUserQuery().userLastNameLike(searchText).list()
* **Response:**
* {
* data: [
* {
* "id": "kermit",
* "firstName": "Kermit",
* "lastName": "the Frog",
* "email": "kermit@server.com"
* }
* ],
* "total": 1,
* "start": 0,
* "sort": "id",
* "order": "asc",
* "size": 1

}

**Create user**

Creates a new user.

* **Request:** PUT /user
* {
* "id": "kermit",
* "firstName": "Kermit",
* "lastName": "the Frog",
* "email": "kermit@server.com",
* "password": "kermit"

}

* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getIdentityService().newUser(); ProcessEngines.getProcessEngine(configuredProcessEngineName).getIdentityService().saveUser();
* **Response:**
* {
* "success": true

}

**Add user to groups**

Adds user to the groups posted to this REST service.

* **Request:** POST /user/{userId}/groups

["management", "sales"]

* **API:** identityService().createMembership(userId, groupId);
* **Response:**
* {
* "success": true

}

**Remove user from group**

Removes user from a group.

* **Request:** DELETE /user/{userId}/groups/{groupId}
* **API:** identityService().deleteMembership(userId, groupId);
* **Response:**
* {
* "success": true

}

**Get user picture**

Returns the task attachment

* **Request:** GET /user/{userId}/picture
* **API:** identityService.getUserPicture(userId);
* **Response:**

The user picture.

**Get Group**

Returns details about a group.

* **Request:** GET /group/{groupId}
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getIdentityService().createGroupQuery().groupId(groupId).singleResult();
* **Response:**
* {
* "id": "admin",
* "name": "System administrator",
* "type": "security-role"

}

**List Group Users**

Returns details about a group's users that can be sorted by "id", "firstName", "lastName" or "email".

* **Paginated Request:** GET /group/{groupId}/users
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getIdentityService().createUserQuery().memberOfGroup(groupId).list()
* **Paginated Response:**
* {
* data: [
* {
* "id": "kermit",
* "firstName": "Kermit",
* "lastName": "the Frog",
* "email": "kermit@server.com"
* }
* ],
* "total": 1,
* "start": 0,
* "sort": "id",
* "order": "asc",
* "size": 1

}

**Search groups**

Returns a list of groups from whom the id or name corresponds to the search text.

* **Paginated Request:** GET /groups?searchText=ad
* **API:** identityService.createGroupQuery().list()
* **Response:**
* {
* data: [
* {
* "id": "admin",
* "name": "System administrator",
* "type": "security-role"
* }
* ],
* "total": 1,
* "start": 0,
* "sort": "id",
* "order": "asc",
* "size": 1

}

**Create group**

Creates a new group.

* **Request:** PUT /group
* {
* "id": "admin",
* "name": "System administrator",
* "type": "security-role"

}

* **API:** identityService.newGroup(); identityService.saveGroup();
* **Response:**
* {
* "success": true

}

**Add users to a group**

Adds users to a group.

* **Request:** POST /group/{groupId}/users

["kermit", "fozzie"]

* **API:** identityService().createMembership(userId, groupId);
* **Response:**
* {
* "success": true

}

**Remove user from group**

Removes user from a group.

* **Request:** DELETE /group/{groupId}/users/{userId}
* **API:** identityService().deleteMembership(userId, groupId);
* **Response:**
* {
* "success": true

}

**Management**

**List Jobs**

Returns a paginated list of jobs that can be sorted by "id", "process-instance-id", "execution-id", "due-date", "retries" or some custom arbitrary property id. The list can also be filtered by process instance id, due date or if the jobs have retries, are executable or only have messages or timers.

* **Paginated Request:** GET /management/jobs?process-instance={processInstanceId?}&with-retries-left={withRetriesLeft=false}&executable={axecutable=false}&only-timers={onlyTimers=false}&only-messages={onlyMessage=false}&duedate-lt={iso8601Date}&duedate-ltoe={iso8601Date}&duedate-ht={iso8601Date}&duedate-htoe={iso8601Date}&start={start=0}&size={size=10}&sort={sort=id}&order={order=asc}
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).createJobQuery().xxx().listPage()
* **Paginated Response:**
* {
* "data": [
* {
* "id": "212",
* "executionId": "211",
* "retries": -1,
* "processInstanceId": "210",
* "dueDate": null,
* "assignee": null,
* "exceptionMessage": "Can\'t find scripting engine for \'groovy\'"
* }
* ],
* "total": 1,
* "start": 0,
* "sort": "id",
* "order": "asc",
* "size": 1

}

**Get Job**

Returns details about a job.

* **Request:** GET /management/job({jobId}
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).createJobQuery().id(jobId).singleResult()
* **Response:**
* {
* "id": "212",
* "executionId": "211",
* "retries": -1,
* "processInstanceId": "210",
* "dueDate": null,
* "assignee": null,
* "exceptionMessage": "Can\'t find scripting engine for \'groovy\'",
* "stacktrace": "org.activiti.engine.ActivitiException: Can't find scripting engine for 'groovy'\n\tat ..."

