开发 SHAREPOINT SERVER 2007 多级审批工作流

Sequential 版 柴晓伟 2007.2.6 (2007.8.5 v1.1 修正)

一.介绍

SharePoint Server 2007 中内置了一个审批工作流,当审批者选择”批准”或者”拒绝”送审者送交的审批请求后工作流就结束了.

然而在实际情况中,我们可能希望在”拒绝”请求之后给送审者一个提示并且使他能够更加方便的再次提交送审,而不需要再次启动工作流;我们也可能希望将审批同时送交给多个审批者或者按照审批者的级别由低到高依序送交.

本文中我们要开发的工作流涉及三层级别的审批,具体的审批流程参见图 1:

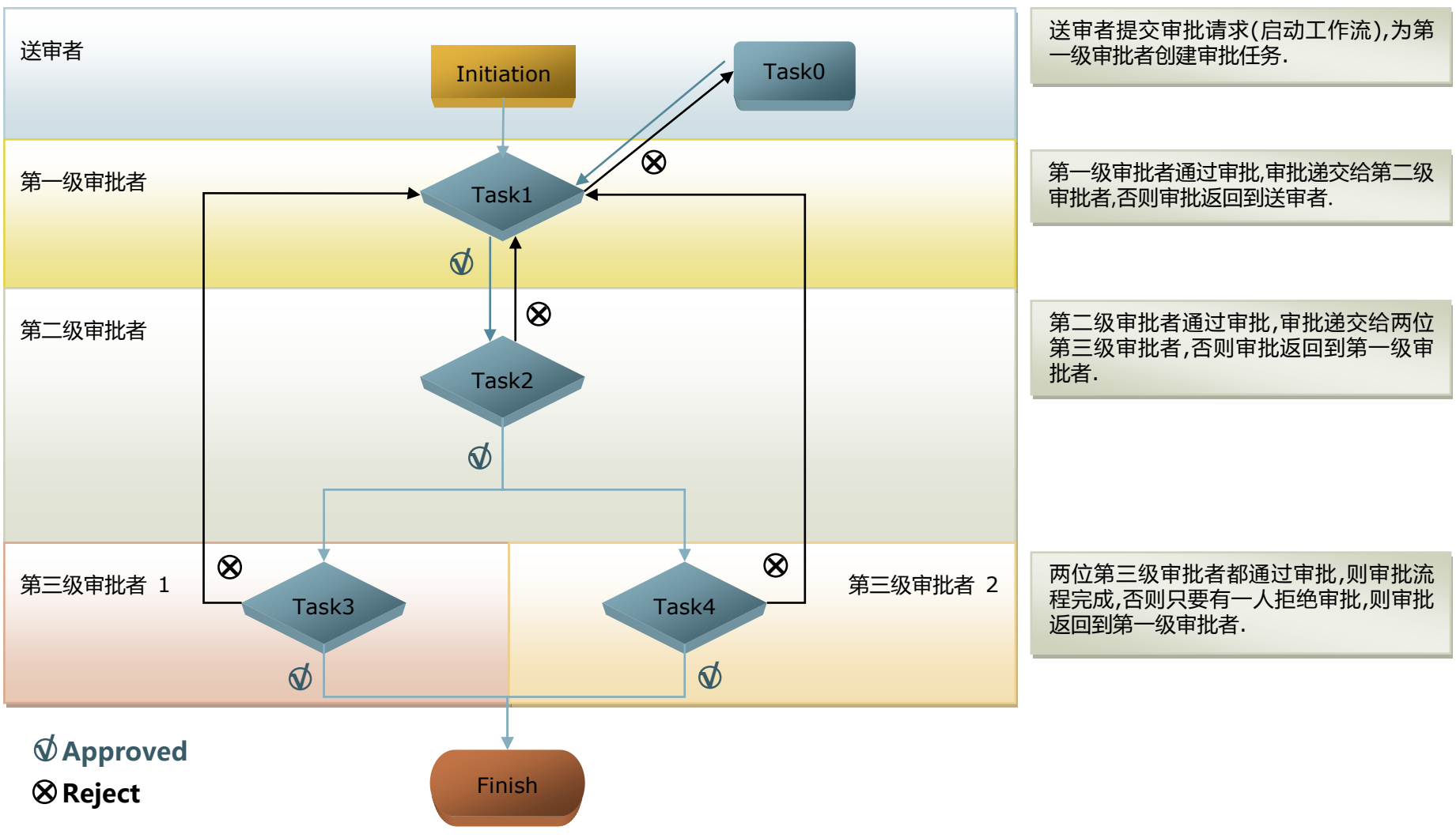


图 1.

二.开发环境&准备工作

- SharePoint Server 2007
- Visual Studio 2005
- InfoPath 2007
- .NET Framework 3.0 (下载)
- Visual Studio 2005 Extensions for Windows Workflow Foundation (下载)
- ECM Starter Kit for Visual Studio 2005 (下载)

为了开发,部署和调试的方便,我们在 SharePoint Server 2007 服务器上开发工作流,在安装好上述软件之后,还需要向 Visual Studio 2005 添加 SharePoint Workflow Actions 组件.

- 在 Visual Studio 2005 工具箱的空白处点击**右键 > 添加选项卡** > 添加一个名为 **SharePoint Workflow Actions** 的选项卡.
- 在 SharePoint Workflow Actions 空白处点击**右键 > 选择项 > .NET Framework 组件**.
- 添加所有命名空间为 **Microsoft.SharePoint.WorkflowActions** 的组件,如图 2 所示:

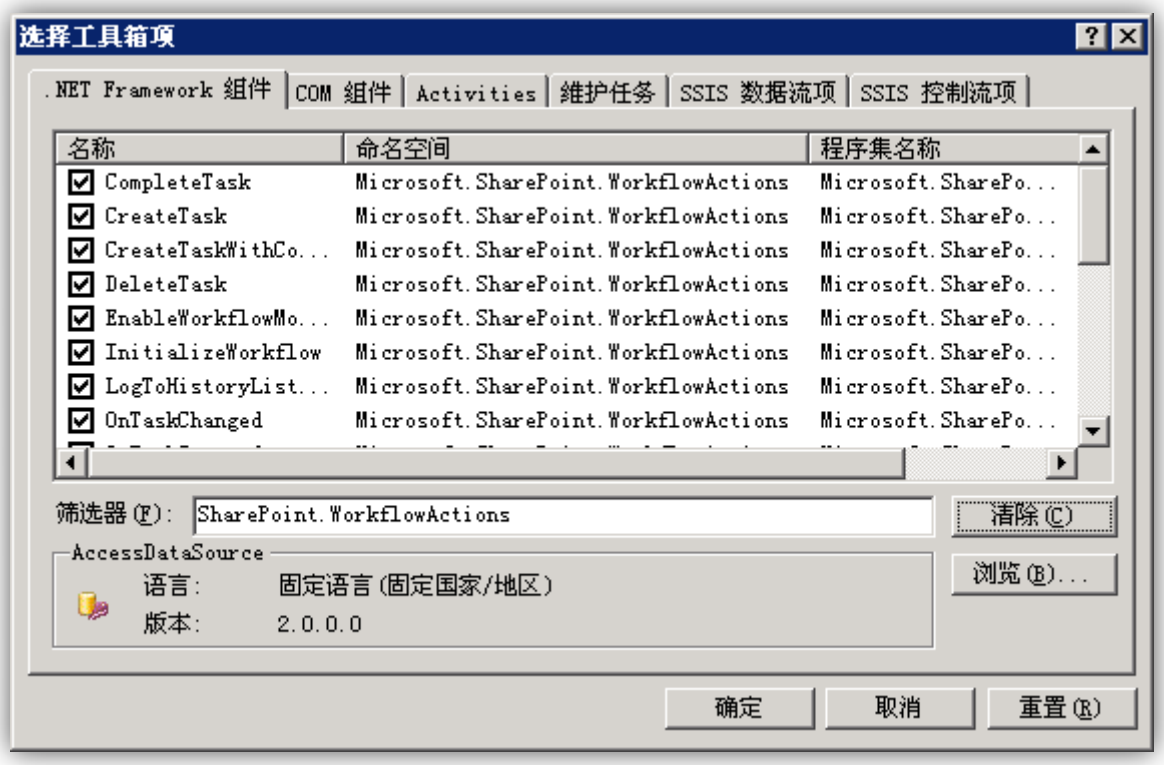


图 2

注: 安装 ECM starter kit 之后如果出现项目模板丢失的情况请参考我的另外一篇文章[<< Visual Studio.net 2005 新建项目对话框中项目模版消失的解决方案>>]

三.在 VISUAL STUDIO 设计工作流流程图

3.1 创建项目

在 Visual Studio 2005 中创建基于 **SharePoint Sequential Workflow** 模版的项目 **ApprovalSequentialEdition**:

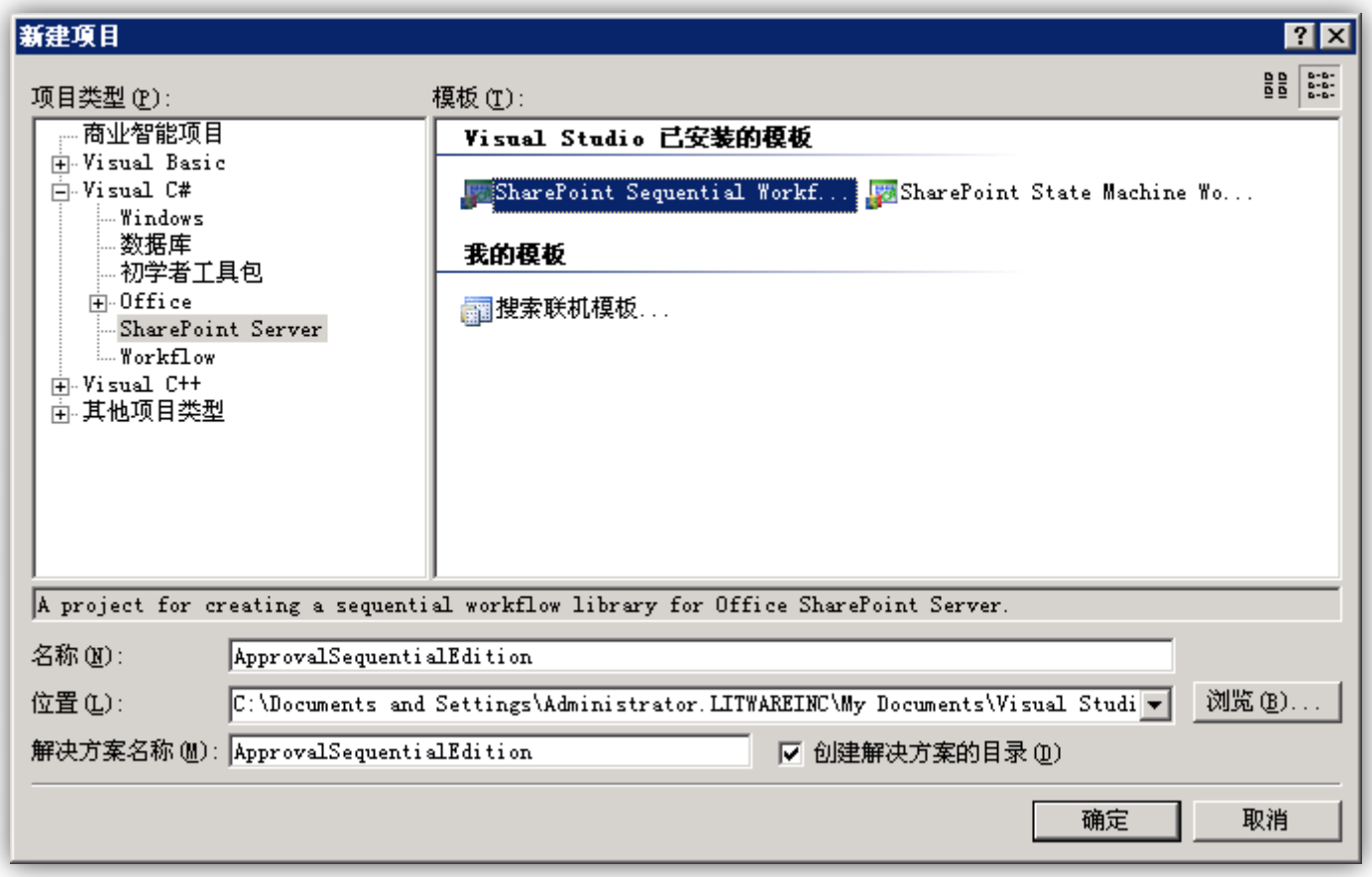


图 3

项目创建完毕之后,可以在解决方案资源管理器中看到当前项目包含如图 4 所示的文件:

图 4

双击打开 Workflow1.cs,进入工作流的 **Design** 模式,将看到如图 5 所示的工作流:

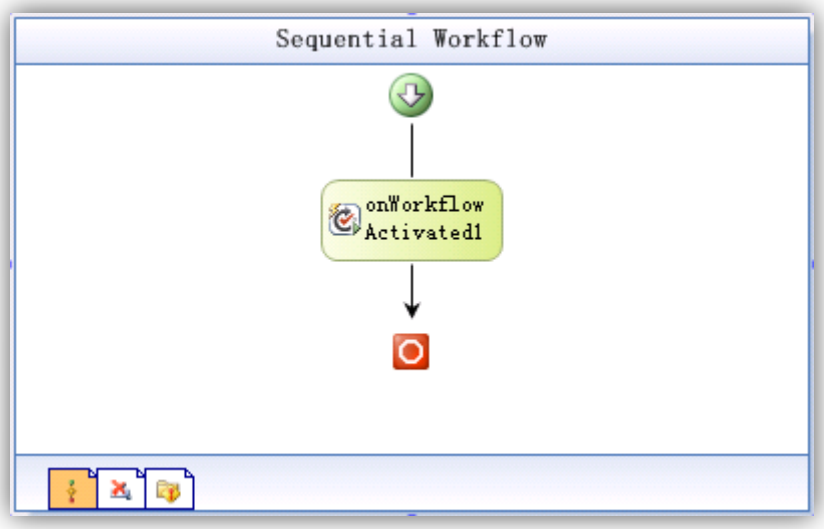


图 5

此时工作流中只有一个 **onWorkflowActivated** 活动,所有 SharePoint 工作流都必须从这个活动开始.

onWorkflowActivated 活动有一些属性值得我们注意:

- **CorrelationToken**：Correlation Token 是将若干相关联的活动映射到同一集合的标识符,其作用与某些编程语言中 RadioButtion 组件的 GroupId 属性类似,注意不要为任务活动和工作流活动指定相同的 CorrelationToken.
    - **OwnerActivityName**： 活动的父容器名称,建议设置为活动的最小父容器名称,尤其当活动被包含在 While 活动中时.
- **WorkflowProperties**： 此属性包含了工作流初始信息,比如启动工作流的人员,列表项,时间等等.

3.2 设计工作流流程图

先整理一下思路,观察图 1,我们要设计的工作流可以循环和并发,而在 Sequential 工作流中活动是以预定的顺序执行的,所以要实现图 1 中可以循环的流程,我们必须借助于 **Windows Workflow** 选项卡中的一个活动:**WhileActivity**.

WhileActivity 类似 C#语法中的 While()循环,不同的是 WhileActivity 的**条件(Condition)**可以是**代码(Code Condition)**,也可以是**规则(Delarative Rule Condition)**.

我们将任务包含在 WhileActivity 中循环执行直到审批通过才跳出循环,可想而知,有多少级审批就需要有多少个 WhileActivity.

由于 WhileActivity 只能包含一个子活动,所以我们还需要在每个 WhileActivity 中添加一个 **SequenceActivity** 作为添加多个活动的”容器”.

