

各种工作流模式的实现

作者：非也 QQ: 20674450 Email: nychen2000@163.com

目 录

1. 概述.....	3
2. Fire Workflow 流程元素介绍.....	3
1) Activity 和 Task:	3
2) Synchronizer、StartNode、EndNode.....	4
3) Transition.....	4
3. 设计约束.....	4
1) 约束 1.....	4
2) 约束 2.....	4
3) 约束 3.....	5
4) 约束 4.....	5
5) 关于设计约束的说明.....	5
4. 顺序、分支、汇聚.....	6
1) 顺序分支汇聚其实是统一的.....	6
2) 顺序业务流程举例.....	8
3) 并行业务流程举例.....	8
4) 分支选择业务流程举例.....	9
5) 汇聚业务流程举例.....	10
5. 子流程.....	11
1) 流程设计.....	11
2) 流程模拟.....	12
3) 关于“Multi-Merge”的探讨.....	13
6. “自由流”(Jump).....	14
1) 流程设计.....	14
2) 流程模拟.....	14
3) 相关 API.....	17
7. 循环(Loop).....	18
1) 流程设计、模拟.....	18
2) 相关 API.....	18
8. 略过(Skip).....	18
1) 流程设计.....	18
2) 流程模拟.....	19
9. 会签.....	20
10. 委派.....	21
11. 任务完成期限.....	21
1) 流程设计、模拟.....	21
2) 相关 API.....	22
12. 监听工作流事件.....	22

1) TaskInstance 事件监听器.....	22
2) ProcessInstance 事件监听器.....	23
13. 表单绑定.....	24
14. 流程元素属性详细说明.....	25
1) 所有流程元素通用属性.....	25
2) WorkflowProcess 的属性.....	25
3) StartNode、Synchronizer、EndNode 属性.....	25
4) Activity 属性.....	25
5) Transition 的属性.....	26
6) Subflow Task 的属性.....	26
7) Tool Task 的属性.....	26
8) Form Task 的属性.....	26

1. 概述

本文最初始的标题是“21 种 workflow 模式的实现”，但是我觉得 21 种 workflow 模式总结的并不科学，“21 种 workflow 模式”只看到了业务现象，没有反应出业务的本质。

例如：“并行模式(Parallel Split)”、“单选 (Exclusive Choice)”、“多选(Multi Choice)”没有实质区别，都是从流程的分支中选择若干个执行。具体选择哪几个是由“转移条件”决定的，如果转移条件计算结果为 true 则执行，否则不执行。因此并行模式只是分支选择模式的一个特例，即所有的转移条件计算结果都是 true。

再例如：21 种 workflow 模式中的“同步 (Synchronization)”、“简单汇聚(Simple Merge)”也没有实质区别，在 Fire Workflow 中这都是“同步器”节点统一处理。

我甚至认为 21 种 workflow 模式存在一些谬误，我简单阐述我的看法，欢迎探讨。

例如：21 种 workflow 模式对“简单汇聚(Simple Merge)”的解释是“流程中的某个节点，使得两个或者多个流程分支汇聚在一起”，之所以将这种汇聚称为“简单汇聚”，是“假设多个流程分支不并行执行，因此汇聚点不需要进行复杂的同步操作”。

这种强行的“假设”看似简化了问题，实际上非常不合理。这种假设要求汇聚点“预知”多个分支中只有一个分支被执行，这种预知是很难实现的；或者限制多个分支中只允许一个被执行，这种后继节点限制前驱节点的行为是很荒谬的。

因此，本文将 workflow 模式按照我自己的理解重新组织。本文档的所有示例流程都在 example_workflow_process.rar 中。

2. Fire Workflow 流程元素介绍

在使用 Fire Workflow 设计业务流程之前，非常有必要介绍一下 Fire Workflow 的 workflow 元素。Fire workflow 的元素分成三类：1)Activity 和 Task， 2)Synchronizer、StartNode、EndNode， 3)Transition

1) Activity 和 Task:

Activity 在中文里一般称为“活动”，但是在我的文档里习惯称之为“环节”。Task 是任务，代表具体的业务逻辑，如录入一张表单、调用一段 java 代码或者调用另外一个流程（子流程 Task）。一个环节中可以有多多个任务。例如某个入职流程中有一个体检环节，体检环节包含了“检查视力”、“检查肝功能”、“检查心血管功能”等等多个任务。在 Fire workflow 中，体检环节建模如下：



2) Synchronizer、StartNode、EndNode

Synchronizer 是同步器。此处的“同步”是一个更加广义的概念，他代表工作流子系统的计算逻辑。开始节点（StartNode）和结束节点（EndNode）是同步器的特例，开始节点是没有输入“边”的同步器，结束节点是没有输出“边”的同步器。

3) Transition

转移（Transition）在 Fire workflow 并不仅仅是一条连接线，他代表控制权在工作流子系统和业务子系统之间交换。

3. 设计约束

在本人看来，可以将整个系统分成业务子系统和工作流子系统。Activity 和 Task 代表业务子系统的计算逻辑；Synchronizer、StartNode、EndNode 代表工作流子系统的计算逻辑；Transition 代表控制权在业务子系统和工作流子系统之间转移。有如下流程设计约束。

1) 约束 1

每个 Activity 的前驱节点、后继节点必须是同步器。即控制权从工作流子系统获得，业务执行完毕后控制权必须交还给工作流子系统。

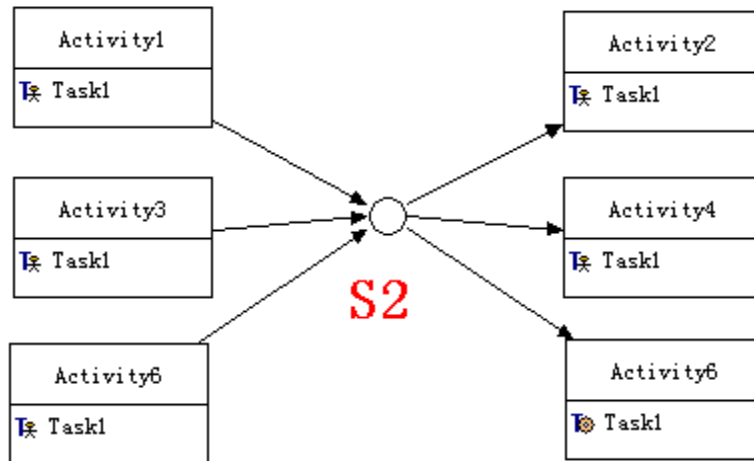
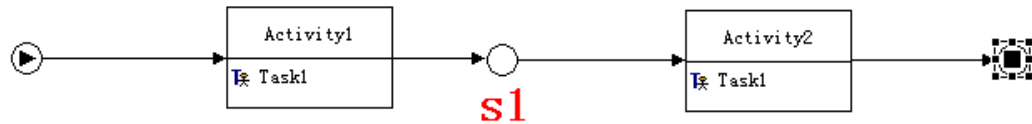
2) 约束 2

每个 Activity 只允许有一个输入 Transition，一个输出 Transition。

为什么呢？如果某个 Activity 有多个输入 Transition，则说明此 Activity 需要执行“汇聚”逻辑，然而“汇聚”并不是 Activity 的职责，而是同步器的职责；同理，如果某个 Activity 有多个输出 Transition，则说明该 Activity 需要执行“分支”逻辑，这也不是 Activity 的职责。

3) 约束 3

每个同步器的前驱节点和后继节点都必须是 **Activity**。即从业务子系统获得控制权后，进行汇聚和分支计算，然后把控制权再交给业务子系统。在 Fire Workflow 中，顺序流程中的同步器与复杂逻辑中的同步器的算法是一致的，如下图，S1 是 S2 的特列。



4) 约束 4

一个流程有且只有一个开始节点，至少有一个结束节点。

5) 关于设计约束的说明

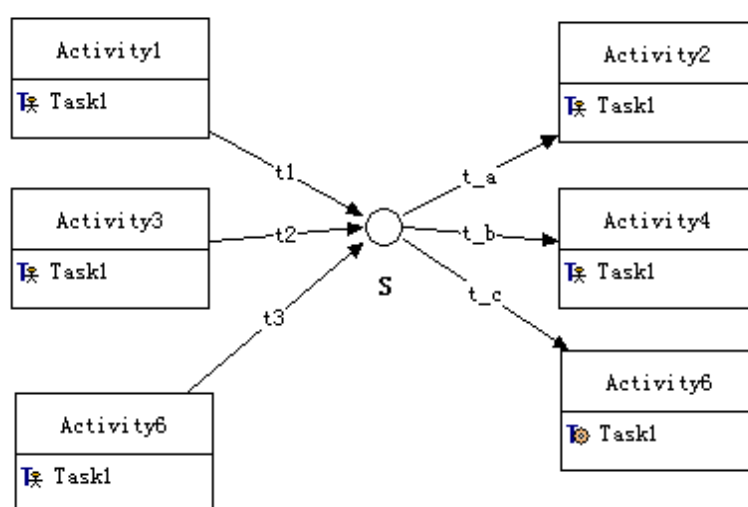
Fire Workflow 是以 Petri Net 作为理论基础的，因此上述设计约束 首先是 Petri Net 的要求。

但是 Fire Workflow 也力求不生搬硬套 Petri Net；我通过学习研究，发现用 Petri Net 可以非常恰当地描述“控制权”在“业务子系统”和“工作流子系统”之间相互转换这样一个业务模型。可以用 Petri Net 中的 token 来刻画“控制权”这个概念，用 T 来刻画“业务子系统”这个概念，用 P 来刻画“工作流子系统”这个概念。因此，上述设计约束可以精确、严密地描述业务流程。

4. 顺序、分支、汇聚

1) 顺序分支汇聚其实是统一的

首先，Fire Workflow 认为，“分支”和“汇聚”都是一种计算逻辑，这种计算逻辑应该由“工作流子系统”负责完成。Fire Workflow 用“同步器”节点在模型中表示这种计算逻辑。如下图，同步器“S”是一个一般意义上的同步器。

