利用 SharePoint Designer 开发可循环工作流

柴晓伟

2007-3-19

介绍

大家都知道 SharePoint 工作流有两种开发(设计)方式:通过 SharePoint Designer 和 Visual Studio.

- 使用 Visual Studio,开发人员可以通过编码的方式灵活的开发 SharePoint 工作流.
- 使用 SharePoint Designer,网站管理人员可以基于定义和规则设计 SharePoint 工作流,无需编码.

使用 SharePoint Designer 设计工作流是件简单的事情,SharePoint Designer 工作流设计器可以帮助我们设计出顺序执行的工作流,然而许多真实场景中的工作流可能并非如此,那么 SharePoint Designer 是否可以设计出非顺序执行的工作流呢?

本文就将"利用"SharePoint Designer 来设计一个可以循环的工作流.

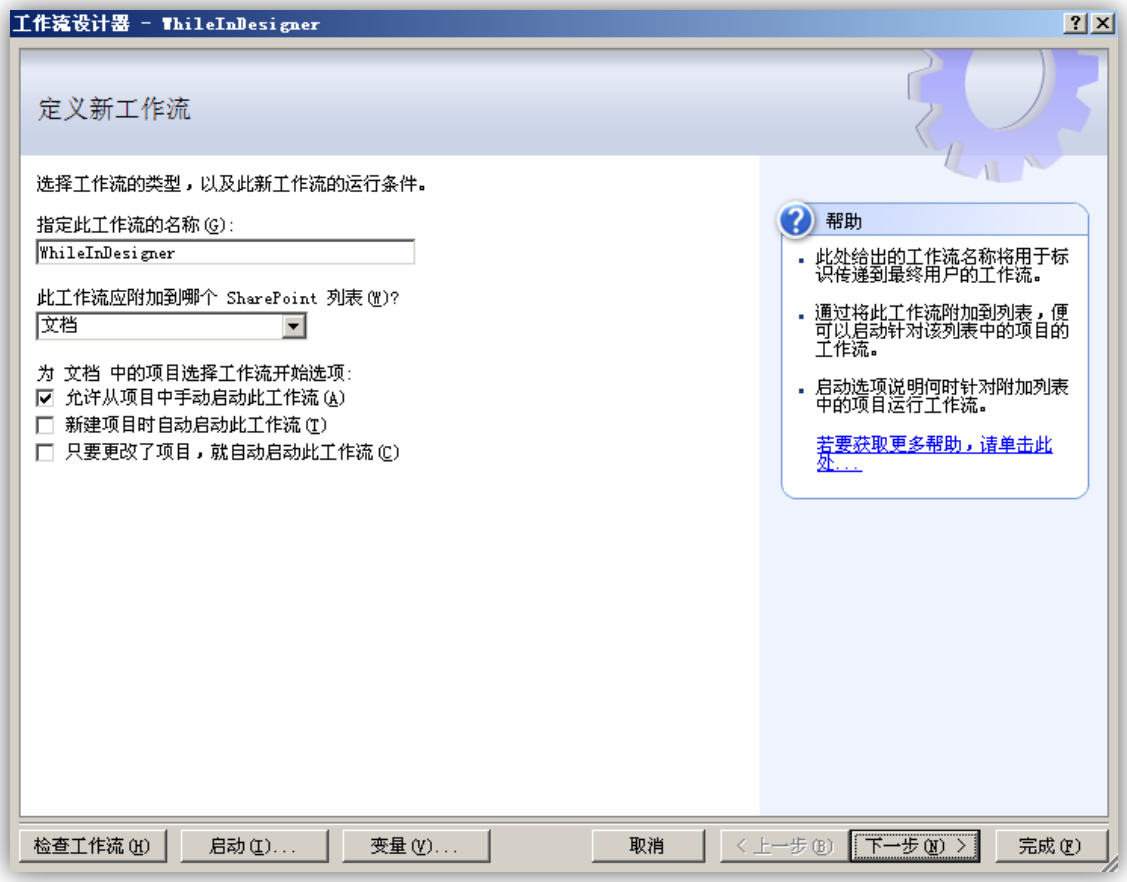
使用 SharePoint Designer 设计一个简单的工作流

我们先利用 SharePoint Designer 来设计一个简单的工作流,这个工作流先判断一个 Bool 型变量是否为 False,如果为 False 则向用户收集一个 Bool 型的数据,并把它赋给之前的变量.

1.用 SharePoint Designer 打开文档中心网站.

2.新建 > 工作流:

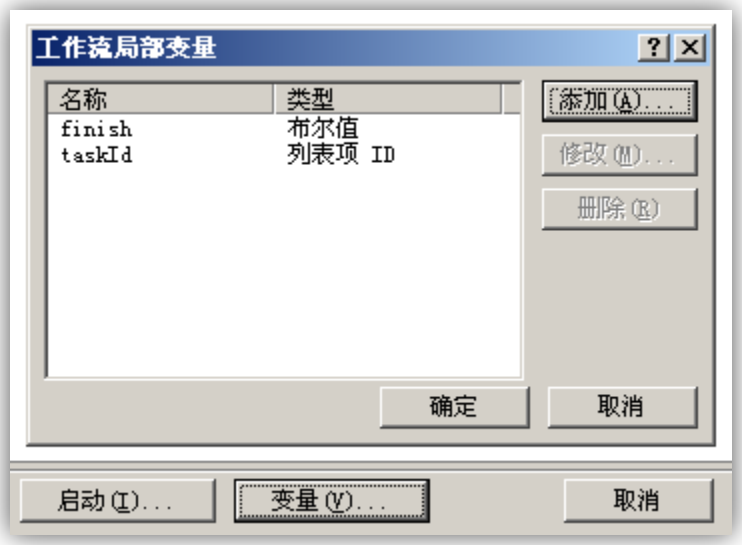
- 名称:**WhileInDesigner**
- 附加到列表:**文档**



点击**下一步.**

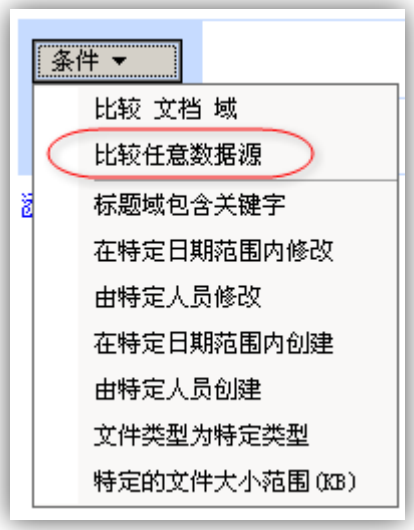
3.新建两个变量.

点击**变量(V)**按钮,添加下列两个变量:

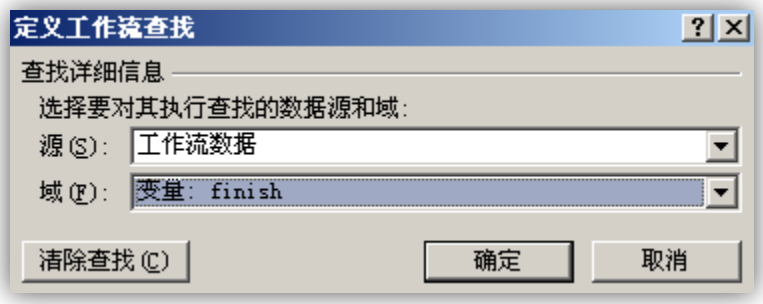


4.添加一个条件.

点击**条件**按钮,选择**比较任意数据源**.



点击**等于**左边的**值** >  > 选择**工作流数据**和**变量:finish**.



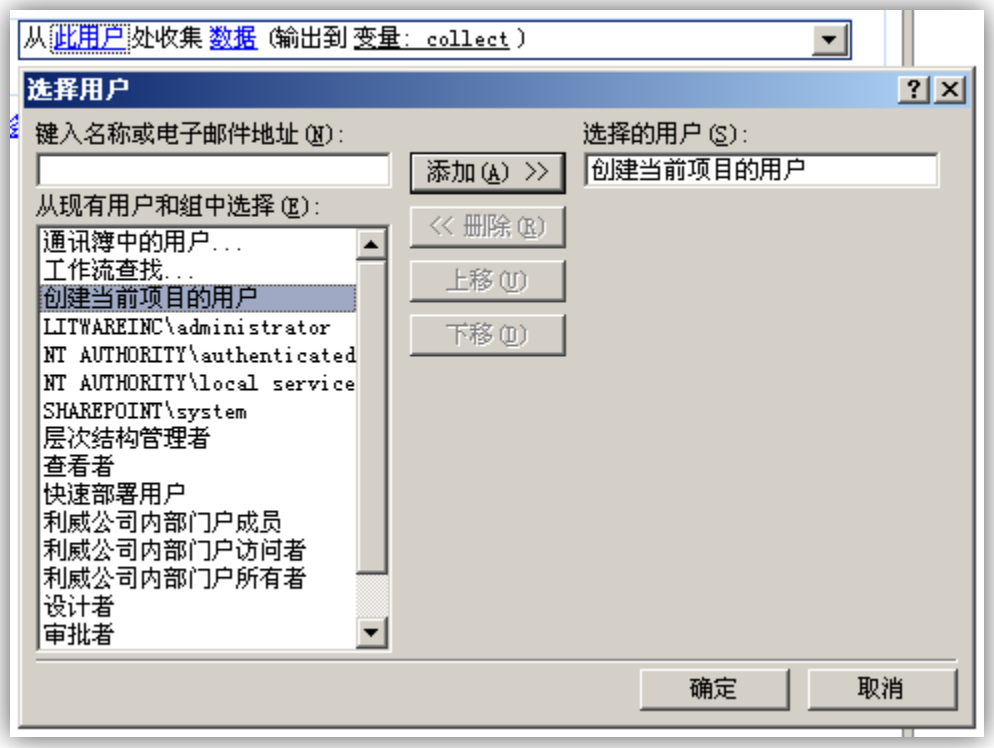
点击**等于**右边的**值**,选择**否**.

5.添加操作:收集用户数据

下面我们来添加一个收集用户数据的操作,事实上就是为用户创建一个任务.

点击**条件 > 从用户处收集数据**.

点击**此用户**,选择**创建当前项目的用户**.