并发的实现则需要借助于 **Windows Workflow** 选项卡中的 **ParallelActivity**,此活动可以实现多条分支的并发,并且当所有分支都结束之后它才会结束.

并发的任务(Task3 和 Task4)完成之后,我们通过 **Windows Workflow** 选项卡中的 **CodeActivity** 执行自定义代码来判断审批结果.

请参照图 6 来设计工作流流程:

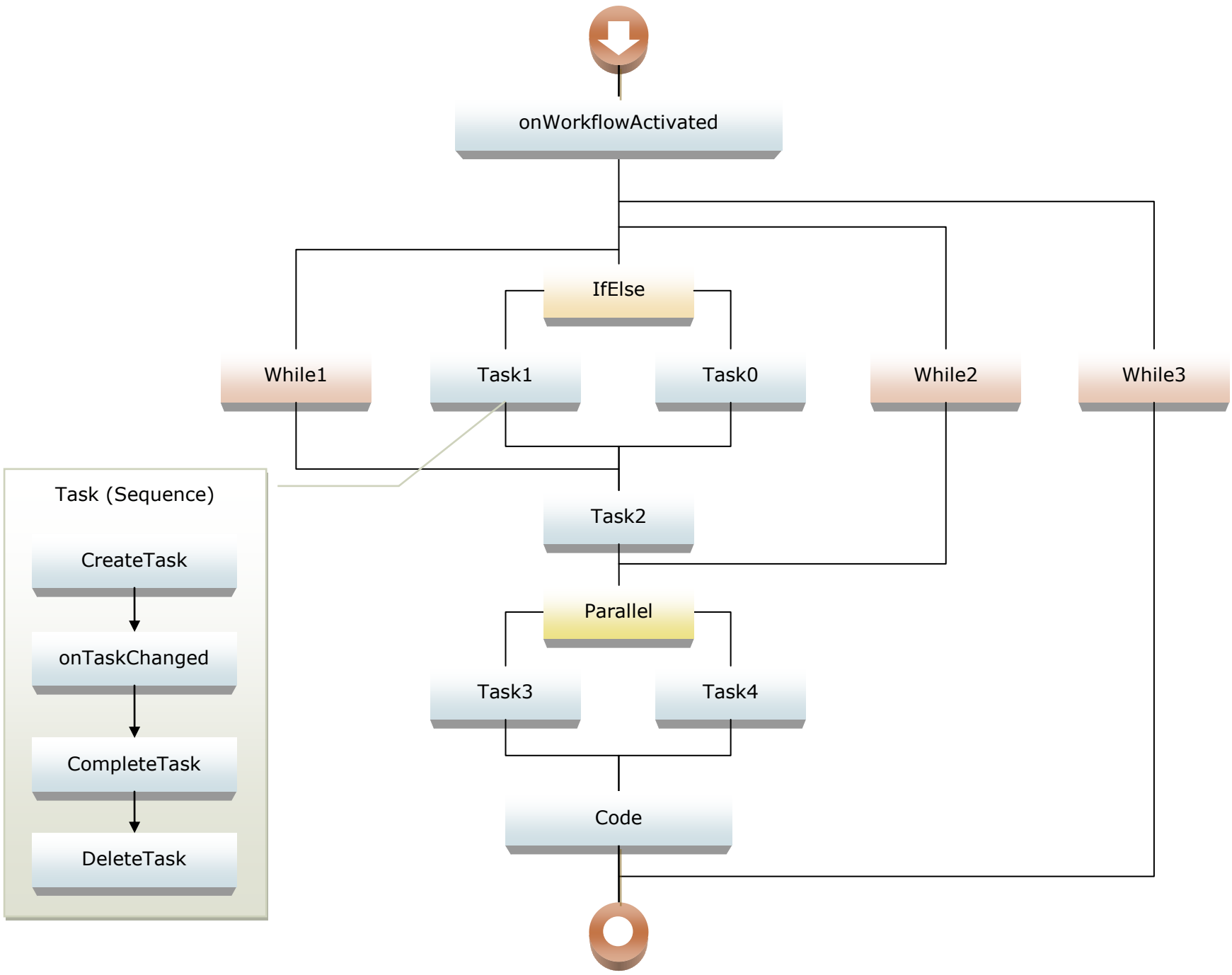


图 6

图 6 中,在 While1 中有一个 **IfElseActivity**,它的两条分支分别是 **Task1**(第一级审批者的任务)和 **Task0**(送审者的任务),这样做的用意是当送审者启动了工作流之后,IfElse 根据我们设定的条件判断,并为第一级审批者创建任务,如果第一级审批者拒绝了审批,那么修改条件并再次循环,IfElse 根据修改过的条件判断并为送审者创建任务,如此循环直到第一级审批者批准了审批才跳出 While1.

除此之外还需要注意以下几点:

- 图中每一个 Task 都是一个 SequenceActivity(IfElseAcitvity 和 ParallelActivity 的分支也是 SequenceActivity), 它依序包含 **CreateTask**,**onTaskChanged**,**CompleteTask** 和 **DeleteTask** 四个活动,代表了一个任务的一次生命周期.
- 同一组 Task 活动都必须设置相同的 **CorrelationToken** 属性和 **TaskId** 属性(在本例中同一组 Task 活动都包含在一个 SequenceActivity 中);而不同组的 Task 活动则需要设置不同的 CorrelationToken 属性和 TaskId 属性.
- 本例中 Task 活动的 OwnerActivityName 必须设置为包含它们的 SequenceActivity 名称.
- 将 onTaskChanged 的 afterProperties 属性设置为和 CreateTask 活动的 TaskProperties 属性相同的变量(这样做的目的是默认保存用户对 Task 的修改).

3.3 示例:TASK1 第一个任务

下面示范创建一组任务并配置它们的属性.

3.3.1 SEQUENCEACTIVITY

从 **Windows Workflow** 选项卡中拖放 **SequenceActivity** 到适当的位置,将其改名为 **task1**.

3.3.1 CreateTask

从我们之前添加的组件中找到 **CreateTask** 组件并将其拖放到 task1 内,下面来配置 CreateTask1 的属性:

- **CorrelationToken**：Task1_Token
- **OwnerActivityName**：task1
- **TaskId**： 点击  右边的  按钮,在弹出的对话框中选择 **Bind to a new member** ,设置 **new member name** 为 **task1_Id**,**Choose a type of member to create** 选择 **Create Field**，点击 **OK** 确定,如图 7 所示:

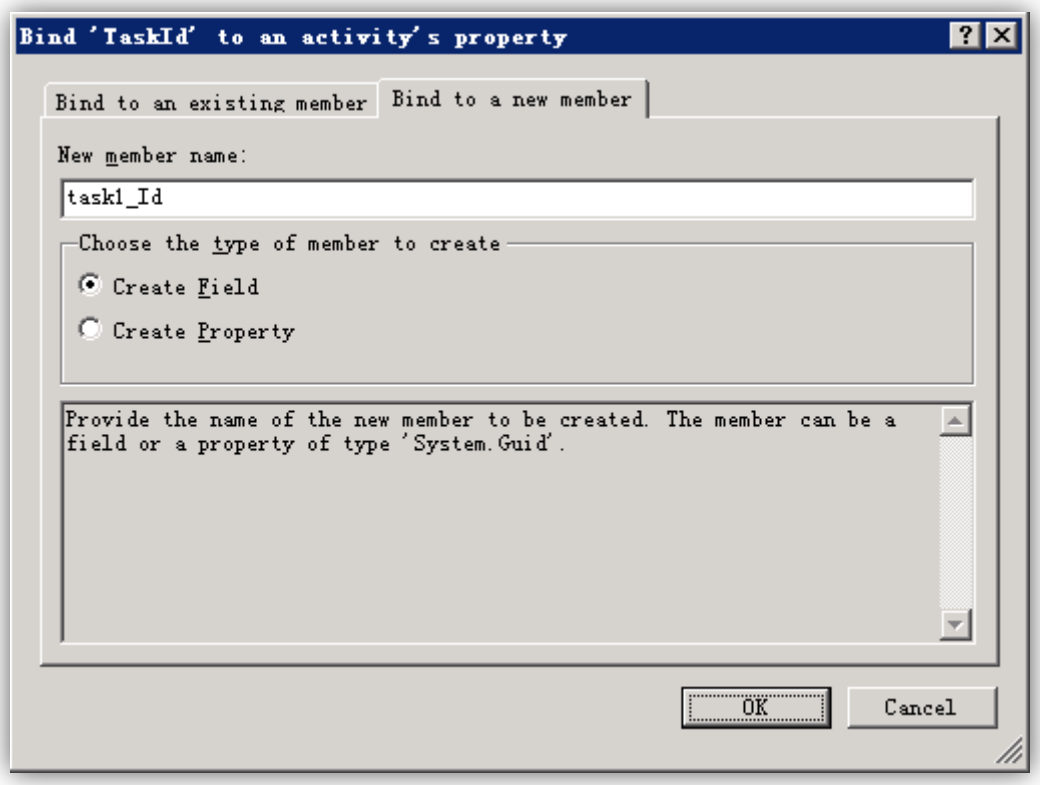


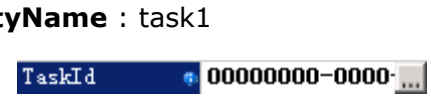
图 7

- 以同样的方法创建名为 **task1_Properties** 的 **TaskProperties** 属性.

3.3.2 OnTaskChanged

CreateTask 配置完毕,以同样的方法添加一个 **OnTaskChanged**.

配置 onTaskChanged1 的属性:

- **CorrelationToken**：Task1_Token
- **OwnerActivityName**：task1
- **TaskId**： 点击  右边的  按钮,在弹出的对话框中选择 **Bind to a existing member** ,选择我们之前创建的 task1_Id,点击 **OK** 确定,如图 8 所示:

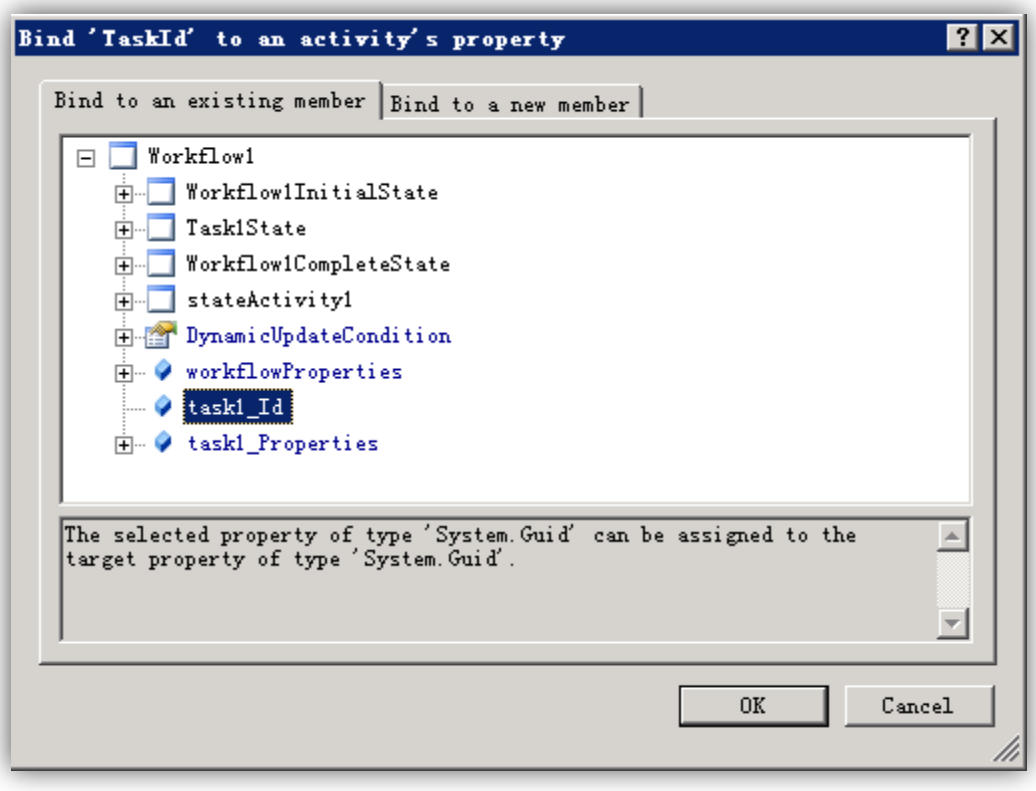


图 8

- 以同样的方法将 **AfterProperties** 属性绑定到 **task1_Properties** .

3.3.3 CompleteTask 和 DeleteTask

配置 CompleteTask 和 DeleteTask 的属性:

- **CorrelationToken**：Task1_Token
- **OwnerActivityName**：Task1
- **TaskId**：task1_Id

按照图 6 设计好工作流流程图并配置每个 task 活动的属性之后,我们可以暂时离开 Visual Studio 2005 来为工作流设计表单了,WhileAcitivity,IfElseAcitivity 和 CodeActivity 还有一些属性我们将在编码的时候完成.

四.InfoPath 表单

4.1 介绍

我们采用 **InfoPath 表单**作为工作流的表单来收集信息,这样做的一个好处是它不仅可以显示在 SharePoint Server 站点上,还可以在 Office 客户端应用程序中显示,这样用户不用打开浏览器就可以参与工作流了.

根据图 1 所示的流程,工作流中不同级别的审批者需要不同的表单:

- 送审者则需要两张,一张用于启动工作流(**Init**),一张用于在审批被拒绝时修改(**Task0**);
- 第一级审批者需要一张(**Task1**);
- 第二级审批者需要一张(**Task2**);
- 两个第三级审批者使用同样的表单(**Task3**)

那么我们一共需要设计 5 张表单.

4.2 Init 表单

下面我们来设计第一张表单,用于启动工作流的 Init 表单.

4.2.1 创建 InfoPath 表单

打开 InfoPath 2007 > **设计表单模板** > 选择 **空白** 模板并勾选 **仅启用浏览器兼容性功能** ,如图 14 所示:

图 9

4.2.2 添加 Contact Selector 控件

Initiation 表单中需要指定第一级审批者,所以我们要添加一个选择联系人的控件 **Contact Selector**,这个控件在 InfoPath 2007 中默认是没有被添加的.