}

**Execute Job**

Executes a job.

* **Request:** PUT /management/job/{jobId}/execute
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getManagementService().executeJob(jobId)
* **Response:**
* {
* "success": true

}

**Execute Jobs**

Executes multiple job.

* **Request:** POST /management/jobs/execute
* {
* "jobIds": [ "212" ]

}

* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getManagementService().executeJob(jobId)
* **Response:**
* {
* "success": true

}

**List Database Tables**

Returns meta data information about all database tables in the engine.

* **Request:** GET /management/tables
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getManagementService().getTableCount()
* **Response:**
* {
* "data": [
* {
* "tableName": "ACT\_GE\_PROPERTY",
* "noOfResults": 2
* }
* ]

}

**Get Table Meta Data**

Returns meta data about a database table.

* **Request:** GET /management/table/{tableName}
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getManagementService().getTableMetaData(tableName))
* **Response:**
* {
* "tableName": "ACT\_GE\_PROPERTY",
* "columnNames": ["REV\_","NAME\_","VALUE\_"],
* "columnNames": ["class java.lang.Integer", "class java.lang.String", "class java.lang.String"]

}

**Get Table Data**

Returns a paginated list of database table data.

* **Paginated Request:** GET /management/table/{tableName}/data
* **API:** ProcessEngines.getProcessEngine(configuredProcessEngineName).getManagementService().createTablePageQuery().tableName(tableName).start(start).size(size).orderXXX(sort).singleResult();
* **Paginated Response:**
* {
* "data": [
* {
* "NAME\_": "schema.version",
* "REV\_": "1",
* "VALUE\_": "5.10"
* },
* {
* "NAME\_": "next.dbid",
* "REV\_": "4",
* "VALUE\_": "310"
* }
* ],
* "total": 2,
* "start": 0,
* "sort": "NAME\_",
* "order": "asc",
* "size": 2

}

**Chapter 16. 集成CDI**

**Table of Contents**

[设置activiti-cdi](http://www.mossle.com/docs/activiti/#N1648E)

[查找流程引擎](http://www.mossle.com/docs/activiti/#N16493)

[配置Process Engine](http://www.mossle.com/docs/activiti/#N164B7)

[发布流程](http://www.mossle.com/docs/activiti/#N164D5)

[基于CDI环境的流程执行](http://www.mossle.com/docs/activiti/#N164E3)

[与流程实例进行关联交互](http://www.mossle.com/docs/activiti/#N164E8)

[声明式流程控制](http://www.mossle.com/docs/activiti/#N16512)

[在流程中引用bean](http://www.mossle.com/docs/activiti/#N16529)

[使用@BusinessProcessScoped beans](http://www.mossle.com/docs/activiti/#N1653D)

[注入流程变量](http://www.mossle.com/docs/activiti/#N16547)

[接收流程事件](http://www.mossle.com/docs/activiti/#N16569)

[更多功能](http://www.mossle.com/docs/activiti/#N165A1)

[已知的问题](http://www.mossle.com/docs/activiti/#N165B9)

activiti-cdi模块提供activiti的可配置型和cdi扩展。 activiti-cdi最突出的特性有：

* 支持@BusinessProcessScoped beans（绑定到流程实例的cdi bean），
* 流程为cdi bean支持自定义EL处理器，
* 使用注解为流程实例提供声明式控制，
* Activiti可以挂接在cdi事件总线上，
* 支持Java EE和Java SE，支持Spring，
* 支持单元测试。

要想在maven项目中使用activiti-cdi，可以添加如下依赖：

<dependency>

        <groupId>org.activiti</groupId>

        <artifactId>activiti-cdi</artifactId>

        <version>5.x</version>

</dependency>

把'x'替换成你的activiti版本（>=5.6）。它会自动加入activiti-entin和spring。

**设置activiti-cdi**

Activiti cdi可以安装在不同环境中。这里，我们会根据配置项一一讲解。

**查找流程引擎**

cdi扩展需要访问到ProcessEngine。为实现此功能， 使用org.activiti.cdi.spi.ProcessEngineLookup接口在运行期进行查找。 cdi模块使用默认的名为org.activiti.cdi.impl.LocalProcessEngineLookup的实现， 它使用ProcessEngines这个工具类来查找ProcessEngine。默认配置下, 使用ProcessEngines#NAME\_DEFAULT来查找ProcessEngine。这个类可能是使用了自定义名称的子类。 注意：需要把activiti.cfg.xml放在classpath下。