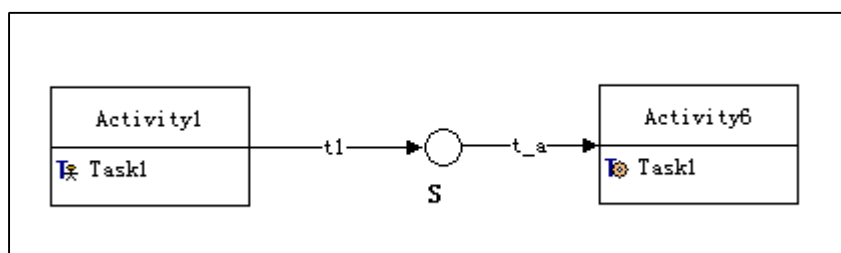


上图中“S”首先对 t1、t2、t3 执行汇聚操作。在 Fire Workflow 中，不管 t1、t2、t3 中哪个或者哪几个流程分支被执行，S 都能正确地进行汇聚。

S 汇聚完成后，根据转移条件的计算结果，启动 t_a, t_b, t_c 中的一个或者几个分支。

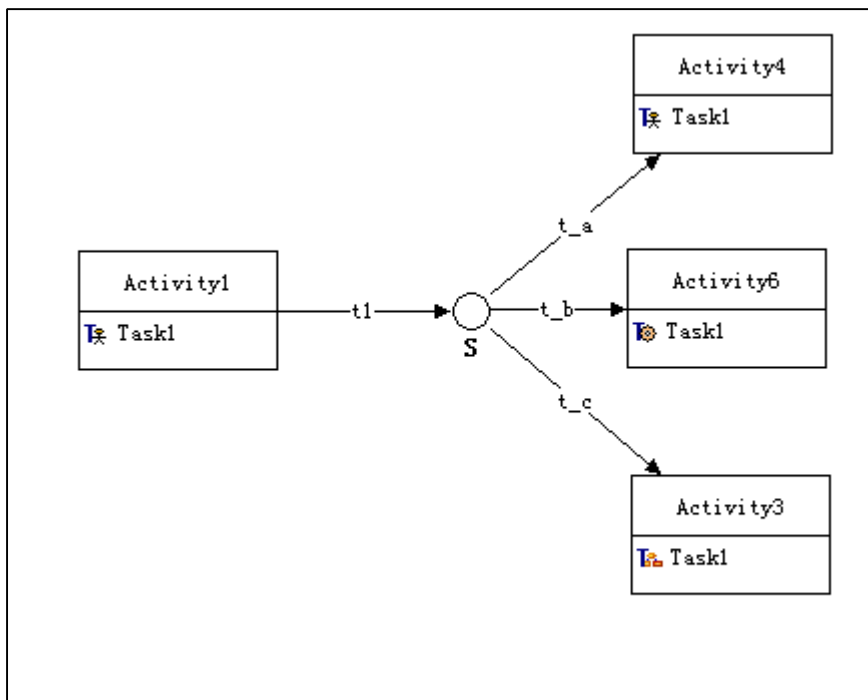
为什么说顺序、分支、汇聚是统一的呢？

A) 在上图中，如果“S”的前驱和后继都是唯一的，则形成一个特例，即顺序流程。如下图。

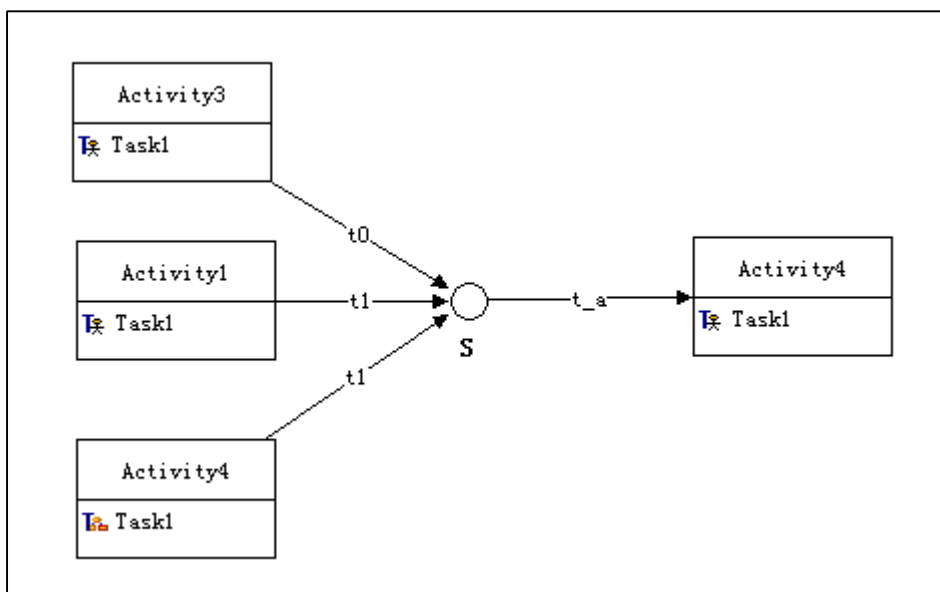




B) 如果上图中“S”的前驱是唯一的，后继有多个，则形成了“21 中工作流模式”中

的“并行模式(Parallel Split)”、“单选 (Exclusive Choice)”、“多选(Multi Choice)””。如下图



C) 如果上图中“S”的前驱有多个，后继仅有一个分支，则形成“汇聚”，如下图。



不仅顺序、分支、汇聚是统一的，同步器、开始节点、结束节点也是统一的，即开始节点和结束节点是同步器的特例。如果同步器的前驱为 0，则形成开始节点，如果同步器的后继节点为 0，则形成结束节点。在 Fire Workflow 中分别用  和  表示这两个节点。

2) 顺序业务流程举例

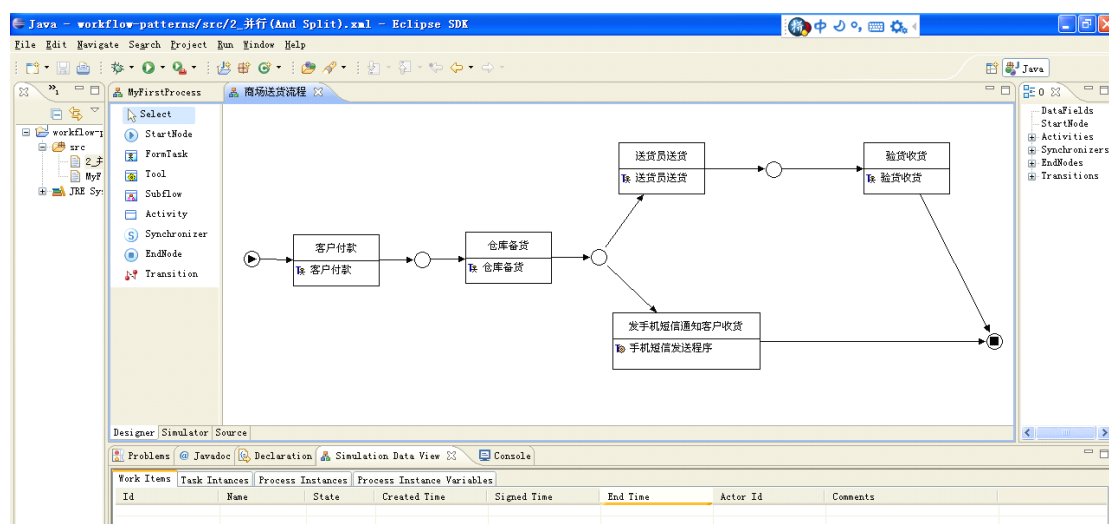
顺序模式的实现以及模拟请参阅《2_通过设计器和模拟器快速了解 Fire Workflow》

流程定义文件是 MyFirstProcess.xml

3) 并行业务流程举例

并行业务流程其实是分支选择的一个特例，即所有的转移条件计算结果为 true。在 Fire workflow 中如果 Transition 的转移条件 EL 表达是为空，则默认其结果为 true。

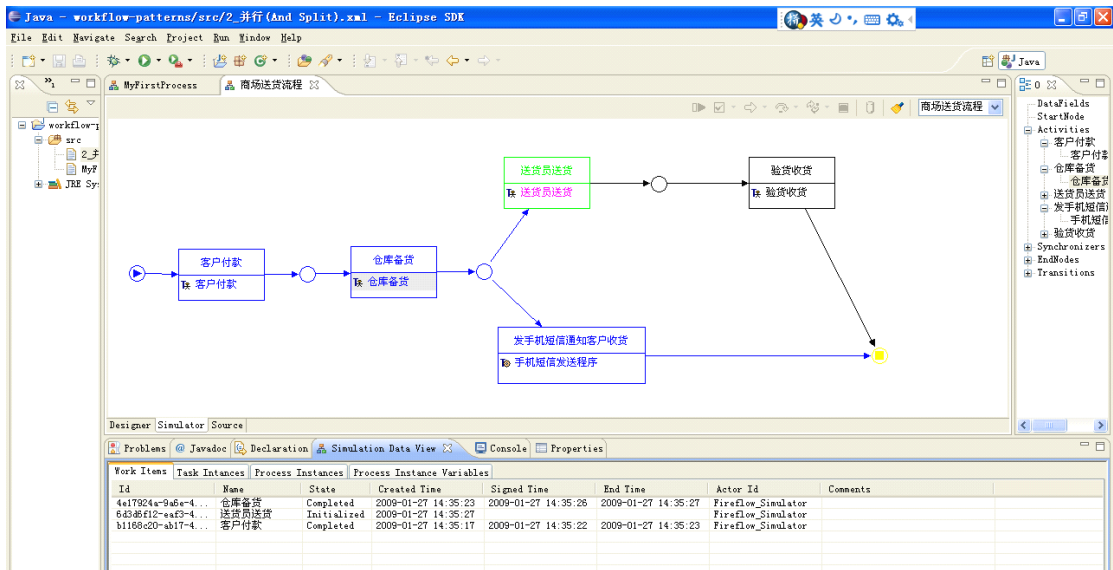
业务举例：某商场送货流程。仓库备货完成后流程分拆为两个并行的分支，一个分支通知送货员送货；另一个线程启动短信发送程序，预约客户在家等候收货。如下图：



