内容

阶段 1 - 简介 Stage 1 - Introduction 6

关于 Tersus 平台 About the Tersus Platform 6

关于本教程 About this Turtorial 6

文件规定 Document Conventions 7

使用教程 Using the tutorial 7

应用程序样本 The Sample Application 8

使用 Eclipse 中文版 Use Eclipse in Chinese version 8

阶段 2 - 建立基本显示模型 Stage 2 - Modeling a Basic Display 9

阶段目标 Stage Goals 9

覆盖的 Tersus 概念 Tersus Concepts Covered 9

建立应用程序功能模型 Application Functionality Modeled. 9

Tersus 工作室 The Tersus Studio 10

生成简单的网页应用程序 Create a Simple Web Application 11

开始建模-生成输入申请单的表单 Start Modeling -Create a Form for Entering a Requisition . 11

生成视图（打开申请单）Create a View (Open Requisition) 12

添加一个按钮（新申请单）Add a Button(New Requisition) 13

生成弹出窗口（输入新申请单）Create a Popup(Enter New Requisition) 14

添加显示单元到弹出窗口里面 Add Display Element to the Popup 16

重新命名模型 Rename a model 18

完成阶段 2 Completing Stage 2 20

输入项目样本 Importing a Sample Project 20

实例 See It Live 27

阶段 3-建立幕后逻辑模型 Modeling the logic behind the Screen 28

阶段目标 Stage Goals 28

覆盖的 Tersus 概念 Tersus Concepts Covered 28

建立应用程序功能模型 Application Functionality Modeled 28

应用程序逻辑建模 Model Application Logic 28

定义数据结构 Define a Data Structure 29

使用过程模板（Process Template）以生成记录标识码（Record identifier） Use a Process Template (to generate a record identifier) 32

生成流程 Create a Flow. 32

使用显示数据单元（Display Data Element）以提取用户输入结果 Use a Display Data Element (to retrieve uswer input) 33

使用插入模板（Insert Template）以将数据保存到数据库里面 Use the Insert Template (to Store data in the database) 36

使用关闭窗口模板（Close Window Template）和确保过程发生顺序正确 Use the Close Window Template (and make sure processes occur in the right order) 37

完成阶段 3 Completing Stage 3 40

实例 See it live 41

阶段 4-建立简单表格显示（Table Display）模型 Stage 4-Modeling a Simple Table Display 42

阶段目标 Stage Goals 42

覆盖的 Tersus 概念 Tersus Concepts Covered 42

建立应用程序功能模型 Application Functionality Modeled 42

建立数据表格显示（Tabular Display）模型 Model a Tabular Display 43

生成表格显示 Create a Table Display 43

重新使用数据结构定义表格内容 Reuse a Data Structure to Define Table Contents 44

使用动作模型定义过程（以数据库数据填写显示表格） Use Action Model to Define a Process(populating the display table with data from the database) 47

定义子过程（在内存中生成表格数据单元）Define a Sub-Process(populating the display table with data from the database) 48
使用“查找”模板 (以定义数据类型和生成表格数据单元) Use the Find template (to retrieve data from the database) 48
使用显示数据单元（以定义数据类型和生成表格数据单元）Use the Display Data Element(to define the data type & create the table data element) 49
从子过程输出数据 Output Data from the Sub-process 50
使用显示数据单元（把数据输出到显示）Use a Display Data Element (to output data to the display) 51
重新使用动作过程（以刷新表格显示）Reuse the Action Process (to refresh the table display)... 52
完成阶段 4 Completing Stage 4 55
实例 See it Live..... 56
阶段 5-建立选取器模型和使用常数 Stage 5 - Modeling Choosers and Using Constants 57
阶段目标 Stage Goals..... 57
覆盖的 Tersus 概念 Tersus Concepts Covered..... 57
建立应用程序功能模型 Application Functionality Modeled 57
建立选取器模型 Model a Chooser..... 57
使用行单元以改善（弹出窗口显示)格式 Use a Row Element for Better Formatting(of the popup display) 57
添加选取器显示(到弹出窗口里面) Add a Chooser Display (to the popup) 58
生成初始化过程(填入选取器数值) Create an Initialization Process (that populates the chooser with values) 60
使用常数(定义选取器显示的数值) Use Constants(to define the values displayed in the chooser). 60
添加字段到数据结构中 Add a Field to the Data Structure 62
重新使用意味着建模速度更快(显示和数据库结构自动更新) Reuse Means Faster Modeling(display &database structure are automatically updated) 63
完成阶段 5 Completing Stage 5..... 64
实例 See It live..... 64
阶段 6 - 建立附加视图模型和更新数据 Stage 6 - Modeling an Additional View and Updating Data..... 65
阶段目标 Stage Goals..... 65
覆盖的 Tersus 概念 Tersus Concepts Covered..... 65
建立应用程序功能模型 Application Functionality Modeled 65
用户建模 User Modeling..... 65
给模型添加一个视图 Add a View to the Model 65
重新使用显示单元 (申请单清单) Reuse a Display Element (Requisition List) 66
在无法完全重新使用时重新创建一个过程(以填充申请单清单) Recreate a Process when it cannot be Reused in full (to populate the requisition list) 67
添加一个更新现有记录并将记录标识为“已批准”的按钮 Add a Button(that updates an existing record, marking it Approved) 68
从表格显示中提取已选定行 Retrieve the Selected row from a Table Display 69
改变字段数值 Change a field’s Value 70
提交更新记录到数据库和刷新表格显示 Commit the Updated Record to the Database and Refresh the Table Display 72
完成阶段 6 Completing Stage 6 76
实例 See It Live..... 76
阶段 7--重新分解 - 改变过程以提高可重用性 Stage 7 - Re-factoring -Changing a process to Enhance Reusability..... 77
阶段目标 Stage Goals..... 77
覆盖的 Tersus 概念 Tersus Concepts Covered..... 77
建立应用程序功能模型 Application Functionality Modeled 77
重新分解现有模型 Re-factor an Existing Model..... 77
过程可重用单元分组 Group Reusable Elements in a Process 79

完成重新分解(以新过程取代单元和流程) Finish Re-factoring (replacing elements and flow with the new process) 80
重新使用部分重新分解模型(取消申请单) Reuse part of the Re-factored Model (Cancel Requisition)82
完成阶段 7 Completing Stage 7 85
实例 See It Live 87
阶段 8 - 过滤提取的数据 Stage 8 - Filtering Retrieved Data 88
阶段目标 Stage Goals 88
覆盖的 Tersus 概念 Tersus Concepts Covered 88
建立应用程序功能建模 Application Functionality Modeled 88
用户建模 User Modeling 88
在查找过程中添加一个触发器（以规定过滤的数值）Add a Trigger to the Find Process (to specify a value by which to filter) 89
使用删除流程（清空表格）Use Remove Flow (to clear the table) 90
在另一个编辑器窗口中打开模型 Opening Models in a separate Editor Window 92
从模型中删除单元 Remove an Element from the Model 94
了解模型包装和命名 Understanding Model Packaging and Naming 95
使用高级查找模板（使用复杂规则过滤记录） Use the Advanced Find template (to filter records using a complex criteria) 98
完成阶段 8 Completing Stage 8 101
实例 See It Live 104
阶段 9--在透视图中排列视图 Stage 9-Arranging Views into Perspectives 105
阶段目标 Stage Goals 105
覆盖的 Tersus 概念 Tersus Concepts Covered 105
建立应用程序功能模型 Application Functionality Modeled 105
用户建模 User modeling 105
添加一个透视图（雇员）Adda perspective (Employee) 107
删除缺省透视图 Remove the Default Perspective 108
添加附加透视图（经理）Add an additional Perspective (Manager) 108
完成阶段 9 Completing Stage 9 110
实例 See It Live 117
阶段 10 从 Excel 输入数据 Stage 10 - importing Data from Excel 118
阶段目标 Stage Goals 118
覆盖的 Tersus 概念 Tersus Concepts Covered 118
建立应用程序功能建模 Application Functiionality Modeled 118
用户建模 User Modeling 120
使用文件输入字段（以选择电子表格文件）Use a file Input Field (to select the spreadsheet file) 120
使用载入 Excel 表格模板（提取数据行）Use a Load Excel table template (to extract data rows). 122
定义从电子表格中提取的行的数据结构 Define the Data Structure of Rows extracted from the Spreadsheet 125
使用文本控制模板（连接文本数值）Using a Text Manipulation Template (to concatenate text values) 128
完成输入过程 Completing the import process) 129
完成阶段 10 Completing Stage 10 132
实例 See It Live 132
阶段 11 -- 控制表格显示 Stage 11-Controlling Table Display 133
阶段目标 Stage Goals 133
覆盖的 Tersus 概念 Tersus Concepts Covered 133
建立应用程序功能模型 Application Functionality Modeled 133
用户建模 User Modeling 133
使用表格模板 Use a Table Template 134
使用数字/文本/日期显示模板 Use Number/Text / Date /Display Templates 134
填充表格 （以申请表）Populate the table (with requisitions) 135

完成阶段 11 Completing Stage 11 139
实例 See It Live 139
阶段 12—控制应用程序流程 Stage 12- Controlling Application Flow 140
阶段目标 Stage Goals 140
覆盖的 Tersus 概念 Tersus Concepts Covered 140
建立应用程序功能模型 Application Functionality Modeled 140
用户建模 User Modeling 140
使用分支模板 Use a Branch Template 140
使用提示模板(向用户显示提示信息) Use an Alert Template (to display an alert to the user) 142
按条件显示弹出窗口 Display a Popup Conditionally 144
完成阶段 12 Completing stage 12 147
实例 See It Live 152
阶段 13-关系数据建模 Stage 13 - Modeling Relational Data 153
阶段目标 Stage Goals 153
覆盖的 Tersus 概念 Tesus Concepts Covered 153
建立应用程序功能模型 Application Functionality Modeled 153
用户建模 User Modeling 154
根据数据结构使用选取器 Use a Chooser Based on a Data Structure 154
使用服务模板(过程必须在服务器上运行时) Use a Service Template (when a process must run on the server) 156
定义（采购订单的）数据库记录 Define a Database Record for the purchase order) 157
正确放置上级索引（不是在服务中） Positioning an Ancestor Reference Correctly (not in a service) 159
填充采购订单记录（以采购订单详情） Populate the Purchase Order Record (with purchase order details) 161
更新申请单 Update the Requisition 164
使用<已完成>（<Done>）出口（以规定运行顺序） Use a <Done> Exit (to specify the order of execution) 166
完成阶段 13 Completing Stage 13 168
实例 See It Live 171
阶段 14 -- 显示多个（连接）表格 Stage 14 - Displaying Multiple (Linked) Tables ... 172
阶段目标 Stage Goals 172
覆盖的 Tersus 概念 Tersus Concepts Covered 172
建立应用程序功能模型 Application Functinality Modeled 172
用户建模 User Modeling 173
<点击>过程（点击某一行时运行一个过程）<On Click> Process (to execute a process when a row is clicked) 173
添加一个过程（以计算每个申请单的采购订单总金额）Add a process (to calculate PO aggregates for each requisition) 181
使用汇总模板(以计算申请单总价) Use a Sum Template (to calculate the total price of a requisition) 183
使用刷新模板(以刷新显示的数据) Use a Refresh template (to refresh data in your display) 184
完成阶段 14 Completing Stage 14 186
实例 See It Live 186
附录 A - Tersus 工作室功能和工具 Appendix A - Tersus Studio Features and Tools 188
附录目标 Appendix Goals 188
Eclipse 平台 The Eclipse Platform 188
创建新项目 Creating a new Project 189
熟悉模型编辑器 Familiarizing Yourself with the Model Editor 193
“选项板”（The Palette） 193
在模型中插入新单元 Inserting a New Element to the Model 194
选择单元 Selecting an Element 194
移动单元 Moving an element 195