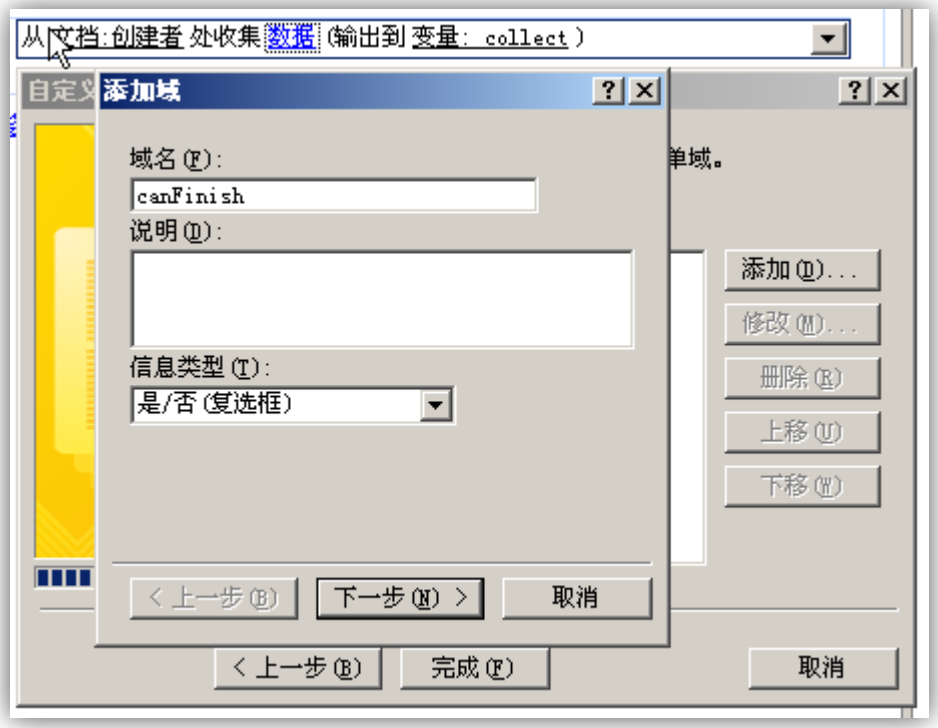


点击**数据 > 下一步**,输入任务名称和说明信息.

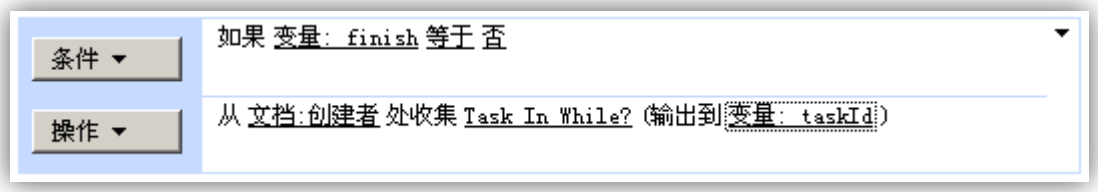
- 任务名称:**Task In While?**

点击**下一步 > 添加**,添加一个向用户收集的数据 canFinish:

- 域名:**canFinish**
- 信息类型:**是/否(复选框)**
- 默认值:**否**



设置**输出到变量**为 **taskID**.

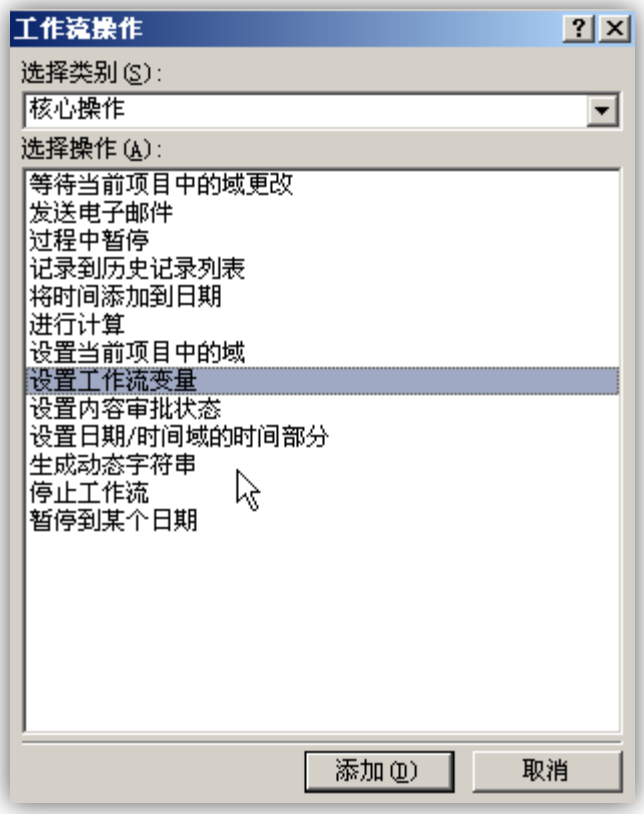


点击**变量(V)**删除自动生成的变量 **collect**.

6.添加操作:获取用户数据

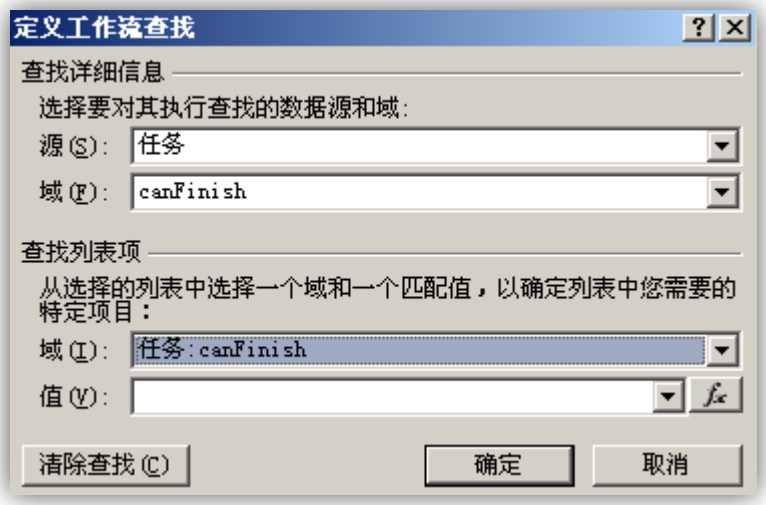
再添加一个操作来获取刚才收集的 canFinish 值并将其赋给变量 finish.