在**设计任务面板**中点击**控件**,打开如图 15 所示的控件面板:

图 10

点击 **添加或删除自定义控件** > **添加** :

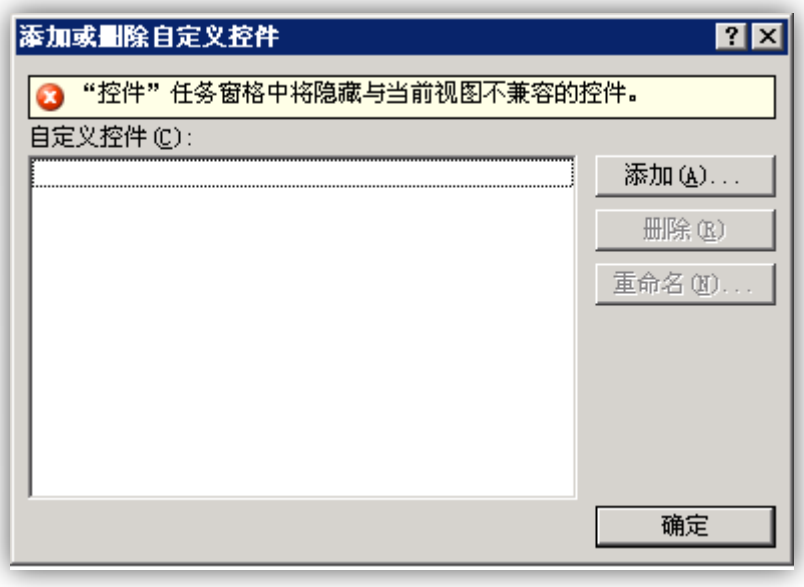


图 11

选择 **ActiveX 控件** > 点击 **下一步** ：

图 12

选择 **Contact Selector** > 点击 **下一步** ：

图 13

选择 **不包括 .cab 文件** > 点击 **下一步**:

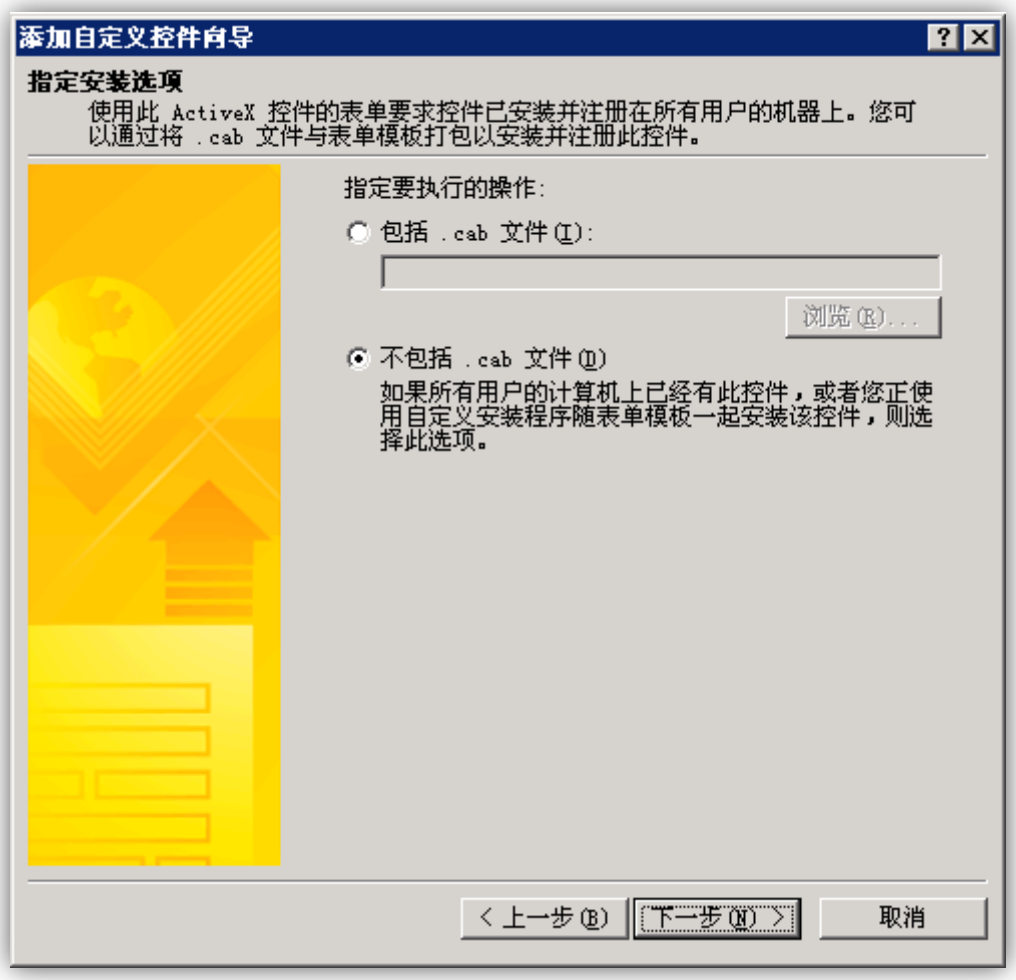


图 14

绑定属性选择 **Value** > 点击 **下一步**:

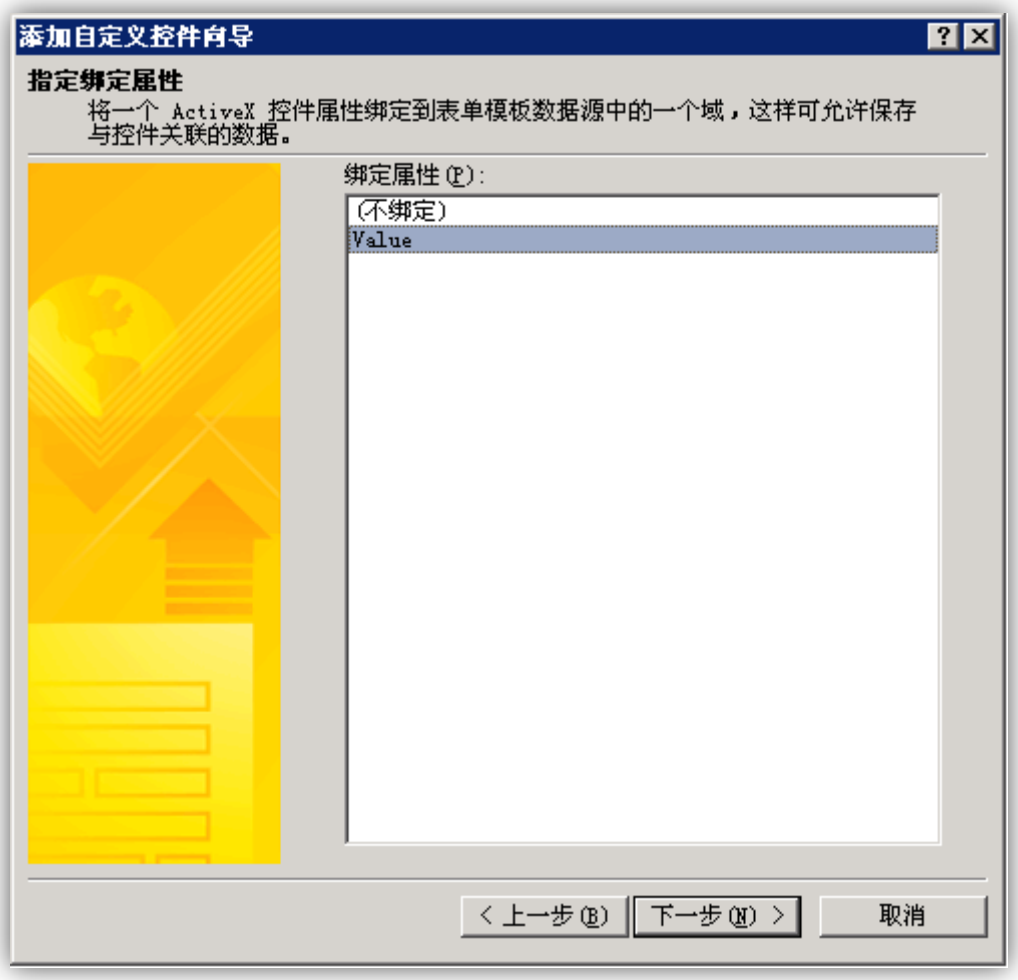


图 15

域或组的类型选择 **域或组(任何数据类型)** > 点击 **完成** > **关闭** > **确定**.

图 16

4.2.3 设计 INIT 表单

现在我们来设计第一张 InfoPath 表单—Init 表单.我们将这张表单作为工作流的初始表单(Initiation Form),它将在工作流实例启动之前显示.

参照图 17 从控件面板中拖放 **Contact Seletor**，**文本框** 和 **按钮**:

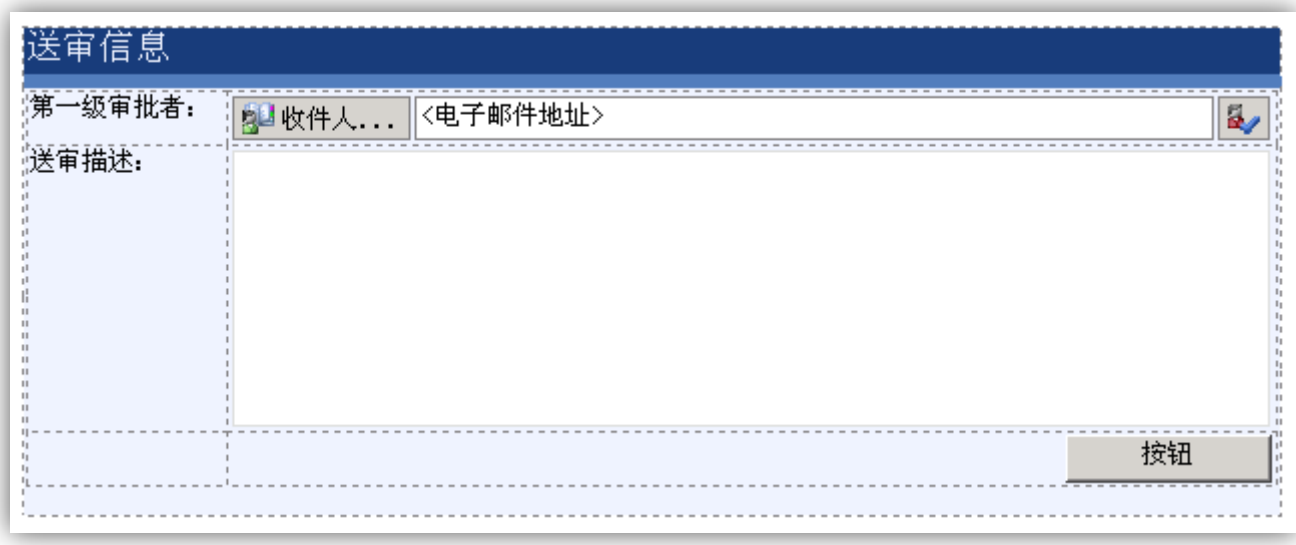


图 17

4.2.4 配置数据源

4.2.4.1 主数据源

在 **设计任务面板** 中点击 **数据源** ,打开 **数据源面板** ,修改域组的名称由 **myFields** 修改为 **Init**.

这样做的原因是 **InfoPath 表单模板中主数据源的根域组名称 = 表单架构的根元素名称 = 表单类名称** ：

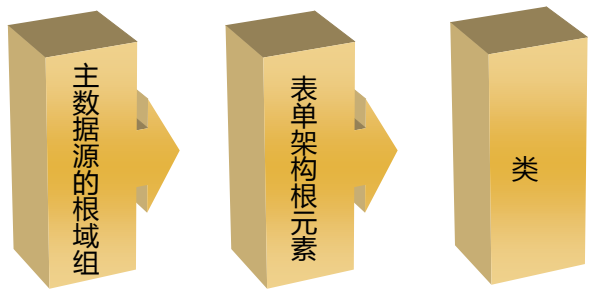


图 18

双击 **文本框** 控件,在 **数据** 页中将其 **域名称** 修改为 **comments**,在 **显示** 页中勾选 **多行**.

双击 **Contact Selector** 控件,将其 **域或组的名称** 修改为 **contact**.

此外,我们还必须为 Contact Selector 控件创建数据架构,Contact Selector 将根据此架构保存联系人信息.

右键单击 **contact** 域组 > **添加** > 添加一个名称为 **Person**，类型为 **组** 并且 **重复** 的域组,如图 24 所示:

图 19

在 **Person** 域组下添加 3 个名称为 **DisplayName**，**AccountId** 和 **AccountType** ,类型为 **文本(string)** 的域,最终表单模板的主数据源域如图 25 所示:

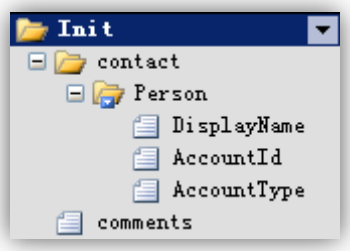


图 20

4.2.4.3 按钮提交规则

双击 **按钮** 控件,将其标签修改为 **启动工作流** ,点击 **规则** ：

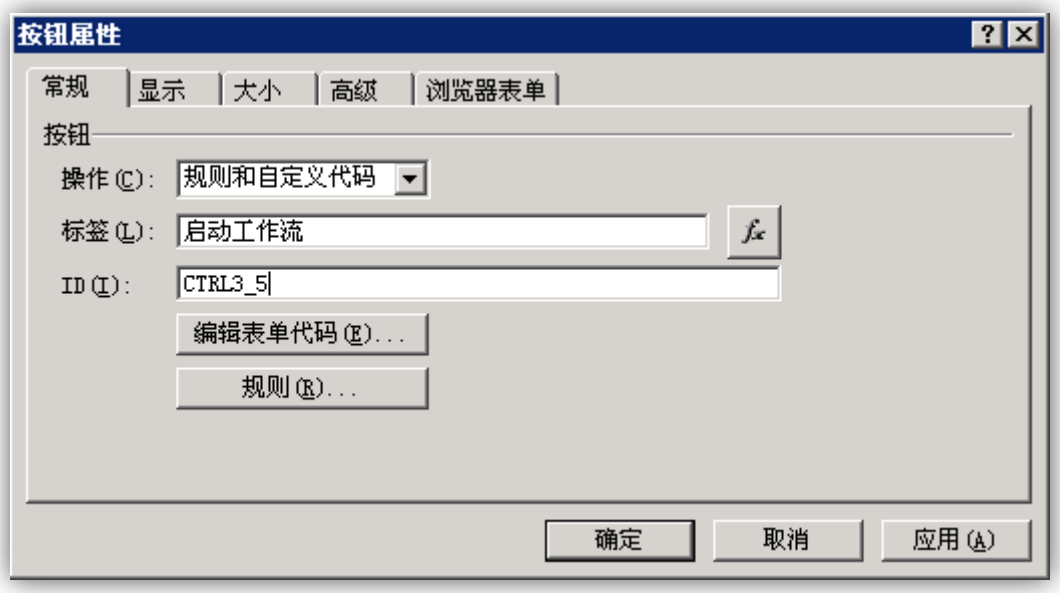


图 21

点击 **添加** ：

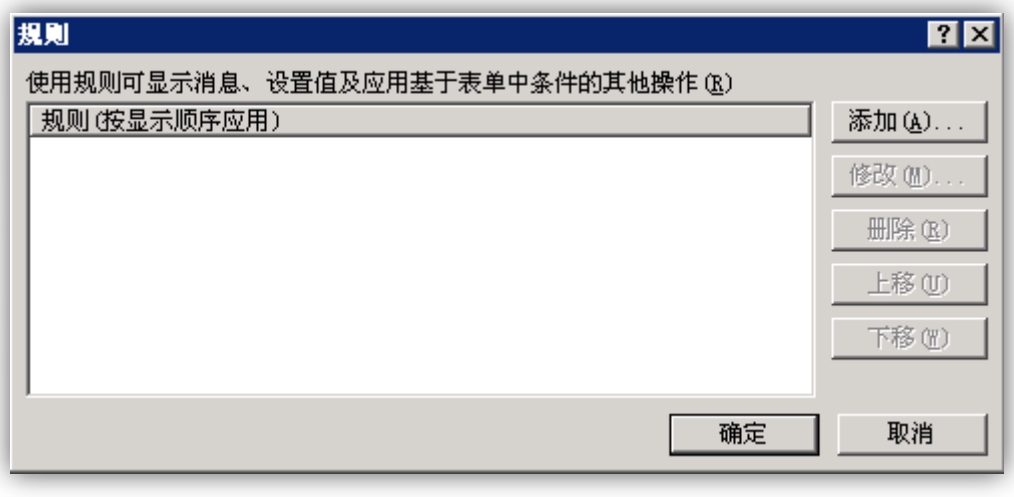


图 22

点击 **添加操作** ：

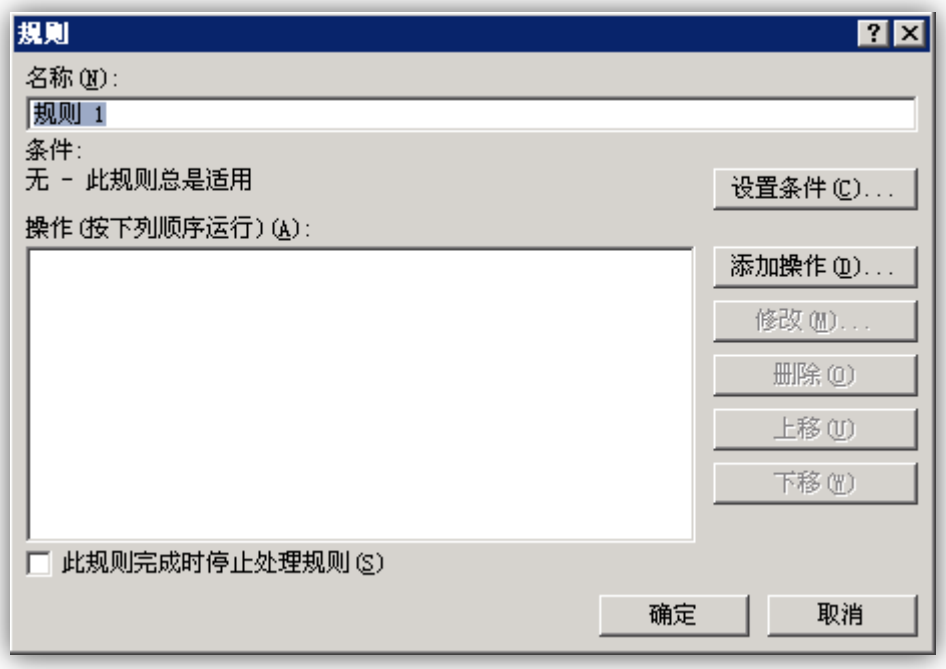


图 23

在 **操作** 下方选择 **使用数据连接进行提交** ,点击 **添加** ：

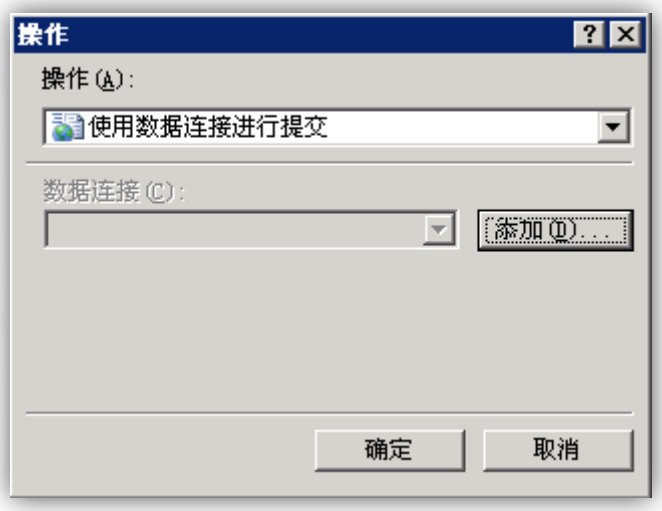


图 24

选择 **新建连接** > **仅提交数据** ,点击 **下一步** ：

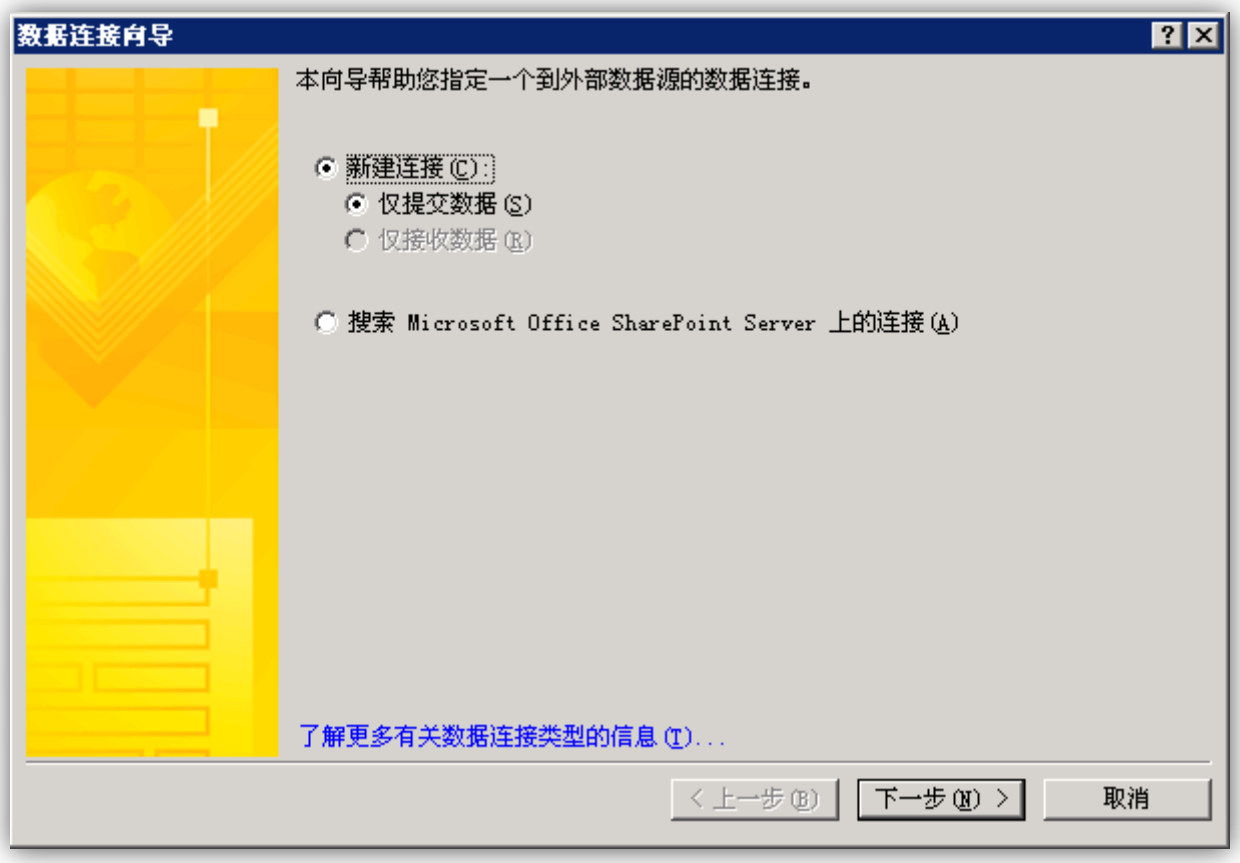


图 25

选择 **提交到宿主环境...**，点击 **下一步** > **完成** > **确定** .

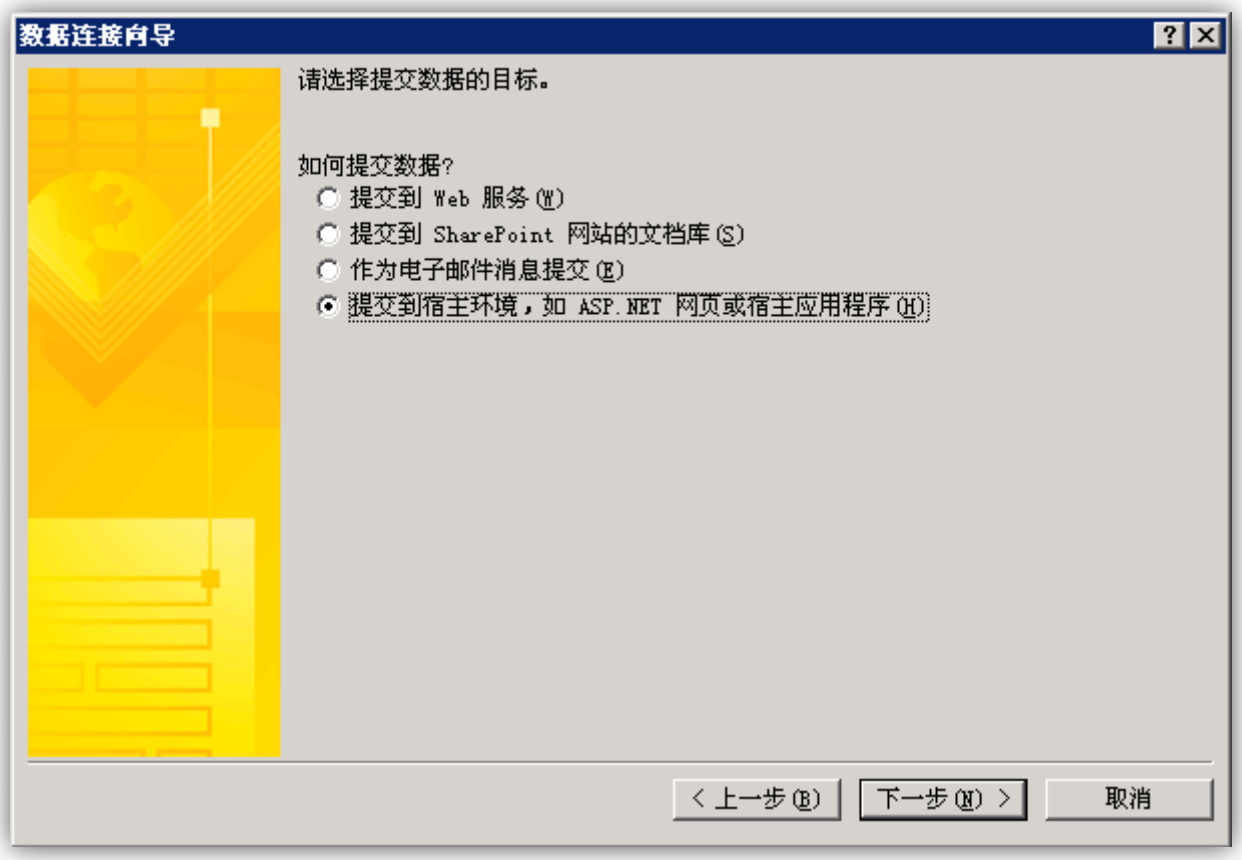


图 26

点击 **添加操作** > 选择 **关闭表单** > 点击 **确定** > **确定** > **确定** > **确定** .

4.2.4.2 辅助数据源 – CONTEXT

为表单添加了 Contact Selector 控件之后,还需要添加一个辅助数据源-**Context**,否则表单在打开时会发生错误.

新建一个 xml 文档,其内容为:

```
<Context
isStartWorkflow="true"
isRunAtServer="false"
provideAllFields="true"
siteUrl=""
/>
```

如果服务器上不只一个 SharePoint 站点集,将 **siteUrl** 设置为想要部署工作流的站点集 URL(可想而知，如果没有此数据源,Contact Selector 控件就不知道从何处验证用户输入的联系人是否存在了).

