Fireflow 设计

内容目录

 QQ：20674450

1、Fireflow 工作流的组成

Fireflow 由模型(model)，引擎(Engine)，和设计器(Designer)三部分组成。如下图：

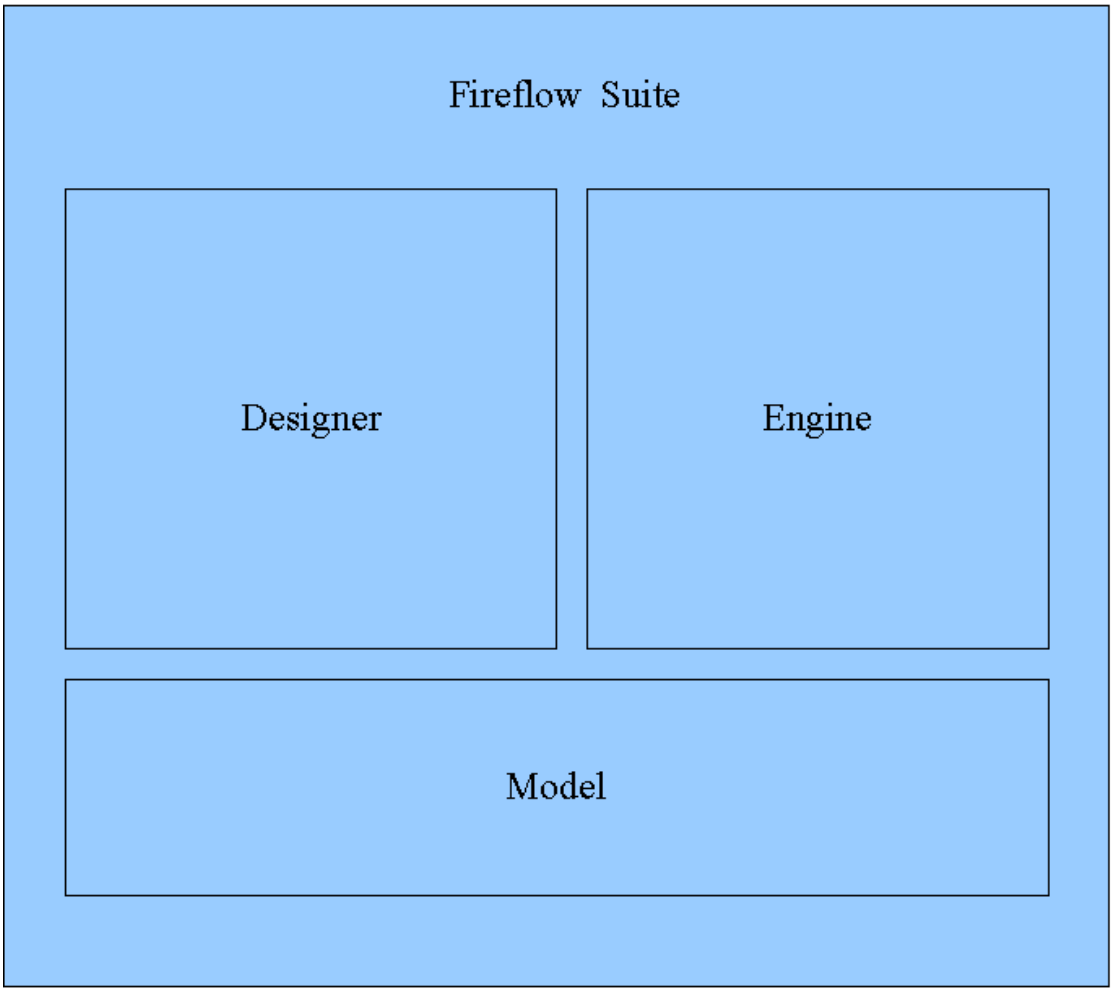

图 1-1

模型部分是整个工作流系统的基石，定义了描述流程的各种元素。Fireflow 工作流模型及其定义语言遵守《Fireflow 技术原理》中描述的一些约束。

设计器是创建 Fireflow 流程定义语言的工具；Fireflow 设计器提供一个模拟器，可以在设计时模拟流程的运行，观察流程的执行路径、工单（workitem)的创建情况，流程变量的变化情况等等。

