上世纪九十年代以后，随着WfMC联盟的成立，BPM市场群雄逐鹿如火如荼，工作流技术得到了突飞猛进的发展，其中IBM、Oracle等大型软件厂商在工作流领域各扯大旗割据一方。2011年BPMN2.0新规范的发布为各工作流产品互容互通提供了统一的标准，结束了各工作流厂商各自为政相互抵斥的局面。

**什么是BPMN、Workflow？**

* BPM（Business Process Management）——“通过建模、自动化、管理和优化流程，打破跨部门跨系统业务过程依赖，提高业务效率和效果”。
* Workflow——“全部或者部分由计算机支持或自动处理的业务过程”（工作流管理联盟WfMC组织对工作流概念的经典定义）

BPM基本内容是管理既定工作的流程，通过服务编排，统一调控各个业务流程，以确保工作在正确的时间被正确的人执行，达到优化整体业务过程的目的。BPM概念的贯彻执行，需要有标准化的流程定义语言来支撑，使用统一的语言遵循一致的标准描述具体业务过程，这些流程定义描述由专有引擎去驱动执行。这个引擎就是工作流引擎，它作为BPM的核心发动机，为各个业务流程定义提供解释、执行和编排，驱动流程“动“起来，让大家的工作“流”起来，为BPM的应用提供基本、核心的动力来源。

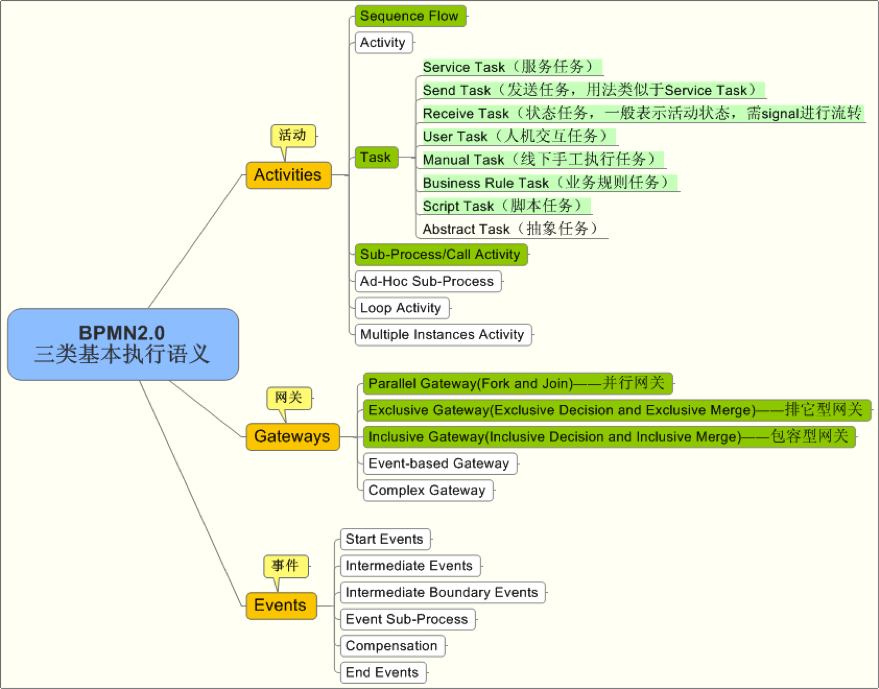
现实工作中，不可避免的存在跨系统跨业务的情况，而大部分企业在信息化建设过程中是分阶段或分部门（子系统）按步实施的，后期实施的基础可能是前期实施成果的输出，在耦合业务实施阶段，相同的业务过程可能会在不同的实施阶段重用，在进行流程梳理过程中，不同的实施阶段所使用的流程描述语言或遵循的标准会有所不同（服务厂商不同），有的使用WfMC的XPDL，还有些使用BPML、BPEL、WSCI等，这就造成流程管理、业务集成上存在很大的一致性、局限性，提高了企业应用集成的成本。