调整单元大小 Resizing an element 195
下钻（Drill-down） 195
放大/缩小(Zoom-in/out) 196
撤消/重复（Undo/Redo） 196
大纲 The Outline 196
与模型编辑器同步化 Synchronization with the Model Editor 197
双击动作（Double-Click Behavior） 197
拖放动作（Drag-and-Drop Behavior） 197
储存库管理器 The Repositiory Explorer 197
使用储存库管理器 Finding your way in the Repository 198
双击动作（Double-click Behavior） 198
拖放动作(Drag-and-drop Behavior) 198
储存库管理器和大纲(The Repository vs. the Outline) 199
嵌入式应用程序和数据库服务器 The Embedded Applicatin and Database Servers 199
附录 B-可视调试 Appendix B - Visual Debugging 201

阶段 1 - 简介 Stage 1 - Introduction

关于 Tersus 平台 About the Tersus Platform

欢迎来到Tersus 平台。

利用Tersus, 你能很容易地通过画图而不是写代码编网页应用程序。

Tersus技术已成功应用于建立各种软件解决方案（从精巧的应用程序到高端的、处理金融交易的、对任务至关重要的系统）。

Tersus 特别适合由内置组件、自开发组件以及网页服务组成的复合应用程序。

Tersus 平台包括三个主要组件 ：

- **Tersus 工作室(Tersus Studio)**，是Eclipse平台的扩展，建模者（开发人员和业务专家）使用Tersus 工作室以图像定义应用程序的功能;
- **Tersus 模型库(Tersus Model Libraries)**，包含组装应用程序的组建模块;
- **Tersus服务器(Tersus Server)**，执行模型化解决方案和进行所需的数据库更新(可在J2EE应用程序服务器上运行)。

通过定义模型层次结构**编写应用程序(Creating an application)**，每个模型都是由较低级组件组成。开发人员从代表整个系统的顶层图表开始，自上到下重复进行细化 – 展开每个模型以规定模型的组件。通过使用代表整个模型的层次结构的“无限画板”，开发人员可以通过视觉和直观的方式，完整和精确地规定所需的商业逻辑。

模型一旦建立，便可立刻**运用应用程序(Deploying the application)**。模型以多层次的XML文件进行保存，XML文件由Tersus运行时间引擎(Tersus Runtime Engine)进行读取。引擎在所有层次（用户接口、服务器端处理和数据库操作）运行模型规定的功能。如果为了审计或者为了分析问题的根本原因需要进行追踪，它还能记录运行的所有详情。

通过修改模型**维护现有应用程序 (Maintaining an existing application)** - 改变业务流程、添加新组件、或者关闭冗余组件。必要的修改完成之后，便可立即重新运用应用程序。

关于本教程 About this Turtorial

本教程使用Tersus建模工具，简单介绍了一个完整的样本应用程序（一个采购申请管理系统）的开发过程。

本教程提供了一个开发这样的系统的分步骤实例，系统从员工发出申请单开始，到所申请的物品送货时结束。

在几分钟内和建立应用程序的每一个步骤，您可以用 Tersus 运行时间引擎从浏览器启动应用程序。

文件规定 Document Conventions

本教程使用以下规定:

规定	内容
双击根模型(Double-click the root model)	你可以遵照和执行的分步骤建模说明
视图 (View)	Tersus 建模工具中的对象
新申请单(***New Requisition***)	你建立的模型的推荐的名字
数据结构为… A data structure is …	详尽阐述的超出当前主题的值得注意的信息

使用教程 Using the tutorial

本教程由多个阶段组成，每个阶段都会谈到Tersus开发方法的几个新概念。

在每个阶段里面，教程都会谈到：

- 开发过程（建模型或“如何建立模型”）;
- 应用程序定义(模型，或“建立的模型完成后是什么样子”)；以及
- 结果（应用程序, “用户得到的能够使用的产品”）。

每个阶段后面都有一个实施所有阶段（到当前阶级为止，包括当前阶段）的预先建好的、可运行项目样本。

项目样本的目的有两个:

1. 在阅读教程每个阶段时提供一致的参考信息。
2. 提供与教程中建立的模型类似的附加功能。

你可以使用样本并跳过附加功能建模部分，也可以自己创建模型(附加功能的描述简短，并没有详细区分步骤，但由于您们在该阶段较早的时候已经进行了学习，您们应该可以根据这些描述自己创建功能模型)。

我们强烈建议您按照本教程的阶段和所列的顺序进行学习。

应用程序样本 The Sample Application

Acme corp (ACME公司) 想以基于网页的、员工能够提出产品（如电脑、家具、办公用品）采购要求的电脑化系统替代人工的、纸张密集型的采购申请流程。

采购申请流程必须至少经过以下几个步骤 ：

1. **员工**输入新的申请;

2. 员工**经理**批准/退回申请;

3. **采购人员**处理已批准的申请单并发出采购订单;

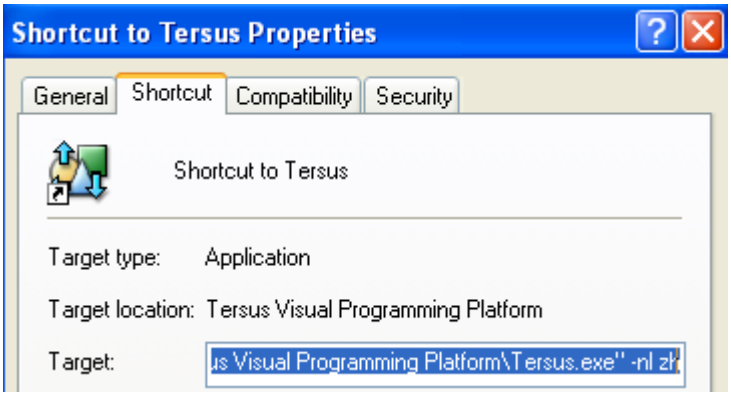
4. 订购的物品送货后，申请视为已完成和已经结束。

使用Eclipse中文版 Use Eclipse in Chinese version

1. 下载Eclipse中文语言包:
   http://www.eclipse.org/downloads/download.php?file=/technology/babel/babel_language_packs/BabelLanguagePack-eclipse-zh_3.4.0.v20091121043401.zip

2. 复制中文语言包的功能(features)和插件(plugins) 到..\Tersus Visual Programming Platform\features 和 ..\Tersus Visual Programming Platform\plugins

3. 添加 “-nl zh”， 然后启动 Tersus 工作室(Tersus Studio)



阶段 2 - 建立基本显示模型 Stage 2 - Modeling a Basic Display

阶段目标 Stage Goals

本阶段介绍**Tersus 工作室(Tersus Studio)**

你将会学习如何生成新的Tersus项目(Tersus project)

覆盖的 Tersus 概念 Tersus Concepts Covered

完成本阶段后，你将会熟悉以下概念：

建模概念(Modeling notions): 模型 (Model)， 显示(Display)， 模板(Template)。

模型与单元名称 (Model vs. Element Name.)

建模技术(Modeling techniques): 生成和管理显示(Creating & managing the display.)

重新命名模型 (Renaming a model.)

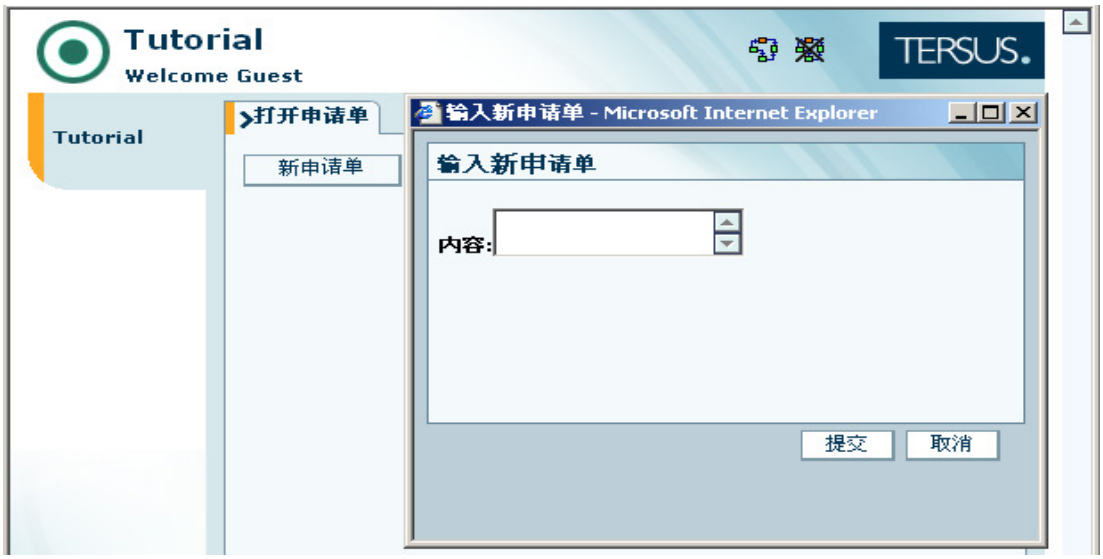
有用的显示模板(Useful display templates): 视图(View)， 按钮(Button)， 弹出窗口(Popup)，

标签(Label)， 文本区(Text area)

建立应用程序功能模型 Application Functionality Modeled.

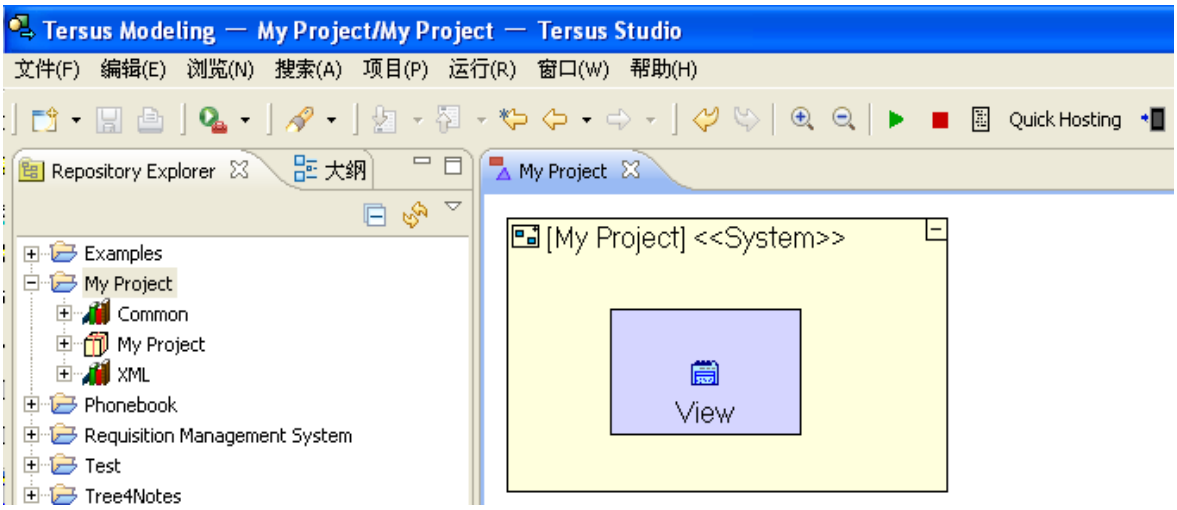
在这个阶段，你将建立用于输入申请单的浏览器表单模型。

建立的网页应用程序(web application)包含一个打开弹出表单的按钮(**button**)，用于输入新的申请单(**enter a new Requisition**)，表单外观与下图类似:



Tersus 工作室 The Tersus Studio

以下屏幕快照显示了Tersus 工作室的缺省外观。它包含右边的带集成**选项版(**integrated **Palette)的模型编辑器(Model Editor)**左边的包含储存库管理器(Repository Explorer)和大纲(Outline)的带标签视图(tabbed view)。