引擎部分是 Fireflow 工作流的执行机，提供 API 供业务子系统调用，同时通过各种模式调用业务子系统。

 QQ：20674450

2、Model

Fireflow 重新定义了工作流模型，其基本原理是 Petri Net，具体论述见《Fireflow 技术原理》。

2.1、WorkflowProcess

Fireflow Model 的 xml 顶层元素是 WorkflowProcess，其结构如下图

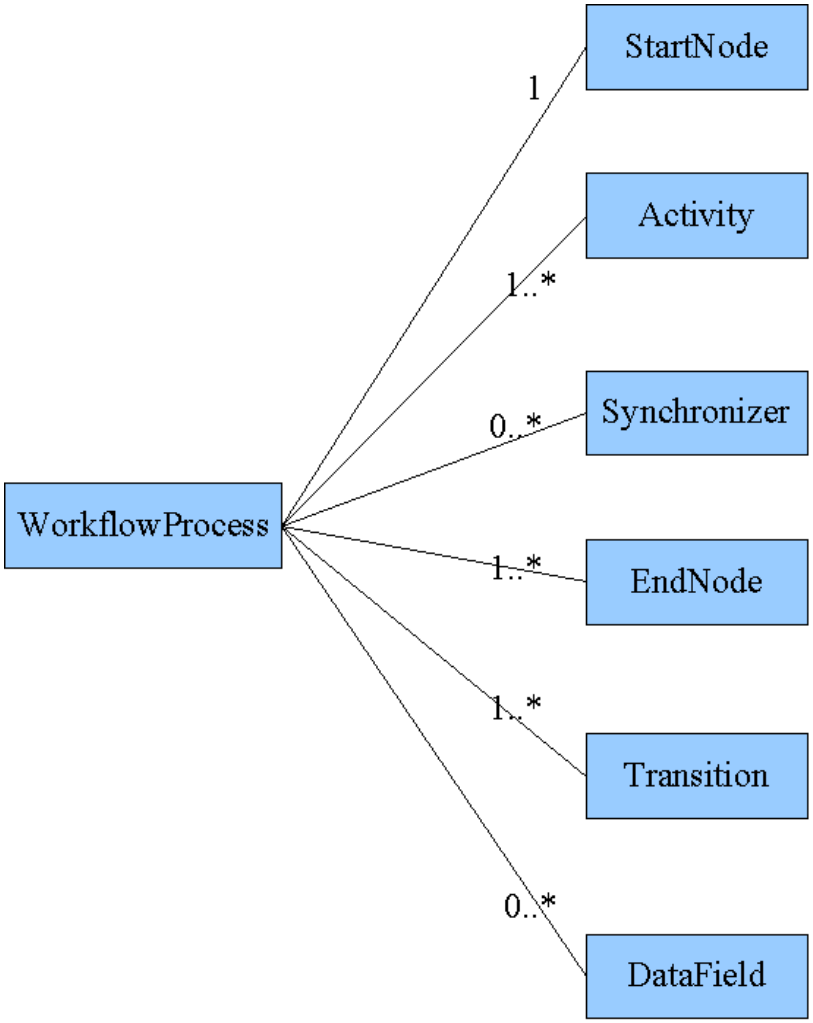

图 2.1-1

从图中可以看出，Firelfow 与 Wfmc 的工作流描述语言有很大的差别。Wfmc 规定的顶层元素是 Package，即流程包；每个流程包可以包含多个流程，还可以包含全局作用域（即对包中的每个流程都可见）的 Application, DataField 等等。

Wfmc 定义的 Package 看上去很合理，但是很不实用。因为，这种方法导致多个流程定义在同一个 xml 文件中，而我们实际的项目开发往往是多个人分别开发不同的流程，多个人共享一个 xml 文件非常不方便。Fireflow 将 WorkflowProcess 作为顶层元素，每个

WorkflowProcess 保存为一个单独的 xml 文件，方便系统开发。另外，在 Fireflow 看来，系统中的流程定义文件和 java 类文件是是等同的，都是系统的源代码。