**BPMN2.0规范的引入**

遵循BPMN2.0新规范的工作流产品能很大程度上解决此类问题。BPMN2.0相对于旧的1.0规范以及XPDL、BPML及BPEL等最大的区别是定义了规范的执行语义和格式，利用标准的图元去描述真实的业务发生过程，保证相同的流程在不同的流程引擎得到的执行结果一致。BPMN2.0对流程执行语义定义了三类基本要素，它们是日常业务流程的“三板斧”：

* Activities（活动）——在工作流中所有具备生命周期状态的都可以称之为“活动”，如原子级的任务（Task）、流向（Sequence Flow），以及子流程（Sub-Process）等
* Gateways（网关）——顾名思义，所谓“网关”就是用来决定流程流转指向的，可能会被用作条件分支或聚合，也可以被用作并行执行或基于事件的排它性条件判断
* Events（事件）——在BPMN2.0执行语义中也是一个非常重要的概念，像启动、结束、边界条件以及每个活动的创建、开始、流转等都是流程事件，利用事件机制，可以通过事件控制器为系统增加辅助功能，如其它业务系统集成、活动预警等

这三类执行语义的定义涵盖了业务流程常用的Sequence Flow（流程转向）、Task（任务）、Sub-Process（子流程）、Parallel Gateway（并行执行网关）、ExclusiveGateway（排它型网关）、InclusiveGateway（包容型网关）等常用图元，如图1：



**图1：BPMN2.0三类基本执行语义要素**

现实业务所有的业务环节都离不开Activities、Gateways和Events，无论是简单的条件审批还是复杂的父子流程循环处理，在一个流程定义描述中，所有的业务环节都离不开Task、Sequence Flow、Exclusive Gateway、Inclusive Gateway（如图1中右侧绿色标记所示元素），其中Task是一个极具威力的元素，它能描述业务过程中所有能发生工时的行为，它包括User Task、Manual Task、Service Task、Script Task等，可以被用来描述人机交互任务、线下操作任务、服务调用、脚本计算任务等常规功能。

User Task:生成人机交互任务，主要被用来描述需要人为在软件系统中进行诸如任务明细查阅、填写审批意见等业务行为的操作，流程引擎流转到此类节点时，系统会自动生成被动触发任务，须人工响应后才能继续向下流转。常用于审批任务的定义。

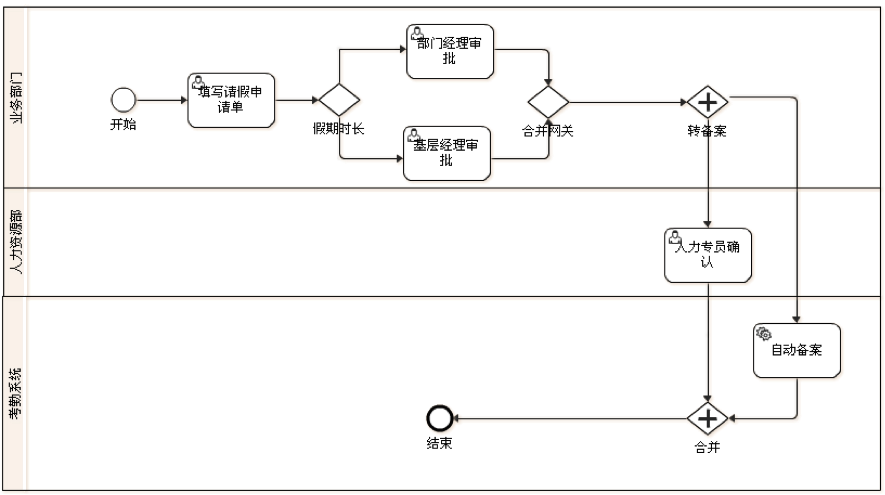
Manual Task:线下人为操作任务，常用于为了满足流程图对实际业务定义的完整性而进行的与流程驱动无关的线下任务，即此类任务不参与实际工作流流转。常用于诸如物流系统中的装货、运输等任务的描述。

Service Task:服务任务，通常工作流流转过程中会涉及到与自身系统服务API调用或与外部服务相互调用的情况，此类任务往往由一个具有特定业务服务功能的Java类承担，与User Task不同，流程引擎流经此节点会自动调用Java类中定义的方法，方法执行完毕自动向下一流程节点流转。另外，此类任务还可充当“条件路由”的功能对流程流转可选分支进行自动判断。常用于业务逻辑API的调用。