Activiti cdi使用java.util.ServiceLoader SPI处理org.activiti.cdi.spi.ProcessEngineLookup的实例。 为了提供接口的自定义实现，我们需要创建一个文本文件，名为 META-INF/services/org.activiti.cdi.spi.ProcessEngineLookup， 在文件中我们需要指定实现的全类名。

**Note**

如果你没有提供自定义的org.activiti.cdi.spi.ProcessEngineLookup实现， activiti会使用默认的LocalProcessEngineLookup实现。这时， 你所需要做的就是把activiti.cfg.xml放到classpath下（看下一章）。

**配置Process Engine**

实际的配置依赖于选用的ProcessEngineLookup策略（参考上章）。 这里，我们主要结合LocalProcessEngineLookup讨论可用的配置， 这要求我们在classpath下提供一个spring的activiti.cfg.xml。

Activiti提供了不同的ProcessEngineConfiguration实现，主要是依赖实际使用的事务管理策略。 activiti-cdi模块对事务的要求不严格，意味着任何事务管理策略都可以使用 （即便是spring事务抽象层）。简单来讲，cdi模块提供两种自定义ProcessEngineConfiguration实现：

* org.activiti.cdi.CdiJtaProcessEngineConfiguration：activiti的JtaProcessEngineConfiguration的子类， 用于在activiti使用JTA管理的事务环境。
* org.activiti.cdi.CdiStandaloneProcessEngineConfiguration：activiti的StandaloneProcessEngineConfiguration的子类， 用于在activiti使用简单JDBC事务环境。

下面是JBoss 7下的activiti.cfg.xml文件的例子：

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

<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">

        <!-- lookup the JTA-Transaction manager -->

        <bean id="transactionManager" class="org.springframework.jndi.JndiObjectFactoryBean">

                <property name="jndiName" value="java:jboss/TransactionManager"></property>

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

        </bean>

        <!-- process engine configuration -->

        <bean id="processEngineConfiguration"

                class="org.activiti.cdi.CdiJtaProcessEngineConfiguration">

                <!-- lookup the default Jboss datasource -->

                <property name="dataSourceJndiName" value="java:jboss/datasources/ExampleDS" />

                <property name="databaseType" value="h2" />

                <property name="transactionManager" ref="transactionManager" />

                <!-- using externally managed transactions -->

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

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

        </bean>

</beans>

这是Glassfish 3.1.1下的例子（假设已经配置了名为jdbc/activiti的datasource）：

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

<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">

        <!-- lookup the JTA-Transaction manager -->

        <bean id="transactionManager" class="org.springframework.jndi.JndiObjectFactoryBean">

                <property name="jndiName" value="java:appserver/TransactionManager"></property>

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

        </bean>

        <!-- process engine configuration -->

        <bean id="processEngineConfiguration"

                class="org.activiti.cdi.CdiJtaProcessEngineConfiguration">

                <property name="dataSourceJndiName" value="jdbc/activiti" />

                <property name="transactionManager" ref="transactionManager" />

                <!-- using externally managed transactions -->

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

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

        </bean>

</beans>

注意上面的额配置需要"spring-context"模块：

<dependency>

        <groupId>org.springframework</groupId>

        <artifactId>spring-context</artifactId>

        <version>3.0.3.RELEASE</version>

</dependency>

在Java SE环境下的配置和[创建ProcessEngine](http://www.mossle.com/docs/activiti/#configuration)章节中提供的例子一样， 使用 "CdiStandaloneProcessEngineConfiguration" 替换 "StandaloneProcessEngineConfiguration"。

**发布流程**

可以使用标准的activiti-api发布流程（RepositoryService）。另外，activiti-cdi提供自动发布 classpath下processes.xml中列出的流程的方式。 下面是一个processes.xml文件的例子：

<?xml version="1.0" encoding="utf-8" ?>

<!-- list the processes to be deployed -->

<processes>

        <process resource="diagrams/myProcess.bpmn20.xml" />

        <process resource="diagrams/myOtherProcess.bpmn20.xml" />

</processes>

**基于CDI环境的流程执行**

这一章，我们简短了解activiti cdi扩展使用的基于环境的流程执行模型。 BPMN业务流程通常是一个长时间运行的操作，包含了用户和系统任务的操作。 运行过程中，流程会分成多个单独的工作单元，由用户和应用逻辑执行。 在activiti-cdi中，流程实例可以分配到cdi环境中，关联展现成一个工作单元。 这是非常有用的，如果工作单元太复杂，比如如果实现的用户任务是不同形式的复杂顺序， 可以在这个操作中保持"non-process-scoped"状态。 默认配置下，流程实例分配到"broadest"激活环境，就会启动交互， 如果交互环境没有激活，就会返回到请求中。

**与流程实例进行关联交互**

处理@BusinessProcessScoped beans，或注入流程变量时，我们实现了激活的cdi环境与流程实例的关联。 Activiti-cdi提供了org.activiti.cdi.BusinessProcess bean 来控制关联，特别是：

