**JBPM流程开发指导文档**

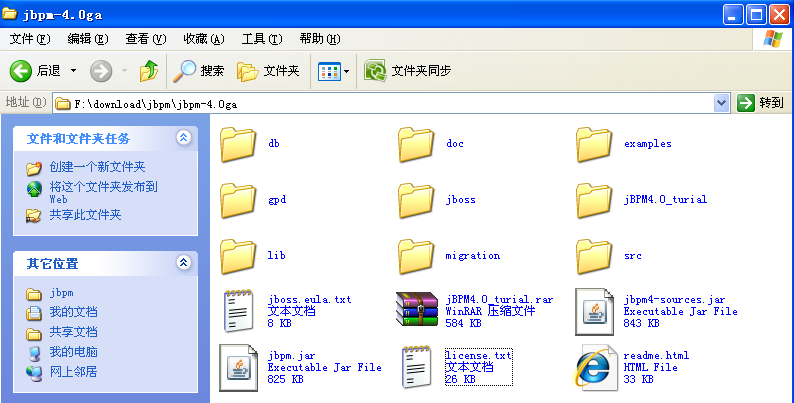
# JBPM工作流引擎介绍

JBPM是JBoss开源的一个比较知名的项目，是一个灵活的、可扩展的工作流引擎，它拥有直观的流程定义语言，可用任务、异步等待状态、定时器、自动化动作等来表示业务流程图，并且把这些业务操作绑定在一起，使之完成流程中的各项操作。由于采用Java语言开发，并且采用了开源的Hibernate框架，所以可以做到跨平台部署，跨数据库。并获得广大软件开发商的认可，并且越来越多的业务系统也采用该框架作为首先的工作流引擎。

# JBPM开发环境准备

首先从JBPM代码托管地址上（http://sourceforge.net/projects/jbpm/

）下载４.０ＧＡ版本，下载完后，解压到本地目录（installDir），如下图所示：



学习资料可以查看其下的doc目录下的devguide与userguide.

以下为该目录的主要说明：

* + - db: JBPM数据创建脚本
    - doc: 用户指导，API文档，开发手册
    - examples: 用户指导中的流程例子
    - gpd: Jbpm流程设计器（eclipse 插件）
    - jboss:Jbpm的Jboss发布插件
    - lib: 第三方依赖包及一些Jbpm的发布包
    - src: Jbpm源代码
    - jbpm.jar:　Jbpm的主要二进制包

## JBPM4环境要求

JDK 1.5 以上版本，如需要执行Ant脚本，需要Ant 1.7以上的版本。

## 安装开发环境

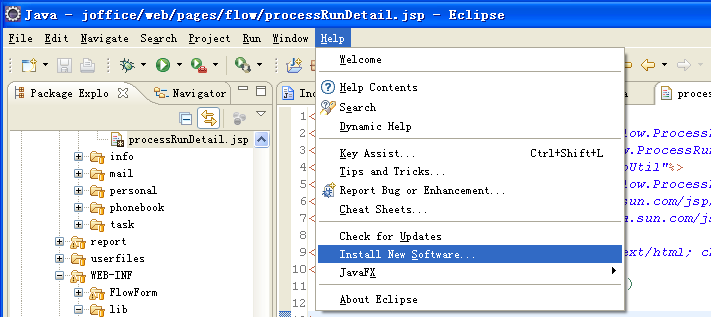
1. 安装eclipse

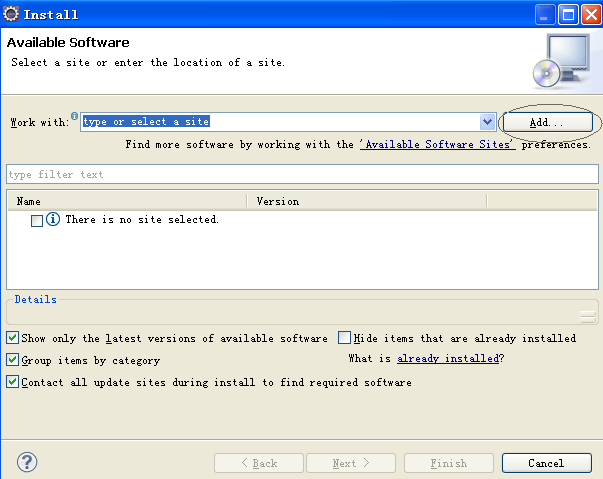
运行配置基于eclipse 3.5开发环境，开发工具可在以下地址下载：

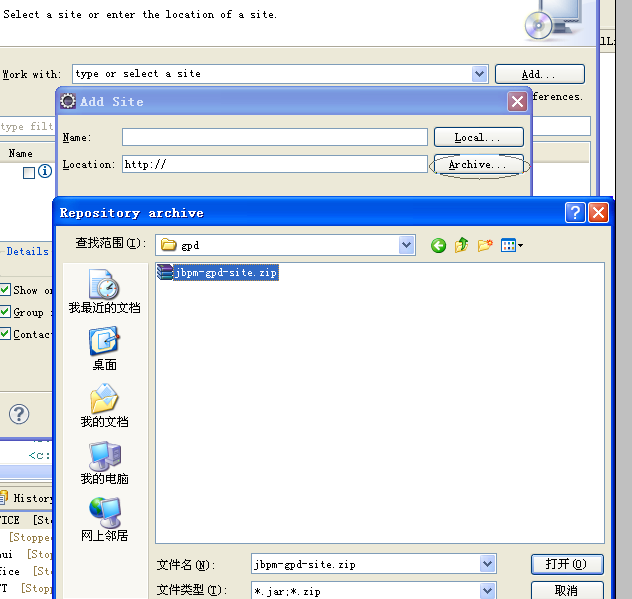
<http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/galileo/SR1/eclipse-jee-galileo-SR1-win32.zip>