点击**操作 > 其他操作 > 设置工作流变量**.

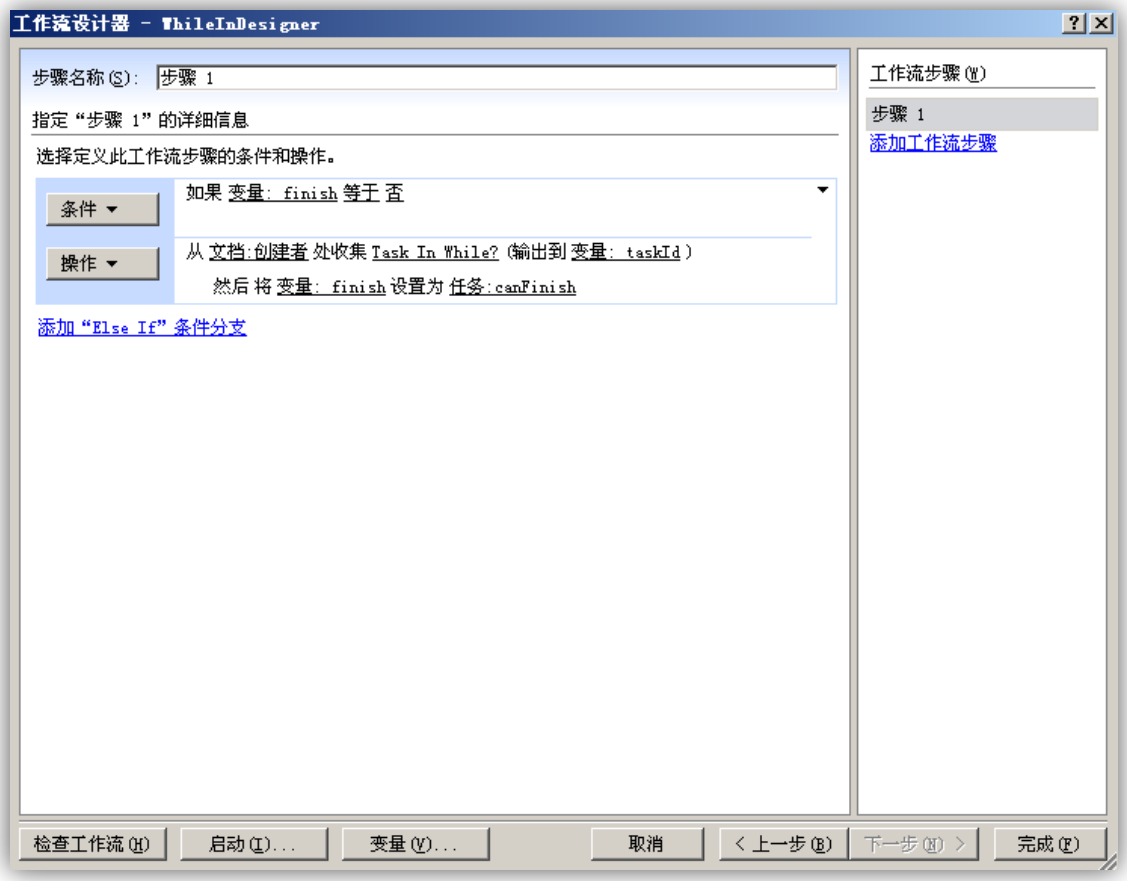


点击**工作流变量**,选择**变量:finish**.

点击**值 >**  **>** 按照下图设置:



最终的工作流如下图所示:



点击**完成**,SharePoint Designer 会**保存**,**验证**并自动将工作流**关联**到之前选择附加的列表上.

7.测试工作流

现在来测试一下刚才设计的工作流.

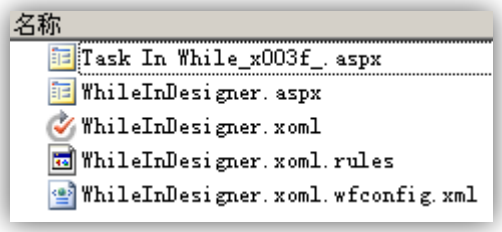
打开 **SharePoint 文档中心 > 文档**,上传一个文档,为其启动 WhileInDesigner 工作流.

打开**任务**,编辑 WhileInDesigner 工作流创建的 **Task In While?**任务,点击**完成任务**,这时 Task In While?任务和 WhileInDesigner 工作流的状态都是**已完成**.