* startProcessBy\*(...)方法，对应activiti的RuntimeService中的相关方法， 允许启动和随后向关联的业务流程，
* resumeProcessById(String processInstanceId)，允许通过提供的id来关联流程实例，
* resumeTaskById(String taskId)，允许通过提供的id来关联任务（扩展情况下，也关联相应的流程实例），

一个工作单元（比如用户任务）完成后，completeTask()方法可以调用来解除流程实例和会话/请求的关联。 这会通知activiti当前任务已经完成，并让流程实例继续执行。

注意，BusinessProcess bean是@Named bean，意思是导出的方法可以通过 表达式语言调用，比如在JSF页面中。下面的JSF 2 代码启动一个新的交互， 把它分配给一个用户任务实例，id作为一个请求参数传递（比如pageName.jsf?taskId=XX）：

<f:metadata>

<f:viewParam name="taskId" />

<f:event type="preRenderView" listener="#{businessProcess.startTask(taskId, true)}" />

</f:metadata>

**声明式流程控制**

Activiti-cdi允许通过注解声明启动流程实例和完成任务。 @org.activiti.cdi.annotation.StartProcess注解允许 通过"key"或"name"启动流程实例。 注意流程实例会在注解的方法返回*之后*启动。比如：

@StartProcess("authorizeBusinessTripRequest")

public String submitRequest(BusinessTripRequest request) {

        // do some work

        return "success";

}

根据activiti的配置，注解方法的代码和启动流程实例 会在同一个事务中执行。 @org.activiti.cdi.annotation.CompleteTask事务的使用方式相同：

@CompleteTask(endConversation=false)

public String authorizeBusinessTrip() {

        // do some work

        return "success";

}

@CompleteTask注解可以结束当前会话。 默认行为会在activiti返回后结束会话。可以禁用结束会话的功能， 实例可以参考上述代码。

**在流程中引用bean**

Activiti-cdi使用自定义解析器把CDI bean暴露到activiti El中。这就可以在流程中引用这些bean：

<userTask id="authorizeBusinessTrip" name="Authorize Business Trip"

                        activiti:assignee="#{authorizingManager.account.username}" />

"authorizingManager"可以是生产者方法提供的bean：

@Inject @ProcessVariable Object businessTripRequesterUsername;

@Produces

@Named

public Employee authorizingManager() {

        TypedQuery<Employee> query = entityManager.createQuery("SELECT e FROM Employee e WHERE e.account.username='"

                + businessTripRequesterUsername + "'", Employee.class);

        Employee employee = query.getSingleResult();

        return employee.getManager();

}

你可以使用同样的方法在服务任务中调用EJB的业务方法， 使用activiti:expression="myEjb.method()"扩展。 注意，这要求在MyEjb类中使用@Named注解。

**使用@BusinessProcessScoped beans**

使用activiti-cdi，bean的生命周期可以绑定到流程实例上。为了扩展，可以提供一个自定义的环境实现， 命名为BusinessProcessContext。BusinessProcessScoped bean的实例会作为流程变量保存到当前流程实例中。 *BusinessProcessScoped bean需要是PassivationCapable（比如序列化）。* 下面是使用流程作用域bean的例子：

@Named

@BusinessProcessScoped

public class BusinessTripRequest implements Serializable {

        private static final long serialVersionUID = 1L;

        private String startDate;

        private String endDate;

        // ...

}

有时，我们需要使用流程作用域bean，没有与流程实例关联，比如启动流程之前。 如果当前流程实例没有激活，BusinessProcessScoped bean实例会暂时保存在局部作用域里（比如，会话或请求， 依赖环境。如果作用域后来与业务流程实例关联了，bean实例会刷新到流程实例里。）

**注入流程变量**

流程变量可以实现用于注入。Activiti-CDI支持以下方式：

* @BusinessProcessScoped使用@Inject [附加修饰] 类型 属性名实现类型安全的注入
* 使用@ProcessVariable(name?)修饰符实现对类型不安全的流程变量的注入：
* @Inject @ProcessVariable Object accountNumber;

@Inject @ProcessVariable("accountNumber") Object account

为了通过EL引用流程变量，我们可以简单实用如下方式：

* @Named @BusinessProcessScoped beans可以直接引用，
* 其他流程变量可以使用ProcessVariables bean来使用：

#{processVariables['accountNumber']}

**接收流程事件**

[**[EXPERIMENTAL]**](http://www.mossle.com/docs/activiti/#experimental)

Activiti可以挂在CDI的事件总线上。这样我们可以使用标准CDI事件机制来监听流程事件。 为了启用activiti的CDI事件支持，需要在配置中启用对应的解析监听器：