1. 安装Jbpm设计器插件

* 添加update site 至Eclipse中
* Help🡪Install New Software
* 点击“Add”
* 在弹出对话框，选择“Archive…”
* 导航到Jbpm的解压目录选择gpd/jbpm-gpd-site.zip，然后点击“打开”。
* 在Add Site 对话框选择“ＯＫ”，这将进入安装
* 选择jPDL 4 GPD的Update Site，点击“Finish”
* 选择Approval the license
* 安装完成后，重启eclipse



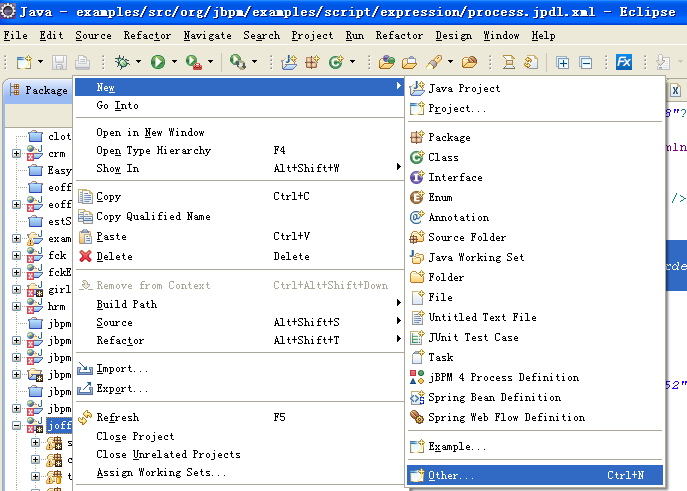


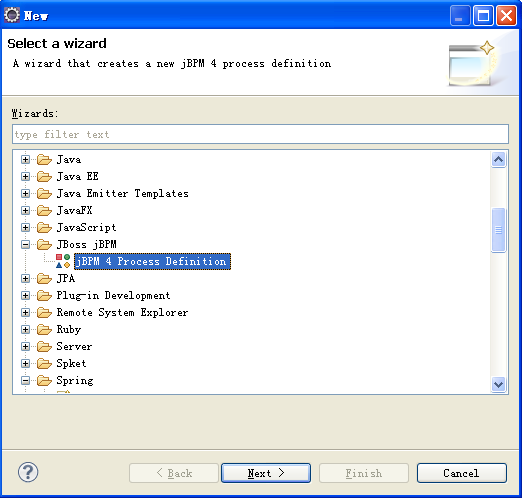


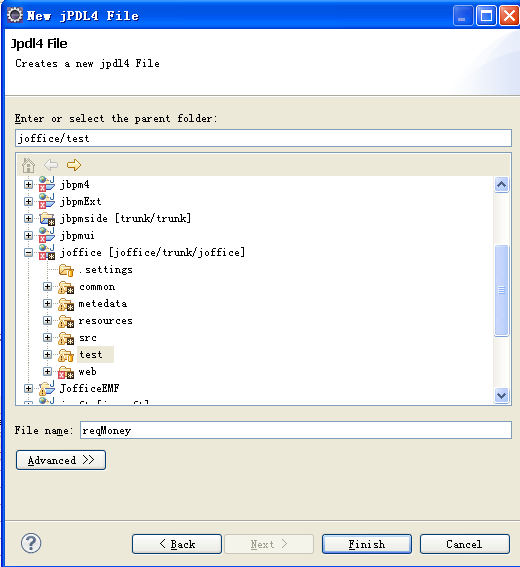
## 设计第一个流程

安装好Jbpm设计器的插件后，就可以开始自己的流程设计之旅。

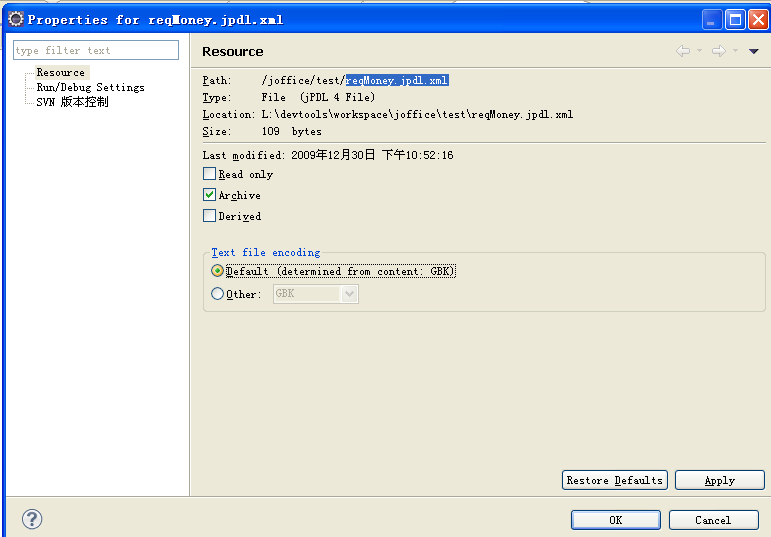
新建流程



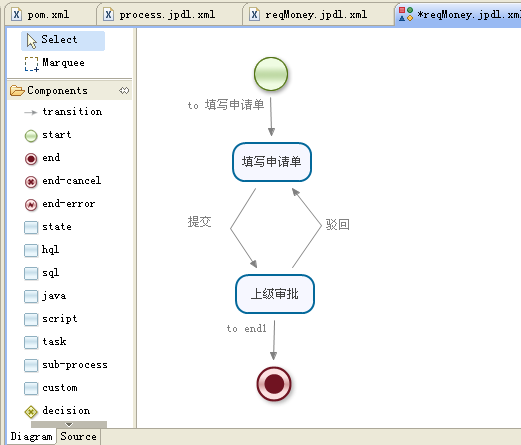




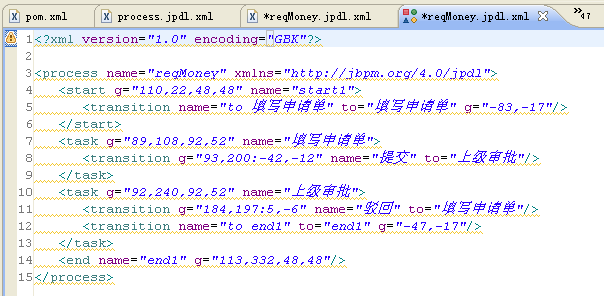
点“Finish”完成后，需要注意自己的开发环境的编码，即reqMoney.jpdl.xml需要设置为GBK以使其支持中文标签。