修改 XOML 实现循环

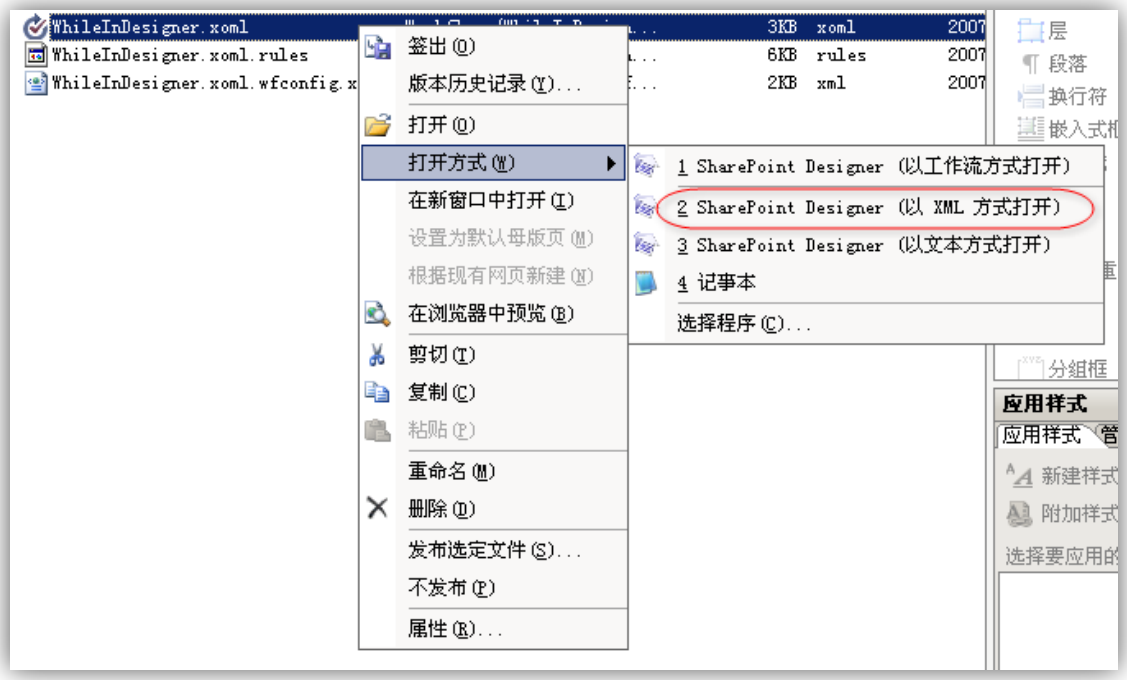
1.SharePoint Designer 生成的文件

回到 SharePoint Designer,我们发现 SharePoint Designer 自动生成了以下文件:



- **Task In While_003f_.aspx**：Task In While?任务的寄主页
- **WhileInDesigner.aspx**： 工作流初始化页
- **WhileInDesigner.xoml**： 工作流定义文件
- **WhileInDesigner.xoml.rules**： 工作流规则文件
- **WhileInDesigner.xoml.wfconfig.xml**： 工作流配置文件

右键单击 **WhileInDesigner.xoml**,选择**以 XML 方式打开**.



可以看到如下 XOML 文件:

```
<ns0:RootWorkflowActivityWithData x:Class="Microsoft.SharePoint.Workflow.ROOT" x:Name="ROOT"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/workflow"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:ns0="clr-namespace:Microsoft.SharePoint.WorkflowActions;Assembly=Microsoft.SharePoint.
WorkflowActions, Version=12.0.0.0, Culture=neutral, PublicKeyToken=null">
    <ns0:RootWorkflowActivityWithData.WorkflowFields>
        <ns0:WorkflowDataField Name="__list" Type="System.String" />
        <ns0:WorkflowDataField Name="__item" Type="System.Int32" />
        <ns0:WorkflowDataField Name="__context"
Type="Microsoft.SharePoint.WorkflowActions.WorkflowContext" />
        <ns0:WorkflowDataField Name="__initParams"
Type="Microsoft.SharePoint.Workflow.SPWorkflowActivationProperties" />
        <ns0:WorkflowDataField Name="__workflowId" Type="System.Guid" />
        <ns0:WorkflowDataField Name="finish" Type="System.Boolean" />
        <ns0:WorkflowDataField Name="taskId" Type="System.Int32" />
        <ns0:WorkflowDataField Name="_x005f_String0" Type="System.String" />
        <ns0:WorkflowDataField Name="_x005f_Boolean0" Type="System.Boolean" />
        <ns0:WorkflowDataField Name="_x005f_Int320" Type="System.Int32" />
    </ns0:RootWorkflowActivityWithData.WorkflowFields>
    <ns0:OnWorkflowActivated WorkflowProperties="{ActivityBind ROOT,Path=__initParams}"
x:Name="ID1">
        <ns0:OnWorkflowActivated.CorrelationToken>
            <wf0:CorrelationToken Name="refObject" OwnerActivityName="ROOT"
```

```
xmlns:wf0="http://schemas.microsoft.com/winfx/2006/xaml/workflow" />
          </ns0:OnWorkflowActivated.CorrelationToken>
     </ns0:OnWorkflowActivated>
     <ns0:ApplyActivation __Context="{ActivityBind ROOT,Path=__context}" x:Name="ID2"
__WorkflowProperties="{ActivityBind ROOT,Path=__initParams}" />
     <IfElseActivity x:Name="ID6" Description="步骤□ 1">
          <IfElseBranchActivity x:Name="ID3">
               <IfElseBranchActivity.Condition>
                    <RuleConditionReference ConditionName="__Rule_ID3_1" />
               </IfElseBranchActivity.Condition>
               <ns0:LookupActivity ListId="{}{A4716DC6-6214-4B4E-A400-BBEEEFBA74CD}"
x:Name="ID9" FieldName="Author" LookupFunction="LookupUser" __Context="{ActivityBind
ROOT,Path=__context}" ListItem="{ActivityBind ROOT,Path=__item}">
                    <ns0:LookupActivity.ReturnValue>
                         <ActivityBind Name="ROOT" Path="_x005f_String0" />
                    </ns0:LookupActivity.ReturnValue>
               </ns0:LookupActivity>
               <ns0:CollectDataTask x:Name="ID8"
ContentTypeId="0x01080100D337657FD8FFC44B8D75294F67F6900E00E892A7A4399FB5499230A2887D54EDD4"
TaskId="{ActivityBind ROOT,Path=taskId}" Title="Task In While?" InProperties="{x:Null}"
__Context="{ActivityBind ROOT,Path=__context}" OutProperties="{x:Null}"
AssignedTo="{ActivityBind ROOT,Path=_x005f_String0}" />
               <ns0:FindValueActivity x:Name="ID12" ReturnValue="{ActivityBind
ROOT,Path=_x005f_Int320}" ExternalFieldName="canFinish" __Context="{ActivityBind
ROOT,Path=__context}" ValueType="System.Boolean"
ExternalListId="{}{B0880D9F-DD1C-47AA-9369-FB1081BF054C}">
                    <ns0:FindValueActivity.FindValue>
                         <ns1:Boolean xmlns:ns1="clr-namespace:System;Assembly=mscorlib,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">false</ns1:Boolean>
                    </ns0:FindValueActivity.FindValue>
               </ns0:FindValueActivity>
               <ns0:LookupActivity ListId="{}{B0880D9F-DD1C-47AA-9369-FB1081BF054C}"
x:Name="ID11" FieldName="canFinish" LookupFunction="LookupBool" __Context="{ActivityBind
ROOT,Path=__context}" ListItem="{ActivityBind ROOT,Path=_x005f_Int320}">
                    <ns0:LookupActivity.ReturnValue>
                         <ActivityBind Name="ROOT" Path="_x005f_Boolean0" />
                    </ns0:LookupActivity.ReturnValue>
               </ns0:LookupActivity>
               <ns0:SetVariableActivity x:Name="ID10" ValueType="System.Boolean">
                    <ns0:SetVariableActivity.Variable>
                         <ActivityBind Name="ROOT" Path="finish" />
                    </ns0:SetVariableActivity.Variable>
                    <ns0:SetVariableActivity.Value>
                         <ActivityBind Name="ROOT" Path="_x005f_Boolean0" />
```

```
                </ns0:SetVariableActivity.Value>
            </ns0:SetVariableActivity>
        </IfElseBranchActivity>
    </IfElseActivity>
</ns0:RootWorkflowActivityWithData>
```