如果您在开始第一个项目之前想了解更多关于Tersus工作室的信息，请参阅附录A(Appendix A)。附录A包括以下主题:

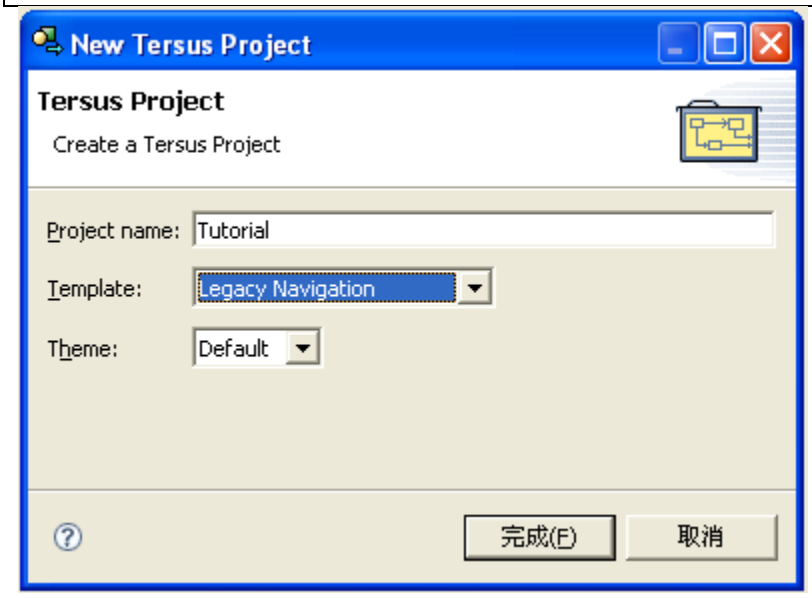
- Tersus工作室(Tersus Studio)和Eclipse平台(Eclipse Platform)
- 选项板 (Pallete)
- 在模型中插入新单元 (New Elements)
- 选定(Selecting)、移动(Moving)和调整单元大小(Resizing Elements)
- 下钻(Down-drill)
- 放大/缩小 (Zoom-in/out)
- 撤消/重复 (Undo/Redo)
- 大纲(Outline)
- 储存库管理器 (Repository Explorer)
- 应用程序服务器 (The Application Server)

生成简单的网页应用程序 Create a Simple Web Application

我们可以从生成一个新的应用程序项目开始：

选择**文件（File）-> 新(New)->**  **Tersus项目(Tersus Project)。**

注意：**Tersus项目**是“**新文件**”分菜单的第一个选项，后面是“**项目**”，“**项目**”是一个普通的eclips选项（如需了解更多信息，请参阅附录A）。



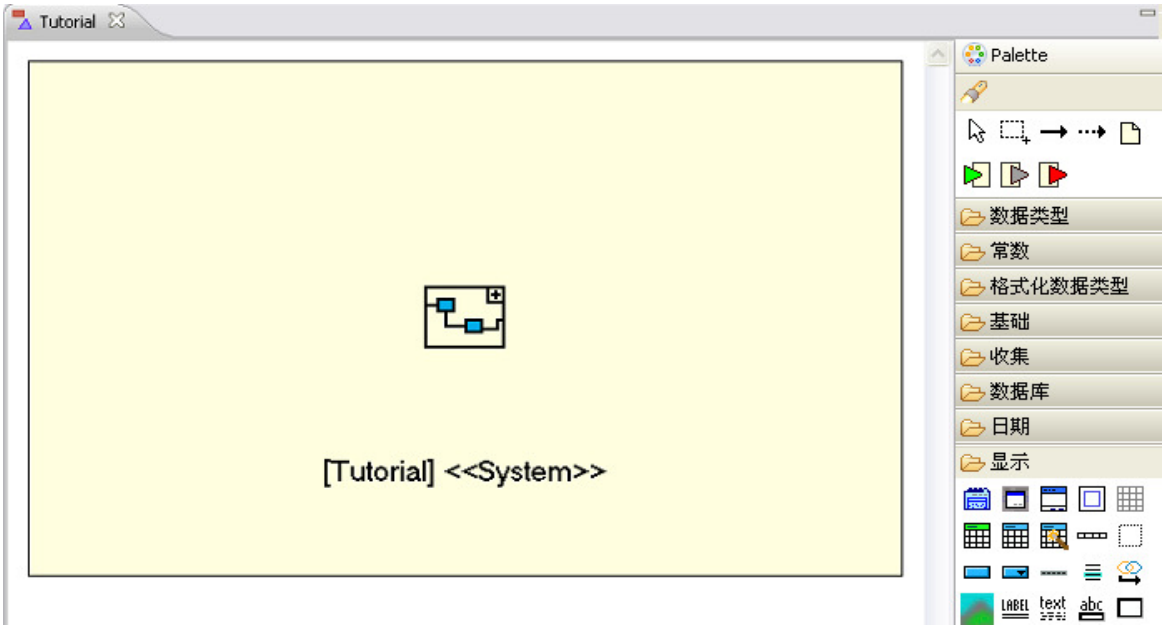
输入新项目的**项目名称 (project name)**：Tutorial。
选择“**模板 (Template)**”：**Legacy Navigation**
点击“**完成 (Finish)**” 按钮。

开始建模-生成输入申请单的表单 Start Modeling -Create a Form for Entering a Requisition

要开始建模，我们必须在模型编辑器(model editor)中打开应用程序根模型(root model)。
如果您已生成新项目**Tutorial**，现在必须在模型编辑器中打开**Tutorial**。如果您还没有生成新项目**Tutorial**，请进行以下操作：

在储存库管理器（Repository Explorer）视图中找到Tutorial项目根（文件夹 folder）并双击根文件夹。

双击根文件夹将打开一个新的编辑器窗口（外观与以下屏幕快照类似）。编辑器窗口里面有一个黄色的长方形，这个长方形代表应用程序。由于我们还没有建模，这个长方形里面没有什么内容，只有文件名称。



生成视图（打开申请单）Create a View (Open Requisition)

您可以在浏览器中看到网页应用程序，所以我们首先必须对**视图(View)**进行定义。
视图模型（View Model）定义浏览器显示的内容，也包含其它显示单元（标签、按钮、表格等）。

要生成名称为“**打开申请单 (Open Requisitions)**”（显示您所有打开的申请单）的视图，请进行如下操作：
在选项板中（palette,在模型编辑器右边），确保“**显示**”类别是打开的（如果“显示”类别还没打开，请点击打开）。
点击“**视图**”模板以选  择“**视图**”模板。
您可以看到，当您把鼠标光标移动到编辑器上时，光标形状改变，表示可以把“**视图**”放在那里（显示一个小的灰色的长方形）。
把光标移到根模型里面，点击以插入视图。
这样就可生成视图模型，编辑器会提示您输入视图的名称：
输入 **“打开申请单”(Open Requisitions)**，完成后点击[Enter]。
您刚生成的简单模型应与下图类似：

“打开申请单”（Open Requisition）视图（绿色/蓝色长方形）现在是根模型**Tutorial**（黄色长方形）的子模型。
多层次结构中的任何模型可包含任何数量的根模型。

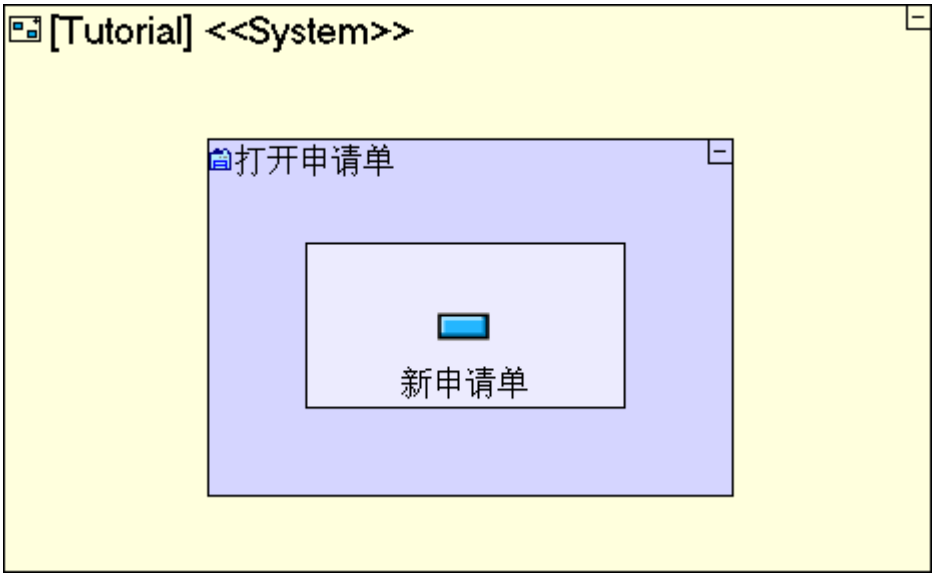
注意：本教程中提供的模型屏幕快照可能与您生成的模型有所不同，这主要是因为模型中的单元位置和大小有所不同，另外一个原因是包含其它单元的单元可扩大（标有“  ”或“  ”）。正如本阶段后面所解释的一样，这种区别不影响应用程序的功能（显示模型除外）。

添加一个按钮（新申请单）Add a Button(New Requisition)

现在我们继续建立视图模型。我们现在需要添加一个“**新申请单**”（**New Requisition**）按钮，让用户可以在弹出表格中输入新申请单)。

在选项板(Palette)的 “**显示**”(**Display**)类别中选择“**按钮**”(**Button**)模板  (点击模板)。
通过在“**打开申请单**”(**Open Requisitions**)窗口内部点击，将按钮插入“**打开申请单**”视图中。
把按钮命名为 “**新申请单**”(**New Requisition**)” (输入“**New Requisition**”)。

现在模型外观应与下图类似:

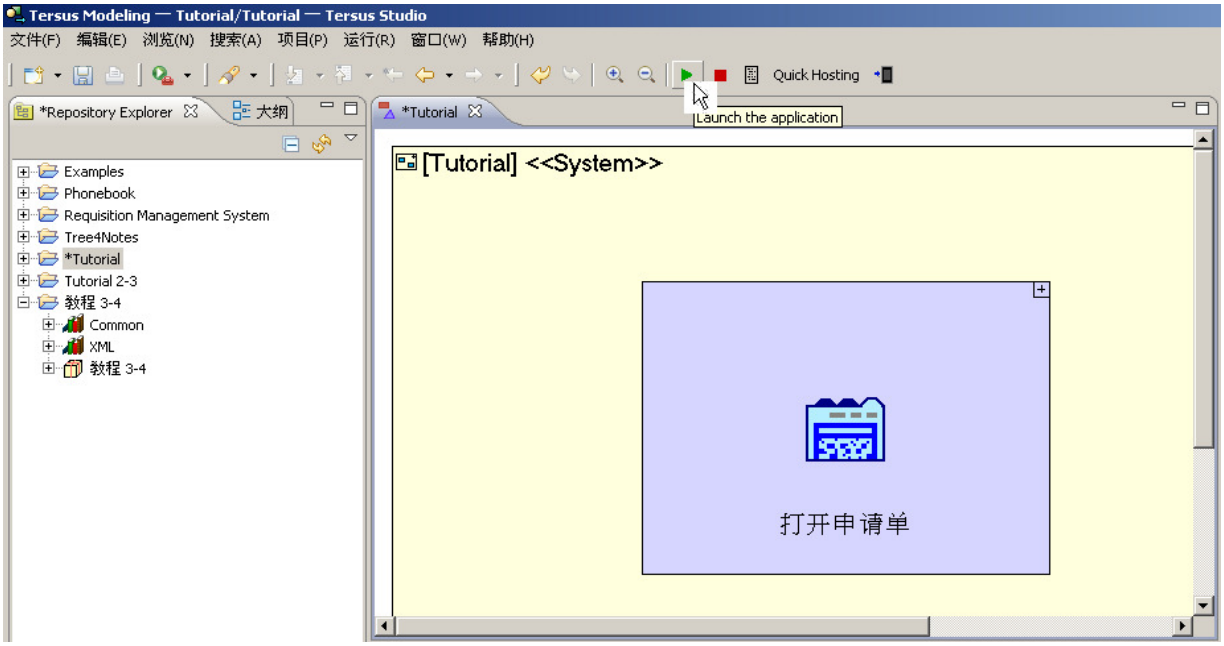


储存库管理器(**Repository Explorer**)和模型编辑器(**Model Editor**)的标题旁边有星号(如 *Tutorial)，这表示最近所做的修改还没有保存。