点击 **数据源面板** 中的 **管理数据连接** ，点击 **添加** ：

图 27

选择 **新建链接** > **仅接收数据** > 点击 **下一步** ：

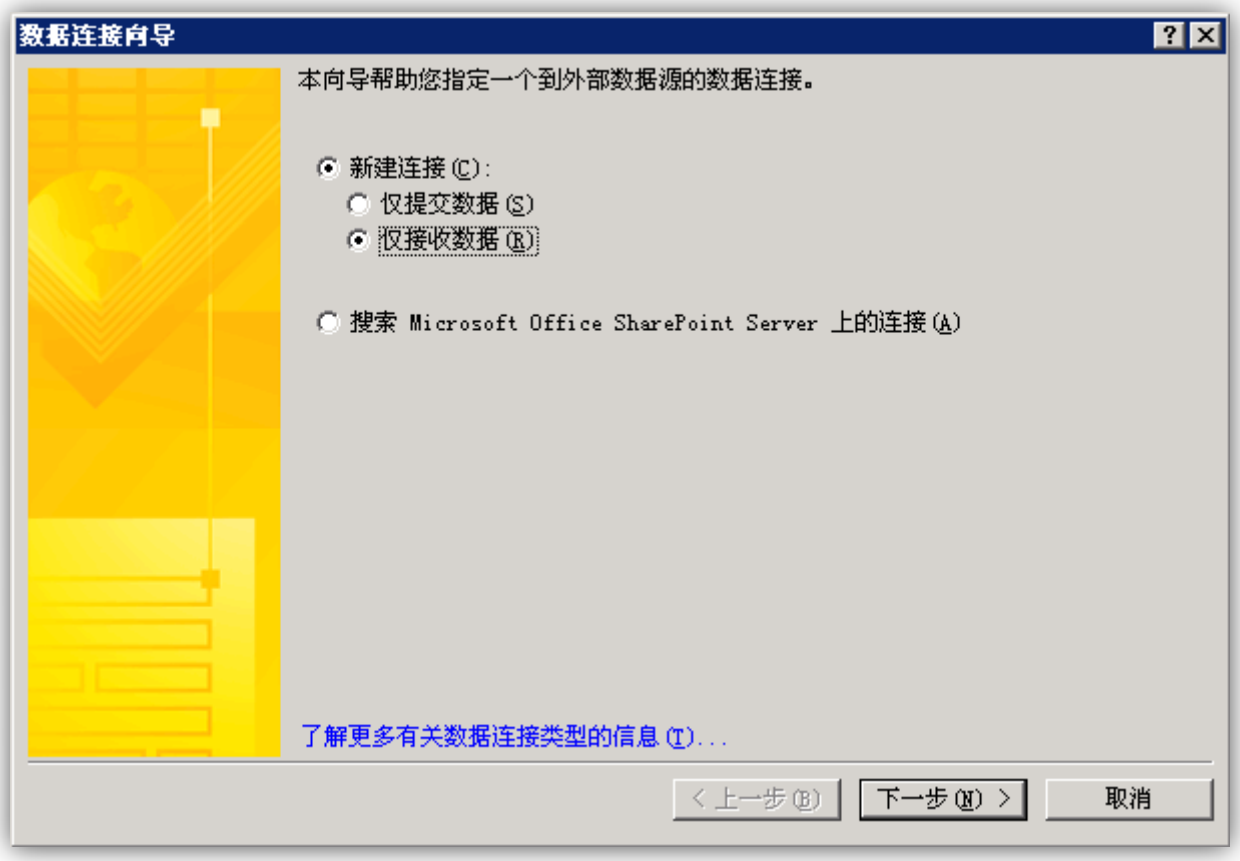


图 28

选择 **XML 文档** > 点击 **下一步** ：

图 29

点击 **浏览** 选择刚才创建的 XML 文档 > 点击 **下一步** ：

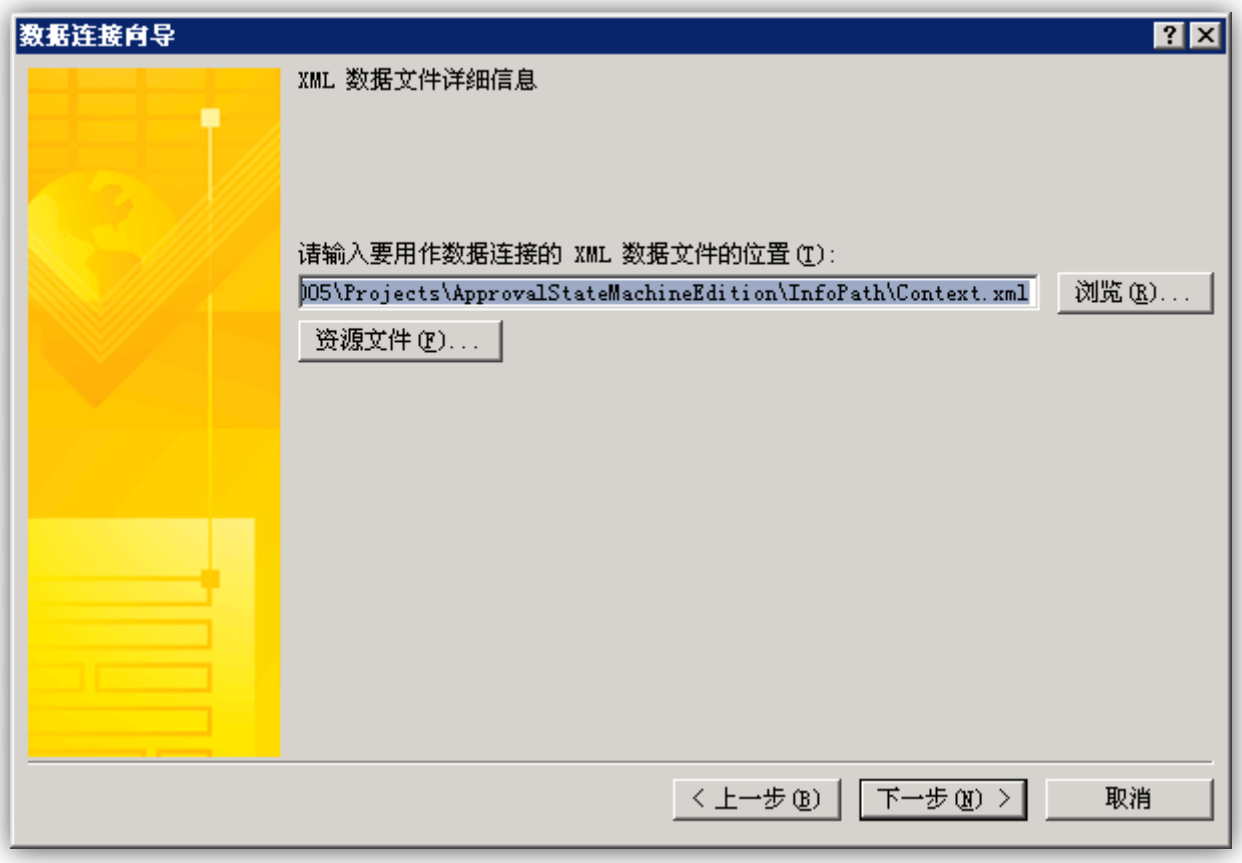


图 30

选择 **将数据包含为表单模板或模板部件中的资源** > 点击 **下一步** > **完成** > **关闭** ：

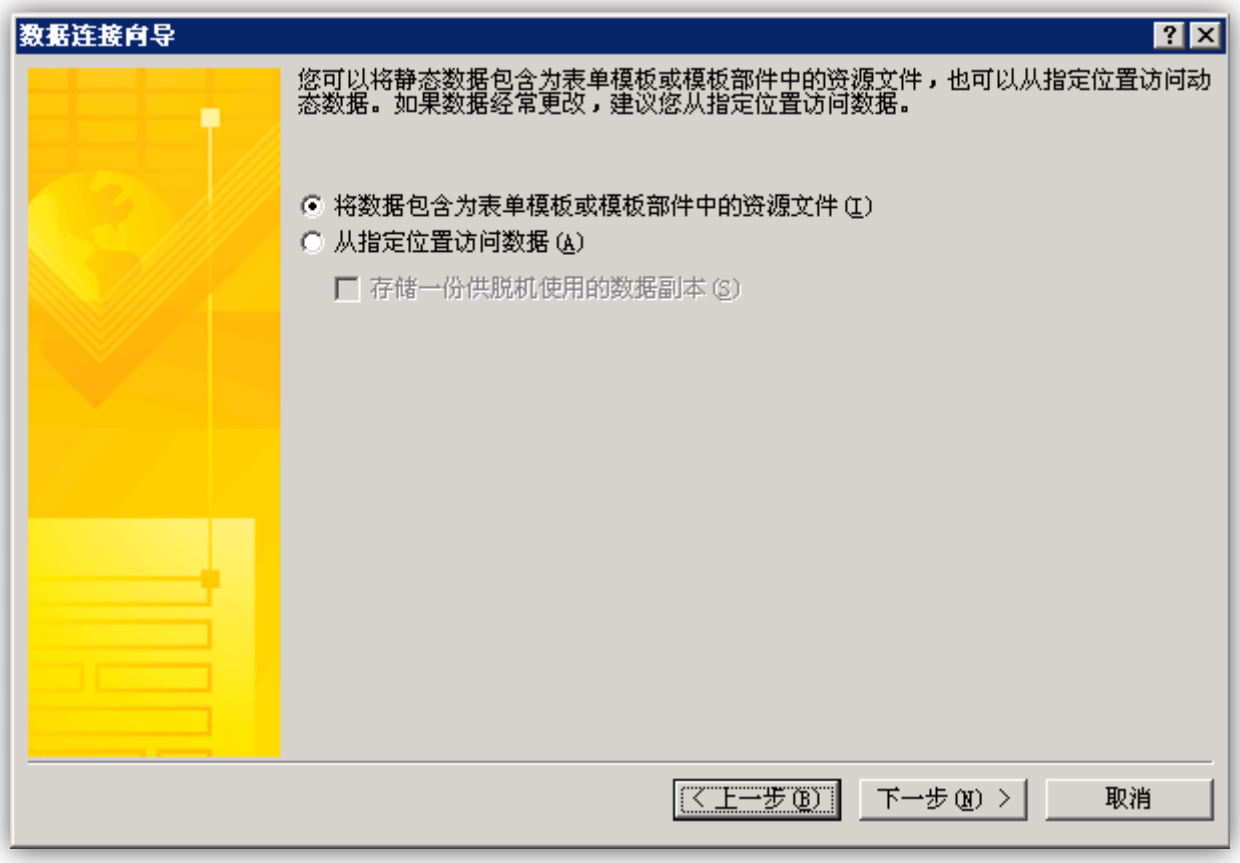


图 31

4.2.5 保存,发布,生成架构和类

选择 **工具** 菜单下的 **表单选项** ，打开 **安全和信任** 页,取消勾选 **自动确定安全级别** ,勾选 **域** 或者 **完全信任** ,如图 32 所示:

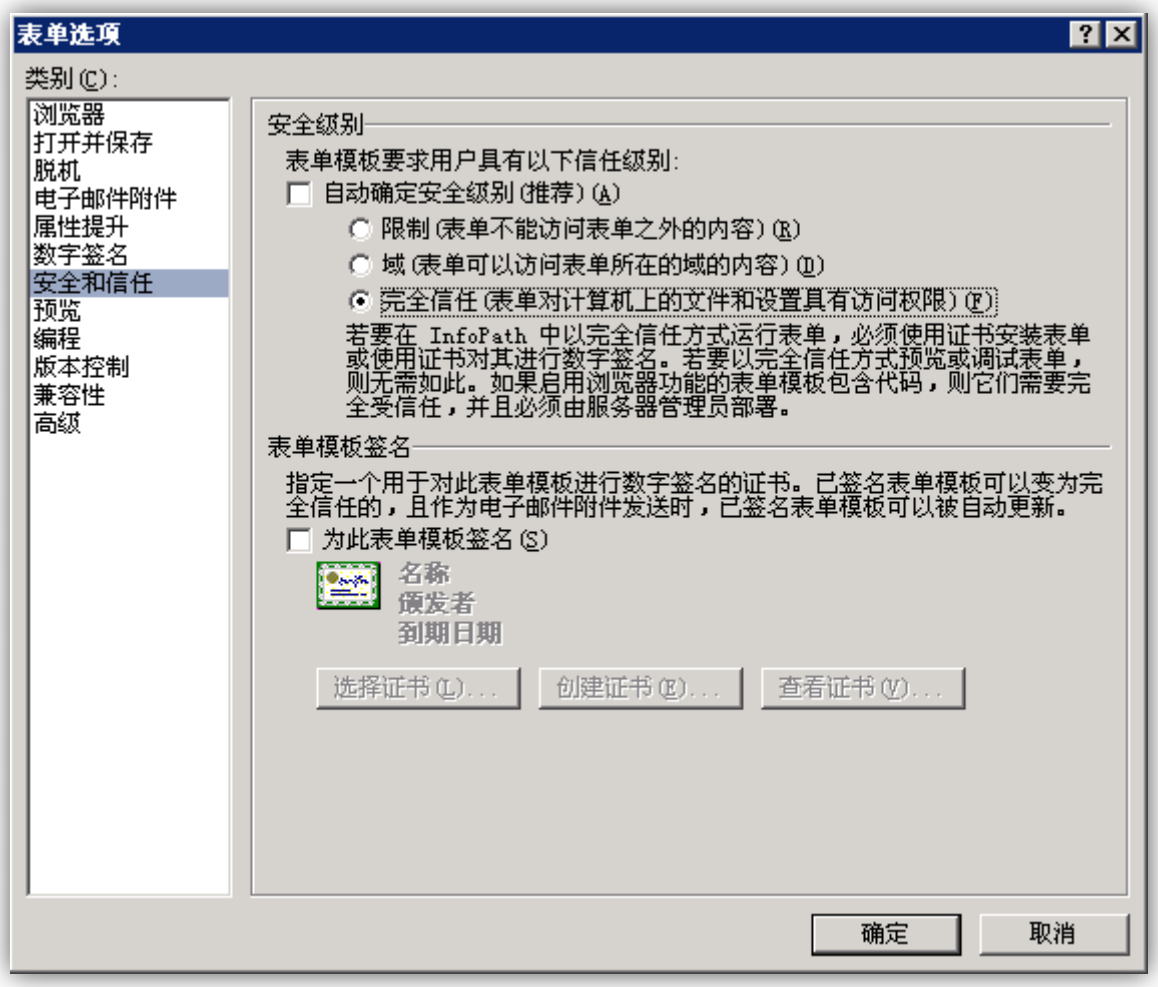


图 32

保存表单模板为 **Init.xsn** .