Package 元素还有一个很完美但是不实用的地方是用来区分 Application、Participant、DataField 的作用域；在 Fireflow 看来，这种区分没有什么实际意义。

WorkflowProcess 可以包含 1 个开始节点（StartNode），1 个或者多个结束节点，0 个或者多个同步器。开始节点和结束节点是同步器的特例，他们的关系如下图：

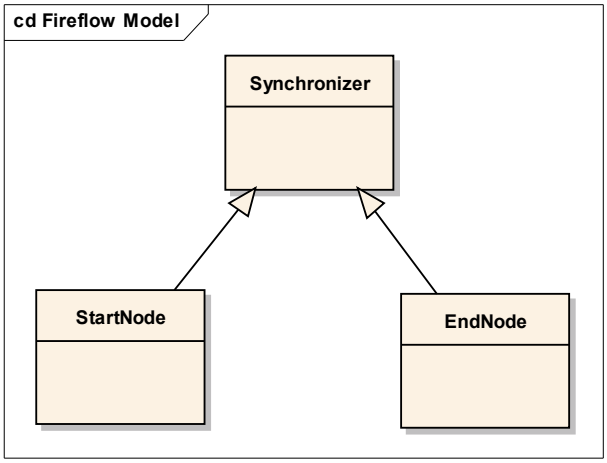

图 2.1-2

WorkflowProcess 可以包含 1 个或者多个 Acitivity，1 个或者多个 Transition，0 个或者多个 DataField(流程变量）。

与 wfmc 的 xpdl 不同，在 Fireflow 中，WorkflowProcess 不包含 Application 和 Participant 元素。因为在 Fireflow 看来，Application、Participant 都是独立于流程定义的全局性的外部资源，WorkflowProcess 只是对他们进行引用而已，具体的引用方法见后面的关于 Task 元素的描述。

WorkflowProcess 定义了如下属性：

属性名称	属性含义
ID	流程定义的 ID，这个 ID 在同一个 Engine 中必须唯一。通俗一点就是在同一个项目中各个流程的 ID 必须唯一
Name	流程的 Name，这个 Name 在同一个 Engine 中必须唯一。通俗一点就是在同一个项目中各个流程的 Name 必须唯一。 （注：目前 fireflow 中流程的 Id 和 name 取值相同)
DisplayName	显示名称。
Description	流程描述

2.2、Activity 和 Task

Fireflow 借鉴 Jbpm，设计了 Activity 和 Task 两个不同的对象来描述流程“环节”。Activity 实际上是工作流逻辑网中的 t，Task 描述的是工作流语义。

一个 Activity 可以包含 0 个或者多个 Task（通常情况下是 1 个 Task），同一个 Acitivity 中的 Task 的执行没有特定的顺序。如下图：

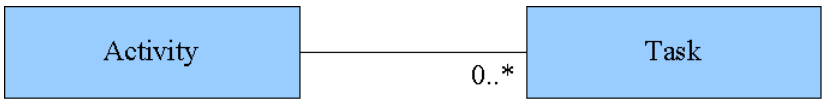

图 2.2-1

Task 分三种类型，Form Task，Application Task，Subflow Task。

Form Task：这是最常见的 Task，表示这个任务是处理一个业务表单。

Application Task：这种 task 在实例化后自动调用一个外部 Application

Subflow Task：task 启动另外一个流程，直到子流程实例完成后，task 才结束。

Activity 的属性如下：

Task 的属性如下：

2.3、Synchronizer、StartNode、EndNode

属性如下：

2.4、Transition

Transition 的属性如下：

2.5、DataField

DataField 的属性如下：

3、Engine

工作流引擎在整个系统中处于什么样的位置？不同的理解 Engine 的设计也大不相同。

 QQ：20674450

有人认为 Workflow Engine 应该像数据库一样，是一个独立的系统，他向业务系统提供流程服务。Fireflow 不赞同这种观点，至少现阶段这种方案不可行。