通过点击工具栏(toolbar)  上的对模型进行保存。

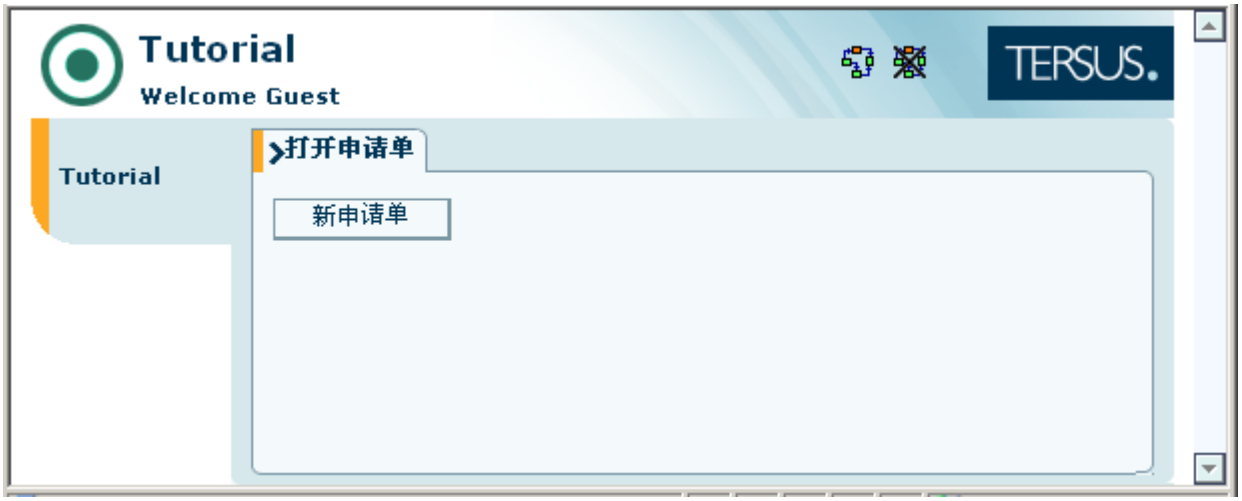
保存模型时，Tersus 工作室（Tersus Studio)会检查(验证)你的模型。如果发现任何错误，就会显示一条信息，而错误会在“验证”视图中显示。有关验证(**Validation**)的更多详细内容，请参阅“完成阶段12”(Completing Stage 12)。

虽然我们建立的模型东西很少，但我们已经可以在浏览器里查看我们的应用程序。
在Studio的主工具栏中点击“**启动应用程序**”(**Launch the application**)按钮，在嵌入式Tersus服务器(Tersus Server)中载入您的应用程序并在网页浏览器中打开。



如果“启动应用程序”（Launch the application）按钮没有启用，点击模型编辑器就可启用“启动应用程序”按钮。

浏览器就会显示与下图类似的页面：



您可以看到“**打开申请单**”(Open Requisition）视图是唯一的选项标签。后面当您插入其它视图，这些视图就会作为其它选项标签显示出来。在视图里面，我们可以看到“**新申请单**”（New Requisition）按钮。

返回Tersus工作室（Tersus Studio）。您可以不关闭浏览器，让浏览器继续在后台运行。

生成弹出窗口（输入新申请单）Create a Popup(Enter New Requisition)

人们经常会想：按了按钮之后，电脑里面发生什么事情呢？如果我们能够穿过这些按钮看到应用程序逻辑的内部就好了，这正是Tersus通过“放大”（zoom in）技术让您能做到的-对按了按钮之后按钮“内部”发生的动作建立模型。

现在我们放大到“**新申请单**”（New Requisition）按钮并为点击按钮后显示的弹出表单（popup form）建模：

双击“**新申请单**”按钮以放大按钮。
从选项板（palette)中选择“**显示/弹出窗**” （Display/Popup）模板。
把它插入到“**新申请单**”单元中。
把它命名为“**输入新申请单**”（Enter New Requisition）。

现在“**新申请单**”按钮模型外观应与下图类似：

“**弹出窗口**”模板是一个模板例子，提供附加的、预先设置好的、独创的功能。
在弹出窗口方面，预先设置好的功能包括：
1. 一个页脚(Footer)，使内容在弹出窗口底部显示出来。
2. 一个“OK”按钮，目前没有任何作用。
3. 一个“取消”(Cancel)按钮，“取消”按钮包含一个“关闭窗口”(Close Windows)模型，点击“取消”按钮，弹出窗口就会关闭。

总之，您可自由按您的意愿修改（或删除）这个功能。

点击 对工作进行保存

现在我们来看一下您的应用程序在浏览器中是什么样子：

转回浏览器界面。

浏览器识别到这个应用程序自从上次载入之后有所变化并自动重新载入这个应用程序。

重新载入应用程序后，点击“**新申请单**”（New Requisition）按钮

就会显示以下结果：

点击“**取消**”（Cancel）按钮，确认弹出窗口是否正如您预料的一样关闭了。

添加显示单元到弹出窗口里面 Add Display Element to the Popup

我们需要一个位置让用户可以输入申请单详情，这个窗口应该足够。

首先，生成一个“**内容：**”标签（“Description：”），提示用户描述申请单内容：

双击“输入新申请单”（Enter New Requisition）弹出窗口以放大窗口。

选择“**显示/标签**” .LABEL. （Display/Label）模板，把它放到弹出窗口上部。

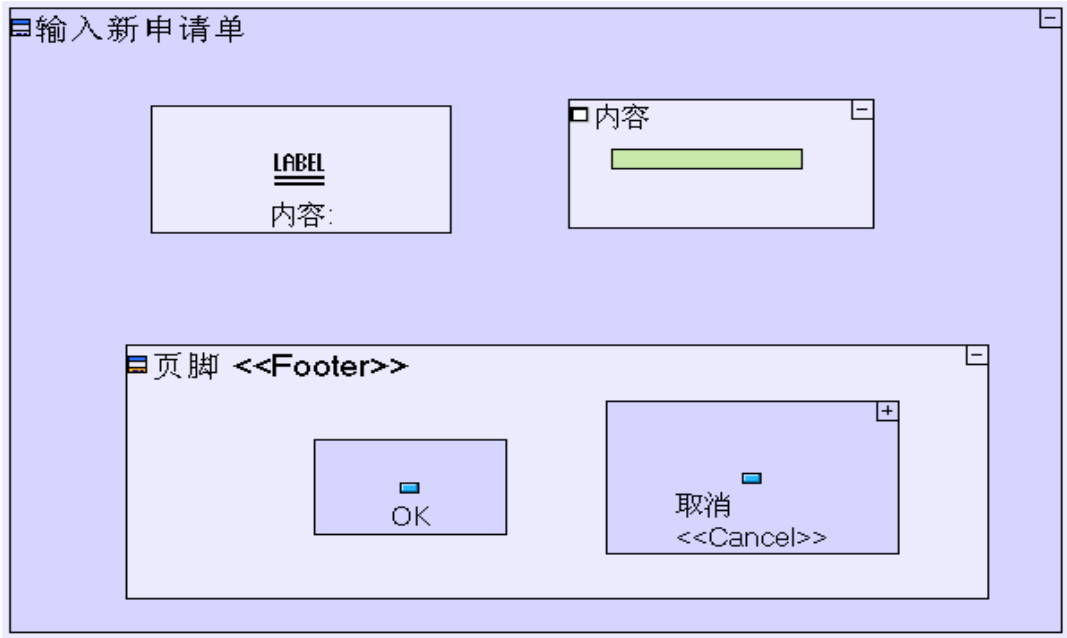
把它命名为 “***内容:”(Description:)*** 冒号(Colon)是名称的一部分

然后插入一个文本区单元，使用户能输入申请单内容：

选择“**显示/文本区**”（Display/Text Area）模板并把它放到标签下面，

把它命名为 “***内容”(Description)***

模型现在外观应与下图类似：



保存您的工作结果，转回浏览器界面。

弹出窗口外观应该与下图类似：

注意：表单中显示的显示单元的顺序由弹出窗口模型中相应的分模型的相对位置决定。总的来说，顺序是从上到下和从左到右（与阅读英文的顺序一样）。
总而言之，模型多层次结构中显示单元的排列规定了显示单元在屏幕上的外观。我们已经看到两个不同的例子：

1. 把弹出窗口放在按钮里面使我们可通过点击按钮显示弹出窗口。
2. 把显示单元放在一起（在同个母模型里面）规定了显示单元显示在屏幕上的顺序。

重新命名模型 Rename a model

“输入新申请单”（Enter New Requisition）弹出窗口里面有一个预先设置好的**“OK”**按钮，在下个阶段我们将使用这个按钮提交新申请单。相应地我们也对**“OK”**按钮重新进行命名：