选择 **文件** 菜单下的 **发布** > 选择 **网络位置** > 点击 **下一步** ：

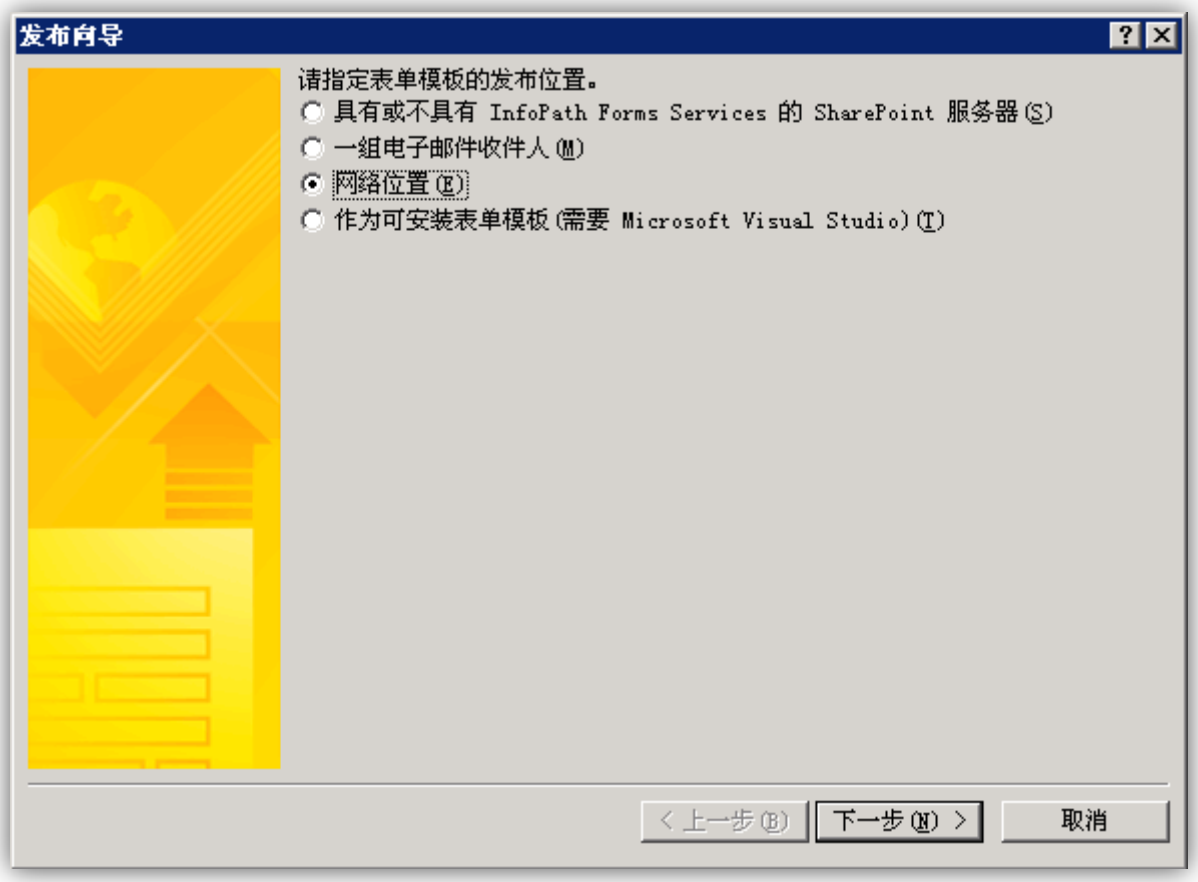


图 33

点击 **浏览** 将表单模板发布到工作流项目文件夹中,点击 **下一步** ：

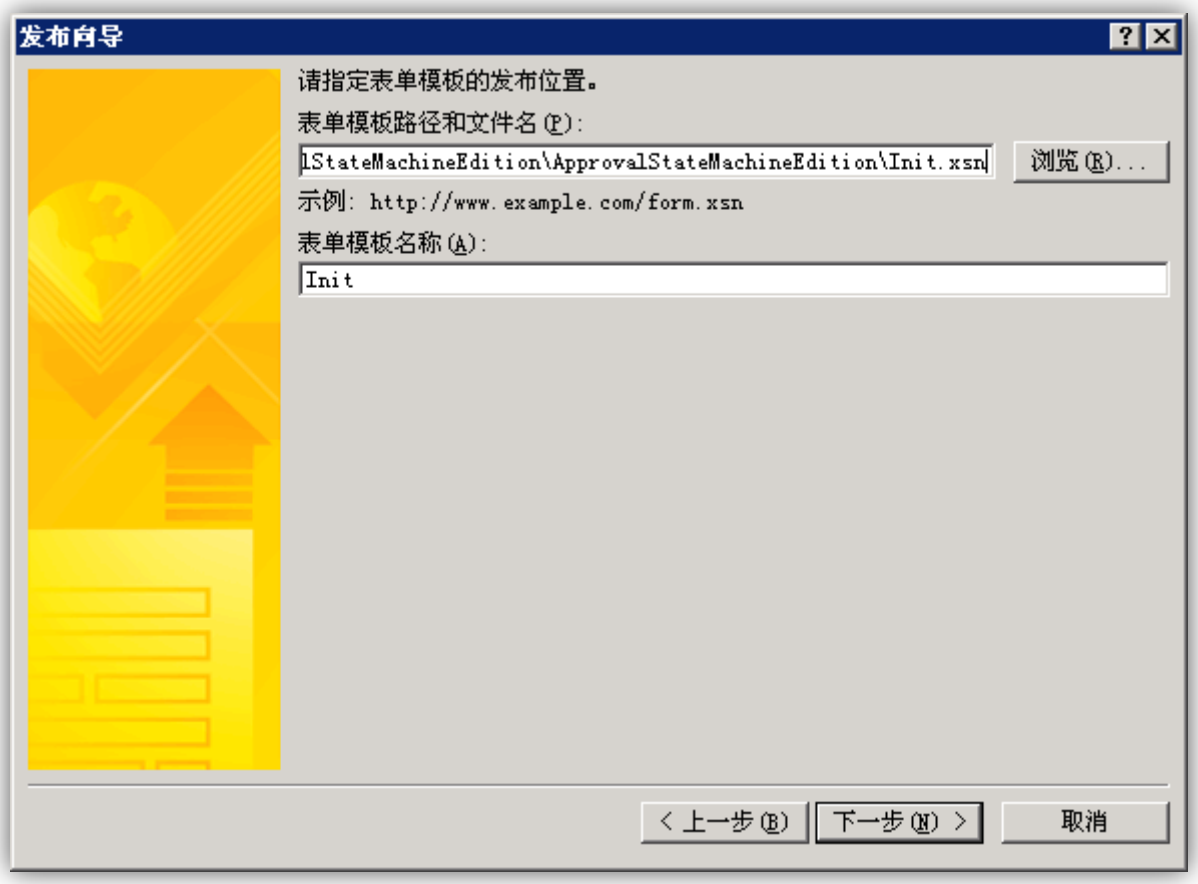


图 34

将可访问路径**清空**,点击 **下一步** > **发布** > **关闭** ：

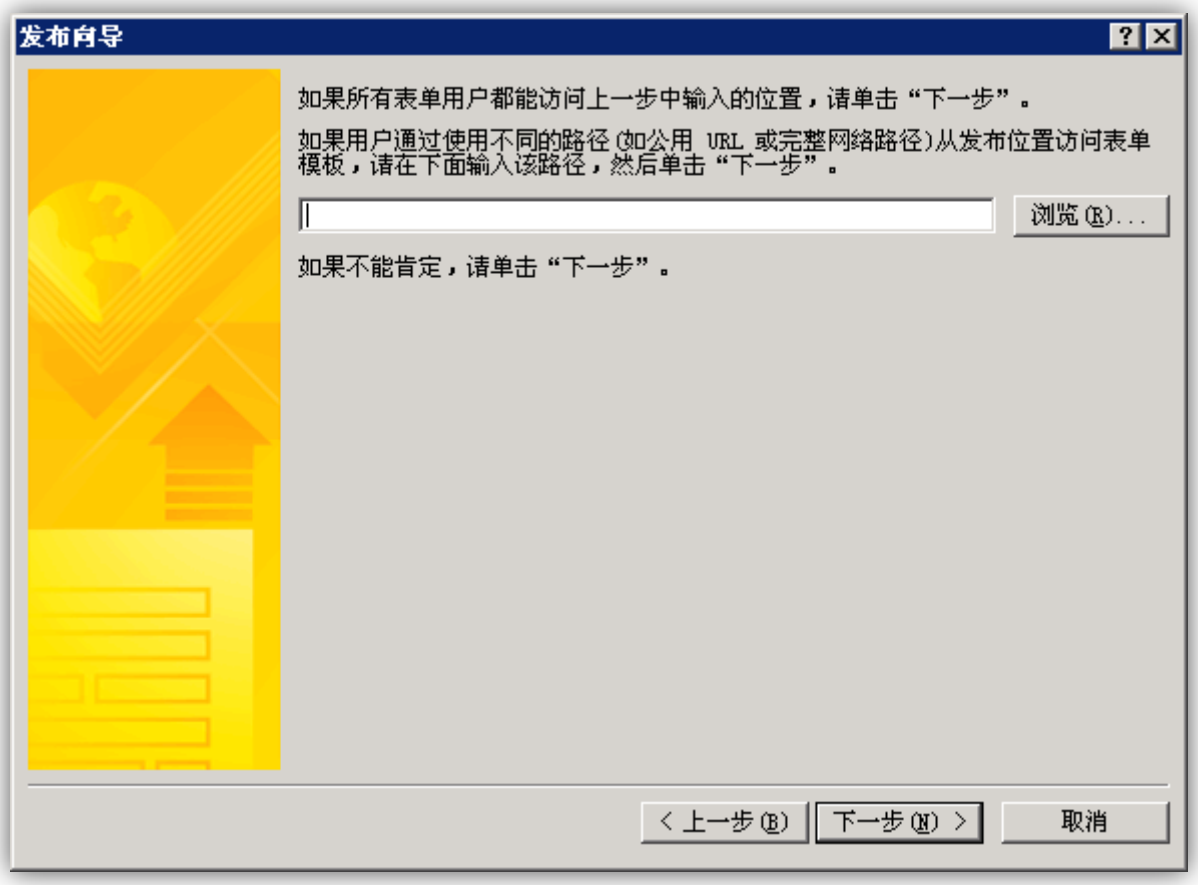


图 35

选择 **文件** 菜单下的 **另存为源文件** ,InfoPath 2007 将在指定的文件夹内生成如图 36 所示的文件:

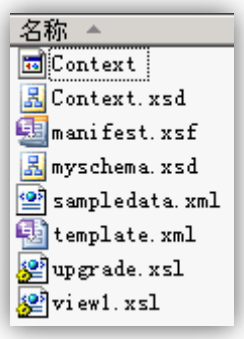


图 36

打开 **Visual Studio 2005 命令提示** 工具 > 运行下面的命令:

```
Xsd “myschema.xsd 的完整路径” /c /o:”工作流项目文件夹的完整路径”
```

注意:工作流项目文件夹的完整路径如果以”\”结尾会报错.

此操作会在工作流项目文件夹中生成一个名为 **myschema.cs** 的文件,该文件包含代表 Init 表单模板的类 **Init**.

将其改名为 **init.cs** 并包含到我们的工作流项目中.这样我们就可以在编码时通过反序列化 onWorkflowActivated.workflowProperties.InitiationData 来访问初始表单中的信息了.

4.3 设计 Task1 表单

4.3.1 设计 Task1 表单

参照图 37 设计来设计第一级审批者使用的表单 – Task1:

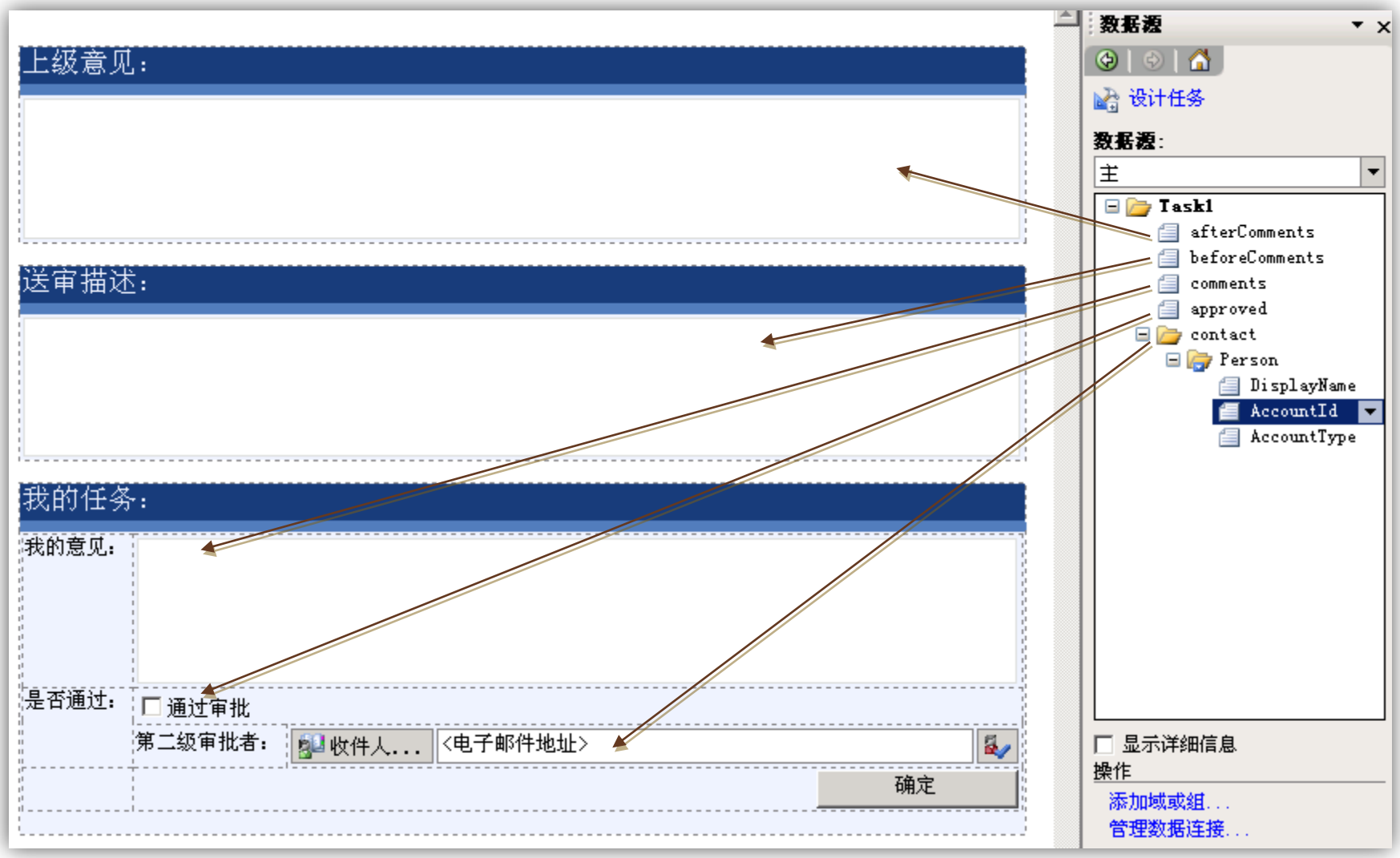


图 37

因为上级意见和送审描述仅供浏览,所以相应的将 afterComments 和 beforeComments 设置为 **只读**.

4.3.2 辅助数据源 – ITEMMETADATA

按照图 37 设计 Task1 表单并配置好数据源,我们还需要添加两个辅助数据源,一个是 Contact Selector 用来接收数据的 **Context.xml**,另外一个是表单用来接收数据的 **ItemMetadata.xml.**

如果没有 ItemMetadata 数据源,表单在显示的时候会发生错误,因为 SharePoint Server 在加载任务表单的时候会将任务数据发送给表单,而表单则需要通过此数据源来接收数据并把数据绑定到相应的域上.

新建一个名为 **ItemMetadata.xml** 的 XML 文档,其内容如下:

```
<z:row xmlns:z="#RowsetSchema"  ows_afterComments="" ows_beforeComments="" ows_comments="" />
```

此文件中参数的格式是"**ows_域名称"**,在这里我们只需要添加表单模板中的文本框控件代表的域即可.

参照添加 Context 数据源的步骤将此文档作为辅助数据源添加到表单模板中.

然后我们来将表单中的域和此辅助数据源关联起来.

双击 **afterComments 文本框**或者**域**名称,在弹出的属性对话框中点击 

右侧的  按钮.

点击 **插入域或组** :

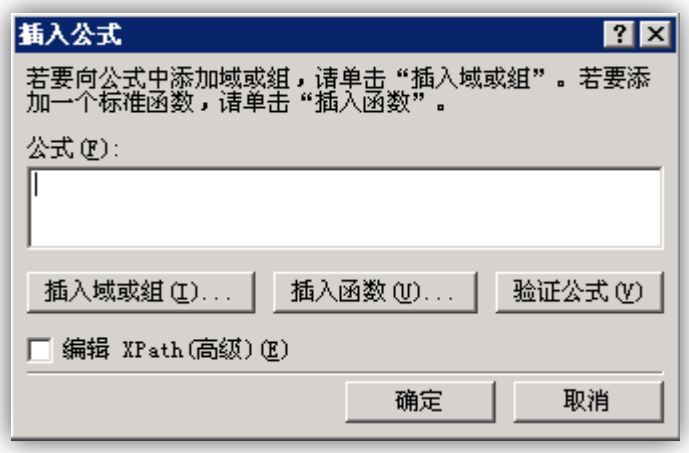


图 38

数据源选择 **ItemMetadata (辅助)** > 选择 **:ows_afterComments** > **确定** > **确定** > **确定**:

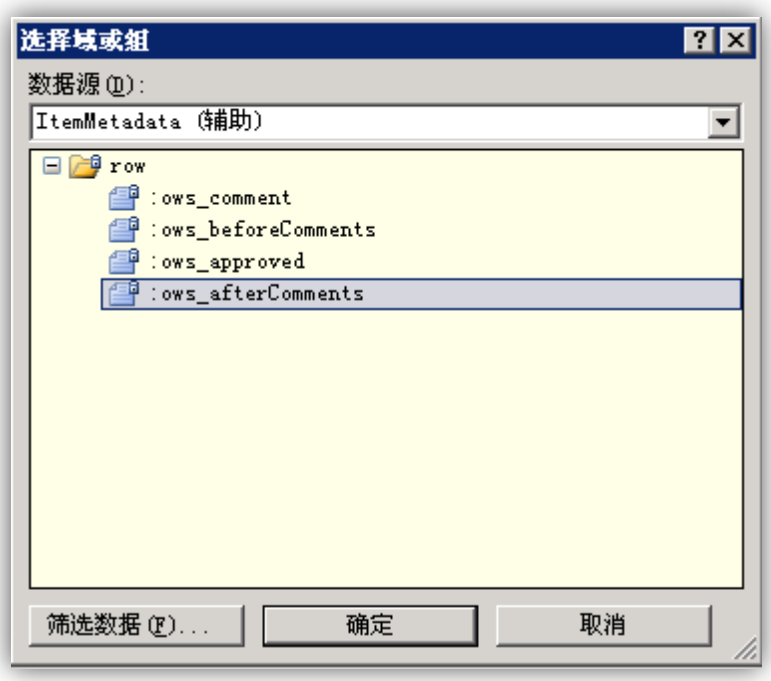


图 39

以同样的方法为表单模板中所有的文本框控件设置相应的默认值.

4.3.3 保存并发布

保存表单模板为 **Task1.xsn**,然后将表单**发布**到工作流项目文件夹内,名称为 **Task1.xsn**.