相关的流程定义文件见：And Split.xml。

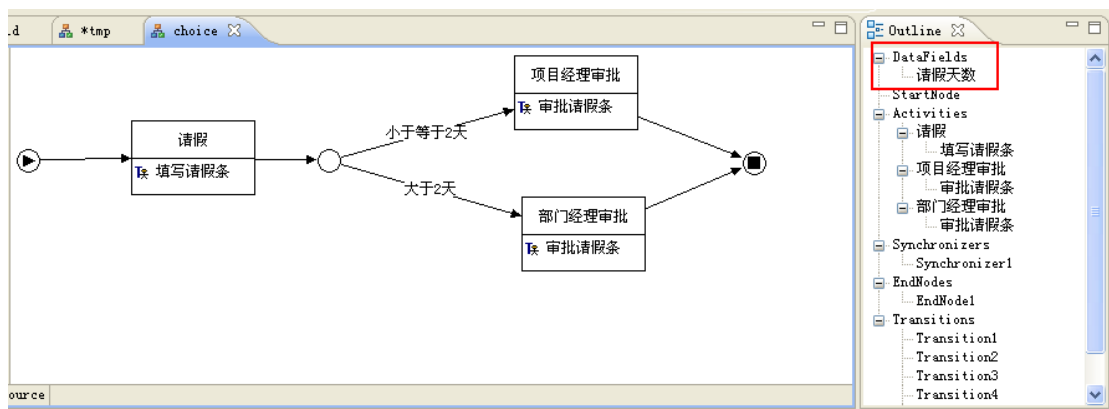
流程模拟

通过模拟器可以模拟该流程，如下图。可以看到，在“仓库备货”环节完成后，两个分支执行线程并行执行。



4) 分支选择业务流程举例

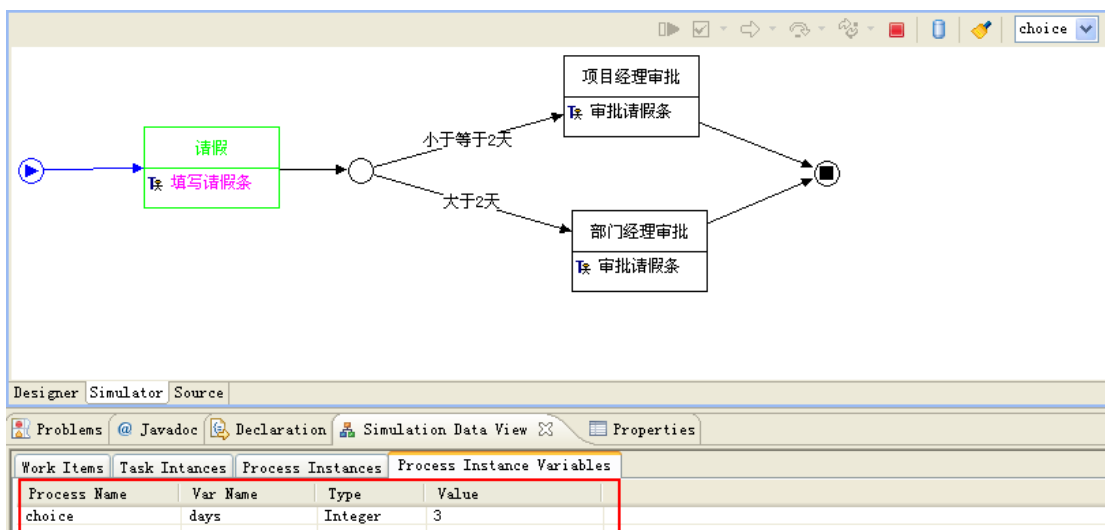
以典型的请假流程为例，如果请假天数小于等于2天，项目经理审批即可，否则需要部门经理审批。该流程图如下。在本流程中定义了一个名称为 `days` 的流程变量。



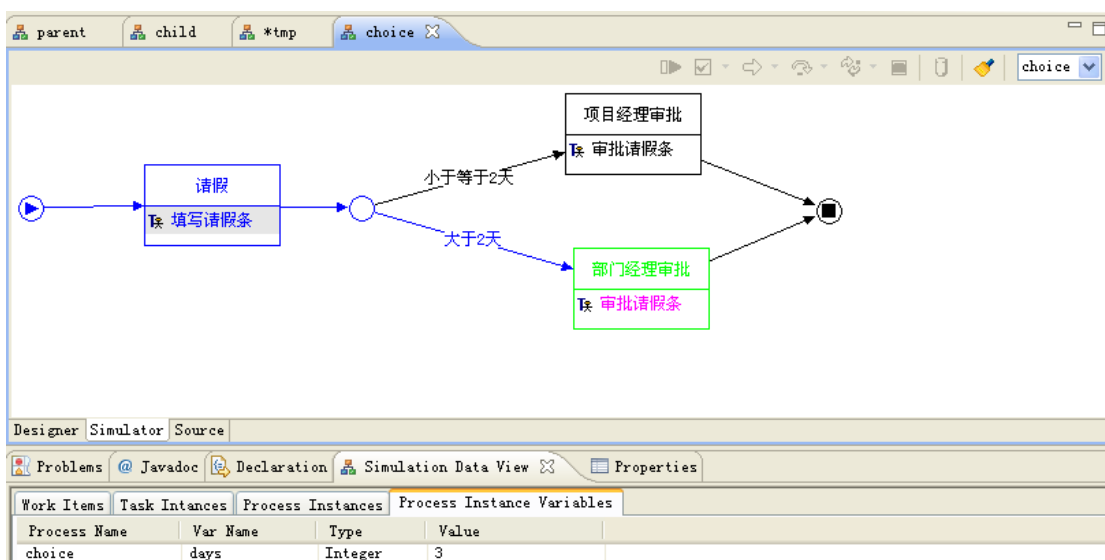
流程定义文件是：choice.xml

流程模拟：

启动流程，在工具栏上点击 ，设置流程变量 `days` 的值为3，如下图



签收并完成“填写请假条”任务后，系统启动了“部门经理审批”环节，而没有启动“项目经理审批”环节，如下图：




读者可以重新模拟该流程，将 days 的值设置为 1 看看结果如何。

5) 汇聚业务流程举例

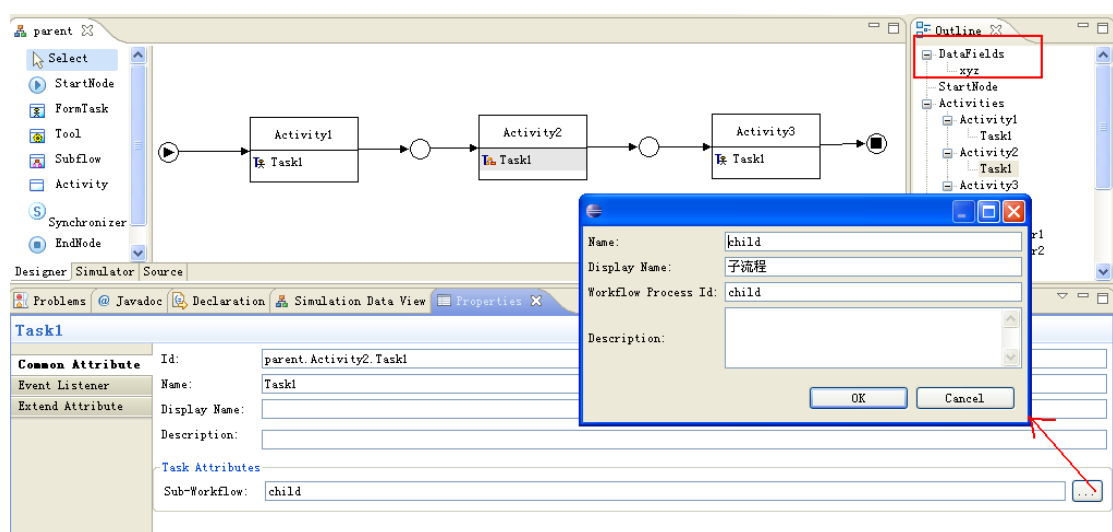
在上述“并行业务业务流程举例”中，结束节点实际上进行了汇聚。我们从模拟器可以看到，当“仓库备货”环节完成后，“手机短信发送程序”被自动执行，此时结束节点的颜色变成黄色，表示正在等待另一个分支完成。当送货分支完成后，结束节点变成蓝色，整个流程实例执行完毕。

5. 子流程

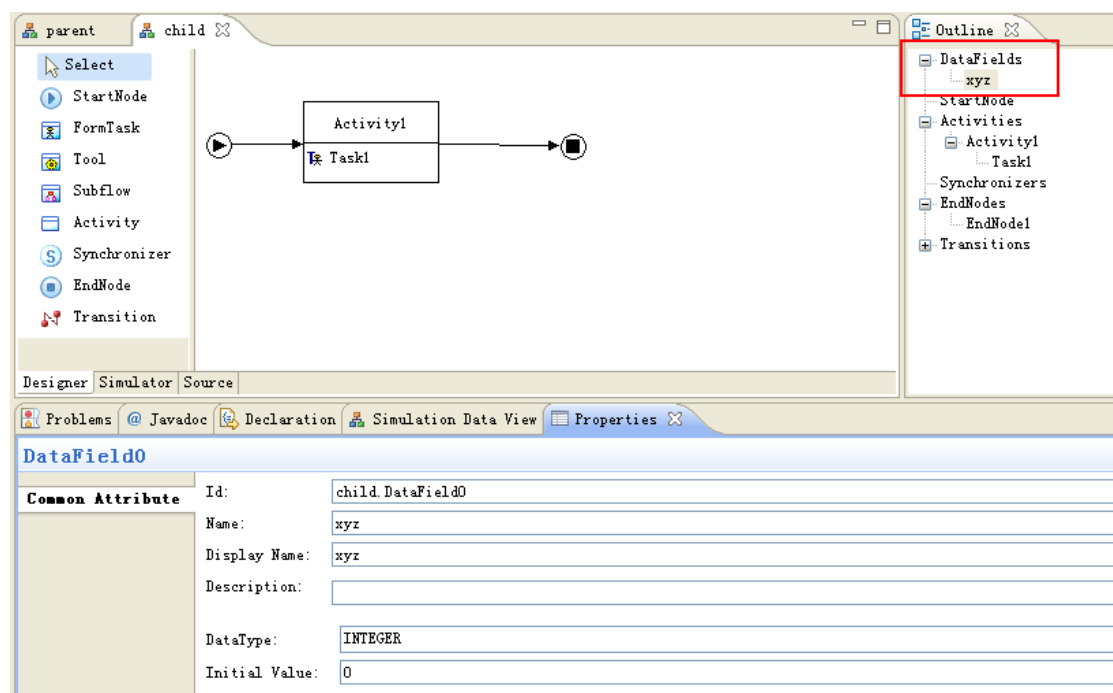
1) 流程设计

在 Fire Workflow 看来，子流程是父流程的一个任务。可以通过  Subflow 创建一个附带子流程任务的环节，也可以在已经存在的环节中增加子流程任务。