移到模型中的**“OK”**按钮处。

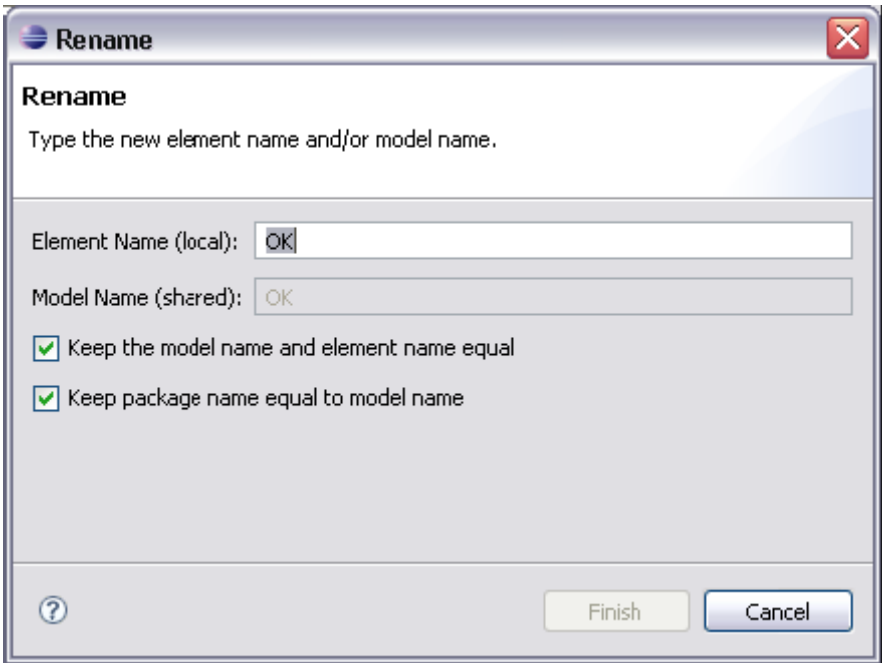
然后

点击**“OK”**按钮或从菜单中选择**“重新命名”**（Rename）。

或

点击以选择模型并按[F2]键。

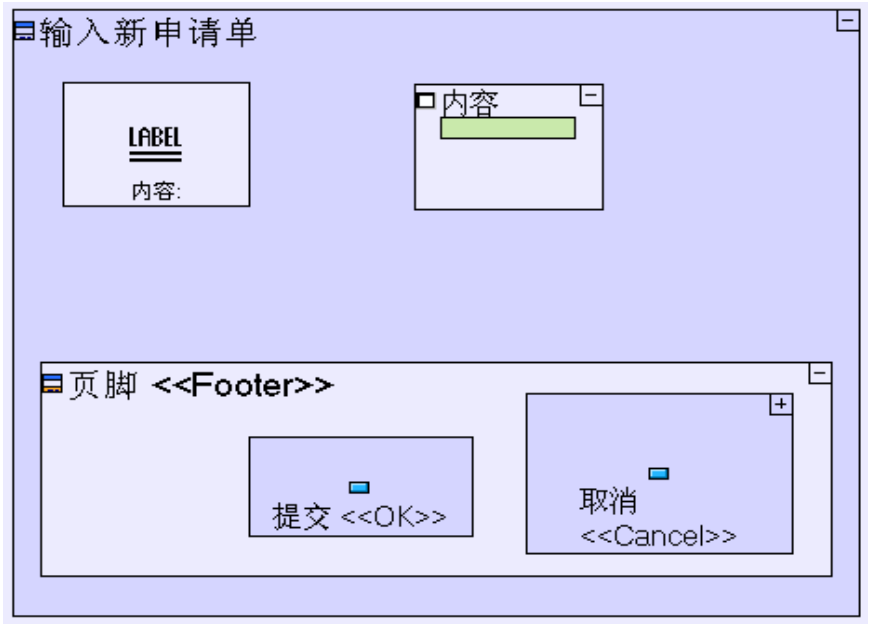
这样就可以打开**“重新命名”**对话框。



输入**“提交”**（Submit）以替换**“OK”**，然后点击**“完成”**（Finish）。

> **“重新命名”**(Rename) 对话框用于为重新命名的模型单元指定两个不同的名称，一个**“单元名称”**（Element Name）和一个**“模型名称”**（Model Name）。
> **“模型名称”**是可重用模型的共享名称（关于重用概念的进一步详情请参阅**阶段4**）。**“单元名称”**在一个指定的数据包里面必须是唯一的。
> **“单元名称”**是模型的本地名称。**“单元名称”**在一个指定的母模型里面必须是唯一的。对于显示模型来说（如我们刚生成的**“视图”**），单元名称是用户看到的名称。
> 一般情况下，这两个名称是一样的；缺省情况下会勾选**“保持模型名称和单元名称一致”**（Keep the model name and element name equal）复选框，这样这两个名称就会同时改变。
> 但在某些情况下，不能使用特定的单元名称或模型名称；在这种情况下，您可能需要不选定复选框并设定不同的名称。想了解更多详情，请参阅阶段14中关于自动生成名称的讨论。
> 如果您重新命名的模型的名称与包含模型的数据包(package)的名称一样，**“保持模型名称和单元名称一致”**(Keep package name equal to model name)复选框就会显示出来。在缺省情况下，这个复选框处于选定状态。如果您不想重新命名数据包的话，您可以不勾选这个复选框。

现在您的模型外观应该与下图类似：



我们重新命名的按钮显示在括号里面的名字仍然是原来的名字（“<<OK>>”),这是为了提醒我们这个按钮来自弹出窗口模板。原来的名字没有什么实际作用，在运行时间不会显示出来，可以不予理会。

保存您的工作成果，转回浏览器 (browswer)界面。

现在您的弹出窗口(popup)外观应该是这个样子：

完成阶段 2 Completing Stage 2

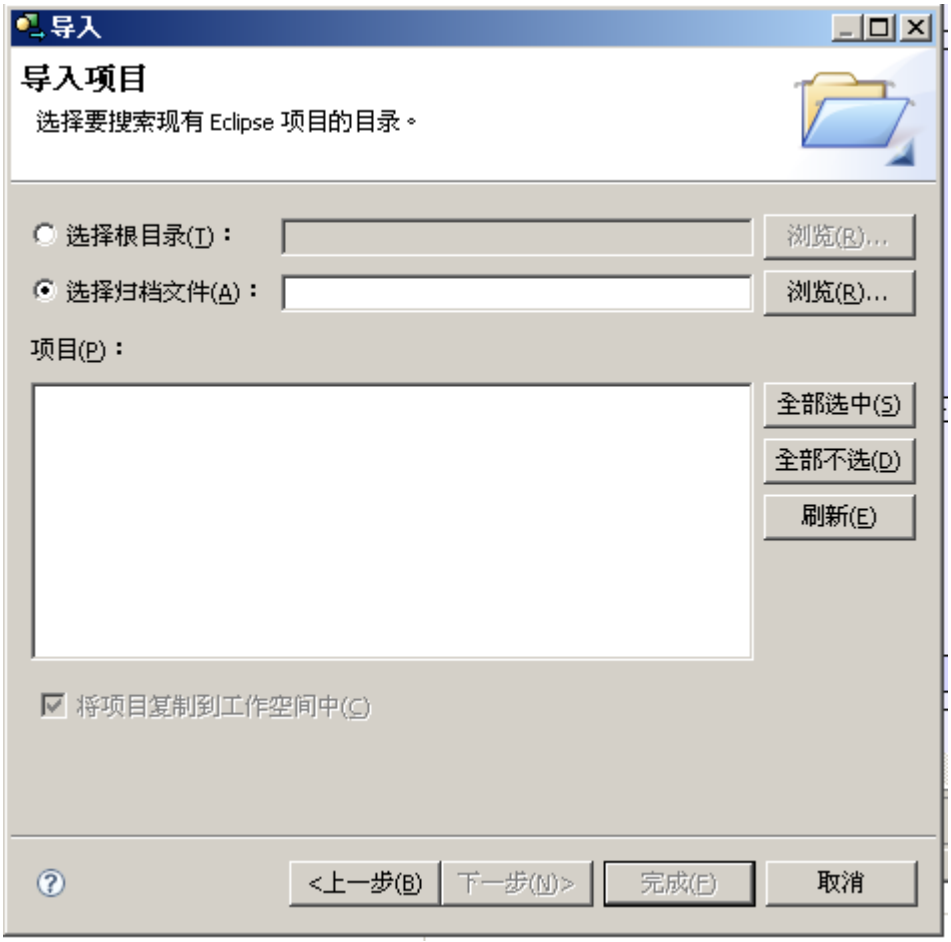
我们已完成表单建模，但按“**提交**”(Submit）按钮并不会把数据保存在任何地方。下个阶段我们会处理这个问题。

输入项目样本 Importing a Sample Project

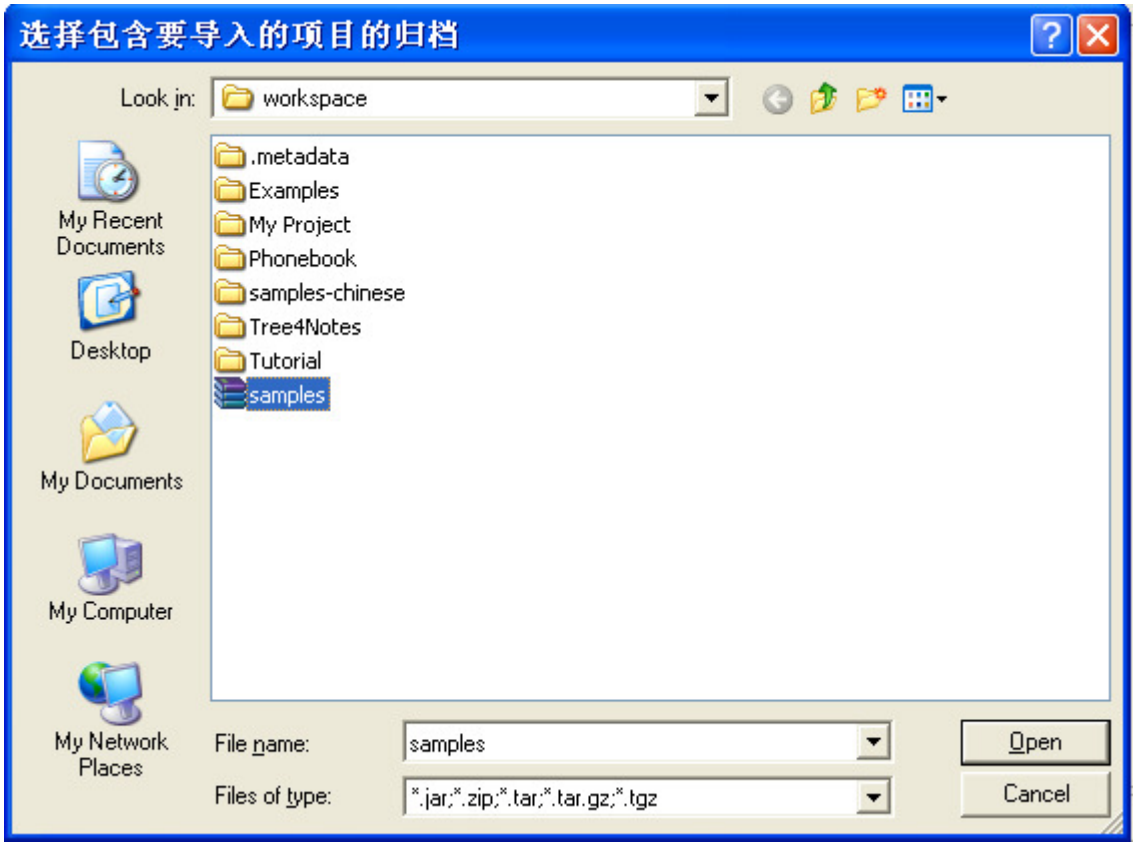
现在我们输入一个现成的项目样本“**Tutorial 2-3**”。这个项目样本具备到目前为止我们建立的模型的所有功能，这些功能是本教程下一个阶段的基础：

选择“**文件**”（File）“**导入**”(Import)…以打开“**导入**”向导(Import wizard)。

选择“**现有项目到工作区空间中**”（Existing Project into Workspace）作为输入来源。
点击“**下一步>**”（next>）。



确保选择“**选择档案文件:**”（Select archive file:)的单选按钮，然后点击“**浏览…**”（Browse...）按钮以打开“**选择包含输入项目的档案**”（Select archive containing the projects to import）对话框。



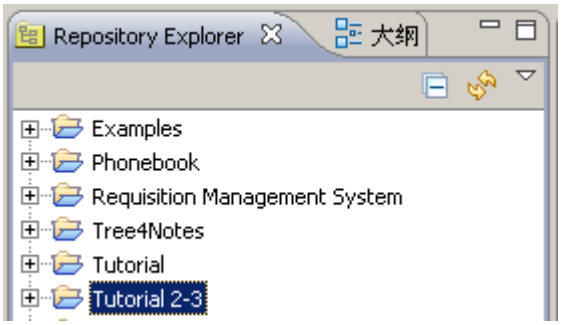
在缺省情况下，对话框应打开包含项目样本档案的“**工作区**”（workspace）文件夹。

选择（或双击 double-click）“**samples.zip**”档案。

注意：
“**samples.zip**”档案含有本教程所有阶段的项目样本。您想输入多少项目就可以输入多少项目，但建议您只输入需要的项目样本（如这个阶段需要的项目样本是“**Tutorial 2-3**”）。
您选择的档案的完整路径可能与上图的路径有所不同，这取决于您把Tersus安装在什么地方。

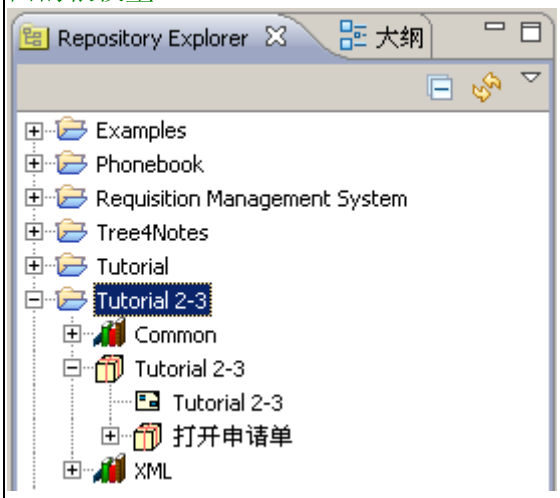
点击“**全部不选**”（Deselect All）按钮。
勾选“**Tutorial 2-3**”项目前面的复选框（check box）。
点击“**完成**”（Finish）以输入项目。