<property name="postBpmnParseHandlers">

        <list>

                <bean class="org.activiti.cdi.impl.event.CdiEventSupportBpmnParseHandler" />

        </list>

</property>

现在activiti已经配置成使用CDI事件总线发布事件。下面给出了如何在CDI bean中处理事件的方式。在CDI，我们可以使用@Observes注解声明特定的事件监听器。事件监听是类型安全的。流程事件类型是org.activiti.cdi.BusinessProcessEvent。下面是一个简单事件监听方法的例子：

public void onProcessEvent(@Observes BusinessProcessEvent businessProcessEvent) {

        // handle event

}

监听器可以监听所有事件。如果想限制监听器接收的事件类型，我们可以添加修饰注解：

* @BusinessProcess：限制指定流程定义的事件。 比如：@Observes @BusinessProcess("billingProcess") BusinessProcessEvent evt
* @StartActivity：限制指定环节的事件。比如： @Observes @StartActivity("shipGoods") BusinessProcessEvent evt 在进入id为"shipGoods"的环节时会触发。
* @EndActivity：限制指定环节的事件。比如： @Observes @EndActivity("shipGoods") BusinessProcessEvent evt 在离开id为"shipGoods"的环节时会触发
* @TakeTransition：限制指定连线的事件。

修饰命名可以自由组合。比如，为了接收"shipmentProcess"流程中所有离开"shipGoods"环节的事件， 我们可以编写如下监听方法：

public void beforeShippingGoods(@Observes @BusinessProcess("shippingProcess") @EndActivity("shipGoods") BusinessProcessEvent evt) {

        // handle event

}

默认配置下，事件监听器是同步调用，并在同一个事务环境中。 CDI事务性监听器（只在JavaEE / EJB环境下有效），可以控制监听器什么时候处理事件， 比如，我们可以保证监听器只在事件中的事务成功之后才处理：

public void onShipmentSuceeded(@Observes(during=TransactionPhase.AFTER\_SUCCESS) @BusinessProcess("shippingProcess") @EndActivity("shipGoods") BusinessProcessEvent evt) {

        // send email to customer.

}

**更多功能**

* 流程引擎和服务都可以注入：@Inject ProcessEngine, RepositoryService, TaskService, ...
* 当前流程实例和任务可以注入：@Inject ProcessInstance, Task,
* 当前业务标识可以注入：@Inject @BusinessKey String businessKey,
* 当前流程实例id可以注入 ：@Inject @ProcessInstanceId String pid,

**已知的问题**

虽然activiti-cdi已经使用了SPI，并设计为“可移植扩展”，但是只在Weld下测试过。

**Chapter 17. 集成LDAP**

**Table of Contents**

[用法](http://www.mossle.com/docs/activiti/#ldap_usage)

[用例](http://www.mossle.com/docs/activiti/#ldap_usecases)

[配置](http://www.mossle.com/docs/activiti/#ldap_configuration)

[属性](http://www.mossle.com/docs/activiti/#ldap_properties)

[为Explorer集成LDAP](http://www.mossle.com/docs/activiti/#lda-explorer)

企业通常已经在LDAP系统各种保存了用户和群组信息。 自从5.13版本开始，Activiti提供了一种解决方案， 通过简单的配置就可以告知activiti如何连接LDAP。

在Activiti 5.13之前，Activiti就已经可以集成LDAP了。 然后，5.13的配置简单了很多。 不过，配置LDAP的“老”办法依然有效。更确切的说， 简化的配置其实是基于“老”方法的封装。

**用法**

要想在你的项目中集成LDAP， 在pom.xml中添加如下依赖：

<dependency>

  <groupId>org.activiti</groupId>

  <artifactId>activiti-ldap</artifactId>

  <version>latest.version</version>

</dependency>

**用例**

集成LDAP目前有两大用例：

* 通过IdentityService进行认证。比如，使用Activiti Explorer 通过LDAP登录。
* 获得用户的组。这在查询用户可以看到哪些任务时非常重要。 （比如，任务分配给一个候选组）。

**配置**

集成LDAP是通过向流程引擎配置章节中的configurators注入 org.activiti.ldap.LDAPConfigurator的实例来实现的。 这个类是高度可扩展的： 如果默认的实现不符合用例的话， 可以很容易的重写方法，很多依赖的bean都是可插拔的。

这时一个实例配置（注意：当然， 通过代码创建引擎时，是非常简单的）。 现在不用担心所有参数，我们会在下一章详细讨论。

    <bean id="processEngineConfiguration" class="...SomeProcessEngineConfigurationClass">

        ...