Script Task:脚本任务，在流程流转期间以“脚本”的声明或语法参与流程变量的计算，目前支持的脚本类型有三种：juel（即JSP EL）、groovy和javascript。在Activiti5.9中新增了Shell Task，可以处理系统外部定义的Shell脚本文件，也与Script Task有类似的功能。常用于流程变量的处理。

**BPMN2.0流程示例**

BPMN2.0为所有业务元素定义了标准的符号，不同的符号代表不同的含义，以OA应用中请假流程为例，使用标准的BPMN2.0图元定义示意如图2：



**图2：BPMN2.0请假流程定义**

在上述的流程示意图中，所涉及到的执行语义图元主要有表1中的8类：



|  |
| --- |
|  |

**表1：请假流程所用图元**

除了上述Start Event、User Task、Exclusive Gateway、Parallel Gateway、Service Task、End Event标准的BPMN2.0图元外，上述流程图还使用了Lane Set（业务部门、人力资源部、考勤系统），分别表示流程活动所涉及到的部门或角色，Lane的概念和jBPM4中“泳道”的概念一样，都用来表示同一类相似任务的归属者。

应用BPMN2.0标准的一个最显著的特色是，不同阶段的人员，无论是需求分析、概要设计、详细设计或是具体的业务实现，都可在一个流程图上开展工作，避免业务理解存在偏差。一个系统的实现，需求分析人员可以利用BPMN2.0标准图元草绘一下搜集到的需求；然后可以拿给设计人员，讨论出具体的业务需求进行功能设计，由设计人员在草图的基础上逐步细化，并得到需求人员的认同；设计人员又将细化后的流程图交给开发人员，罗列要实现的功能点，指出流程图上各活动节点所具备的行为，设计人员与开发人员依据此图达成共识，进入具体的开发阶段；如果后期请假流程发生更改，仍然是在现有流程图上更改，随着项目的推进，流程图也在不断的演进，但至始至终，项目受众都使用同一个流程图交流，保障需求理解的一致性，一定程度上推动了项目的敏捷性。

**Activiti5支持最新的BPMN2.0规范**

作为支持最新BPMN2.0规范的开源工作流引擎Activit5，实现了对规范的绝大多数图元的定义，能够满足企业工作流的各种复杂应用。它是一个无侵入的、支持嵌入式和独立部署的开源工作流引擎，是Tom Bayen离开jBoss加入Alfresco公司后的另立山头之作，共同开发Activit5的除了Alfresco外还有SpringSource、MuleSoft、Salves、FuseSource、Signavio等公司。从Activiti5.0到当前的5.9（今年3月份发布），版本更新迭代速度很快，新版本功能稳定，性能良好，为开源社区提供了商业工作流之外非常具有竞争力的选择。

**与jBPM5的差别**

值得一提的是，Activiti5与jBPM5都属于业界优秀的开源工作流引擎，都支持BPMN2.0最新规范，均基于Apache License，符合J2EE规范，提供工作流建模、执行以及对流程生命周期过程监控。但两者设计理念和技术组成却有很大不同，见下表2：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 技术组成 | Activiti | jBPM |
| 1 | 数据库持久层ORM | MyBatis3 | Hibernate3 |
| 2 | 持久化标准 | 无 | JPA规范 |
| 3 | 事务管理 | MyBatis机制/Spring事务控制 | Bitronix，基于JTA事务管理 |
| 4 | 数据库连接方式 | Jdbc/DataSource | Jdbc/DataSource |
| 5 | 支持数据库 | Oracle、SQL Server、MySQL等多数数据库 | Oracle、SQL Server、MySQL等多数数据库 |
| 6 | 设计模式 | Command模式、观察者模式等 |  |
| 7 | 内部服务通讯 | Service间通过API调用 | 基于Apache Mina异步通讯 |
| 8 | 集成接口 | SOAP、Mule、RESTful | 消息通讯 |
| 9 | 支持的流程格式 | BPMN2、xPDL、jPDL等 | 目前仅只支持BPMN2 xml |
| 10 | 引擎核心 | PVM（流程虚拟机） | Drools |
| 11 | 技术前身 | jBPM3、jBPM4 | Drools Flow |
| 12 | 所属公司 | Alfresco | jBoss.org |