4.4 设计其它表单

4.4.1 设计 Task0 表单

请参照图 40 来设计 Task0 表单:

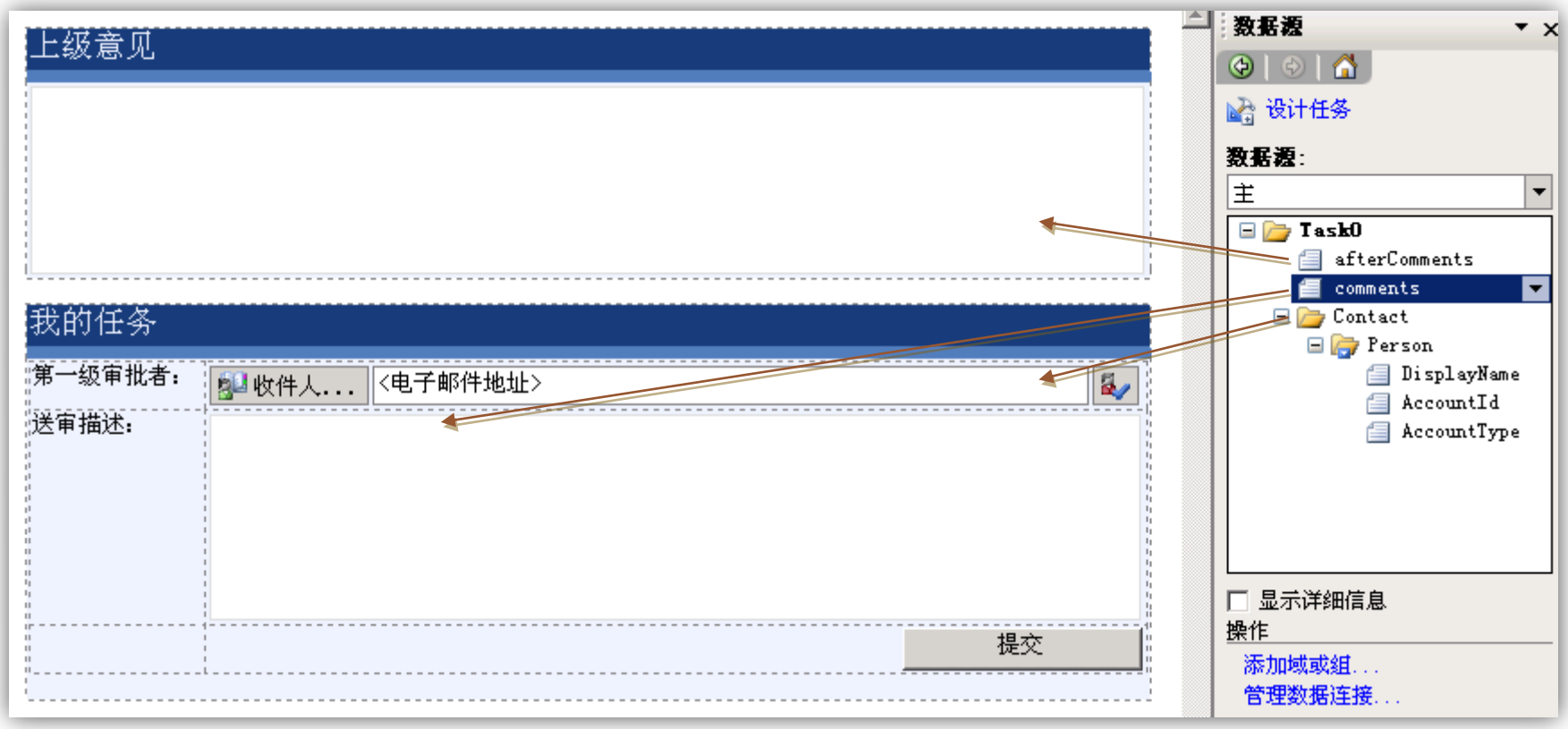


图 40

4.4.2 设计 Task2 表单

请参照图 41 来设计 Task2 表单:

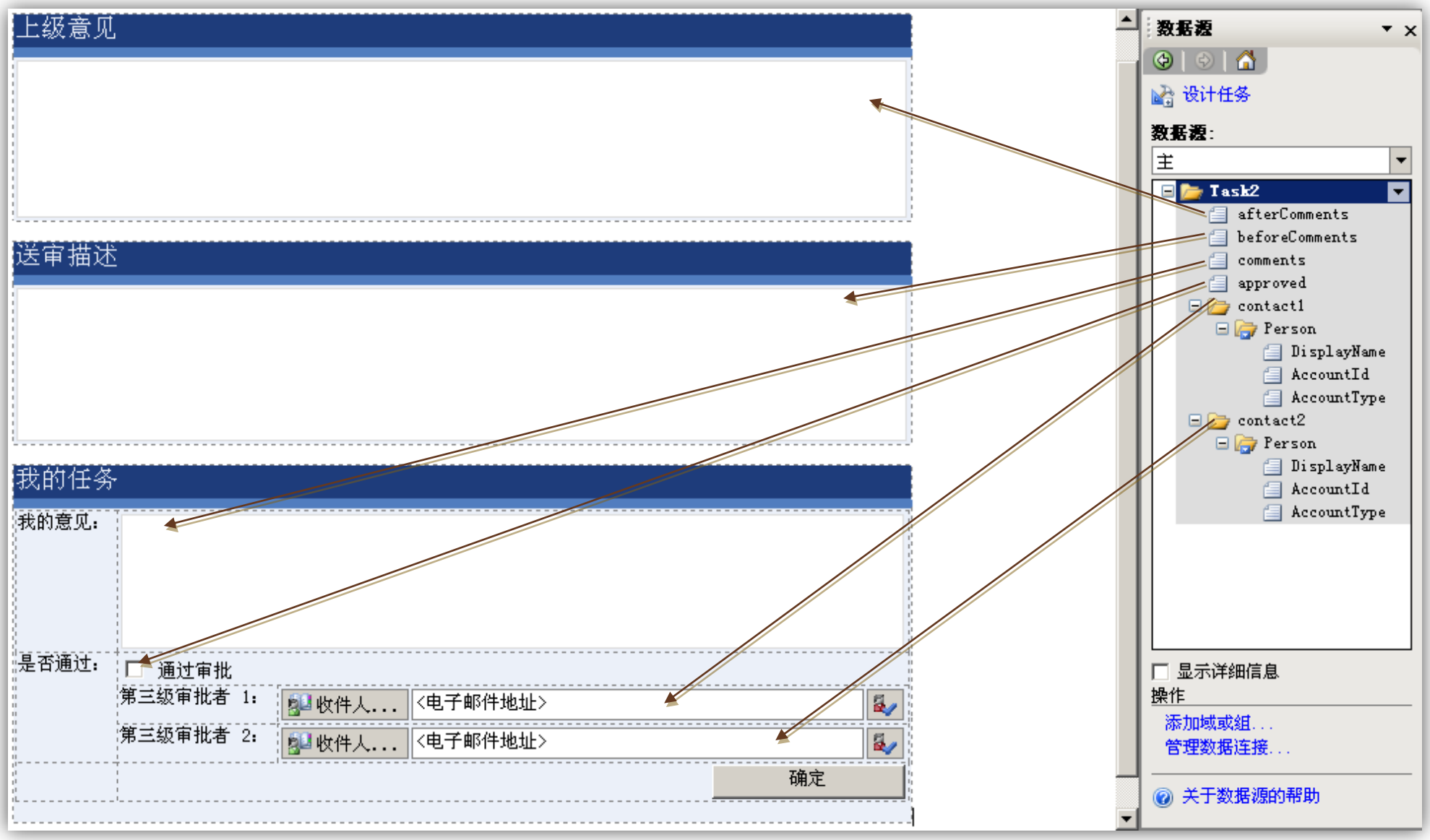


图 41

注意因为第二级审批人要将审批提交给两位第三级审批者,所以我们需要添加两个 Contact Selector 控件 **contact1** 和 **contact2**.

在为 contact1 添加好**重复组 Person** 之后,在 **Person** 上点击**右键** > **引用** > 选择 **contact2** > **确定** :

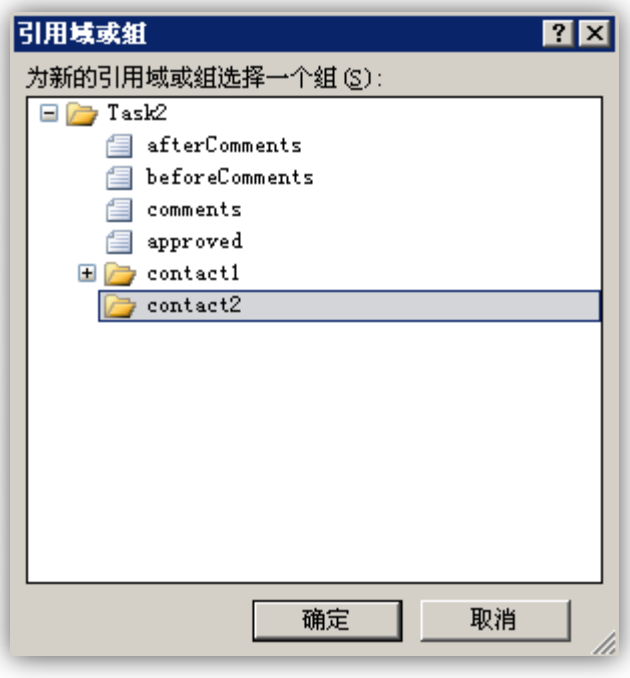


图 42

4.4.3 设计 Task3 表单

请参照图 43 来设计 Task3 表单

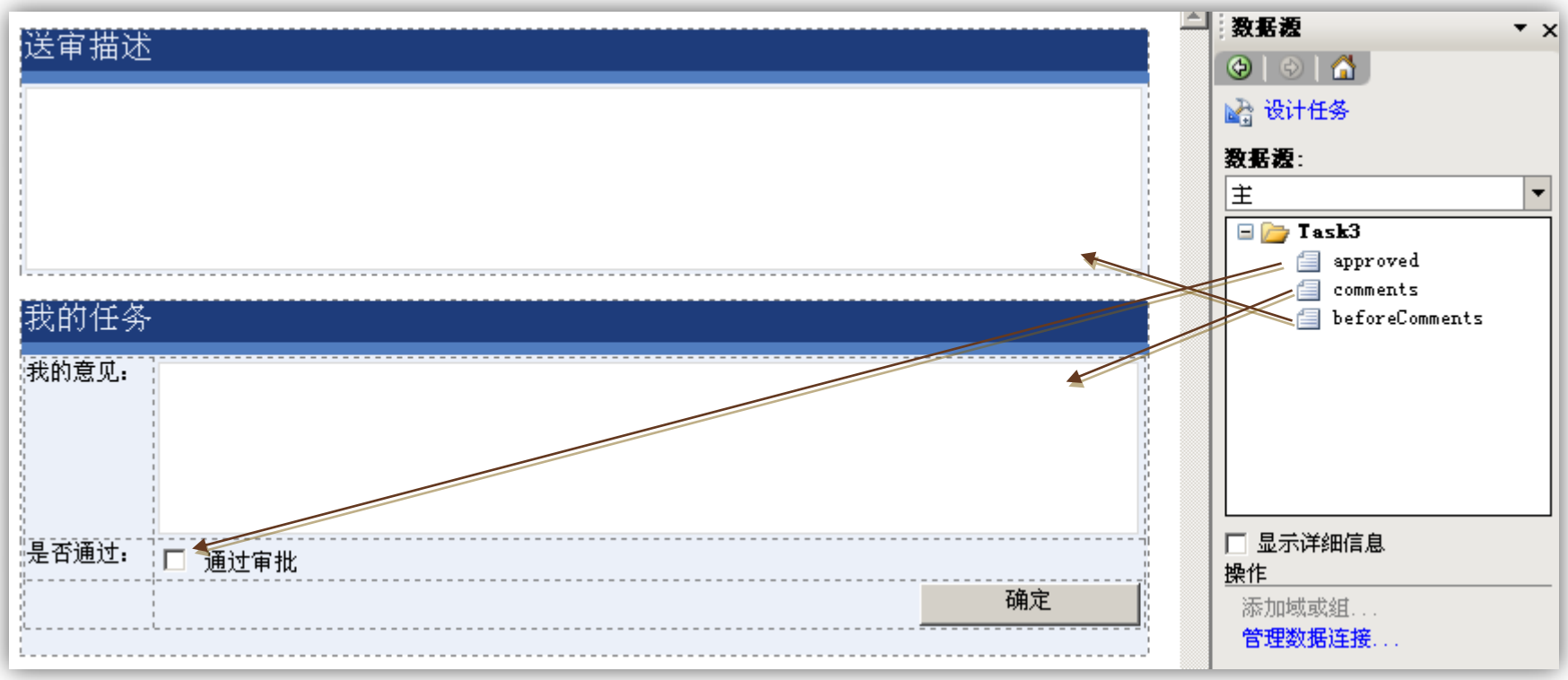


图 43

五.VISUAL STUDIO 项目(2) -- 编码

5.1 onWorkflowActivated

我们已经设计好了工作流流程图和所需的表单模板,下面我们来进入工作流的开发阶段.

在 Visual Studio 2005 中打开工作流项目 > 打开 Workflow1.cs 的 Design 视图 > 右键 > **查看代码** > 插入下列全局变量:

```
//因送审而创建的任务标题
private string approveItemTitle;

//因审批被拒绝而创建的任务标题
private string rejectItemTitle;
```

```
//标识是否创建 Task1，为真时创建 Task1，否则创建 Task0
private bool runTask1 = true;

//标识 Task1 是否通过审批
private bool task1Approved = false;

//标识 Task2 是否通过审批
private bool task2Approved = false;

//标识 Task3 是否通过审批
private bool task3Approved = false;

//标识 Task4 是否通过审批
private bool task4Approved = false;
```

回到工作流,右键单击 onWorkflowActivated1 > **Generate Handlers**.

在 **onWorkflowActivated1_Invoked** 事件中插入一下代码:

```
workflowId = workflowProperties.WorkflowId;

//定义任务的名称
approveItemTitle = "请审批 " + workflowProperties.Item.DisplayName;
rejectItemTitle = workflowProperties.Item.DisplayName + " 被退回，请审批。";

//反序列化 workflowProperties.InitiationData 以得到初始窗体的实例
XmlSerializer xs = new XmlSerializer(typeof(Init));
XmlTextReader xtr = new XmlTextReader(new System.IO.StringReader(workflowProperties.InitiationData));
Init init = (Init)xs.Deserialize(xtr);

//将初始窗体的信息赋给 Task0
task0_Properties.Title = rejectItemTitle;
task0_Properties.AssignedTo = workflowProperties.Originator;
task0_Properties.ExtendedProperties["comments"] = init.comments;

//设置 Task1 的标题和分配对象
task1_Properties.Title = approveItemTitle;
task1_Properties.AssignedTo = init.contact[0].AccountId;
```

5.2 WHILE1

设置 while1 的 **Condition** 属性为 **Code Condition** ,然后在其子属性 **Condition** 中输入 **task1_Approved**：

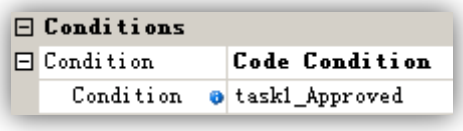


图 44

回车,在 task1_Approved 事件中插入以下代码:

```
e.Result = !task1Approved;
```

当 e.Result 为真的时候(也就是 Task1 没有被批准的时候)WhileActivity 将执行循环.

5.3 IFELSE

设置 ifElse 的 **Condition** 属性为 **Code Condition** ,然后在其子属性 **Condition** 中输入 **task1_Run**,回车,在 task1_Run 事件中插入以下代码:

```
e.Result = runTask1;
```

当 e.Result 为真时为第一级审批者创建任务,否则为送审者创建任务.

5.4 TASK1

5.4.1 CREATETASK1

首先添加一个方法 PersonSayComment,这个方法用来返回一个格式为"某人:意见"的字符串.

```
private string PersonSayComment(Microsoft.SharePoint.Workflow.SPWorkflowTaskProperties task)
{
    if (task.ExtendedProperties.Contains("comments"))
    {

        Microsoft.Office.Workflow.Utility.Contact c = Microsoft.Office.Workflow.Utility.Contact.FromName(task.AssignedTo, workflowProperties.Web);
        return c.DisplayName +
            " : " +
            task.ExtendedProperties["comments"] +
            "\n";
    }
    else
        return "";
}
```

通过上面的代码可以发现,任务表单中的数据可以使用 taskProperties.ExtendedProperties["域名称"]来访问,而不需要像初始表单一样需要反序列化.

双击 createTask1,在 **createTask1_MethodInvoking** 事件中插入以下代码:

```
//为 Task1 指派一个新的 GUID
task1_Id = Guid.NewGuid();

//指定 Task1 使用的表单编号
task1_Properties.TaskType = 1;

//设置 Task1 表单中的送审描述信息
task1_Properties.ExtendedProperties["beforeComments"] = PersonSayComment(task0_Properties);

//设置 Task1 表单中的上级意见信息
task1_Properties.ExtendedProperties["afterComments"] = PersonSayComment(task2_Properties) +
PersonSayComment(task3_Properties) +
PersonSayComment(task4_Properties);
```

5.4.2 onTaskChanged1

当 Task1 发生改变时,我们需要获取审批结果并根据结果决定是否指定下一级审批任务的信息.