        <property name="configurators">

          <list>

              <bean class="org.activiti.ldap.LDAPConfigurator">

                <!-- Server connection params -->

                <property name="server" value="ldap://localhost" />

                <property name="port" value="33389" />

                <property name="user" value="uid=admin, ou=users, o=activiti" />

                <property name="password" value="pass" />

                <!-- Query params -->

                <property name="baseDn" value="o=activiti" />

                <property name="queryUserByUserId" value="(&(objectClass=inetOrgPerson)(uid={0}))" />

                <property name="queryUserByFullNameLike" value="(&(objectClass=inetOrgPerson)(|({0}=\*{1}\*)({2}=\*{3}\*)))" />

                <property name="queryGroupsForUser" value="(&(objectClass=groupOfUniqueNames)(uniqueMember={0}))" />

                <!-- Attribute config -->

                <property name="userIdAttribute" value="uid" />

                <property name="userFirstNameAttribute" value="cn" />

                <property name="userLastNameAttribute" value="sn" />

                <property name="groupIdAttribute" value="cn" />

                <property name="groupNameAttribute" value="cn" />

              </bean>

          </list>

        </property>

    </bean>

**属性**

下面是org.activiti.ldap.LDAPConfigurator可以配置的属性：

**Table 17.1. LDAP配置属性**

| 属性名 | 描述 | 类型 | 默认值 |
| --- | --- | --- | --- |
| server | LDAP服务器地址。比如'ldap://localhost:33389' | String |  |
| port | LDAP运行的端口 | int |  |
| user | 连接LDAP使用的账号 | String |  |
| password | 连接LDAP使用的密码 | String |  |
| initialContextFactory | 连接LDAP使用的InitialContextFactory名称 | String | com.sun.jndi.ldap.LdapCtxFactory |
| securityAuthentication | 连接LDAP时设置的'java.naming.security.authentication'属性值 | String | simple |
| customConnectionParameters | 可以设置那些没有对应setter的连接参数。 参考http://docs.oracle.com/javase/tutorial/jndi/ldap/jndi.html中的自定义属性。 这些属性用来配置连接池，特定的安全设置，等等。 所有提供的参数都会用来创建LDAP连接。 | Map<String, String> |  |
| baseDn | 搜索用户和组的基“显著名称”（DN） | String |  |
| searchTimeLimit | 搜索LDAP的超时时间，单位毫秒。 | long | 一小时 |
| queryUserByUserId | 使用用户id搜索用户的查询语句。 比如：(&(objectClass=inetOrgPerson)(uid={0})) 这里，LDAP中所有包含'inetOrgPerson'类的 匹配'uid'属性的值都会返回。 如例子中所示，{0}会被用户id替换。 如果只设置一个查询无法满足特定的LDAP设置， 可以选择使用LDAPQueryBuilder， 这样就会提供比单纯使用查询增加更多功能。 | string |  |
| queryUserByFullNameLike | 使用全名搜索用户的查询语句。 比如：(& (objectClass=inetOrgPerson) (|({0}=\*{1}\*)({2}=\*{3}\*)) ) 这里，LDAP中所有包含'inetOrgPerson'类的 匹配first name和last name的值都会返回。 注意{0}会替换为firstNameAttribute（如上所示），{1}和{3}是搜索内容， {2}是lastNameAttribute。 如果只设置一个查询无法满足特定的LDAP设置， 可以选择使用LDAPQueryBuilder， 这样就会提供比单纯使用查询增加更多功能。 | string |  |
| queryGroupsForUser | 使用搜索指定用户的组的查询语句。 比如：(&(objectClass=groupOfUniqueNames)(uniqueMember={0})) 这里，LDAP中所有包含'groupOfUniqueNames'类的 提供的DN（匹配用户的DN）是'uniqueMember'的记录都会返回。 像例子中演示的那样，{0}会替换为用户id。 如果只设置一个查询无法满足特定的LDAP设置， 可以选择使用LDAPQueryBuilder， 这样就会提供比单纯使用查询增加更多功能。 | string |  |
| userIdAttribute | 匹配用户id的属性名。 这个属性用来在查找用户对象时 关联LDAP对象与Activiti用户对象之间的关系。 | string |  |
| userFirstNameAttribute | 匹配first name的属性名。 这个属性用来在查找用户对象时 关联LDAP对象与Activiti用户对象之间的关系。 | string |  |
| userLastNameAttribute | 匹配last name的属性名。 这个属性用来在查找用户对象时 关联LDAP对象与Activiti用户对象之间的关系。 | string |  |
| groupIdAttribute | 匹配组id的属性名。 这个属性用来在查找组对象时 关联LDAP对象与Activiti组对象之间的关系。 | string |  |
| groupNameAttribute | 匹配组名的属性名。 这个属性用来在查找组对象时 关联LDAP对象与Activiti组对象之间的关系。 | String |  |
| groupTypeAttribute | 匹配组名的属性类型。 这个属性用来在查找组对象时 关联LDAP对象与Activiti组对象之间的关系。 | String |  |