向导(wizard)就会输入项目，并把项目添加到“**储存库管理器**”（Repository Explorer）。

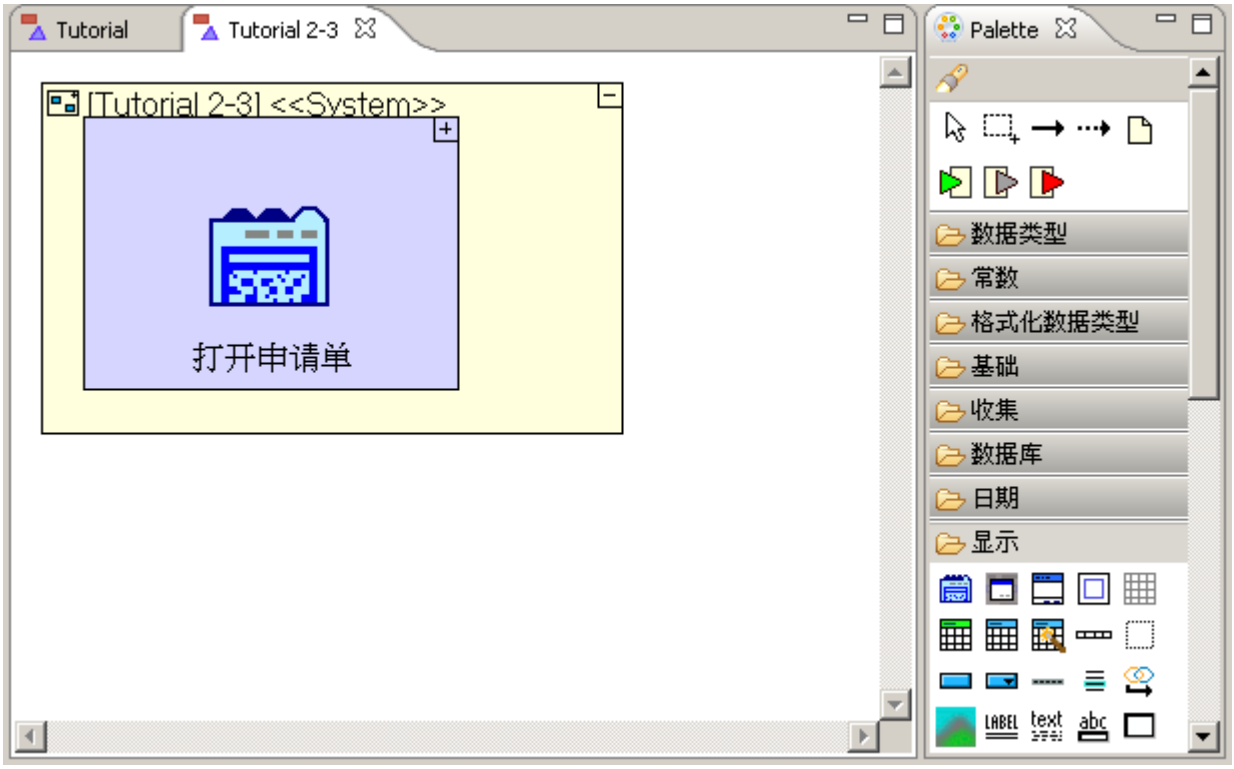


双击“**Tutorial 2-3**”项目以打开模型编辑器(model editor) 中的项目的根模型(root model)。

您也可以展开输入的项目“**Tutorial 2-3**”，里面有“**Tutorial 2-3**”数据包（package），然后双击里面的根模型。

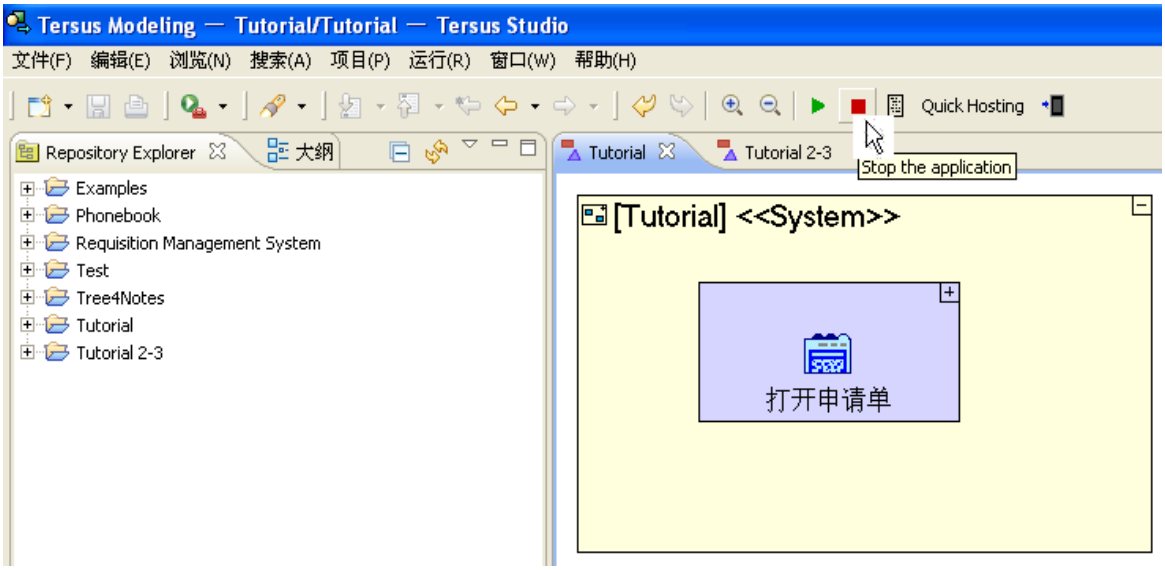


这样就会在一个新的模型编辑器窗口(model editor window)内打开模型。

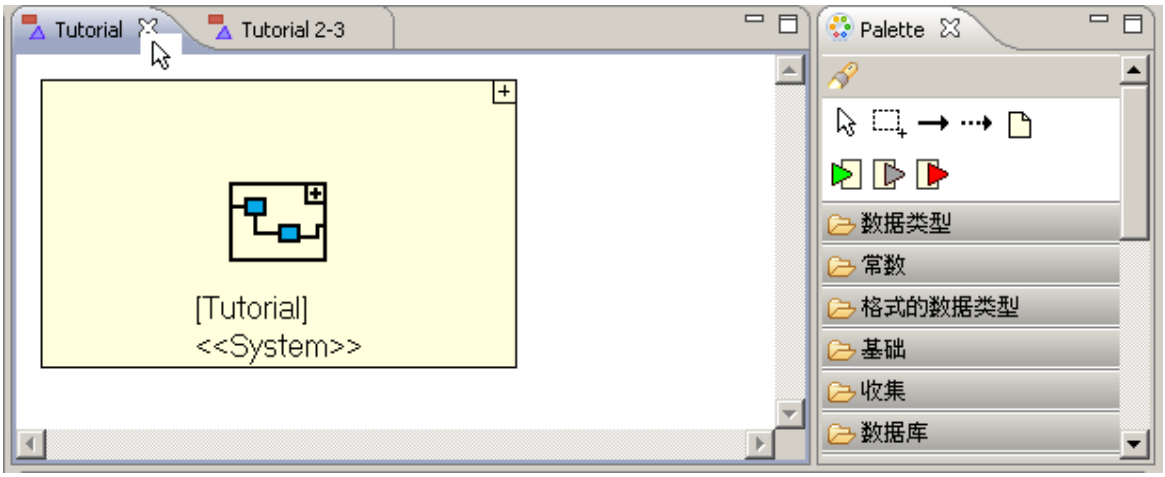


您可以看到，您之前处理的项目“**Tutorial**”在另一个模型编辑器和应用程序服务器中仍然是打开的，并可进行编辑（标签在模型编辑器的最左边，见以上屏幕快照）。

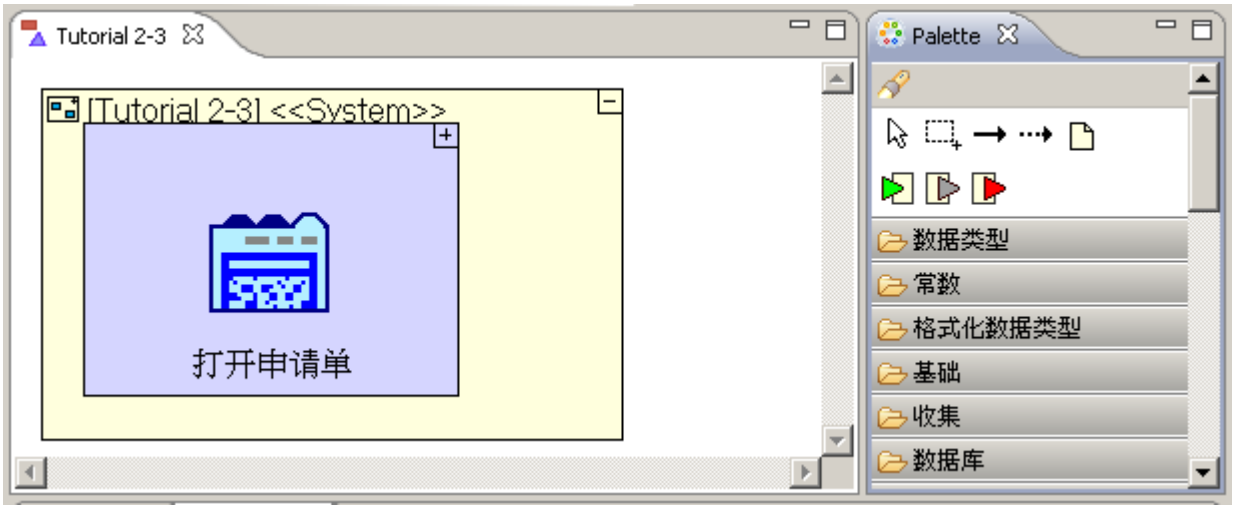
点击“**Tutorial**”标签(tab)以转回“**Tutorial**”模型编辑器(model editor)。
点击工作室（Studio）主工具栏中的“**停止应用程序**”（Stop the application）以从嵌入式Tersus服务器上(server)载应用程序。



关闭运行应用程序的浏览器窗口。
按编辑器标签(editor’s tab)上的“关闭”（close）按钮以关闭“**Tutorial**”模型编辑器:



模型编辑器(model editor)现在外观应该是这样的:

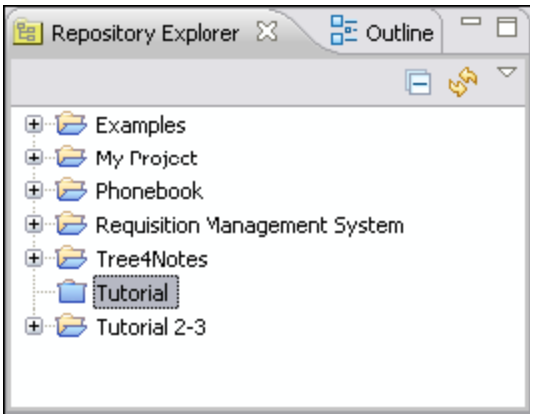


建议您关闭您在本阶段开始时创建的（原来的old）“**Tutorial**”项目：

在储存库管理器（Repository）中，右击(right-click)“**Tutorial**”项目。

从菜单中选择“**关闭项目**”（Close Project）。

储存库管理器外观应与下图类似：



您现在可以进入**阶段3 (Stage 3)**。在**阶段3**里面，我们将执行一个把已提交申请单保存到数据库里面的过程。

实例 See It Live

点击此处在另一个窗口里面打开实际项目。

阶段 3-建立幕后逻辑模型 Modeling the logic behind the Screen

阶段目标 Stage Goals

覆盖的 Tersus 概念 Tersus Concepts Covered

完成本阶段后，您将会熟悉以下概念：

建模概念 (Modeling notion)：数据-单元(Data-element)、数据类型(Data Types)、流程(Flow)、显示数据-单元(Display Data-element)、原始参考资料（Ancestor Reference)、保留名称(reserved Names)、原型(Phototypes)