在本文提供的子流程实例如下图，在父流程和子流程中都定义了一个流程变量 xyz，在下一节流程模拟中可以看到，父流程的流程变量的值传递给了子流程。




父流程图

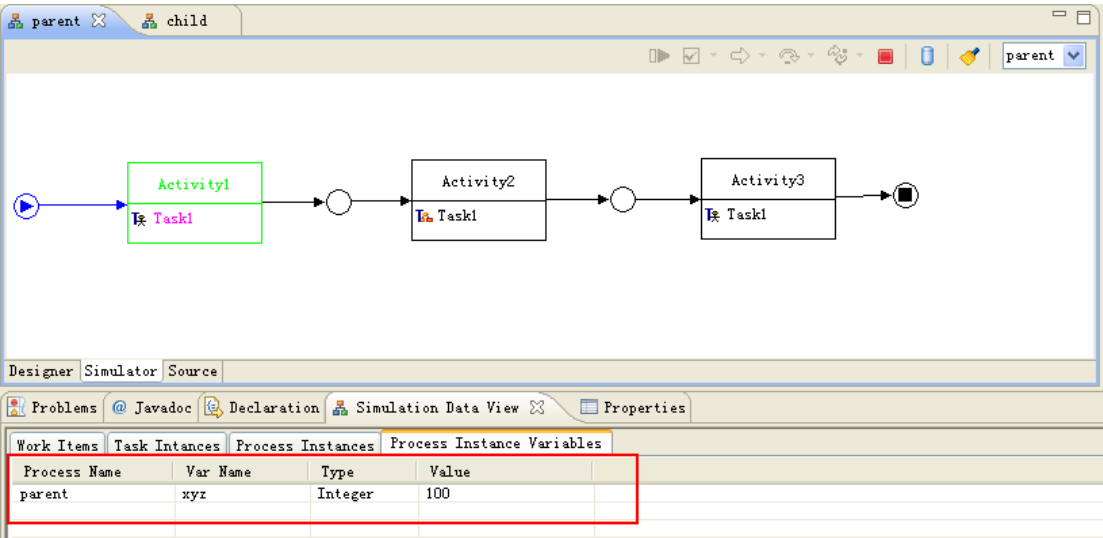


子流程图

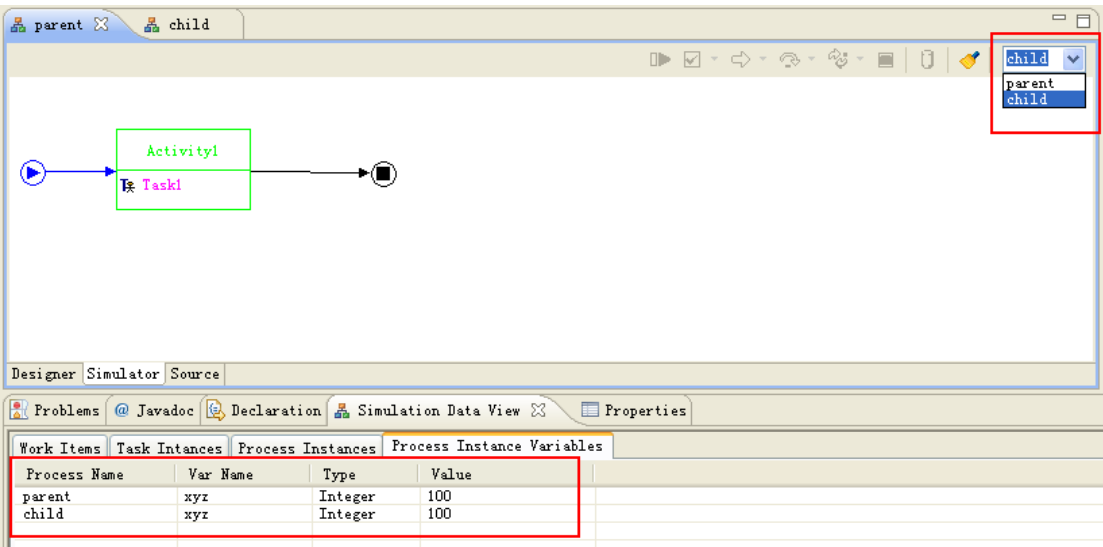
流程定义文件是：parent.xml，child.xml

2) 流程模拟

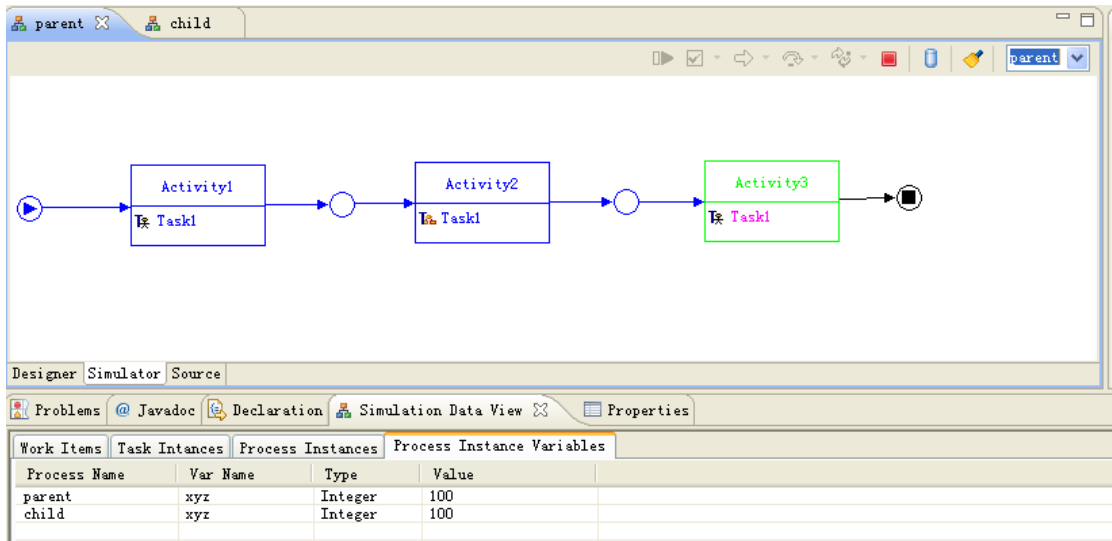
首先在父流程的模拟器界面启动父流程，并点击 设置流程变量 xyz 的值为 100，如下图所示



签收并通过 Activity1.Task1 后，系统弹出选择文件的对话框，定位到 child.xml 流程定义文件，然后确定。系统启动 Activity2.Task1，流程执行进程视图切换到 child 流程，同时父流程的流程变量的值赋给了子流程。如下图。你可以通过工具栏右边的下拉列表在执行进程视图中切换不同的流程。

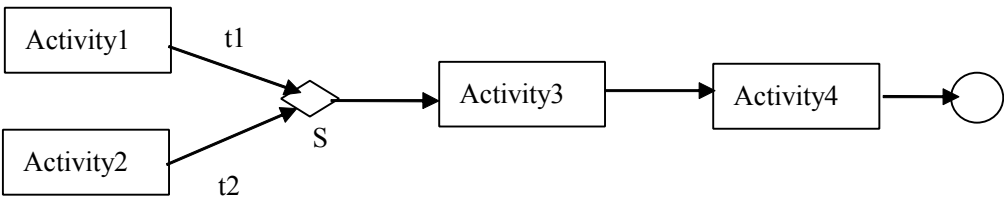


签收并结束子流程，将视图切换当 parent，可以看到父流程的 Activity2.Task1 也结束了，并且启动了父流程的 Activity3.Task1。如下图



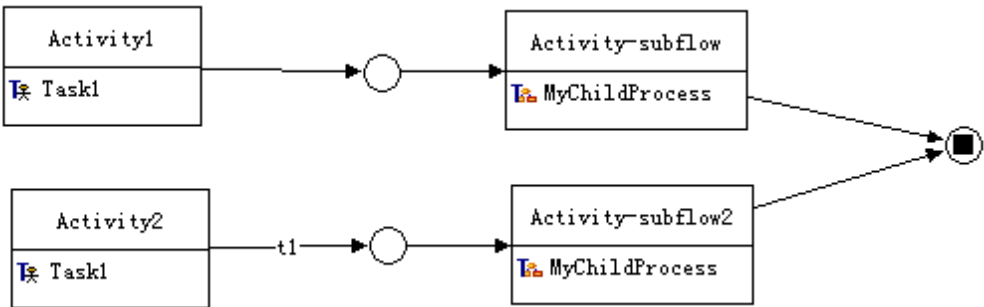
3) 关于“Multi-Merge”的探讨

在“21 种”工作流模式中有一个种模式叫做“Multi-Merge”，其含义是在“汇聚点”后的环节可以被执行多次，如下图：



在上图中如果流程分支 t1 执行到汇聚点 S 时，Activity3 和 Activity4 会被执行一遍；同理当 t2 执行到汇聚点 S 时，Activity3 和 Activity4 也会被执行一遍。显然 S 只是图形上的汇聚，在运行时并未起到同步的作用。

在这种情况下，用 Fire workflow 的子流程来建模可能更加合理。将 Activity3 和 Activity4 置于子流程中(假设子流程名称为 MyChildProcess)，父流程如下图。



6. “自由流” (Jump)