进入设计器设计流程，如下所示：



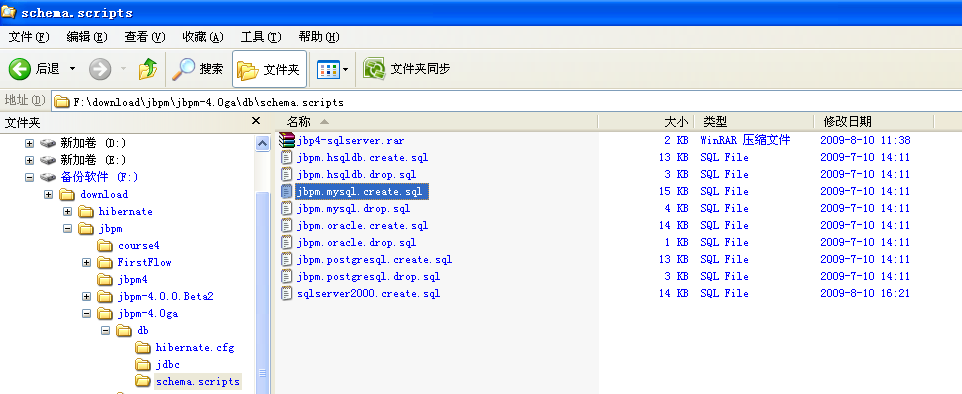
由于使用了中文，因而需要进入source模式下，把XML的encoding设置为”GBK”， 否则保存时，会出现乱码。



# JBPM整合

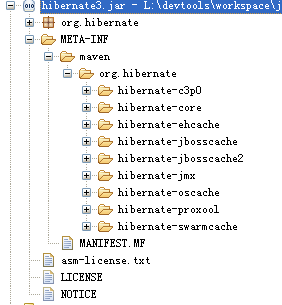
执行创建的脚本

执行db目录下的对应的数据库类型的创建脚本，如下图所示：



## 加入依赖包

1. Jbpm.jar
2. 由于Jbpm使用Hibernate作为ORM底层，因而需要把Hibernate的依赖包也加入来。Hibernate依赖的库可以从hibernate.jar包看到：



1. 加入流程有脚本引擎支持，即在Jbpm4的流程定义中引入类似如下的脚本方式，如条件表达式：

<script name=*"invoke script"*

expr=*"Send packet to* ***#{order.address}****"*

var=*"text"*　g=*"96,16,104,52"*>

这需要引入juel-engine.jar,juel-impl.jar,juel.jar。

注意：引入juel.jar包后，在tomcat6的环境下，运行时，会与其本身的el-api.jar有冲突，解决办法是：把tomcat6下的lib的el-api.jar替换为juel.jar，同时不要在应用程序下加入juel.jar。

## 整合Jbpm至应用的事务管理

Jbpm底层使用了自己Hibernate的SessionFactory，而我们的程序中也有自己Hibernate的SessionFactory，若在执行我们的程序时，需要调用JBPM的相关流程业务，就会相当在两个Connection完成对数据库的操作，这对于事务管理上带来了困难。因此，我们把Jbpm中的SessionFactory指向我们的SessionFactory来。

### 引入Spring的配置管理(app-jbpm.xml)

同时，由于Jbpm提供对于Spring的较好支持，所以我们加入app-jbpm.xml，其配置如下：

**<?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-2.5.xsd"***

**default-lazy-init=*"true"*>**

**<bean id=*"jbpmConfiguration"* class=*"org.jbpm.pvm.internal.cfg.SpringConfiguration"*>**

**<constructor-arg value=*"jbpm.cfg.xml"* />**

**<property name=*"sessionFactory"* ref=*"sessionFactory"*/>**

**</bean>**

**<bean id=*"processEngine"* factory-bean=*"jbpmConfiguration"* factory-method=*"buildProcessEngine"* />**

**<bean id=*"repositoryService"* factory-bean=*"processEngine"* factory-method=*"getRepositoryService"* />**

**<bean id=*"executionService"* factory-bean=*"processEngine"* factory-method=*"getExecutionService"* />**

**<bean id=*"taskService"* factory-bean=*"processEngine"* factory-method=*"getTaskService"*/>**

**<bean id=*"historyService"* factory-bean=*"processEngine"* factory-method=*"getHistoryService"*/>**

**<bean id=*"identityService"* factory-bean=*"processEngine"* factory-method=*"getIdentityService"*/>**

**</beans>**

### 引入jbpm.cfg.xml配置文件

以上jbpmConfiguration Bean 作为Jbpm与Spring的整合的总入口，其配置来自jbpm.cfg.xml文件，我们把该文件定义为：

<jbpm-configuration>

<import resource=*"jbpm.default.cfg.xml"* />

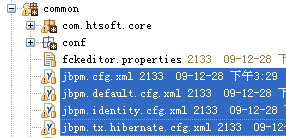
<import resource=*"jbpm.tx.hibernate.cfg.xml"* />

<import resource=*"jbpm.jpdl.cfg.xml"* />

<import resource=*"jbpm.identity.cfg.xml"* />

</jbpm-configuration>

以上其引用的jbpm的xml文件均可以在jbpm.jar包中找到，也可以在Jbpm的解压目录下的src找到。把对应的文件拷至开发目录下的src或其他源代码的根目录。如：



### 重写jbpm.default.cfg.xml

去掉jbpm.default.cfg.xml中的自带的Hibernate配置，使其直接用我们Spring中的Hibernate配置，如下所示：

<!--

<hibernate-configuration>

<cfg resource="jbpm.hibernate.cfg.xml" />

</hibernate-configuration>

<hibernate-session-factory />

-->

<script-manager default-expression-language=*"juel"*

default-script-language=*"juel"*>

<script-language name=*"juel"* factory=*"org.jbpm.pvm.internal.script.JuelScriptEngineFactory"* />