2.根元素:RootWorkflowActivityWithData 元素

从根元素 **RootWorkflowActivityWithData** 的三个名称空间可以看出,SharePoint Designer 设计的工作流需要 **Windows Workflow Foundation 和 Windows SharePoint Services 3.0** 共同支持.

3.替换 IfElseActivity

继续往下看,**IfElseActivity** 元素表示我们添加的条件,它对应于 WinWF 中的 IfElseActivity 活动.在 XOML 中,WinWF 元素和活动的名称是相同的.

在 SharePoint Designer 设计的工作流中,**步骤**总是 **SequenceActivity** 元素或者 **IfElseActivity** 元素.

所以这里我们将 IfElseActivity 元素替换为 SequenceActivity 元素.

IfElseActivity 活动的每一条分支都是一个 **IfElseBranchActivity** 活动,我们不再需要它,把它修改为 **WhileActivity**.

将 **IfElseBranchActivity.Codition** 改为 **WhileActivity.Codition**.

这就是我们之前设计工作流时添加条件的目的,利用条件来生成我们需要的规则文件,因为在本文中,WhileActivity 循环的条件也是 **finish=false**.

现在原来的 IfElseActivity 已经被修改为 SequenceActivity 和及其子元素 WhileActivity 了.

4.WhileActivity 的子活动

在 WinWF 中, WhileActivity 有且只能有一个子活动,然而我们现在的 WhileActivity 中却包含了多个子活动(基本上一个元素就是一个子活动),所以我们还需要添加一个 SequenceActivity 将其余的子活动包含在内.

然后我们来看看这些 SequenceActivity 的子活动们.

5.LookUpActivity 与 SharePoint Designer 自动生成的变量

LookUpActivity 的名称空间前缀是 ns0,表示它是一个 SharePoint Workflow Action.

有过 Visual Studio 工作流开发经验的朋友一定会想起 CreateTask 等活动,不错,LookUpActivity 与 CreateTask 活动同属于 SharePoint Workflow Actions,但却有所不同, CreateTask 等活动被 Visual Studio 2005 Designer for Windows Workflow Foundation 支持,支持编码开发;而 LookUpActivity 等活动基于定义和规则,且不被 Visual Studio 2005 Designer for Windows Workflow Foundation 支持.

LookUpActivity 的作用是从指定的列表项中获取某一域的值,这里,它从我们上传的文档中获取上传者的名称,并将其存储在变量**_x005f_String0** 中.

_x005f_String0 是 SharePoint Designer 自动生成的变量.其前缀**_x005f_**表示它是由 SharePoint 自动生成的,如果没有此前缀,那么我们可以在 SharePoint Designer 工作流设计器中看到此变量,中间的 **String** 表示其类型,末位 **0** 表示序号(这也是 SharePoint Designer 自动生成的中间变量的命名规则).

6.CollectDataTask

CollectDataTask 用来收集用户数据(创建任务),它的属性描述了该任务的内容类型,分配对象,标题,甚至 GUID(CollectDataTask 将任务的 **TaskID** 绑定到了变量 **taskID** 上).