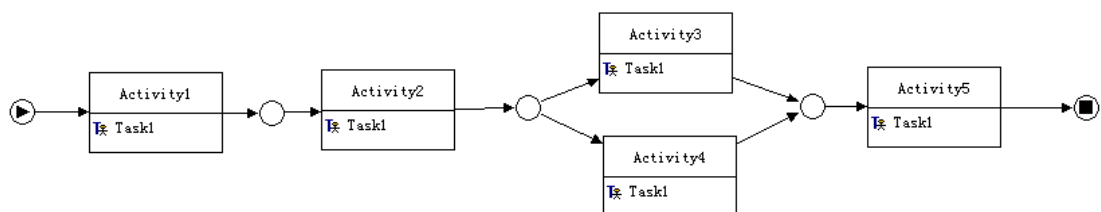
1) 流程设计

“自由流”是一种中国特色的业务流程模式，即在某些特殊情况下，流程不按照既定顺序执行，在环节间自由跳转。

在 Fire Workflow 中，跳转操作还不能在设计器中通过图形表达，只能调用接口 `IWorkItem.jumpTo(String nextActivityId, List<String> nextActorIds)` 完成。`jumpTo` 方法首先结束当前 `WorkItem`、`TaskInstance` 和 `ActivityInstance`，然后启动目标 `Activity`，创建相应的 `TaskInstance` 和 `WorkItem`。

自由跳转是有条件的，其首要条件是当前环节和目标环节在同一个“执行线”上，如果 Fire Workflow 检查到当前环节和目标环节不在同一个执行线上，则拒绝执行跳转操作。

“执行线”是我创造的一个工作流名词，在图论中的定义是 $Li(activity1)=Li(activity2)$ 。即：如果 `activity1` 的所有前驱节点和后继节点构成的集合等于 `activity2` 的所有前驱节点和后继节点构成的集合，则称 `activity1` 和 `activity2` 在同一个执行线上。如下图，`Activity1` 和 `Activity2` 在同一个执行线上，`Activity1` 和 `Activity5` 也在同一个执行线上；但是，`Activity1` 和 `Activity3` 不在同一个执行线上，`Activity3` 和 `Activity4` 也不在同一个执行线上。



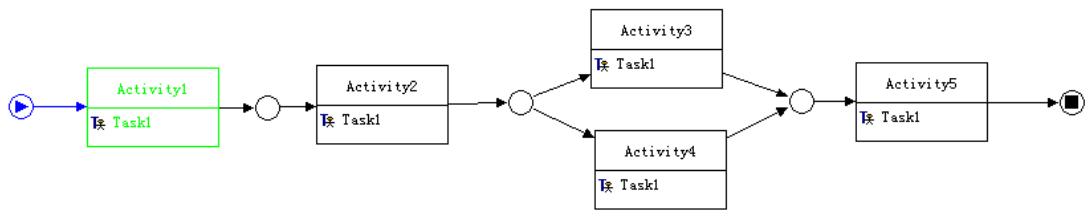
流程进行跳转的第二个条件是当前环 `WorkItem` 的结束操作可以触发对应的 `TaskInstance` 和 `ActivityInstance` 也正确结束，否则 Fire workflow 拒绝进行跳转操作。例如，在会签的情况下，如果还有操作员没有完成他的工单，则不能执行跳转操作，只有最后一个完成工单的操作员才具备执行跳转操作的条件。


自由流相关的流程定义文件见：`Jump.xml`

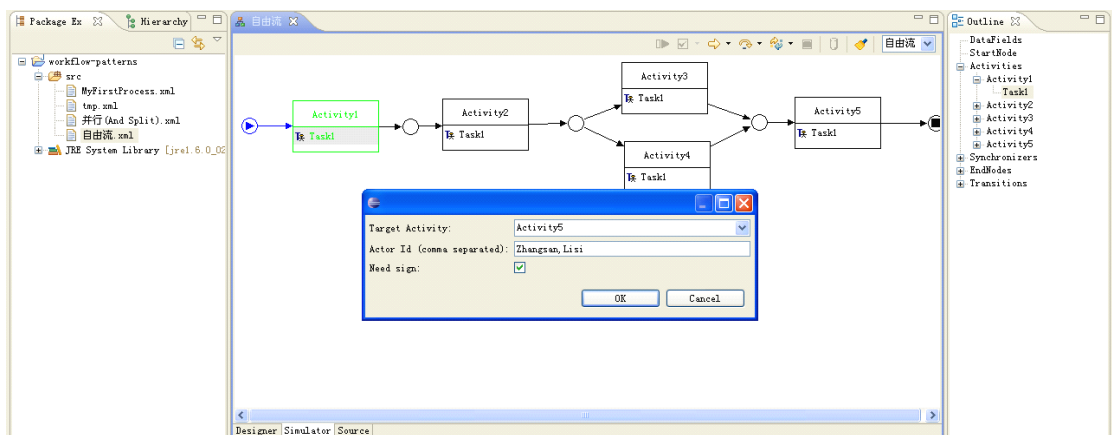
2) 流程模拟

下面模拟从环节 `Activity1` 直接跳转到 `Activity5` 的情形。

A) 首先创建流程实例，并签收 Activity1.Task1。得到的流程执行进程视图如下




B) 然后点击工具栏按钮 ，弹出对话框。在该对话框中输入目标环节的名称和操作员 ID，多个 ID 通过逗号分隔。在有多个 ID 的情况下，一般都需要签收，因此需要选中“Need Sign”复选框。如下图，有两个操作员，分别是“Zhangsan”和“Lisi”。



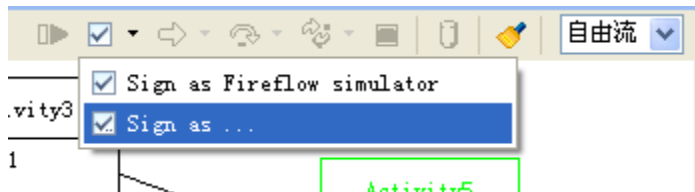
确定对话框后，流程跳转到 Activity5；同时从模拟数据视图中可以看到，系统生成了两个 WorkItem，分别赋值给 Zhangsan 和 Lisi 如下图。

Id	Name	State	Created Time	Signed Time	End Time	Actor Id	Co
94ce3572-1682-4...	Task1	Completed	2009-01-30 14:54:15	2009-01-30 14:54:18	2009-01-30 14:54:35	Fireflow Simulator	
4b19a8eb-c0c2-4...	Task1	Initialized	2009-01-30 14:54:35			Lisi	
3e11ef1c-8cfe-4...	Task1	Initialized	2009-01-30 14:54:35			Zhangsan	

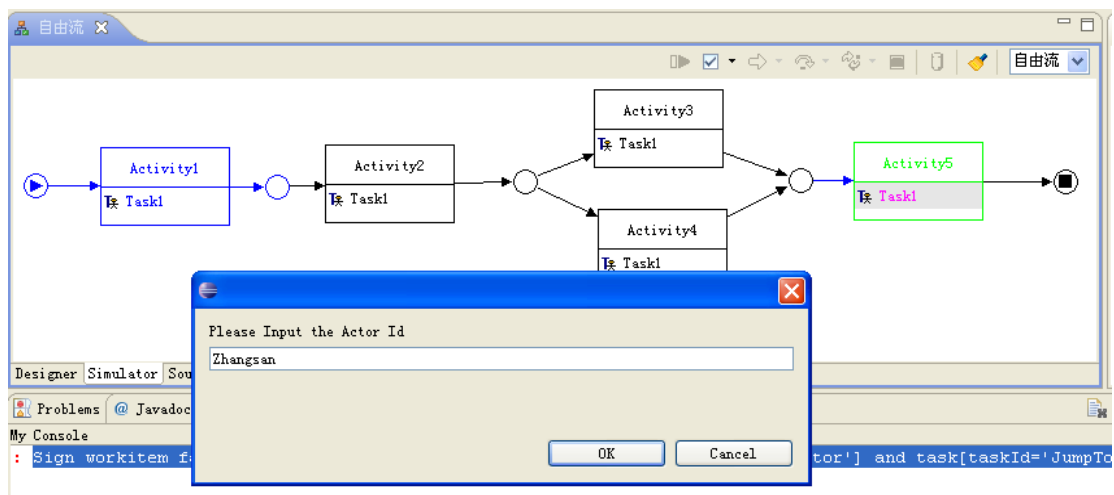
C) 签收 Activity5.Task1。

在本例中，如果点击  来签收 Activity5.Task1，系统会报告如下错误：Sign workitem failed, NO todo work items for actor[actorId='Fireflow_Simulator'] and task[taskId='JumpTo.Activity5.Task1']。即，操作员 Fireflow_Simulator 在 Activity5.Task1 上没有待办任务。