</script-manager>

*说明：在这里也可以看到，其是指定使用juel的脚本引擎库，所以需要把juel的依赖库加至系统中来。*

### 重写jbpm.tx.hibernate.cfg.xml

修改jbpm.tx.hibernate.cfg.xml的配置为以下

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

<jbpm-configuration>

<process-engine-context>

<command-service>

<retry-interceptor />

<environment-interceptor />

<spring-transaction-interceptor />

</command-service>

</process-engine-context>

<transaction-context>

<hibernate-session current=*"true"*/>

</transaction-context>

</jbpm-configuration>

### 重写jbpm.identity.cfg.xml配置

修改jbpm.identity.cfg.xml的配置如下，以使其识别我们的系统用户为流程的参与者。

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

<jbpm-configuration>

<transaction-context>

<object class=*"com.htsoft.core.jbpm.UserSession"*/>

</transaction-context>

</jbpm-configuration>

说明：

UserSession必须实现IdentitySession（org.jbpm.pvm.internal.identity.spi.IdentitySession）接口，对于使流程识别我们的系统的用户与角色，则需要用户(AppUser)实现其org.jbpm.api.identity.User接口，角色(AppRole)实现其org.jbpm.api.identity.Group接口。

【说明】

完成以上配置后，现在我们可以基于Spring来进行JBPM的流程管理，在应用程序中，我们是通过app-jbpm.xml中的Bean提供的接口来完成的。以下这些Bean的说明：

ProcessEngine：流程引擎服务总入口，可以获得其他Service

RepositoryService：流程定义库服务类

ExecutionService：流程执行服务类

TaskService：流程任务服务类

HistoryService：历史任务服务类

IdentityService：用户与角色服务类

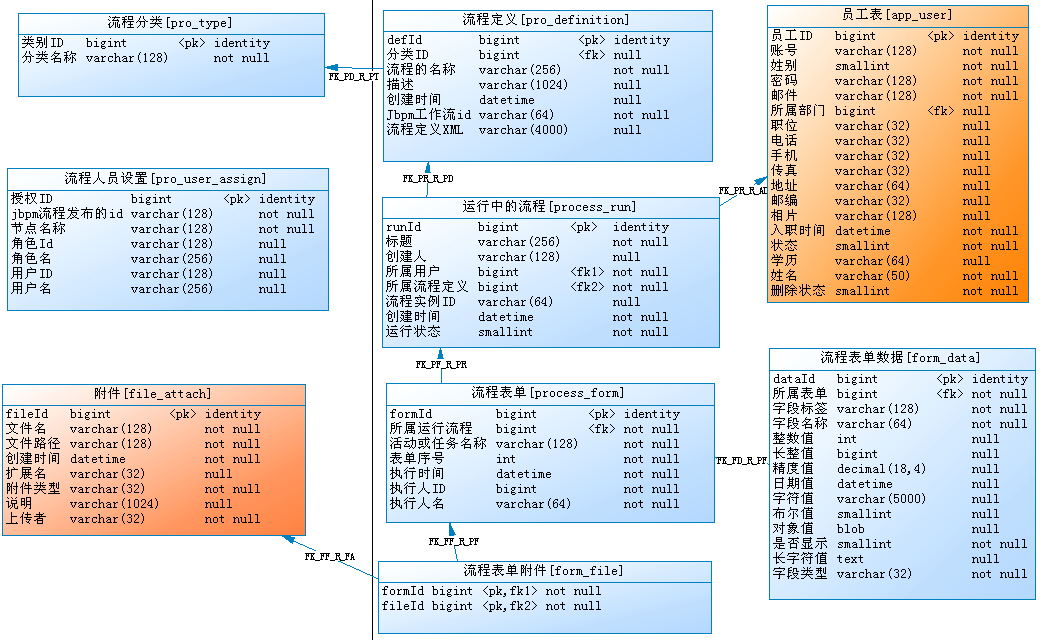
# JBPM工作流扩展

## JBPM常用术语

1. 工作流引擎组件：核心组合负责生成流程运行的各实例及数据，以及管理流程的运行及结束。
2. 流程定义：预先定义好的业务逻辑流程。
3. 流程实例：业务流程实际运行过程
4. 流程库（Repository）存放所有的流程定义，也可以认为是流程发布的存放库。
5. 任务：逻辑上定义给需要流程的参与人员处理的工作
6. 用户组：代表一组用户，可认为某一类型或某些角色的所有用户。
7. 用户：参与流程任务的特定用户
8. 候选用户：参与流程任务的候选用户，即某些任务在某些情况下可以转给候选用户来处理。
9. 待办任务：指流程参与者目前需要处理的任务
10. 任务表单：用户在处理某任务时，需要填写内容，称之为表单。

由于JBPM只是处理流程，所以在流程的处理过程中，其任务表单需要由开发人员来定制开发，而表单的数据也没有进行较好的存储，这为查询及管理流程任务带来很大不便，因而需要进行扩展。

流程的表设计如下：



## 流程扩展表说明

1. 流程分类[pro\_type]：为了方便流程管理而进行了分类。
2. 流程定义[pro\_definition]：是本系统的流程定义，其中字段“流程定义XML”是JBPM中的流程定义的格式。
3. 流程运行[process\_run]：当流程运行时，会有相应的运行的显示，特别是在国内企业的使用习惯中，都希望看到是哪一个流程（流程的标题），哪一个用户启动的，其运行状态如何。该表就是记录这些信息以方便查询。
4. 流程表单[process\_form]运行过程中，流程会进行不同的任务，每个任务都会有相应的表单及数据需要存储，而该表则存储表单的信息，同时也存储不同的执行人。
5. 流程表单数据[form\_data]任务表单的数据。
6. 流程表单附件[form\_file]放置流程表单的数据。