7.FindValueActivity

FindValueActivity 的作用大概和 LookUpActivity 差不多,但是 FindValueActivity 会提供一个默认值,本文的实验中它的存在与否似乎无关紧要,于是删之.

8.又一个 LookUpActivity

接着又是一个 LookUpActivity,将其 ListItem 属性值中的**_x005f_Int320** 修改为 **taskID**,这样 LookUpActivity 就会从 Task In While?任务中获取 **canFinish** 域的值并将其存储在变量**_x005f_Boolean0** 中

9.SetVariableActivity

SetVariableActivity 是一个给变量赋值的活动,这里它将**_x005f_Boolean0** 的值赋给变量 **finish.**

10.最终的 XOML 文件及其逻辑

OK,所有活动已经介绍并且修改完毕,最终的 XOML 文件内容如下:

```
<ns0:RootWorkflowActivityWithData x:Class="Microsoft.SharePoint.Workflow.ROOT" x:Name="ROOT"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/workflow"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:ns0="clr-namespace:Microsoft.SharePoint.WorkflowActions;Assembly=Microsoft.SharePoint.
WorkflowActions, Version=12.0.0.0, Culture=neutral, PublicKeyToken=null">
    <ns0:RootWorkflowActivityWithData.WorkflowFields>
        <ns0:WorkflowDataField Name="__list" Type="System.String" />
        <ns0:WorkflowDataField Name="__item" Type="System.Int32" />
        <ns0:WorkflowDataField Name="__context"
Type="Microsoft.SharePoint.WorkflowActions.WorkflowContext" />
        <ns0:WorkflowDataField Name="__initParams"
Type="Microsoft.SharePoint.Workflow.SPWorkflowActivationProperties" />
```

```
            <ns0:WorkflowDataField Name="__workflowId" Type="System.Guid" />
            <ns0:WorkflowDataField Name="finish" Type="System.Boolean" />
            <ns0:WorkflowDataField Name="taskId" Type="System.Int32" />
            <ns0:WorkflowDataField Name="_x005f_String0" Type="System.String" />
            <ns0:WorkflowDataField Name="_x005f_Boolean0" Type="System.Boolean" />
            <ns0:WorkflowDataField Name="_x005f_Int320" Type="System.Int32" />
        </ns0:RootWorkflowActivityWithData.WorkflowFields>
        <ns0:OnWorkflowActivated WorkflowProperties="{ActivityBind ROOT,Path=__initParams}"
x:Name="ID1">
            <ns0:OnWorkflowActivated.CorrelationToken>
                <wf0:CorrelationToken Name="refObject" OwnerActivityName="ROOT"
xmlns:wf0="http://schemas.microsoft.com/winfx/2006/xaml/workflow" />
            </ns0:OnWorkflowActivated.CorrelationToken>
        </ns0:OnWorkflowActivated>
        <ns0:ApplyActivation __Context="{ActivityBind ROOT,Path=__context}" x:Name="ID2"
__WorkflowProperties="{ActivityBind ROOT,Path=__initParams}" />
        <SequenceActivity x:Name="ID6" Description="步骤□ 1">
            <WhileActivity x:Name="ID3">
                <WhileActivity.Condition>
                    <RuleConditionReference ConditionName="__Rule_ID3_1" />
                </WhileActivity.Condition>
                <SequenceActivity x:Name="subSequence">
                    <ns0:LookupActivity ListId="{}{A4716DC6-6214-4B4E-A400-BBEEEFBA74CD}"
x:Name="ID9" FieldName="Author" LookupFunction="LookupUser" __Context="{ActivityBind
ROOT,Path=__context}" ListItem="{ActivityBind ROOT,Path=__item}">
                        <ns0:LookupActivity.ReturnValue>
                            <ActivityBind Name="ROOT" Path="_x005f_String0" />
                        </ns0:LookupActivity.ReturnValue>
                    </ns0:LookupActivity>
                    <ns0:CollectDataTask x:Name="ID8"
ContentTypeId="0x01080100D337657FD8FFC44B8D75294F67F6900E00E892A7A4399FB5499230A2887D54EDD4"
TaskId="{ActivityBind ROOT,Path=taskId}" Title="Task In While?" InProperties="{x:Null}"
__Context="{ActivityBind ROOT,Path=__context}" OutProperties="{x:Null}"
AssignedTo="{ActivityBind ROOT,Path=_x005f_String0}" />
                    <ns0:LookupActivity ListId="{}{B0880D9F-DD1C-47AA-9369-FB1081BF054C}"
x:Name="ID11" FieldName="canFinish" LookupFunction="LookupBool" __Context="{ActivityBind
ROOT,Path=__context}" ListItem="{ActivityBind ROOT,Path=taskId}">
                        <ns0:LookupActivity.ReturnValue>
                            <ActivityBind Name="ROOT" Path="_x005f_Boolean0" />
                        </ns0:LookupActivity.ReturnValue>
                    </ns0:LookupActivity>
                    <ns0:SetVariableActivity x:Name="ID10" ValueType="System.Boolean">
                        <ns0:SetVariableActivity.Variable>
                            <ActivityBind Name="ROOT" Path="finish" />
```

```
                    </ns0:SetVariableActivity.Variable>
                    <ns0:SetVariableActivity.Value>
                        <ActivityBind Name="ROOT" Path="_x005f_Boolean0" />
                    </ns0:SetVariableActivity.Value>
                </ns0:SetVariableActivity>
            </SequenceActivity>
        </WhileActivity>
    </SequenceActivity>
</ns0:RootWorkflowActivityWithData>
```