这里我们添加一个方法来更方便的获得表单中 Contact　Selector 中的联系人信息:

```
private string GetNextAssigneeDisplayname(string assigneeXml)
{
    Microsoft.Office.Workflow.Utility.Contact[] contacts = Microsoft.Office.Workflow.Utility.Contact.ToContacts(assigneeXml,
    workflowProperties.Web);
    if (contacts[0] != null)
        return contacts[0].DisplayName;
    else
        return "";
}
```

双击 onTaskChanged1,在 **onTaskChanged1_Invoked** 事件中插入以下代码:

```
//删除 Task1 时将此标识设置为 false，则下次进入 IfElse 会为 Task0 创建任务
runTask1 = false;

//获取 Task1 的审批结果
task1Approved = bool.Parse(task1_Properties.ExtendedProperties["approved"].ToString());

//设置 Task2 的标题和分配对象
if (task1Approved)
{
    task2_Properties.Title = approveItemTitle;
    task2_Properties.AssignedTo = GetNextAssigneeDisplayname(task1_Properties.ExtendedProperties["contact"].ToString());
}
```

5.5 Task0

5.5.1 createTask0

双击 createTask0,在 **createTask0_MethodInvoking** 事件中插入以下代码:

```
task0_Id = Guid.NewGuid();
task0_Properties.TaskType = 0;
task0_Properties.ExtendedProperties["afterComments"] = PersonSayComment(task1_Properties) +
    PersonSayComment(task2_Properties) +
    PersonSayComment(task3_Properties) +
    PersonSayComment(task4_Properties);
```

5.5.2 onTaskChanged0

双击 onTaskChanged0,在 **onTaskChanged0_Invoked** 事件中插入以下代码:

```

```

```
//删除 Task0 时将此标识设置为 true，则下次进入 IfElse 会为 Task1 创建任务
runTask1 = true;
task1_Properties.Title = approveItemTitle;
task1_Properties.AssignedTo = GetNextAssigneeDisplayname(task0_Properties.ExtendedProperties["contact"].ToString());
```

5.6 WHILE2

设置 while2 的**Condition** 属性为 **Code Condition** ,然后在其子属性 **Condition** 中输入 **task2_Approved** ,回车后在 task2_Approved 事件中插入以下代码:

```
e.Result = !task2Approved;
```

5.7 TASK2

5.7.1 CREATETASK2

双击 createTask2,在 **createTask2_MethodInvoking** 事件中插入以下代码:

```
task2_Id = Guid.NewGuid();
task2_Properties.TaskType = 2;
task2_Properties.ExtendedProperties["beforeComments"] = PersonSayComment(task0_Properties) +
    PersonSayComment(task1_Properties);
task2_Properties.ExtendedProperties["afterComments"] = PersonSayComment(task3_Properties) +
    PersonSayComment(task4_Properties);
```

5.7.2 ONTASKCHANGED2

双击 onTaskChanged2,在 **onTaskChanged2_Invoked** 事件中插入以下代码:

```
task2Approved = bool.Parse(task2_Properties.ExtendedProperties["approved"].ToString());
if (task2Approved)
{
    task3_Properties.Title = approveItemTitle;
    task3_Properties.AssignedTo = GetNextAssigneeDisplayname(task2_Properties.ExtendedProperties["contact1"].ToString());
    task4_Properties.Title = approveItemTitle;
    task4_Properties.AssignedTo = GetNextAssigneeDisplayname(task2_Properties.ExtendedProperties["contact2"].ToString());
}
else
{
    runTask1 = true;
    task1Approved = false;
    task1_Properties.Title = rejectItemTitle;
}
```

5.8 WHILE3

设置 while3 的 **Condition** 属性为 **Code Condition** ,然后在其子属性 **Condition** 中输入 **final_Approved** ,回车后在 final_Approved 事件中插入以下代码:

```
e.Result = ! (task3Approved && task4Approved);
```

5.9 TASK3

5.9.1 CREATETASK3

双击 createTask3,在 **createTask3_MethodInvoking** 事件中插入以下代码:

```
task3_Id = Guid.NewGuid();
task3_Properties.TaskType = 3;
task3_Properties.ExtendedProperties["beforeComments"] = PersonSayComment(task0_Properties) +
    PersonSayComment(task1_Properties) +
    PersonSayComment(task2_Properties);
```

5.9.1 ONTASKCHANGED3

双击 onTaskChanged3,在 **onTaskChanged3_Invoked** 事件中插入以下代码:

```
task3Approved = bool.Parse(task3_Properties.ExtendedProperties["approved"].ToString());
```

5.10 TASK4

5.10.1 CREATETASK4

双击 createTask4,在 **createTask4_MethodInvoking** 事件中插入以下代码:

```
task4_Id = Guid.NewGuid();
task4_Properties.TaskType = 3;
task4_Properties.ExtendedProperties["beforeComments"] = PersonSayComment(task0_Properties) +
    PersonSayComment(task1_Properties) +
    PersonSayComment(task2_Properties);
```

5.10.2 onTaskChanged4

双击 onTaskChanged4,在 **onTaskChanged4_Invoked** 事件中插入以下代码:

```
task4Approved = bool.Parse(task4_Properties.ExtendedProperties["approved"].ToString());
```

5.10 code

设置 code 的 ExecuteCode 属性为 **customCode**,回车后在 **customCode** 事件中插入以下代码:

```
if (!(task3Approved && task4Approved))
{
    runTask1 = true;
    task1Approved = false;
    task2Approved = false;
    task1_Properties.Title = rejectItemTitle;
}
```

六.部署

在图 4 中我们看到 3 个关于部署的文件 **Feature.xml**，**Workflow.xml** 和 **Install.bat**.

6.1 强签名

因为工作流的程序集需要部署到 GAC 里,所以我们需要为程序集生成一个强签名.

打开项目属性面板 > 切换到 **签名** 页 > 选择 **为程序集签名** > 选择 **新建** ：

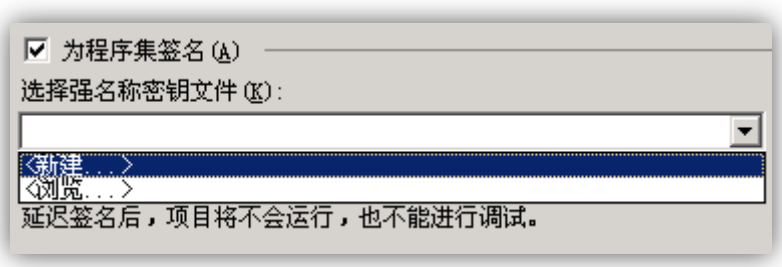


图 45

输入 **密钥文件名称** > **确定** > **保存**更改:

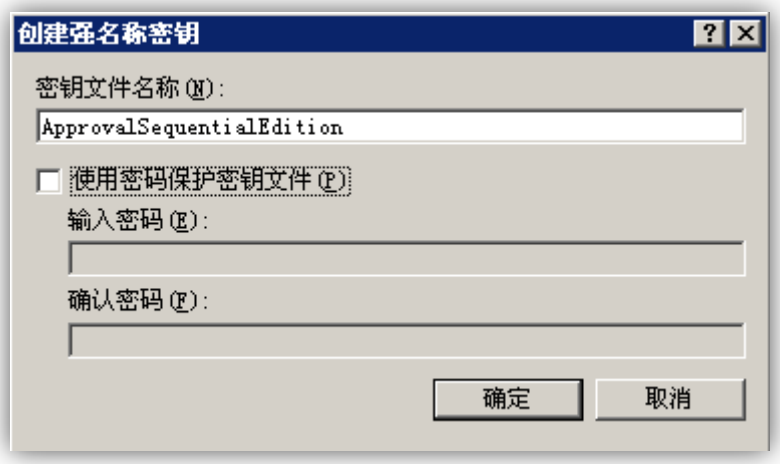


图 46

启用强签名之后重新生成程序集.

6.2 FEATURE.XML

在 SharePoint Server 2007 中,工作流是作为一个 Feature 来安装的,所以我们需要配置 Feature.xml 文件,这个文件用来描述工作流 feature 的基本信息 (比如 feature 名称和它所包含的工作流定义文件路径等).

打开 **feature.xml** > **点击右键** > **插入代码段** > **SharePoint Server Workflow** > **Feature.xml Code**.

注:如果你的右键菜单中并没有上述的代码段,请参见我的另外一篇文章:<< Visual Studio 2005.net 代码段(Snippet)丢失的解决方案>>.

本例的 Feature.xml 中我们只需要输入以下信息:

- **Id**： 全局唯一标示符 GUID,生成方法见下文.
- **Title**： 工作流 Feature 的标题.
- **Description**： 工作流 Feature 的描述信息.

其中 Id 是一个 GUID,点击 **工具** 菜单下的 **创建 GUID**：

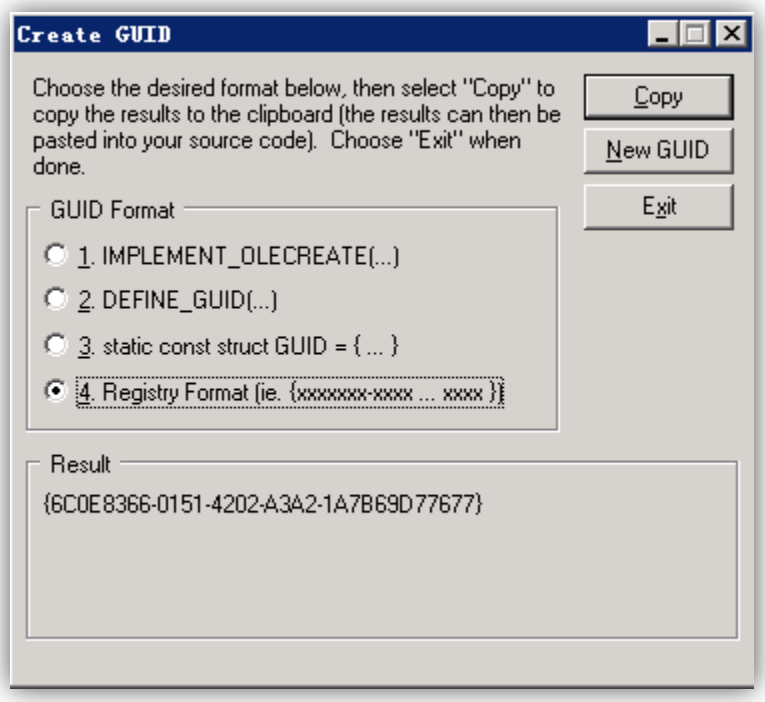


图 47

选择 **4.Registy Format**，点击 **Copy**，**粘贴**到 Id 处.

6.3 WORKFLOW.XML

Workflow.xml 是工作流定义文件,它描述了工作流的基本信息(如名称,描述等),程序集和所包含表单的 URN.

打开 **workflow.xml** > **点击右键** > **插入代码段** > **SharePoint Server Workflow** > **Workflow.xml Code**.

本例中,我们需要输入以下信息:

- **Name**： 工作流的名称.
- **Description**： 工作流的描述信息.
- **Id**： 工作流的 GUID,不可以和 feature 的 Id 相同.

- **CodeBdsideClass**： 工作流的类名,格式是如下:

```
项目名称.工作流类名
```

- **CodeBesideAssembly**： 工作流程序集信息.格式如下:

```
项目名称,Version =版本号, Culture=neutral, PublicKeyToken=公钥标记
```

版本号可以在工作流程序集文件(.DLL)的属性信息中获取,获取程序集 PublicKeyToken(公钥标记)则需要在 Visual Studio 命令提示工具中执行以下命令:

```
sn -T “工作流程序集文件(.DLL)完整路径”
```

- **Instantiation_FormURN**： 初始表单(Init.xsn)的 URN,获取表单 URN 的方法如下:

以设计模式打开表单模板 > 选择 文件 菜单中的 属性 :

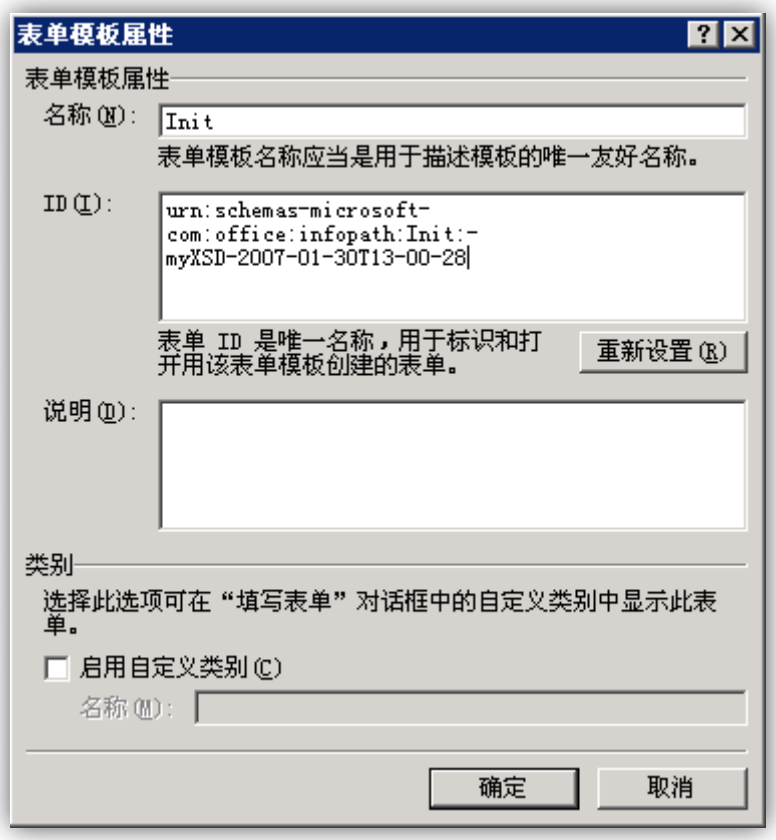


图 48

图 48 中 ID 里的内容就是我们需要的 URN.

- **Task0_FormURN**：Task0.xsn 的 URN.
  - 按照 Task0_FormURN 的格式依序添加 **Task1_FormURN, Task2_FormURN** 和 **Task3_FormURN**,并指定其值.

因为本例中没有为工作流指定**关联表单**(**Association Form**)和**修改表单**(**Modification Form**),所以我们还需要在 Workflow.xml 文件中删除以下信息:

- AssociationUrl="_layouts/CstWrkflIP.aspx"
- ModificationUrl="_layouts/ModWrkflIP.aspx"
- <Association_FormURN>associationURN<Association_FormURN>
- <Modification_GUID_FormURN>modificationURN</Modification_GUID_FormURN>

6.4 INSTALL.BAT

这是一个批处理文件,它的功能是:

- 拷贝并注册工作流表单.
- 将程序集添加到 GAC.

- 安装并激活工作流 feature

我们需要对这个文件做一些修改来适应我们的项目,打开这个文件:

- 将所有的 **MyFeature** 替换为 **ApprovalSequentialEdition**.
- 将所有的 **http://localhost** 替换为**站点 URL**.
- 可能还需要根据服务器的配置做一些相应的修改(比如服务器的 CPU 是 32 位还是 64 位等).

到此我们已经完成了工作流开发的所有步骤,双击运行 Install.bat 就可以将工作流部署到站点上了.

本文以及范例示范了一个多级审批 Sequential 工作流的开发过程,这个工作流很简单,但初次接触 SharePoint 和 Workflow Foundation 的朋友或许会和我一样摸不着头脑,希望本文能够给予大家一些帮助,也欢迎大家指正本文的谬误(如果存在的话).

Email: xiaoshatian@gmail.com
Blog: xiaoshatian.cnblogs.com