Fireflow 认为工作流引擎是是整个系统中的一个子系统，Engine 一定要 Embeded 到系统中去。不仅如此，业务子系统和工作流引擎之间有很多交互，这种交互是双向的；既有业务子系统调用 Engine 的 Api，也有 Engine 调用业务子系统的接口（如调用存储子系统保存流程实例等）。

鉴于 Fireflow 对 Engine 的定位，以及 Engine 与外部子系统的复杂交互。Fireflow Engine 的整体结构如下：

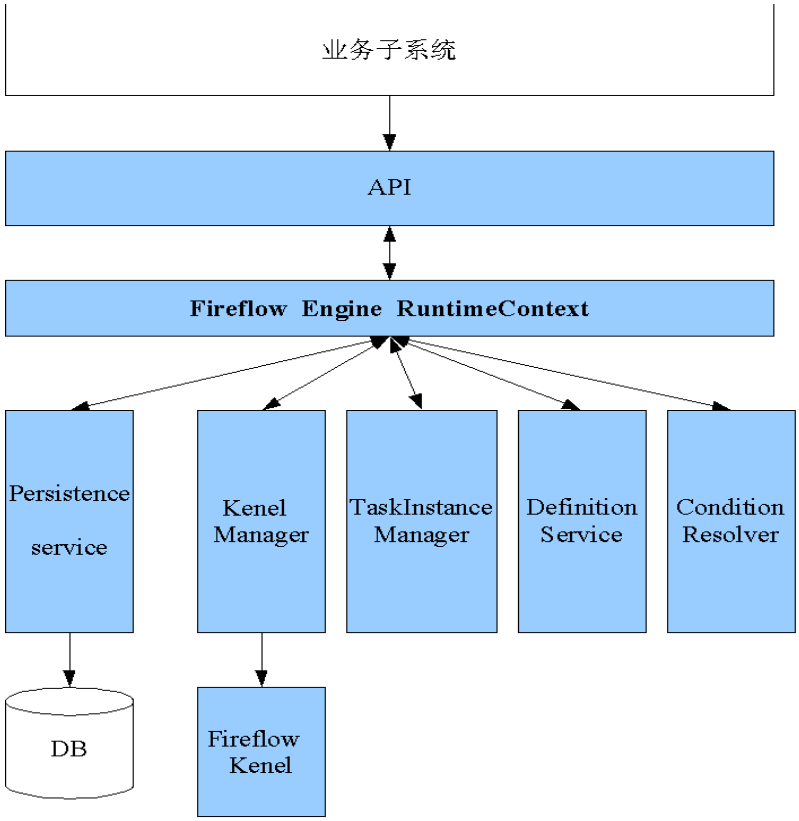

图 3-1

如图所示，Engine 由 API，Fireflow Engine Context 以及多个服务组成，各个部件通过 Spring 框架组装在一起。下面逐一说明各部分。

3.1、API

Fireflow Engine API 由如下对象组成，与 API 相关的描述请见我的 blog http://feiye.blog.51cto.com/126688/63875 ，为节省时间，不再 copy 了。客户端代码可以通过 Spring 的方式获得 FireflowSession，ex：

```
ClassPathResource resource = new ClassPathResource(
                "/test/engine/testContext.xml");
XmlBeanFactory beanFactory = new XmlBeanFactory(resource);
IfireflowSession fireflowSession = (IFireflowSession) beanFactory
                .getBean("fireflowSession");
```

IProcessInstance processInstance =
fireflowSession.createProcessInstance(workflowProcess.getName());