## 流程任务的人员授予：

流程任务最终都需要指定人员来执行，任务的执行人员可以在流程的定义文件（ＸＭＬ）中来指定，这种方式我们也称之为前绑定角色或人员，但是那样会导致流程在运行起来修改指定的执行人员并不方便，要加强这种灵活，则需要进行使用后绑定的方式，则流程发布后，可以再通过程序或界面去设置其每个任务执行的人员或角色，或在任务执行的过程中，指定其对应的执行人员。

后设置的每个任务指定的执行人员或角色的信息则存储在pro\_user\_assign表中，其设置的界面如下所示：



指定了执行人后，当流程进入该任务后，会自动分配给该用户来执行。

也可以在提交任务时，指定一个参数（flowAssignId），设置为用户的id，执行时会优先设置，这种方式适合于在任务表单中指定下一个任务执行的人员。

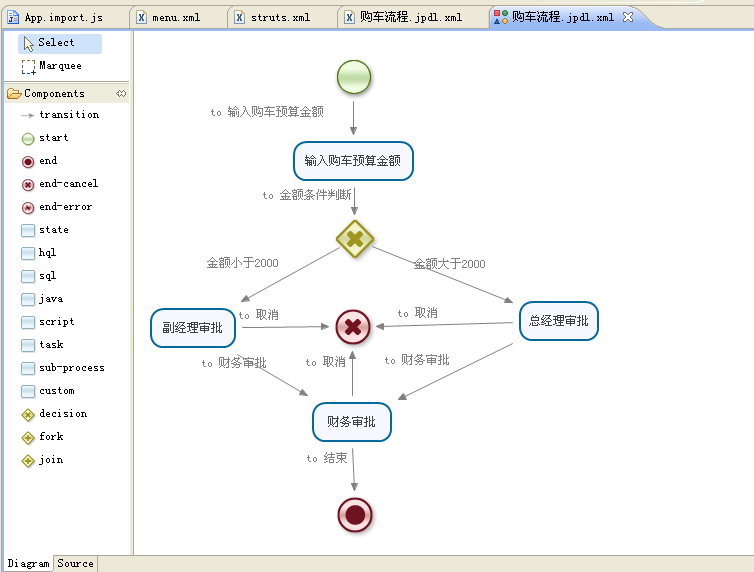
其实现的原理请参考JbpmServiceImpl.java中的assignTask方法，其任务的指派过程如下所示：



## 最终的流程发布

### 定义流程

使用Jbpm Eclipse plugin定义流程如下所示：



其定义的XML文件如下所示：

<?xml version=*"1.0"* encoding=*"GBK"*?>

<process key=*"buyCarFlow"* name=*"购车流程"* xmlns=*"http://jbpm.org/4.0/jpdl"*>

<start g=*"196,22,48,48"* name=*"开始"*>

<transition g=*"-143,-11"* name=*"to 输入购车预算金额"* to=*"输入购车预算金额"*/>

</start>

<task g=*"153,104,133,52"* name=*"输入购车预算金额"*>

<transition g=*"-92,-11"* name=*"to 金额条件判断"* to=*"金额条件判断"*/>

</task>

<decision g=*"197,184,48,48"* name=*"金额条件判断"*>

<transition g=*"-83,-17"* name=*"金额小于2000"* to=*"副经理审批"*>

<condition expr=*"#{reqAmount&lt;2000}"*/>

</transition>

<transition g=*"-28,-16"* name=*"金额大于2000"* to=*"总经理审批"*>

<condition expr=*"#{reqAmount&gt;2000}"*/>

</transition>

</decision>

<task g=*"10,271,98,52"* name=*"副经理审批"*>

<transition g=*"-47,-17"* name=*"to 取消"* to=*"取消"*/>

<transition g=*"-71,-17"* name=*"to 财务审批"* to=*"财务审批"*/>

</task>

<task g=*"379,264,92,52"* name=*"总经理审批"*>

<transition g=*"-47,-17"* name=*"to 取消"* to=*"取消"*/>

<transition g=*"-71,-17"* name=*"to 财务审批"* to=*"财务审批"*/>

</task>

<task g=*"172,365,92,52"* name=*"财务审批"*>

<transition g=*"-47,-17"* name=*"to 取消"* to=*"取消"*/>

<transition g=*"-47,-17"* name=*"to 结束"* to=*"结束"*/>

</task>

<end-cancel g=*"195,272,48,48"* name=*"取消"*/>

<end g=*"197,460,48,48"* name=*"结束"*/>

</process>

说明：

1．注意以下这部分的定义

<transition g=*"-83,-17"* name=*"金额小于2000"* to=*"副经理审批"*>

<condition expr=*"#{reqAmount&lt;2000}"*/>

</transition>

<transition g=*"-28,-16"* name=*"金额大于2000"* to=*"总经理审批"*>

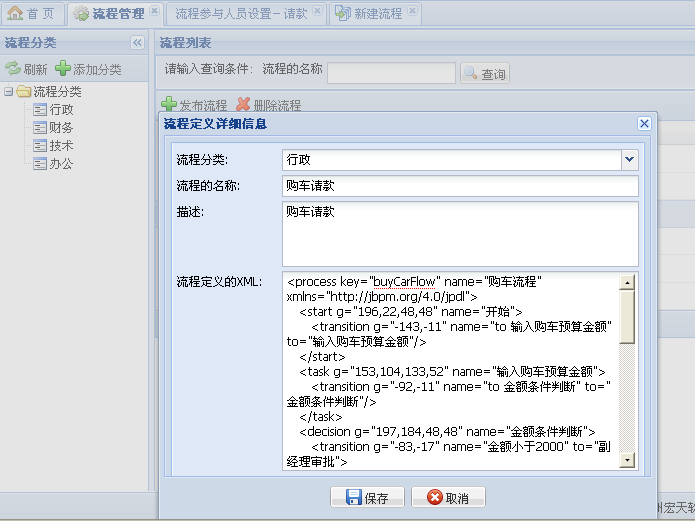
<condition expr=*"#{reqAmount&gt;2000}"*/>