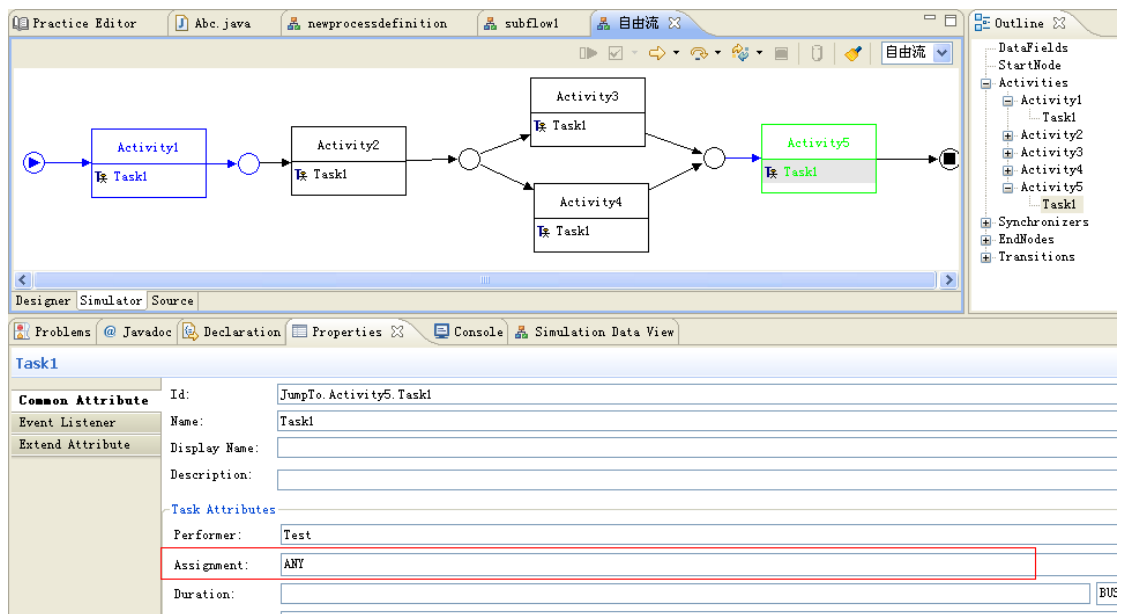
必须点击“Sign as ...”菜单来执行签收操作。如下图



点击“Sign as...”后，系统弹出对话框，要求输入操作员 ID，此时输入 Zhangsan（或者 Lisi）成功签收任务。如下图：

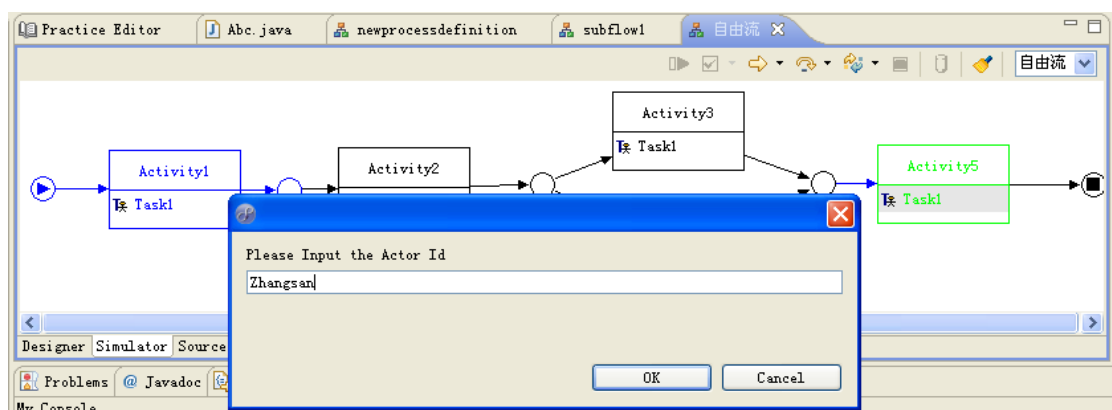
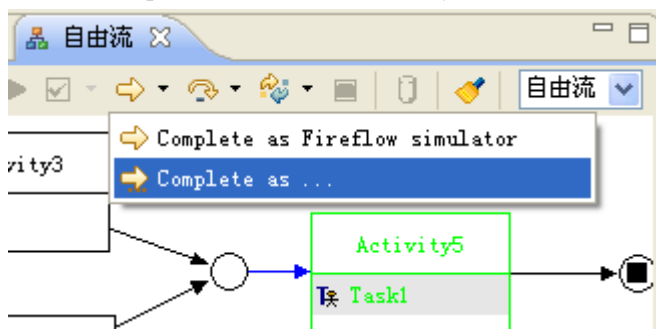


签收任务后，在模拟数据视图中可以看到“Zhangsan”的工单变成了 Started 状态，而 Lisi 的工单被删除了。这是因为 Activity5.Task1 不是的任务分配策略是“ANY”，如下图；即任何一个操作员完成该任务即可，这种情况下，只要一个操作员签收了他的工单，其他的工单将被删除。



D) 完成 Activity5.Task1

点击 “Complete As...”，完成 Activity4.Task1，如下图：



3) 相关 API


7. 循环(Loop)

1) 流程设计、模拟

循环是“自由流”的特例，即跳转到已经被执行过的环节上。Fire Workflow 自动把工单分派给上一次完成该环节任务的操作者。

在 Fire workflow 中，循环也不能在设计器中通过图形表达，只能是调用 `IWorkItem.loopTo(String nextActivityId)` 实现。

循环操作除了必须遵守“自由流”的条件外，还必须满足如下条件：目标环节已经被执行过。

读者可以通过模拟器工具栏上的  按钮对任意流程模拟循环操作。

2) 相关 API

8. 略过(Skip)

1) 流程设计


如下业务场景非常常见。

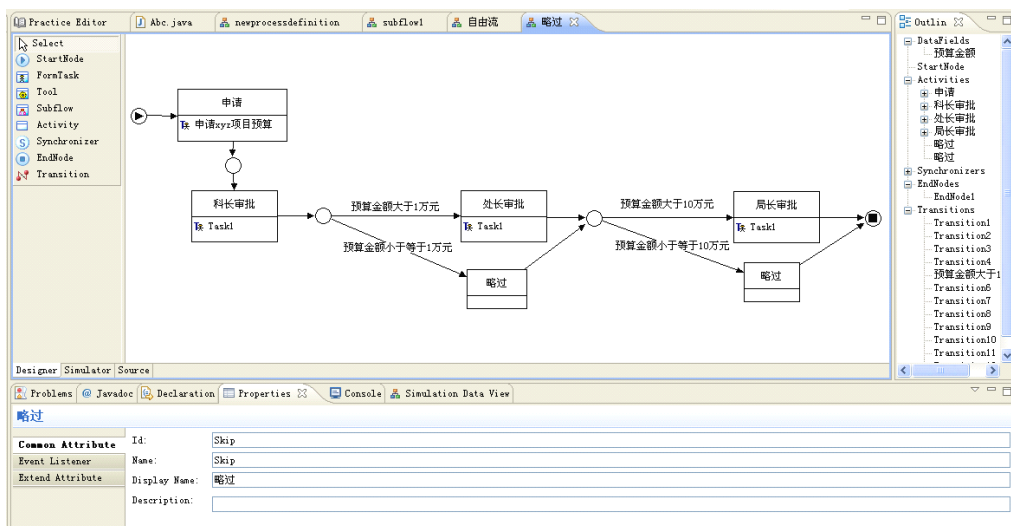
某审批业务规定：当金额小于等于 1 万元的情况下需要科领导审批；当大于 1 万元小于等于 10 万元时，除了科领导审批外还要送处领导审批；如果金额大于 10 万元，则需要再往上送到局领导审批。

对于这种业务模式有三种流程设计方法。

第一种方法是设计一个顺序流程“申请-->科领导批-->处领导批-->局领导批”，然后由业务代码根据申请的金额调用 `IWorkItem.jumpTo(String nextActivityId, List<String> nextActorIds)` 跳过不必要的审批级别。

第二种方法是使用“委派”机制，操作员根据业务管理规则人为地将任务呈送（委派）给上级领导。（该方案是否妥当还带斟酌...）


第三种方法是利用“空环节”（ Activity）实现略过，业务流程如下图

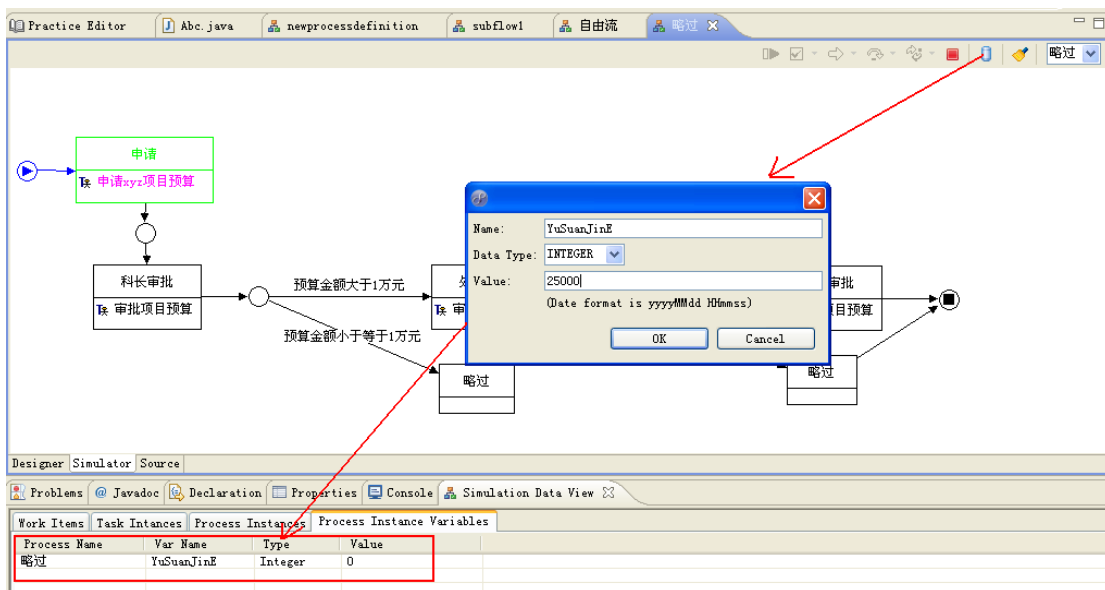


在该流程中预先定义了一个“预算金额”流程变量，在“科长审批”后的相关流程分支中根据该变量进行判断。这个变量也可以不预先定义，因为 Fire Workflow 允许在任何时候设置任何名称的流程变量，如果有同名的流程变量，则后值覆盖前值。