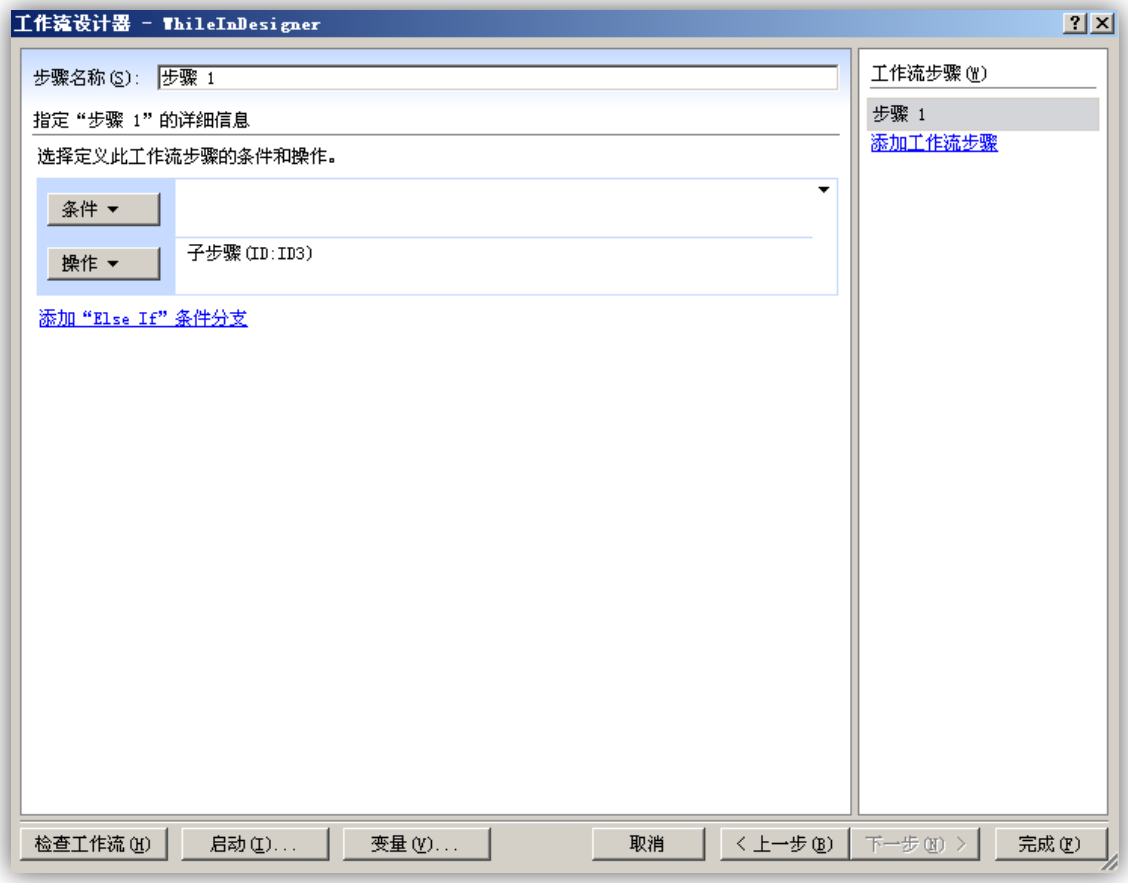
如我们之前的描述,工作流启动之后,WhileActivity 将检查变量 finish 的值,如果为 false(默认就是 false),则执行 WhileActivity 的子活动.

WhileActivity 的子活动的执行过程是:LookUpActivity 获取文档的创建者, CollectDataTask 为文档创建者分配任务,待任务完成后, LookUpActivity 获取任务中 canFinish 域的值, SetVariableActivity 将其值赋给变量 finish.

子活动执行完毕,WhileActivity 再次检查 finish 的值,而此时 finish 的值可能被 SetVariableActivity 改变,从而根据此值来继续或停止循环.

11.工作流的关联与测试

现在保存 XOML 文件并关闭,然后双击以工作流方式打开,因为我们修改了 XOML 文件,所以相比之前的图片有所区别.



步骤 1 中只包含一个操作:**子步骤(ID:ID3)**.在该工作流的 XOML 文件中可以看到,ID3 正是 WhileActivity 的 ID,说明 SharePoint Designer 可以识别 WhileActivity,但无法正确的将其表示出来.

点击**完成**来完成工作流与文档的关联,如果忽略了这一步,SharePoint 中的工作流将不发生任何变化.

最后来**测试**一下,编辑任务时,不要勾选 canFinish,则工作流会自动再分配一项任务,直到你勾选了 canFinish.

标题	分配对象	状态
Task In While? !新	系统帐户	已完成
Task In While? !新	系统帐户	已完成
Task In While? !新	系统帐户	未启动

总结

通过修改 SharePoint Designer 生成的 XOML 文件,使 SharePoint Designer 设计的工作流能够支持 WhileActivity,那我们也有理由相信它可以支持更多的 WinWF 活动.

而且,SharePoint Workflow Actions 中的两类活动虽然使用方法不尽相同,但是实现的功能却是一样的(如 CreateTask 和 CollectDataTask),这是不是意味着使用 Visual Studio 开发的工作流也可以使用 SharePoint Designer 间接的设计出来?

虽然经过修改的工作流已经无法在 SharePoint Designer 中再设计,那么我们可不可以扩展 SharePoint Designer 的工作流设计器使其支持更加复杂的工作流?或者干脆开发第三方工作流设计工具?

留待慢慢研究...