</transition>

其使用了表达式来进行条件的跳转, *reqAmount代表为流程表单中提交的数据字段名称，在后面我们会在表单中使用该字段名称。*

### 在系统中发布流程

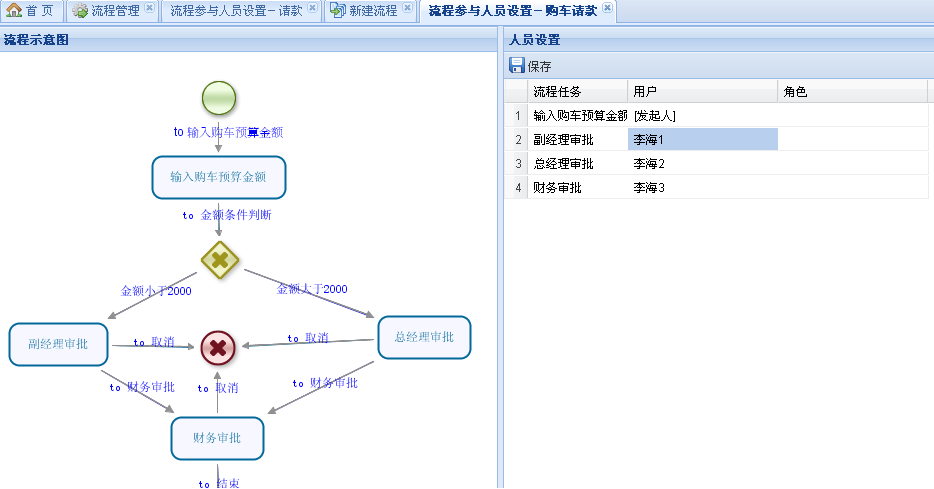
点击流程管理菜单中的流程管理，点击左流程左分类其中一项，再点“发布流程”按钮，如下所示：



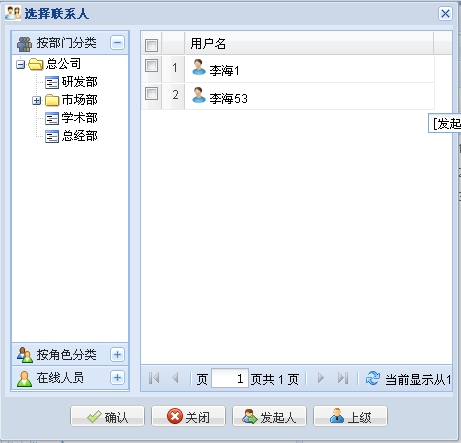
把以上XML文件拷至该对话框。并且填写好流程名称为“购车请款”，注意，流程名称写好后，最好不要更改，否则后面的流程表单的路径也需要更改。再点击“保存”按钮，则成功发布了一个流程定义。

### 分配任务节点的处理人员或角色

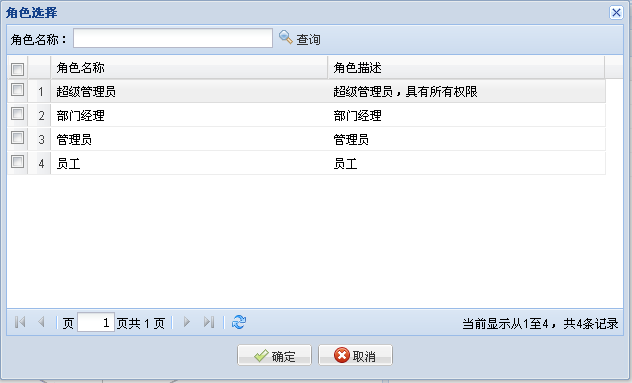
在流程管理列表中，点击“购车请款”流程行中的“”按钮，进入流程任务人员的设置界面，如下图所示：



双击“人员设置”面板中的用户列下的表格，会相应弹出该任务的人员选择对话框，如下图所示：



双击“人员设置”面板中的角色列，选择相应的角色。

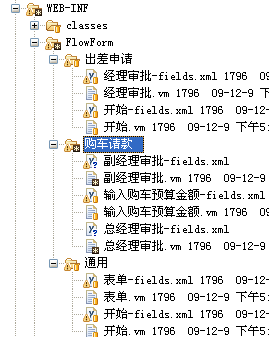


说明：

在该设置中，人员的选择可以为发起人，即该任务会自动转给流程发起人来处理，另外还可以设置“上级”，即流程会流转至我的上级，若我的上级有多个人，即可以由其中一个锁定任务来执行。任务在执行过程中还可以转移由其他人来执行，如某人出差了，可以转至其他人来执行

进入程序的WEB-INF/FlowForm/目录，为该流程定义好表单及表单的字段映射，

以流程的名称建立一个目录，如“购车请款”，如图所示。



并且以各个任务的名称定义表单，

如流程开始前填写的表单统一命名为“开始.vm”,其映射的字段文件为“开始-fields.xml”,若没有为开始或其他任务定义这两个对应的文件,其会默认使用“通用”目录下的表单及映射文件。

在请款流程里，我们需要在“输入购车预算金额”任务里填写请款内容，所以我们在“购车请款”目录下建立了“输入购车预算金额.vm”文件及“输入购车预算金额-fields.xml”

以下为两个文件的对应代码：

输入购车预算金额.vm

[

{

fieldLabel:'预购车日期',

xtype:'datefield',

allowBlank:false,

name:'reserveDate',

format:'Y-m-d',

value:'$dateTool.format("yyyy-MM-dd",$reserveDate)'

},

{

fieldLabel:'预算金额(RBM)',

xtype:'numberfield',

allowBlank:false,

name:'reqAmount',

value:'${reqAmount}'

},

{

fieldLabel:'购车描述',

xtype:'fckeditor',

height:300,

name:'desc',

value:'${desc}'

}

]