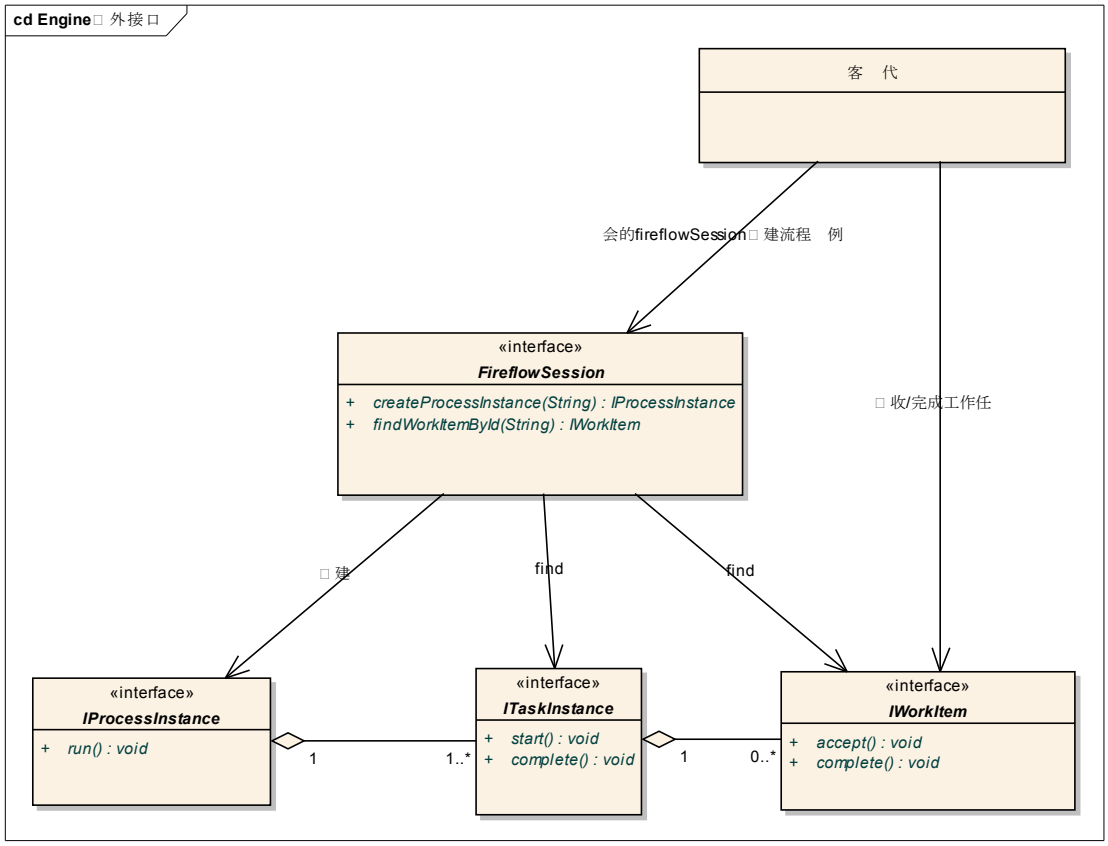

图 3.1-1

3.2、Fireflow RuntimeContext 及其作用

RuntimeContext 是 Engine 的“中枢”，Persistence Service、Definition Service 等服务都 Register 在该对象中。各个服务的注册是通过 Spring 框架实现的，代码如下。

```
……
<bean id="runtimeContext"
class="org.fireflow.engine.RuntimeContext"
init-method="initialize"
factory-method="getInstance">

    <property name="persistenceService">
        <ref local="persistenceService"/>
    </property>
    <property name="definitionService">
```

```
            <ref local="definitionService"/>
        </property>
        <property name="kenelManager">
            <ref local="kenelManager"/>
        </property>
        <property name="taskInstanceManager">
            <ref local="taskInstanceManager"/>
        </property>
        <property name="conditionResolver">
            <ref local="conditionResolver"/>
        </property>
    </bean>

    <bean id="conditionResolver"
class="org.fireflow.engine.condition.ConditionResolver">
    </bean>

    <bean id="taskInstanceManager"
    class="org.fireflow.designer.simulation.taskinstance.TaskInstanceManager4Simulation"
></bean>
    ……
```

3.3、Kenel

Kenel 是工作流逻辑网的执行机，Kenel 不关心任何业务逻辑，仅负责把 Petri Net 从初态 M_0 执行到终态 M_end。

Kenel 由如下对象构成……

Kenel 执行算法……

Kenel 通过 Observer 模式驱动业务逻辑……

3.4、Persisitence Service

Persistence Service 负责保存工作流实例中的各种对象：ProcessInstance ，TaskInstance， WorkItem 等等。

Persisitence Service 缺省实现是 Hibernate 方式。

3.5、Definition Service

Definition Service 负责管理流程定义。

 QQ：20674450