**表2：Activiti5与jBPM5技术组成**

Activiti5使用Spring进行引擎配置以及各个Bean的管理，综合使用IoC和AOP技术，使用CXF作为Web Services实现的基础，使用MyBatis进行底层数据库ORM的管理，预先提供Bundle化包能较容易的与OSGi进行集成，通过与Mule ESB的集成和对外部服务（Web Service、RESTful等）的接口可以构建全面的SOA应用；jBPM5使用jBoss.org社区的大多数组件，以Drools Flow为核心组件作为流程引擎的核心构成，以Hibernate作为数据持久化ORM实现，采用基于JPA/JTA的可插拔的持久化和事务控制规范，使用Guvnor作为流程管理仓库，能够与Seam、Spring、OSGi等集成。

需要指出的是Activiti5是在jBPM3、jBPM4的基础上发展而来的，是原jBPM的延续，而jBPM5则与之前的jBPM3、jBPM4没有太大关联，且舍弃了备受推崇的PVM（流程虚拟机）思想，转而选择jBoss自身产品Drools Flow作为流程引擎的核心实现，工作流最为重要的“人机交互”任务（类似于审批活动）则由单独的一块“Human Task Service”附加到Drools Flow上实现，任务的查询、处理等行为通过Apache Mina异步通信机制完成。

优劣对比：

从技术组成来看，Activiti最大的优势是采用了PVM（流程虚拟机），支持除了BPMN2.0规范之外的流程格式，与外部服务有良好的集成能力，延续了jBPM3、jBPM4良好的社区支持，服务接口清晰，链式API更为优雅；劣势是持久化层没有遵循JPA规范。

jBPM最大的优势是采用了Apache Mina异步通信技术，采用JPA/JTA持久化方面的标准，以功能齐全的Guvnor作为流程仓库，有RedHat(jBoss.org被红帽收购)的专业化支持；但其劣势也很明显，对自身技术依赖过紧且目前仅支持BPMN2。

**Activiti5设计模式**

命令模式能将命令的发出与执行分开，委派给不同的对象，每一个命令都代表一个指令，其最大的好处是提供了一个公共接口，使得用户可以用同一种方式调用所有的事务，同时也易于添加新事务以扩展系统。

Activiti5大量采用了命令模式，在流程运行期间，所有的指令执行（比如流程部署、流程流转、获取任务等）都使用此模式实现， 其中涉及到四个重要的概念：

Command:Activiti5的命令定义接口，仅有一个execute方法，所有运行期要执行的指令都要实现该接口，定义要执行的具体行为。

CommandContext:命令执行的上下文环境，每个Command的执行都依赖其上下文环境，CommandContext创建了命令执行期间的引擎会话与数据库会话，每个CommandContext都是一个单独的ThreadLocal，执行期间不会受其它线程干预，是线程安全的。

CommandExecutor:命令执行器，负责执行所有的运行时Command。引擎中各项指令的执行（即命令的产生者可能来源于多种对象）都托CommandExecutor处理，仅有一个接口方法：execute(Command command)。 ActivityBehavior:活动行为定义，用于定义BPMN2.0执行语义层的各图元在流程引擎的行为，或称之为所具备的图元特征。与Command的概念类似，仅仅描述“待执行”的指令是什么，会发生什么样的行为，但真正要执行时则由引擎负责驱动。

人机交互任务是业务流程应用中最常用的业务类型，以BPMN2.0中定义的“Task”这个典型元素说明一下命令模式在Activiti5中的应用：

Activiti5针对BPMN2.0的Task Element定义了Task接口，并依据Semantic.xsd执行语义定义了相关任务元素所具有的行为特性，此行为特性通过setActivityBehavior方法进行行为与元素的绑定，这些Behavior在流程引擎驱动流转到活动节点时将被触发，通过execute(ActivityExecution execution)执行ActivityBehavior中指定的操作；