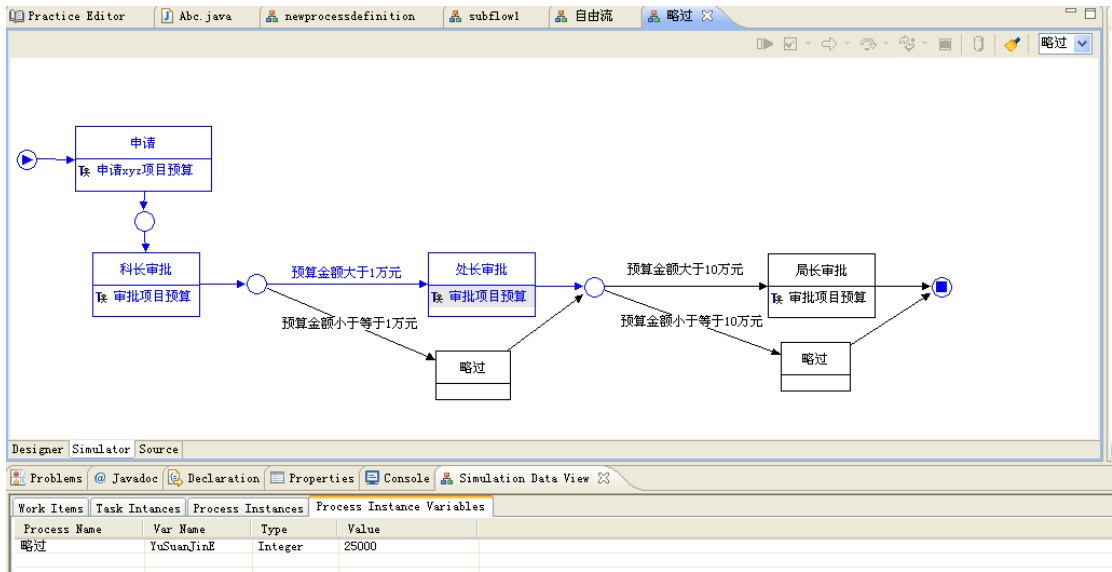
流程定义文件是：Skip.xml

2) 流程模拟

首先启动流程，然后点击，设置流程变量“预算金额”的值。假设我们设置的值为2500，如下图。

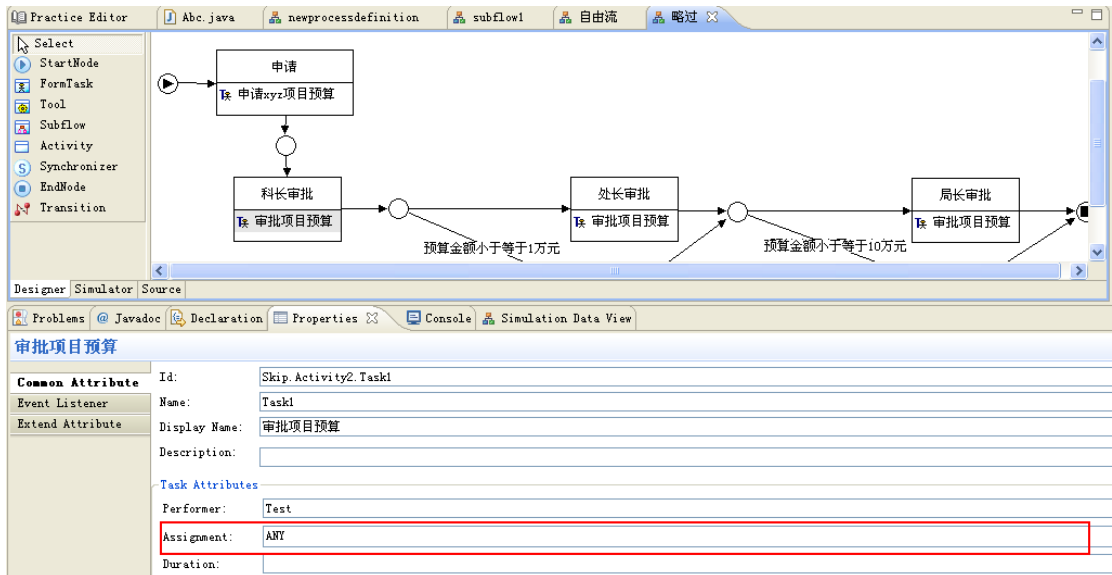


在预算金额为 25000 的情况下，流程执行完毕后的视图如下，可以看到“处领导审批”环节被执行了，而“局领导”审批环节被跳过了。



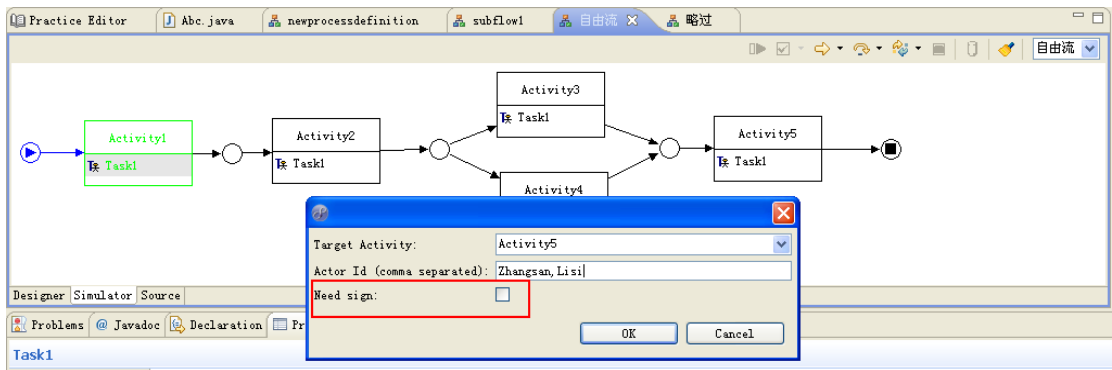
9. 会签

通常情况下，会签是通过设置 Task 的 Assignment 属性来决定的，如果 Assignment 属性为 ALL，则该任务实例必须等到所有的 WorkItem 结束后才会结束。Assignment 属性设置如下图：



另一种情况会导致会签。在跳转的时候，如果下一个环节的 Actor Id 有多个，且不需要签收，则工作流引擎会忽略 Task 的 Assignment 属性。读者可以在“自由流”章节按照如下

图所示的参数模拟跳转，即可看到会签的效果。



10. 委派

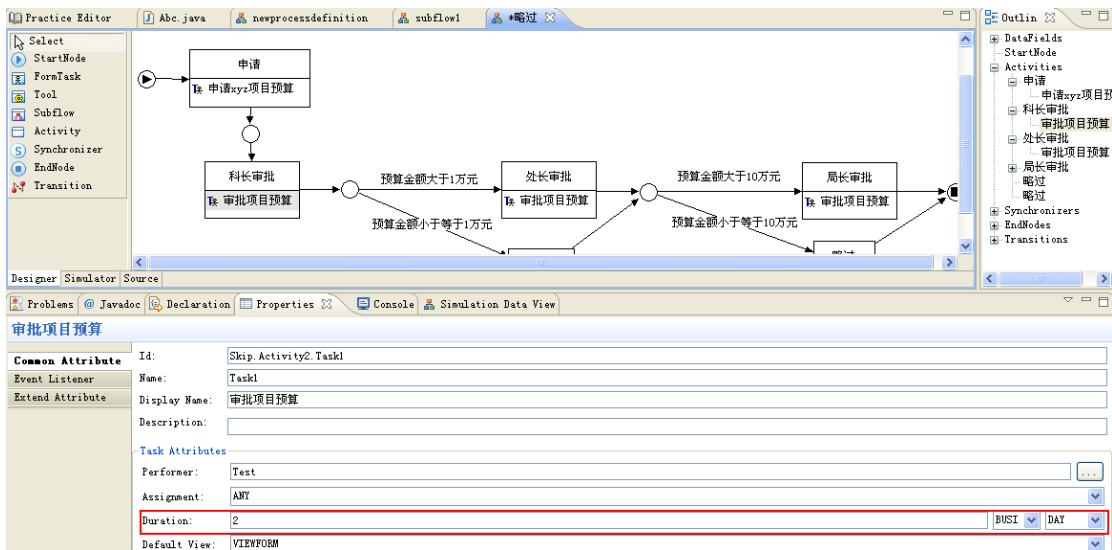
系统提供了如下接口完成任务委派 `IWorkItem.reassignTo(String actorId)` , `IWorkItem.reassignTo(String actorId,String comments)`。

Fire Workflow 仅提供完成委派的接口，并不会按照某种业务规则真正执行委派操作。业务系统应该按照特定业务规则调用该接口实现委派。

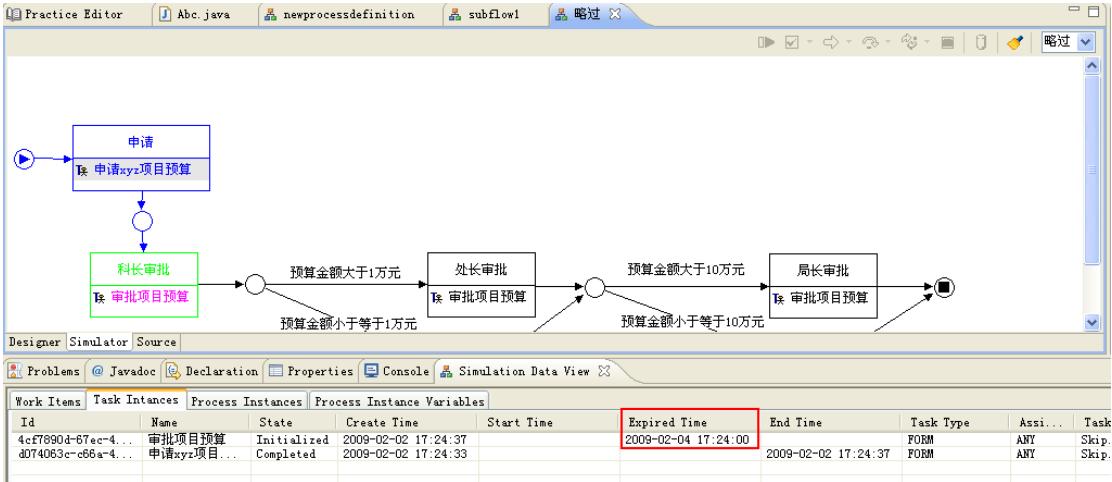
11. 任务完成期限

1) 流程设计、模拟

如果 Task 指定了 Duration 属性，则工作流引擎会自动计算任务完成的期限。以“Skip.xml”流程为例，如果“科长审批”的工作期限是 2 个工作日，则 Duration 属性设置如下图，



启动该流程后，从模拟数据视图可以看到，相关 Task Instance 的 Expired Time 被自动计算出来。如下图



2) 相关 API

工作流引擎把获得系统时间、计算任务完成期限等工作委派给日历服务 `org.fireflow.engine.calendar.ICalendarService`，Fire workflow 提供了该接口的一个缺省实现。业务系统可以实现该接口或者扩展 `org.fireflow.engine.calendar.DefaultCalendarService` 来解决用户自行调整上班时间、调整节假日的问题。更多信息请阅读：6_工作流引擎的结构及其扩展.pdf