输入购车预算金额-fields.xml

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

<fields>

<field name=*"reserveDate"* label=*"预购车日期"* type=*"datetime"*/>

<field name=*"reqAmount"* label=*"预算金额(RBM)"* type=*"decimal"*/>

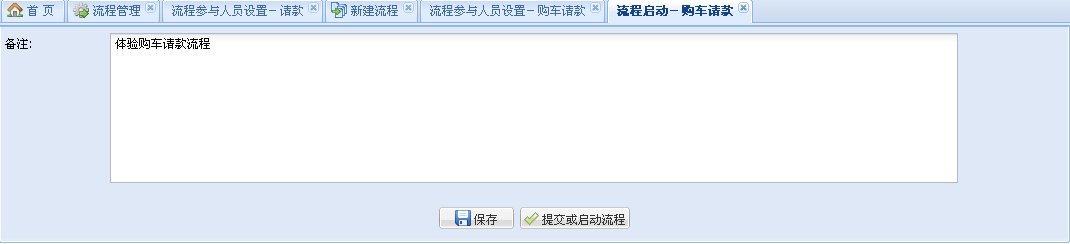
<field name=*"desc"* label=*"描述"* type=*"text"*/>

</fields>

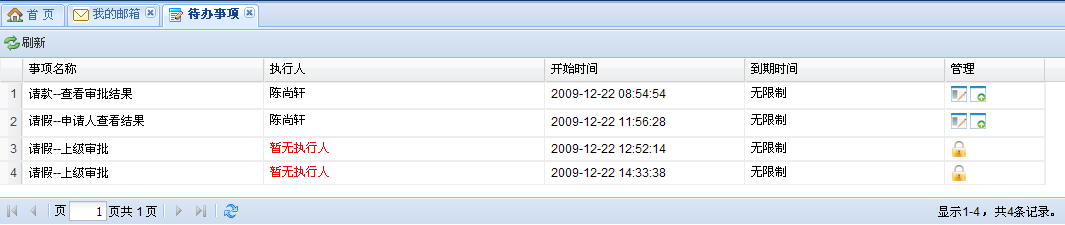
用同样的方法，为其他流程配置其他任务文件。

*说明：在后续的版本中，该流程任务中的表单设计将采用数据库进行管理，并且提供可视化的操作界面。*

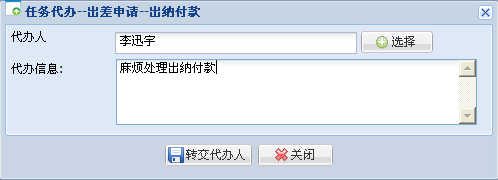
### 在JOffice中启动流程



在“流程管理”目录下“我的待办事项”，可以看到刚才提交的购车请款事项：



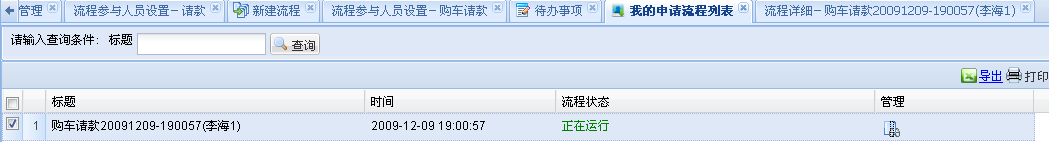
点击管理列中的“”按钮，同时，还可以在此把任务转为其他人员来代办，这时可以选择“”代办流程按钮。