每个活动有若干个Command与之对应，比如ClaimTaskCmd、CompleteTaskCmd、DelegateTaskCmd、SaveTaskCmd、DeleteTaskCmd等，分别表示任务的领取、完成、转交、保存、删除等，这些操作指令的执行结果通过命令执行上下文（CommandContext）得到DAO层的TaskManager将任务对象的变更持久化到数据库中；

引擎不关心要执行什么，凡是实现了Command接口的类都可以通过CommandExecutor执行，除了引擎提供的这些原生的任务指令外，如果业务系统有额外的特性化操作，也可以自定义一组Command，在Command.execute()中自由调用外部服务、发送手机短信、附加任务属性、调用DAO操作数据库等，封装完毕后交由引擎去执行，即可得到希望的结果。同样，如果在业务系统中需要自定义BPMN元素或属性，仅需同步增加ActivityBehavior接口的实现，在解析流程定义文件时将自定义的行为实现与元素（属性）帮定，并缓存之，待引擎驱动到达节点时自动执行。在ActivityBehavior. execute()中依然可以调用各种各样的API已实现特定的业务目的。

此处需要注意的是，Activiti5的CommandContext是包含事务处理的，在每次关闭上下文环境时，会执行事务的提交，但在实际业务系统中，业务事务、引擎事务以及数据库事务应该是被统一到一个事务中去管理，这就需要将Activiti5的事务与业务系统的事务合并。Activiti5通过Spring注入提供了该方式的可行性，引擎内部的事务控制可以委托给业务层去处理，在初始化引擎配置时，将业务系统中定义的DataSource和TransactionManager传递给流程配置的dataSource、transactionManager属性后，Activiti5内部会使用Spring提供的TransactionAwareDataSourceProxy来封装传进来的DataSource，并利用外部的事务管理来接管Activiti5的事务控制，确保了从该DataSource获取的数据库连接与 Spring 定义的事务能够完美地结合，从而实现业务系统与引擎系统事务的集成。

**Activiti5对BPMN2.0执行语义的解析**

Activiti5通过BpmnParse使用SAX方式进行BPMN2.0 XML流程定义文件的解析，是解析的核心类，从根节点开始解析，依次对DefinitionsAttributes、Imports、ItemDefinitions、Messages、Interfaces、和Errors以及ProcessDefinitions各个元素进行解析（以上均是标准的BPMN2.0元素），最后解析负责流程可视化定义的DiagramInterchangeElements元素。每解析一个元素都会判断元素类型，如果是“活动“类型（包括Task、Gateway等），则会为活动设置相应的ActivityBehavior，同时如果流程定义文件中定义了额外属性，Activiti5会自行利用反射机制注入到ActivityBehavior。

除了Command和ActivityBehavior外，Activiti5还大量引入了监听机制（拦截器的概念），目前引擎主要包含四类监听：

* BPMN解析监听——BpmnParseListener，负责对BPMN2.0规范的流程定义文件进行解析控制；
* 任务监听——TaskListener，负责对各类任务的状态以及任务创建、指派责任人、完成任务三类事件进行响应；
* 执行监听——ExecutionListener，对执行过程添加辅助管控功能，对引擎中发生的启动、流转、结束事件进行响应；
* 事务监听——TransactionListener，负责事务控制监听。

PVM流程虚拟机中包含三类事件：Start、End、Take，分别表示流程的启动、流转和结束，流程启动后，流引擎会从Start事件开始执行，通过Take事件，驱动流程流向下一个环节，该“流向”的动作会被PVM运行时的AtomicOperationTransitionNotifyListenerTake监听，该监听会将附加到该流向的所有执行监听依次执行。任务有也有三类事件可以被监听：Create、Assignment、Complete，如果希望在任务被创建或指定了相关责任人或任务完成后增加些额外的辅助功能，可以创建TaskListener接口的实现类，并将其定义到执行定义元素中，Activiti5会处理这一切。这些监听本质上都算是活动的附加代理，在现有操作的基础上额外增加一个管理控制手段以达到特殊的目的，ActivityBehavior从另一个角度来看也是一种代理，都是由DelegateInvocation负责调用执行，它主要用来提供用户代码调用的上下文环境并负责控制实际调用，Activiti5为其提供了五个实现：ActivityBehaviorInvocation、ExecutionListenerInvocation、ExpressionInvocation、JavaDelegateInvocation、TaskListenerInvocation。