建模技术(Modeling techniques)：从显示中提取用户输入数据，把数据保存到数据库表格(Database table)里面。

有用的过程模板(useful process templates)：序列（Sequence)、插入（Insert）

建立应用程序功能模型 Application Functionality Modeled

在这个阶段里面，您将会建立一些基本应用程序逻辑的模型-（通过弹出表单）提取用户输入的申请单内容并把内容保存到数据库表格中。

本阶段建模在上个阶段最后输入的“**Tutorial 2-3**”项目中进行。

应用程序逻辑建模 Model Application Logic

在上个阶段，我们学习了怎样建立表单模型，以输入简单数据。我们提到缺少一样重要的东西-保存用户输入的数据。

现在我们就来建立保存申请单内容的模型（当用户点击了“**提交**”(Submit）按钮时)。

正如上面所提到一样，Tersus 使用“放大”(Zoom in）技术，这种技术在触发应用程序逻辑的显示单元“内部(inside)”定义应用程序逻辑。就目前的情况而言，点击按钮时应该保存申请单详情，所以建模在“**提交**”按钮内部进行。

在模型编辑器中（或在**大纲**（Outline）视图中）双击“**提交**”按钮以放大到“**提交**”按钮。

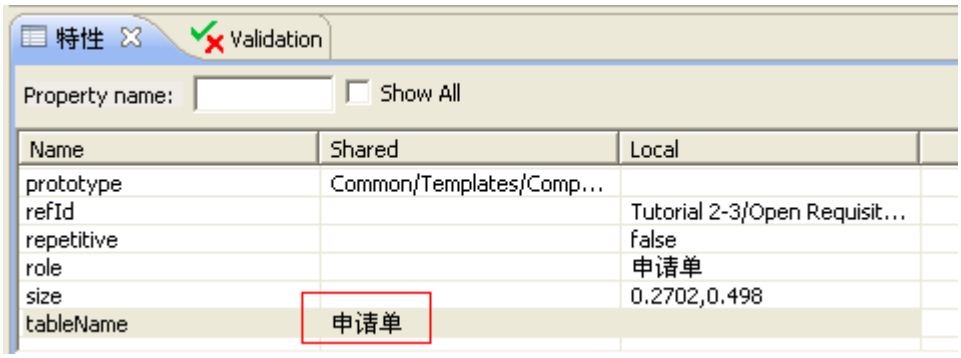
定义数据结构 Define a Data Structure

我们现在对数据结构进行定义，以保存申请单详情：

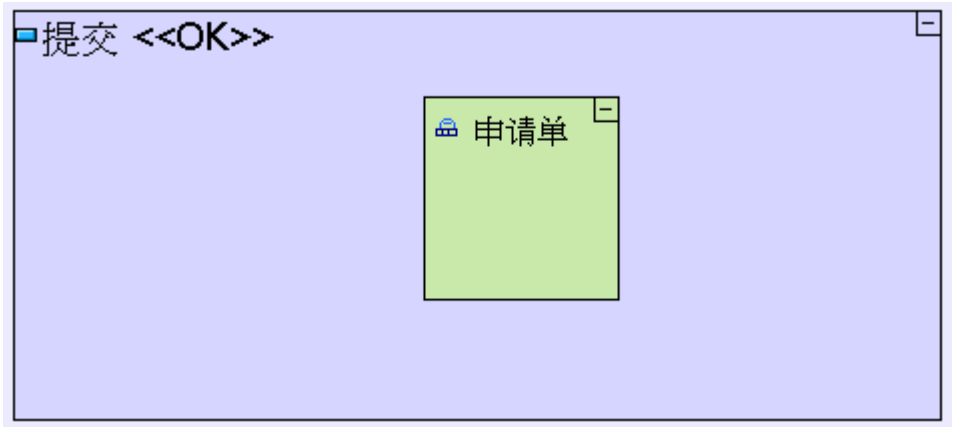
选择 “数据数型/数据库记录”(Data Type/Database Record）模板并把模板放到按钮里面。把它命名为 **“申请单”(*Requisition*)**

数据结构是一系列数据，可以是“基元”（如数字、日期或其它数据结构）。数据结构在模型中显示为灰色长方形。
您可能已经注意到在选项板（Palette）里面,**数据结构**模板(Data Structure template)显示在**数据库记录**模板(Data Record template)的旁边。这两个模板实际上是一样的，唯一的区别是**数据库记录**结构自动与匹配的数据库表格关联，由于我们想把**申请单(Requisition)**数据保存到数据库里面，所以我们把**申请单**数据建立在**数据库记录**的基础上。

选择**申请单->特性(properties),**把 **tablename** 命名 **“申请单”(Requisition)**



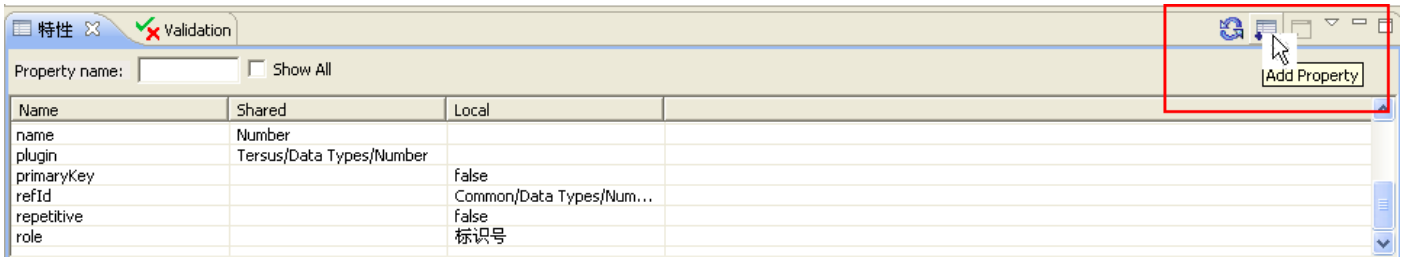
“**提交**”模型现在外观应与下图类似：

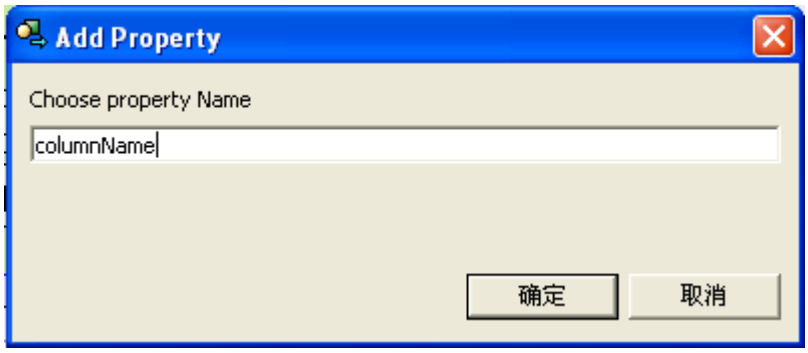


我们接着定义包含数据结构的字段。要将数据单元插入到数据结构中，我们只需拖放适当的数据类型（数字、文本等）。

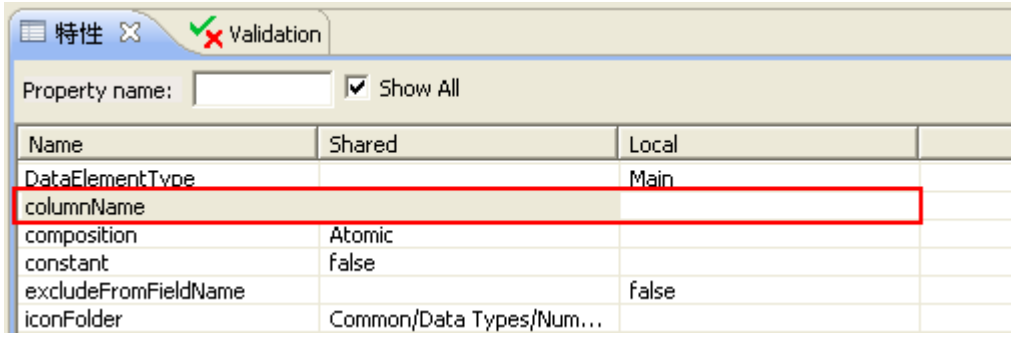
选择 **数据类型/数字（Data Types/Number）**模板并把它放到**“申请单”**数据结构里面。把它命名为 **“标识号”**（*Id*）