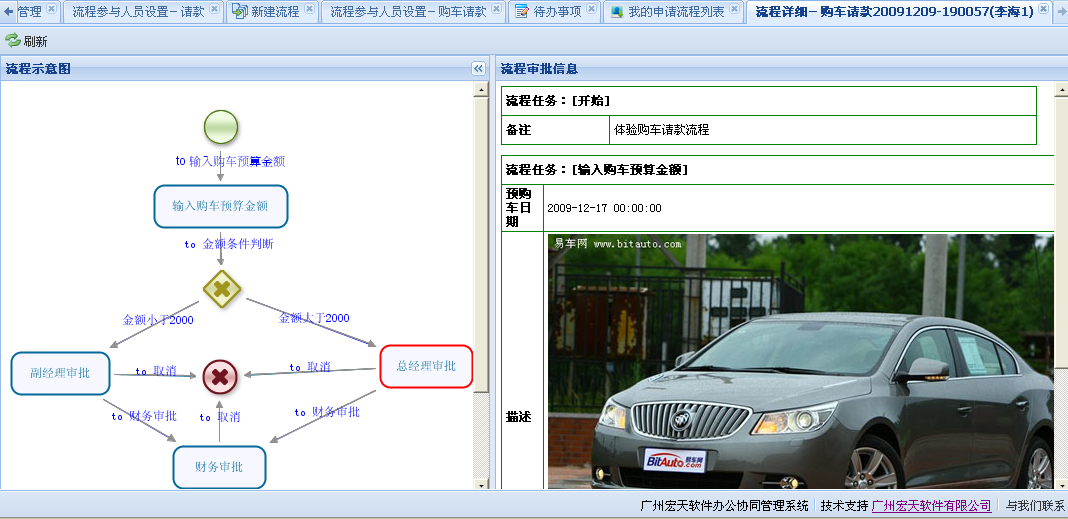




点击“转至：金额条件判断”，则流程进入下一个任务，如下所示：

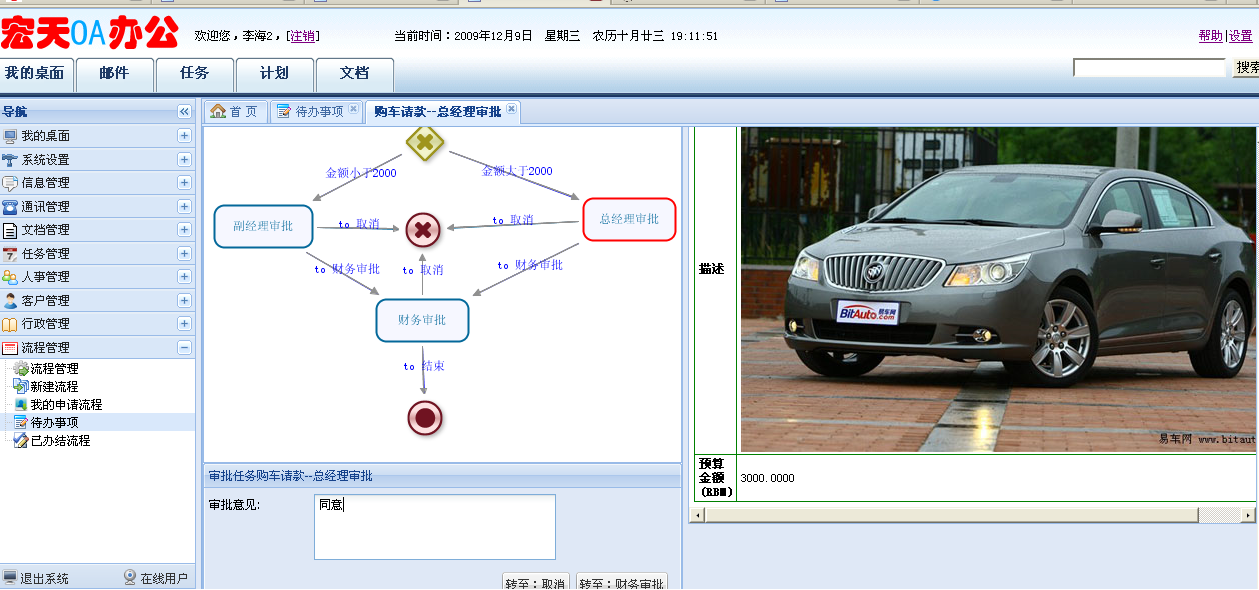
进入我的申请流程中，可以看到该任务已经提交并转至“总经理审批”环节，如下所示：





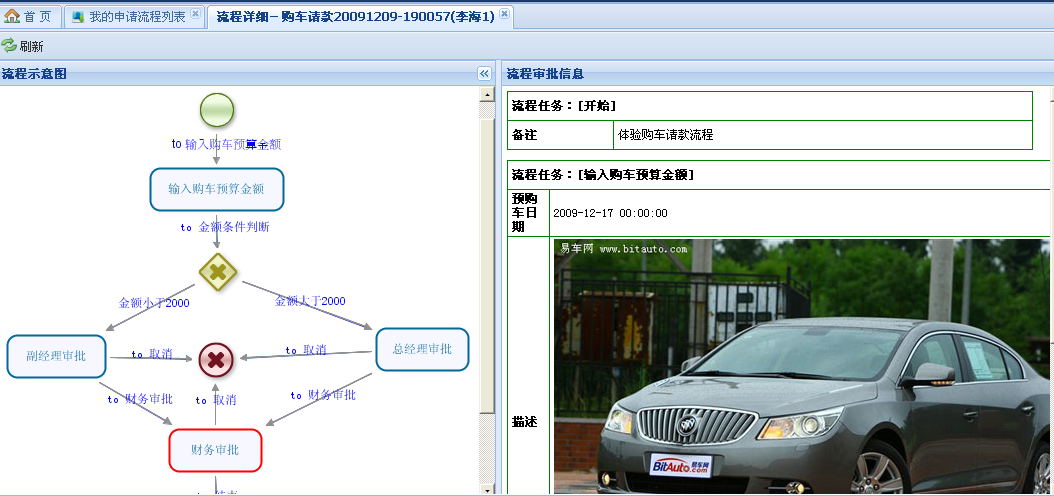
现在以李海2 用户登录，进入我的待办事项，如下所示：





提交至“账务审批”，即完成此次审批。

其他执行方式也相同，在此不介绍，往下执行的效果如下所示：



## 实现说明

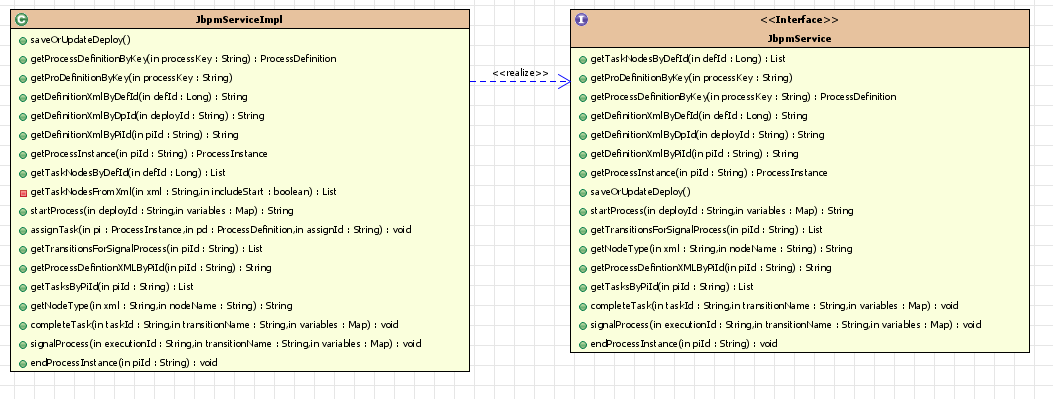
* 为了实现的方便，系统把工作流的相关操作放置在JbpmServiceImpl.java类中，该类提供了流程的相关处理方法，如发布流程：

**jbpmService.saveOrUpdateDeploy(proDefinition);**

删除流程：

**jbpmService.doUnDeployProDefinition(new Long(id));**

该类是整个工作流引擎的最核心的类，其下层是Jbpm的底层操作接口，对外还同时保存了流程的相关信息放置Jbpm流程外的表中进行存储，以使得查询更方便。



* 流程的定义发布由ProDefinitionAction中提供了实现，添加流程或删除流程定义等，其通过JbpmService接口来完成的。
* 流程的人员分配由ProUserAssignAction来完成，如保存流程任务的人员分配设置时，由该类的save方法完成
* 流程的启动及执行过程由ProcessActivityAction类提供实现，其实现细节请参与其源代码
* 流程的任务操作，如锁定任务，任务代办，释放任务由TaskAction提供实现
* 流程运行过程中的详细信息由ProcessRunDetailAction提供了实现