Activiti5提供的Command、ActivityBehavior、Listener等接口为引擎的功能扩展提供了方便，如果业务系统的功能不能满足时可以实现这些接口，以无侵入的方式扩展Activiti5，利用这些扩展接口，可以在其执行方法中完成很多业务逻辑，如权限校验、与业务系统的交互、与外部系统集成调用，甚至替换原有功能偷梁换柱暗度陈仓。

**Activiti5 API应用**

ProcessEngine是Activiti系统的核心接口，七类基础服务接口通过ProcessEngine获取，均采用链式API方式，直观明了，易于使用:

**RepositoryService：**

流程资源服务的接口，主要用于对流程定义的部署、查询和删除操。新流程的部署使用createDeployment().addResourceXXX().deploy()方法；已部署流程的查询使用createDeploymentQuery()附加查询条件的方式获取；另外可以使用deleteDeployment和deleteDeploymentCascade方法进行流程的删除或级联删除。

**TaskService：**

任务服务接口，该接口暴露了管理人机交互任务的操作，如任务领取（claiming）、任务完成（completing）和任务指派（assigning），还包括对任务的创建、查询、保存、删除等。

**RuntimeService：**

运行时服务主要用于启动或查询流程实例，以及流程变量、当前激活状态活动的查询、流程实例的删除等。流程在运行过程中所产生的东西都可以使用该接口进行相关处理。

**HistoryService：**

流程历史的服务接口。提供对历史流程实例、历史任务的查询和删除操作，从提供的API来看，历史流程的查询其提供了finished和unfinished流程的查询，即是说，HistoryService提供了对已完成和当前正在执行流程的活动/任务查询，这一点似乎与runtimeService提供的查询有些冲突，但其实是有差别的，运行时的信息仅包含任意时刻活动的实际运行状态信息（是从流程运行执行性能上考虑的），而历史信息是对已经固化的信息做简单查询而优化的，其所持有的对象是不同的。

**IdentityService：**

用户、组管理服务接口，用于管理Group、User的增删改查，并维护Membership，涉及到的API有newUser、newGroup、saveUser、saveGroup、createMembership以及相关的deleteXXX方法。

**FormService：**

表单服务用于访问表单数据以及在启动新的流程实例时或完成任务时所需的渲染后的表单，提供UI界面辅助用户填写相关值以保存至流程变量。该服务在实际业务应用中并不常用，属于引擎的非核心服务。

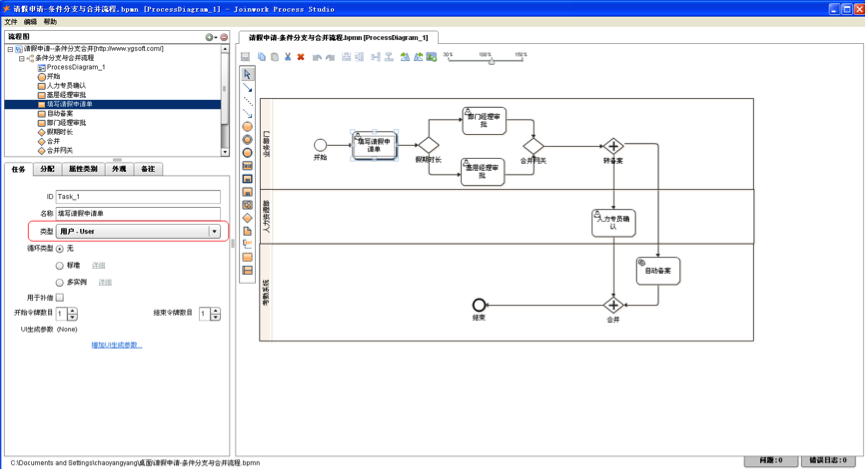
**ManagementService：**

提供流程管理和控制操作的接口服务，和业务流程的运行没有关联关系，比如查询数据库本身的内容、Activiti的版本及序列生成ID规则等，属于引擎的非核心服务。

Activiti5工作流引擎应用需要首先掌握的是配置及API基础应用，下面以上述BPMN2.0请假流程为例，简述Activiti5在具体系统中的应用。

Step1：绘制请假流程图

请假流程图使用标准的BPMN2.0图元进行流程定义，可以使用任何XML编制工具编写（导入XSD可以为编写过程提供代码提示），建议使用Joinwork Process Studio进行可视化编制，如图3:



**图3：Step2配置Activiti5环境**

流程引擎环境主要涉及到三个方面：数据源、事务管理以及流程引擎配置实例。通常流程引擎仅是业务系统的一个核心模块，其数据源和事务都要委托给业务平台，在流程引擎配置定义中，可以通过ref将业务系统的dataSource和transactionManager注入给Activiti5的引擎配置：

<beanid="processEngineConfiguration"class="org.activiti.spring.SpringProcessEngineConfiguration">

<propertyname="dataSource"ref="dataSource" />

<propertyname="transactionManager" ref="transactionManager"/>

<propertyname="databaseSchemaUpdate"value="true"/>

<propertyname="jobExecutorActivate"value="false"/>

</bean>

如果是在OSGi环境中应用Activiti5，还需要将业务环境中注册的dataSource和transactionManager作为OSGi Service引入到当前Bundle，然后再进行processEngineConfiguration的配置：

<osgi:referenceid="dataSource"interface="javax.sql.DataSource"/>

<osgi:referenceid="transactionManager"interface="org.springframework.transaction.PlatformTransactionManager"/>

Step3：部署请假流程定义文件到Activiti5环境

利用流程引擎提供的RepositoryService接口实现流程的部署：

//通过ProcessEngine获取repositoryService

RepositoryServicerepositoryService = processEngine.getRepositoryService();

//使用repositoryService进行新流程部署

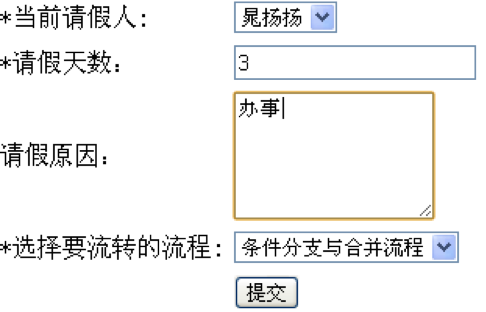
repositoryService.createDeployment()

.addClasspathResource("请假申请-条件分支与合并流程.bpmn20.xml")

.deploy();

Step4：创建请假单页面输入请假天数及原由，启动流程

编写html表单输入界面，然后使用Ajax提交请求，由Servlet根据请求参数创建新流程实例，启动流程后界面如图4：



**图4：流程启动后的界面**

输入请假天数及原因，如果天数大于等于3天，则走“部门经理审批路由“分支，利用jQuery绑定”提交“按钮的操作：

$('#startProcess').click(function(){

varurl = '/com.ygsoft.process.demo/ProcessEngineServlet?operate=start&'+$('#inputform').serialize();

//以UTF8方式提交：

$.ajax({

url:url,

type:"POST",

dataType:"json",

contentType:"application/x-www-form-urlencoded;charset=utf-8",//此参数避免中文乱码

success:function(data){

if(data.success){

alert('您的单据已提交，流程ID：'+data.id);

$('#inputform').hide();

$('#viewTodo').show(2000);

}else{

alert('您的单据未提交成功');

}

}

});

})

Backend端利用RuntimeService接口创建新的流程实例：

// 通过ProcessEngine获取runtimeService

RuntimeServiceruntimeService = processEngine.getRuntimeService();

// 使从Request中获取请求参数，用于构造流程启动参数

Map<String, Object>params = newHashMap<String, Object>();

String processKey = request.getParameter("processKey");

int day = Integer.parseInt(request.getParameter("day"));

String reason = request.getParameter("reason");

params.put("day", day);

params.put("user", user);

params.put("reason", reason);