下列属性用在希望修改默认行为 或修改组缓存的情况：

**Table 17.2. 高级属性**

| 属性名 | 描述 | 类型 | 默认值 |
| --- | --- | --- | --- |
| ldapUserManagerFactory | 设置LDAPUserManagerFactory的自定义实例，如果默认实现不满足需求。 | LDAPUserManagerFactory的实例 |  |
| ldapGroupManagerFactory | 设置LDAPGroupManagerFactory的自定义实例，如果默认实现不满足需求。 | LDAPGroupManagerFactory的实例 |  |
| ldapMemberShipManagerFactory | 设置LDAPMembershipManagerFactory的自定义实例，如果默认实现不满足需求。 注意它不常用，因为正常情况下LDAP会自己管理关联关系。 | LDAPMembershipManagerFactory的实例 |  |
| ldapQueryBuilder | 设置自定义查询构造器，如果默认实现不满足需求。 LDAPQueryBuilder实例用在LDAPUserManager和LDAPGroupManager中， 执行对LDAP的查询。 默认实现会使用配置的queryGroupsForUser和queryUserById属性。 | org.activiti.ldap.LDAPQueryBuilder的实例 |  |
| groupCacheSize | 组缓存的大小。 这是一个LRU缓存，用来缓存用户的组， 可以避免每次查询用户的组时，都要访问LDAP。 如果值小于0，就不会创建缓存。 默认为-1，所以不会进行缓存。 | int | -1 |
| groupCacheExpirationTime | 设置组缓存的过期时间，单位为毫秒。 当获取特定用户的组时，并且组缓存也启用了， 组会保存到缓存中，并使用这个属性设置的时间。 例如，当组在00:00被获取，过期时间为30分钟， 那么所有在00:30之后进行的查询都不会使用缓存， 而是再次去LDAP查询。因此，所以在00:00 - 00:30 进行的查询都会使用缓存。 | long | 1小时 |

**为Explorer集成LDAP**

* 把上面的LDAP配置添加到activiti-standalone-context.xml中。
* 把activiti-ldap jar放到WEB-INF/lib目录下
* 删除demoDataGenerator bean，因为它会尝试插入数据（集成LDAP不允许这么做）
* 将下面的配置添加到activiti-ui.context的explorerApp bean中：
* <property name="adminGroups">
* <list>
* <value>admin</value>
* </list>
* </property>
* <property name="userGroups">
* <list>
* <value>user</value>
* </list>
* </property>

请使用你自己的配置替换其中的值。需要用到的数据是组的id（通过groupIdAttribute配置）。 上述配置会让'admin'组下的所有用户都成为Activiti Explorer的管理员，用户组也一样。 所有不匹配的组都会当做“分配”组， 这样任务就可以分配给他们。

**Chapter 18. 高级功能**

**Table of Contents**

[监听流程解析](http://www.mossle.com/docs/activiti/#advanced_parseHandlers)

[支持高并发的UUID id生成器](http://www.mossle.com/docs/activiti/#advanced.uuid.generator)

[启用安全的BPMN 2.0 xml](http://www.mossle.com/docs/activiti/#advanced.safe.bpmn.xml)

下面内容将介绍使用Activiti的高级用例，它会超越BPMN 2.0流程的范畴。 因此，对于Activiti的明确目标和经验有利于理解这里的内容。

**监听流程解析**

bpmn 2.0 xml文件需要被解析为Activiti内部模型，然后才能在Activiti引擎中运行。 解析过程发生在发布流程或在内存中找不到对应流程的时候， 这时会从数据库查询对应的xml。

对于每个流程，BpmnParser类都会创建一个新的BpmnParse实例。 这个实例会作为解析过程中的容器来使用。解析过程很简单： 对于每个BPMN 2.0元素，引擎中都会有一个对应的org.activiti.engine.parse.BpmnParseHandler实例。 这样，解析器会保存一个BPMN 2.0元素与BpmnParseHandler实例的映射。 默认，Activiti使用BpmnParseHandler来处理所有支持的元素， 也使用它来提供执行监听器，以支持流程历史。

可以向Activiti引擎中添加自定义的org.activiti.engine.parse.BpmnParseHandler实例。 经常看到的用例是把执行监听器添加到对应的环节，来处理一些事件队列。 Activiti在内部就是这样进行历史处理的。 要想添加这种自定义处理器，需要为Activiti添加如下配置：

<property name="preBpmnParseHandlers">

  <list>

    <bean class="org.activiti.parsing.MyFirstBpmnParseHandler" />

  </list>

</property>

<property name="postBpmnParseHandlers">

  <list>

    <bean class="org.activiti.parsing.MySecondBpmnParseHandler" />

    <bean class="org.activiti.parsing.MyThirdBpmnParseHandler" />

  </list>

</property>

配置到preBpmnParseHandlers的BpmnParseHandler实例 会添加在默认处理器的前面。与之类似，postBpmnParseHandlers会加在后面。 当自定义处理器内部逻辑对处理顺序有要求时就很重要了。

org.activiti.engine.parse.BpmnParseHandler是一个很简单的接口：

public interface BpmnParseHandler {

  Collection<Class>? extends BaseElement>> getHandledTypes();

  void parse(BpmnParse bpmnParse, BaseElement element);