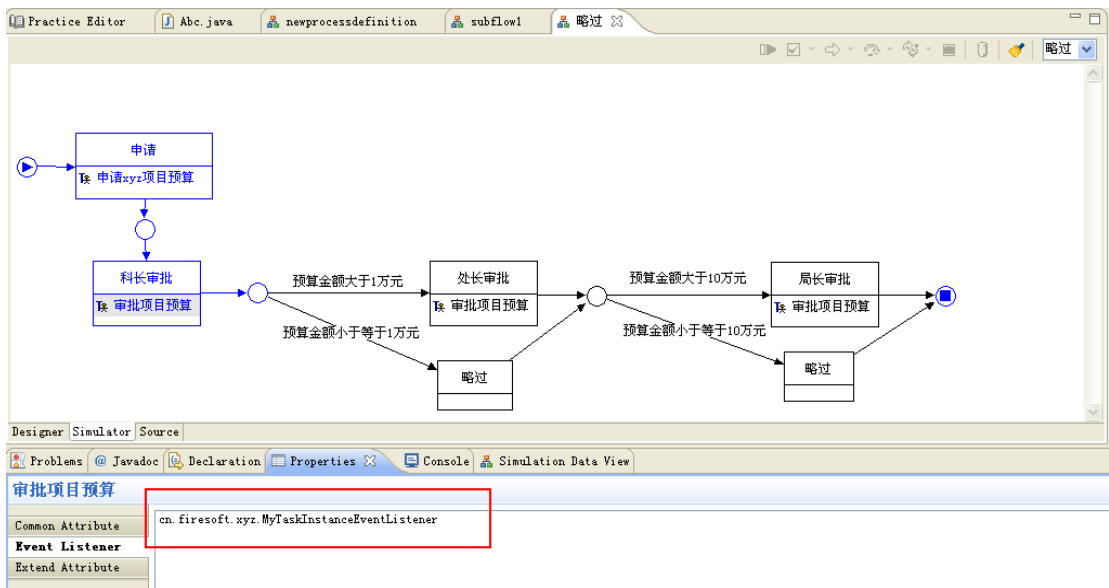
12. 监听 workflow 事件

Fire Workflow 的流程实例、任务实例在状态发生变化时都会产生相应的事件，业务系统可以在流程定义文件中注册相应的事件监听器来响应这些事件。

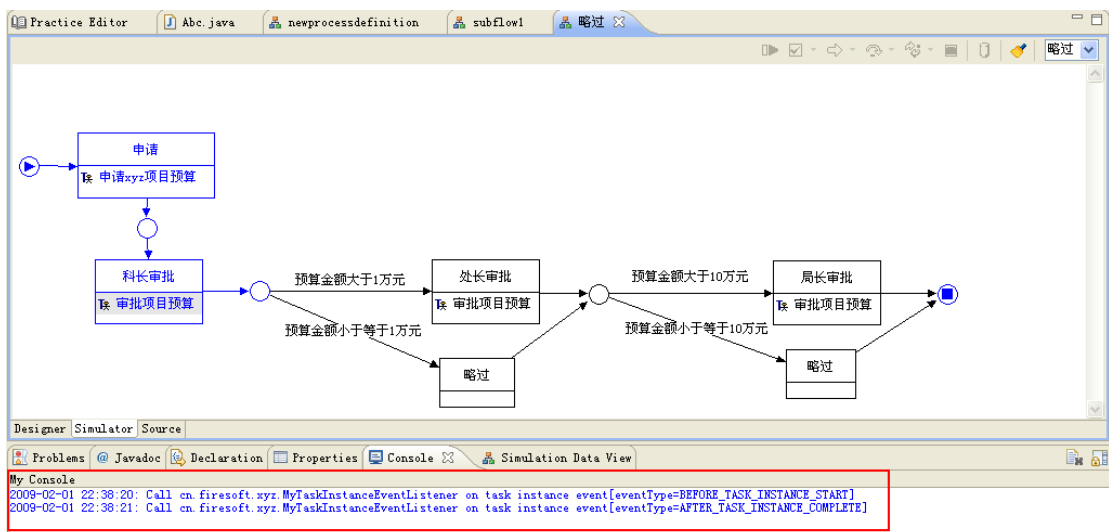
1) TaskInstance 事件监听器

Task Instance 的事件监听器必须实现 `org.fireflow.engine.event.ITaskInstanceEventListener`。该接口只有一个方法 `public void onTaskInstanceFired(TaskInstanceEvent e) throws EngineException`。可以通过 `e.getEventType()` 来确定时间的类型。

事件监听器必须在设计时注册到 Task 的 Event Listener 属性中，如下图



在模拟器中进行流程模拟时，如果 Task 定义了事件监听器，则会在控制台中打印出监听器的名称，如下图



2) ProcessInstance 事件监听器

Process Instance 的事件监听器必须实现接口 `org.fireflow.engine.event.IProcessInstanceEventListener`，该接口也只有一个方法：`public void onProcessInstanceFired(ProcessInstanceEvent e) throws EngineException`。可以通过 `e.getEventType()` 来确定事件的类型。

事件监听器必须注册在 Workflow Process 的 Event Listener 属性中，如下图



注：目前版本的模拟器并没有在模拟时将流程实例的事件监听器名称打印在控制台上。

13. 表单绑定

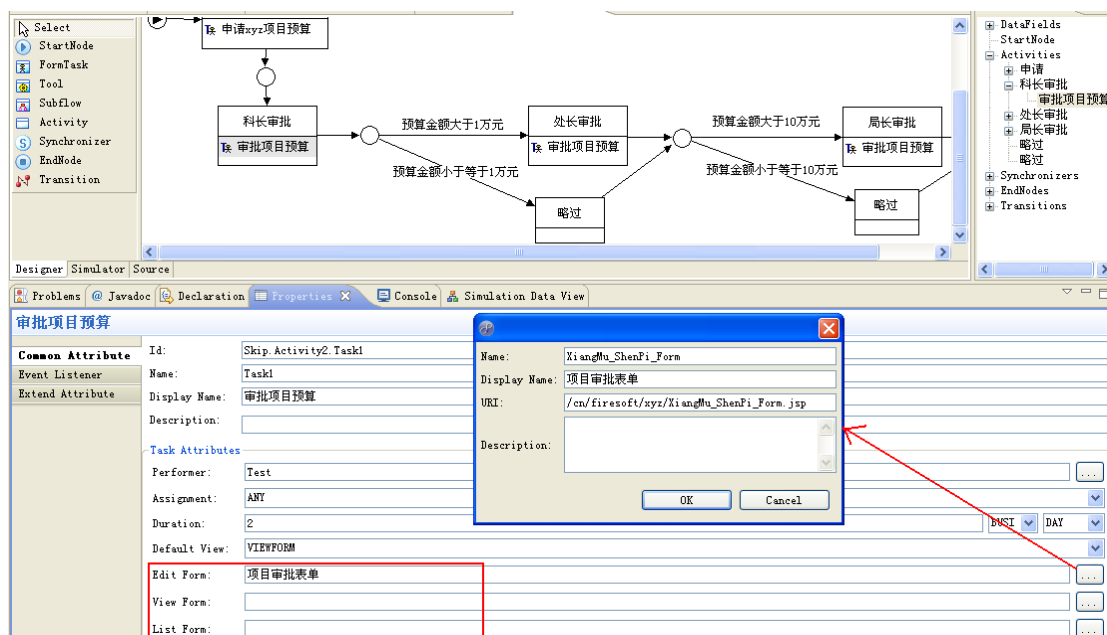
Fire Workflow 不提供所谓自定义表单功能，但是 Form 类型的 Task 提供了相关的属性来存储表单信息。他们分别是：

Edit Form: 该成员变量存储了任务的编辑表单的 url 等相关信息

View Form: 该成员变量存储了任务的只读表单的 url 等相关的信息。

List Form: 该成员变量存储了任务的列表表单的 url 等相关信息。

各表单的属性设置界面如下



14. 流程元素属性详细说明

1) 所有流程元素通用属性

属性名称	属性说明
ID	元素的 ID。不可编辑，由设计器自动生成
Name	元素的名称
Display Name	元素的显示名称，例如在中文应用系统中可以为元素命名一个通俗易懂的中文名称。
Description	元素描述
Extend Attribute	扩展属性，是一个 key-value 组成的 Map。您可以根据需要给元素增加任意多的扩展属性。

2) WorkflowProcess 的属性

Workflow Process 除了上述通用属性外，还有 Event Listener 属性

3) StartNode、Synchronizer、EndNode 属性

目前只有通用属性

4) Activity 属性

除了通用属性外，还有如下属性

属性名称	属性说明
Complete Strategy	<p>Activity Instance 的完成策略。可以取值 ANY 或 ALL，缺省值为 ALL</p> <p>当取值为 ANY 时，他的任何一个 TaskInstance 完成时都可以触发 Activity Instance 执行 Complete()操作。</p> <p>当取值为 ALL 时，只有当他所有的 Task Instance 完成时，才出发 Activity Instance 的 Complete()操作。</p>

5) Transition 的属性

Transition 除了通用属性外，还有如下属性。

属性名称	属性说明
Condition	<p>转移条件。是一个 EL 表达式，如果此 EL 表达式的计算结果为 true，则此转移分支将被执行，否则不执行。</p> <p>缺省值为空，为空时其值为 true。</p> <p>转移条件除了 EL 表达式外，还可以取值为一个字符串常量 “DEFAULT”，只有当其他所有条件的值都是 false 时，此转移才被执行，否则此转移不被执行。</p>

6) Subflow Task 的属性

子流程任务除了上述通用属性外，还有如下属性

属性名称	属性说明
Sub-Workflow	子流程的相关信息，如：子流程的 ID、Name、Display Name 等。

7) Tool Task 的属性

属性名称	属性说明
Execution	取值为 SYNCHR 或者 ASYNCHR，缺省为 SYNCHR SYNCHR 为同步执行对应 Application，ASYNCHR 为异步执行对应的 Application
Duration	(目前的 engine 对此属性没有处理。)
Application	Task 执行的 application，该 application handler 必须实现 org.fireflow.engine.taskinstance.IApplicationHandler

8) Form Task 的属性

Form Task 除了通用属性外，还有如下属性

属性名称	属性说明
Performer	操作者
Assignment	<p>任务分配策略，取值为 ANY 或 ALL，默认值是 ANY ANY 表示任何一个操作员完成该其工单则任务实例可以结束。</p> <p>ALL 表示收到工单的所有操作员都必须完成其工单，任</p>

	务实例才能结束，即会签。
Duration	任务完成期限
Default View	缺省视图，取值为 EditForm、ViewForm、ListForm，缺省值为 ViewForm
Edit Form	可编辑的表单信息，在此填入相关的 url 可以用于链接业务表单
View Form	只读表单信息
List Form	列表表单信息。