// 使用runtimeService启动流程实例（将参数做为流程变量处理）

ProcessInstanceprocessInstance = runtimeService.startProcessInstanceByKey(processKey,params) ;

Step5：获取审批人待办任务

利用TaskService接口可是实现指配给自己的以及候选任务：

// 通过ProcessEngine获取taskService

TaskServicetaskService = processEngine.getTaskService();

// 使用taskService根据用户ID获取候选任务

List<Task> tasks = taskService.createTaskQuery()

.taskAssignee(user)

.orderByTaskCreateTime()

.desc()

.list();

将查询到的List通过Gson转换成json数组传递到前端，由jQuery解析并显示到界面。

还有一种情况是查询分配给某个组或某个人的候选任务：

List<Task> tasks = taskService.createTaskQuery()

.processInstanceId(processInstance.getId())

.taskCandidateGroup("xxxGrp")

.list();

// 或

List<Task> tasks = taskService.createTaskQuery()

.processInstanceId(processInstance.getId())

.taskCandidateUser("xxxUser")

.list();

Step6：审批人查看任务明细

任务明细除了包含Task本身的信息（如任务名称、描述以及流程变量等）外，还要动态显示当前激活任务的可视化流程图。

Task信息可以从List中获取，可视化流程图可以利用以下方式输出至前端：

// 根据当前Task获取流程定义对象 ProcessDefinitionEntityprocessDefinition = (ProcessDefinitionEntity) ((RepositoryServiceImpl) repositoryService) .getDeployedProcessDefinition(task.getProcessDefinitionId()); // 利用ProcessDiagramGenerator生成当前激活任务的图片流 InputStreamdefinitionImageStream = ProcessDiagramGenerator.generateDiagram(processDefinition, "png", runtimeService.getActiveActivityIds(task.getProcessInstanceId())); // 将图片流生成byte[]数组 byte[] diagramBytes = IoUtil.readInputStream(definitionImageStream,null); response.setContentType("image/png");// 设置浏览器响应的ContentType ServletOutputStream out = response.getOutputStream(); out.write(diagramBytes);// 输出至前端 out.close();

Step7：完成审批任务

审批人在查看请假申请单后，填写审批意见后以Ajax方式提交“完成任务“请求；Servlet利用TaskService进行任务的提交：

//先完成当前任务：

Map<String, Object>params = taskService.getVariables(taskId);

String reviewMessage = request.getParameter(“msg”);

String choice = request.getParameter(“choice”);

params.put(“msg”, choice+“-”+user+“-”+reviewMessage); taskService.complete(taskId, params);

待第一个的审批工作完成后，流程引擎会产生Task事件，经由并行网关处理后，系统将生成“人力专员确认“的UserTask任务和”自动备案“的ServiceTask任务，其中ServiceTask任务将由系统自动执行，”人力专员确认“任务依然通过Step5、6、7完成，待这两个任务都完成后，两条路由分支由”合并“路由流转到”结束“节点，至此，流程结束。

通过以上API的应用分析，Activiti5 API构成清晰，针对性更强，不同的功能由相应的服务接口完成，访问接口更友好。

**总结**

BPMN2.0是一个工作流业界标准，规范了大型厂商和开源工作流产品的实现，Activiti5实现了该标准的大部分图元定义和执行语义解释，功能强大，Activiti5可以与IBM、Oracle等大型商用工作流产品流程引擎节点的核心功能媲美，并且为了简化应用、扩充原有功能，Activiti5又自定义了6个扩展元素和15个扩展属性，这些元素和属性能够与BPMN规范相互组合可以实现更多、更实用的业务功能。

笔者通过技术组成、对BPMN规范的覆盖率、API应用友好性、社区支持度、第三方组件依赖程度以及可扩展性六个方面进行分析和比对，Activiti5的综合实力较强。对于如何选型符合BPMN标准的工作流产品，这是一个仁者见仁智者见智的问题，一方面依赖于各个公司对工作流技术方面的历史积累，另一方面也要针对具体项目具体情况区别对待。但如果对于一个全新的项目或对jBPM3、4设计理念认同的公司，不妨考虑Activiti5