}

getHandledTypes()方法会翻译这个解析器处理的所有类型的集合。 它们都是BaseElement的子类，返回集合的泛型限制也说明了这一点。 你也可以继承AbstractBpmnParseHandler类并重写getHandledType()方法， 这样就只需要返回一个类型，而不是一个集合。这个类也包含了需要 默认解析处理器所需要的帮助方法。 BpmnParseHandler实例只有在解析器访问到这个方法返回的类型时才会被调用。 在下面的例子中， 当BPMN 2.0 xml包含process元素时， 就会执行executeParse方法中的逻辑（这是一个已经完成类型转换的方法， 它替换了BpmnParseHandler接口中的parse方法。）

public class TestBPMNParseHandler extends AbstractBpmnParseHandler<Process> {

  protected Class<? extends BaseElement> getHandledType() {

    return Process.class;

  }

  protected void executeParse(BpmnParse bpmnParse, Process element) {

     ..

  }

}

**重要提示：**在编写自定义解析处理器时， 不要使用任何解析BPMN 2.0结构的内部类。这会很难找到问题。 安全的方法是实现*BpmnParseHandler*接口或集成内部抽象类 *org.activiti.engine.impl.bpmn.parser.handler.AbstractBpmnParseHandler*。

可以（但不常用）替换默认的BpmnParseHandler实例 把解析BPMN 2.0元素解析为Activiti内部模型。 可以通过下面的代码来实现：

<property name="customDefaultBpmnParseHandlers">

  <list>

    ...

  </list>

</property>

举个简单的例子，强行把所有服务任务都设置为异步的：

public class CustomUserTaskBpmnParseHandler extends ServiceTaskParseHandler {

  protected void executeParse(BpmnParse bpmnParse, ServiceTask serviceTask) {

    // Do the regular stuff

    super.executeParse(bpmnParse, serviceTask);

    // Make always async

    ActivityImpl activity = findActivity(bpmnParse, serviceTask.getId());

    activity.setAsync(true);

  }

}

**支持高并发的UUID id生成器**

在一些（非常）高并发的场景，默认的id生成器可能因为无法很快的获取新id区域而导致异常。 所有流程引擎都有一个id生成器。默认的id生成器会在数据库划取一块id范围， 这样其他引擎就不能使用相同范围的id。 在引擎奥做期间，当默认的id生成器发现已经越过id范围时，就会启动一个新事务来获得新范围。 在（非常）极限的情况下，这会在非常高负载的情况下导致问题。 对于大部分情况，默认id生成就足够了。默认的org.activiti.engine.impl.db.DbIdGenerator 也有一个idBlockSize属性，可以配置获取id范围的大小， 这样可以改变获取id的行为。

另一个可以选用的默认id生成器是org.activiti.engine.impl.persistence.StrongUuidGenerator， 它会在本地生成一个唯一的[UUID](http://en.wikipedia.org/wiki/Universally_unique_identifier)， 把它作为所有实体的标识。因为生成UUID不需要访问数据库，所以它在高并发环境下的表现比较好。 要注意默认id生成器的性能（无论好坏）都依赖于运行硬件。

UUID生成器可以像下面这样配置到activiti中：

<property name="idGenerator">

    <bean class="org.activiti.engine.impl.persistence.StrongUuidGenerator" />

</property>

使用UUID id生成器需要以下依赖：

 <dependency>

    <groupId>com.fasterxml.uuid</groupId>

    <artifactId>java-uuid-generator</artifactId>

    <version>3.1.3</version>

</dependency>

**启用安全的BPMN 2.0 xml**

大多数情况下，BPMN 2.0流程发布到Activiti引擎是在严格的控制下的，比如开发团队。 然后，一些情况下，可能需要把比较随意的BPMN 2.0 xml上传到引擎。 这种情况，要考虑恶意用户会攻击服务器， 参考[这里](http://www.jorambarrez.be/blog/2013/02/19/uploading-a-funny-xml-can-bring-down-your-server/)。

为了避免上面链接描述的攻击， 可以在引擎配置中设置：*enableSafeBpmnXml*：

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

**默认这个功能没有开启！**这样做的原因是它需要使用 [StaxSource](http://download.java.net/jdk7/archive/b123/docs/api/javax/xml/transform/stax/StAXSource.html)类。 不幸的是，一些平台（比如，JDK 6，JBoss，等等）不能用这个类（因为老的xml解析实现） 所以不能启用安全BPMN 2.0 xml。

如果Activiti运行的平台支持这项功能，请打开这个功能。