选择**标识号->特性（properties），**点击 把它命名为 **“columnName”**



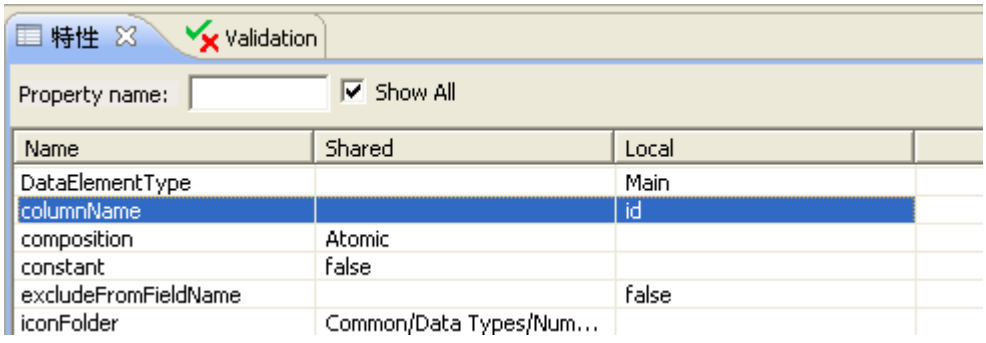


现在“特性”模型的外观应该是这个样子：



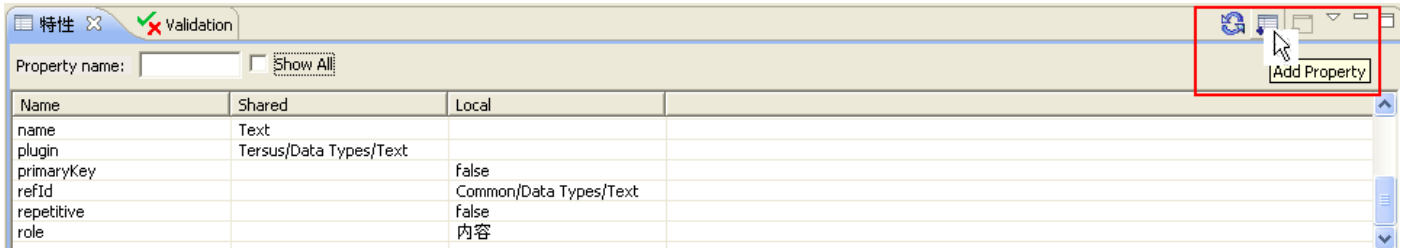
把 **“columnName”**命名为 **“Id”**

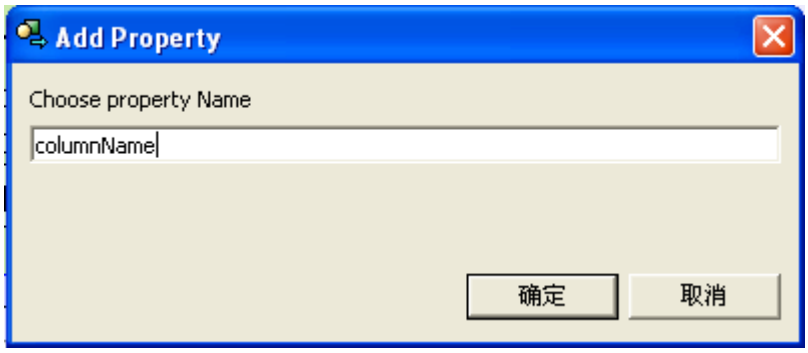
现在“特性”模型的外观应该是这个样子：



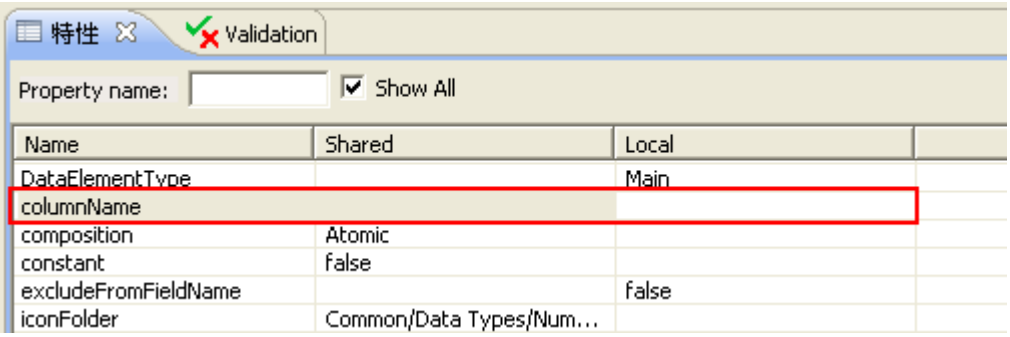
选择**数据 （ab） 类型/文本(Data Types)**模板并把它也放到**“申请单”**里面。把它命名为 **“内容”*(Description)***。

选择**内容->特性(properties)**，点击  把它命名为 **“columnName”**



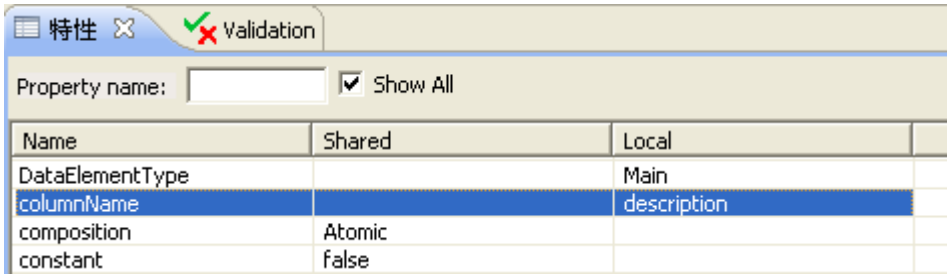


现在“特性”模型的外观应该是这个样子：

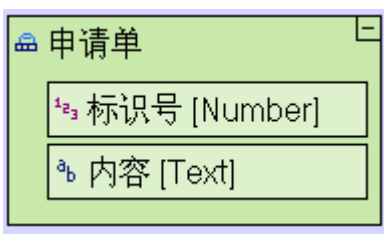


把 **“columnName”**命名为 **“description”**

现在“特性”模型的外观应该是这个样子：



“**申请单**”数据结构外观应与下图类似：



“申请单”数据结构现在里面有两个字段：一个是“**Id**”，这是每个申请单自动生成的独一无二的标识码，另一个是“**Description**”，这是用户可自由输入的申请单的文本描述。

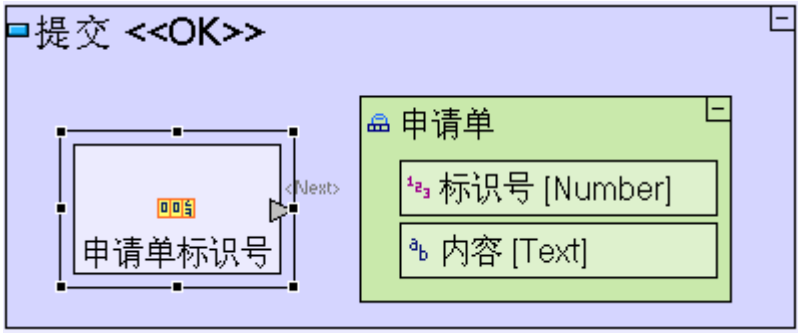
使用过程模板（Process Template）以生成记录标识码（Record identifier） Use a Process Template (to generate a record identifier)

正如之前解释过的，“**Id**”必须是一个自动生成的独一无二的标识码。“**Id**”必须独一无二，因为我们不想任何两个申请单有同一个标识码。

可以通过使用“**序列号**”（Sequence Number）模板做到这一点。

选择（）**数据库/序列号（Database/Sequence Number）**模板并把它放到“**提交**”(Submit)模型里面。把它命名为“**申请单标识号**”(**Requisition Id**)。

模型外观应与下图类似：



序列号是生成新的独一无二标识码的内置过程（动作action）一个例子。过程是正在被运行的东西。过程可以是Tersus平台提供的内置动作并可从选项板（Palette）中进行运用（如基本的数学计算），也可以是用户（您或其他人）建立的复杂的过程。
过程建模是生成解决方案必须用到的主要活动（解决方案由显示模型、数据模型和过程模型组成）。

过程（就当前的情况而言也就是**序列号**）生成输出结果（独一无二的标识码）时，输出结果需要“退出 exit”过程才能被另一个过程使用和填写数据结构的字段。建立“输出”槽（slot）模型是一个显示在过程模型边框上的灰色的三角形()。

“槽Slots”有两种主要类型：一种是过程启动时(后面会谈到这一点)接收数据的**触发器**(Triggers)，另一种是显示过程执行时生成的数据的**出口**(Exits)。

序列号(Sequence Number)模板通过预先定义好的**“<下一个>”**(Next)出口，输出生成的独一无二的标识码。

生成流程 Create a Flow

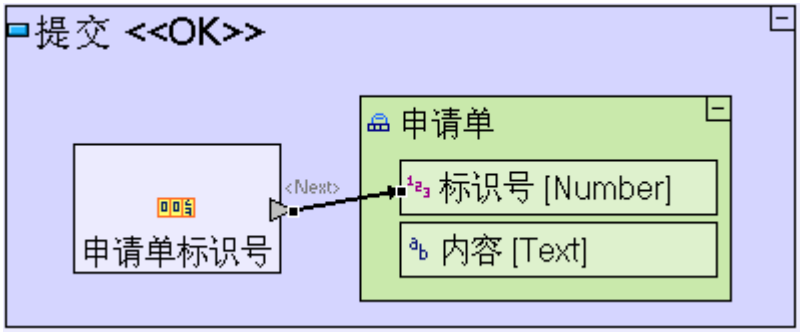
当然，我们还需要将这个生成的标识码从**序列号**过程的**“<下一个>”**(Next)出口传到**申请单（Requisition）**

数据结构的 **Id** 字段。要做到这一点，我们会用到**流程**（Flow）。

选择选项板（palette）最上面那一行的“**流程 Flow**”（  ）工具，点击 **“申请单标识号”（Requisition Id）**的**“<下一个>”（Next）** 出口槽(exit slot)（  ）以指定**来源**（Source），然后点击 申请单（Requisition）中的 Id 字段以指定**目标（Target）**，就会出现一个把槽和字段连接起来的箭头。

使用**流程**工具时，光标形状有所变化以表示当前位置是否可以作为流程定义的**来源**或**目标**。

现在“**提交**”模型的外观应该是这个样子：



流程通过两个模型单元（**来源**和**目标**）之间的箭头建立了模型。**流程**通过两种方式定义**来源**和**目标**之间的关系：

执行顺序(Order of Execution)：每个过程什么时候被执行（和取决于什么条件）。

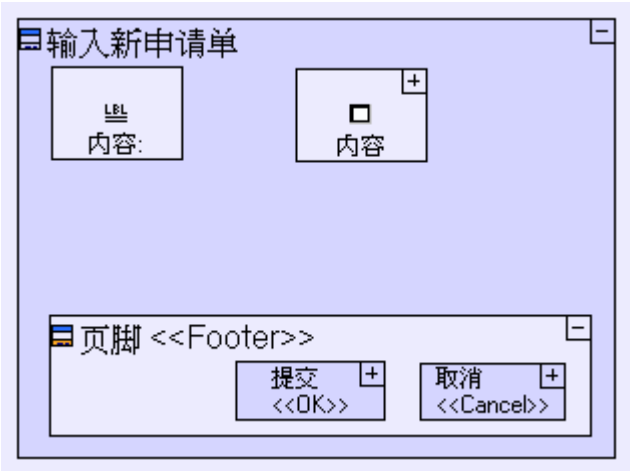
数据流动(Data Flow)：数据什么时候和如何传递。

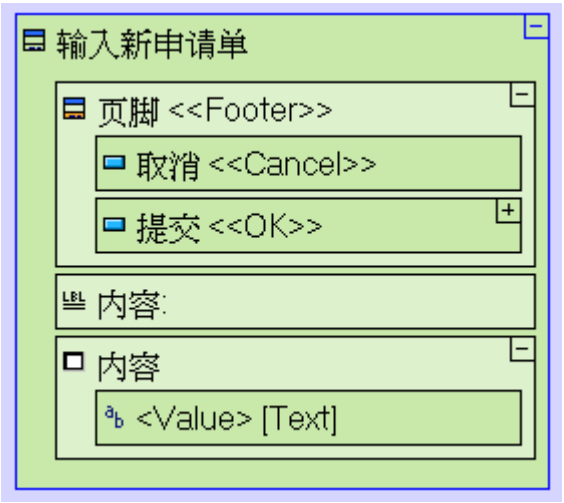
使用显示数据单元（Display Data Element）以提取用户输入结果 Use a Display Data Element (to retrieve uswer input)

现在我们已经在**申请单**数据结构中保存了自动生成的标识码，我们需要把**内容**（Description）保存在同一个数据结构中，毕竟**内容**才是我们感兴趣的真实信息。要做到这一点，我们将在名叫“**显示数据单元**”（Display Data Element）的模型中定义一种新型的单元。

作为数据结构，“**显示数据单元**”是**显示**单元（和**显示**单元的子单元）的另一种表现形式，它的主要作用是使用显示内容，从而使“**显示数据单元**”可以被读取或填写。

例如，我们可以看一下我们正在建模的弹出窗口的显示层次结构，这个层次结构包含一个标签(label)、一个文本区(text area)、一个页脚(footer)和两个按钮(2 bottons)（见“**显示”（Display）**的屏幕快照-以下左图）。弹出窗口的数据结构（包含文本区、按钮和标签的数据结构）代表这个层次结构；文本区包含文本区数值的另一个数据结构（见**“显示数据单元”（Display Data Element）**的屏幕快照-以下右图）。





显示数据单元的屏幕快照（以上右图）中显示单元的顺序可能与您看到的模型中的显示单元的顺序不一样。显示的顺序由单元添加的顺序所规定。只要所有的单元都显示出来（不管顺序如何），教程的运行时间（Runtime）动作都是一样的。

我们想看到“**内容**”(Description)文本区的内容，所以我们必须向“**提交**”(Submit)按钮添加一个指向“**输入新申请单**”(Enter New Requisition)弹出窗口的显示数据单元(Display Data Element)。由于“**输入新申请单**”是“**提交**”按钮的“父项”，所以我们进行“**添加原始参考数据**”(Add Ancestor Reference)操作：

右击“**提交**”按钮(right-click on submit button)，从菜单中选择“**添加原始参考数据**”(Add Ancestor Reference)并选择“**输入新申请单**”(Enter New Requisition)。

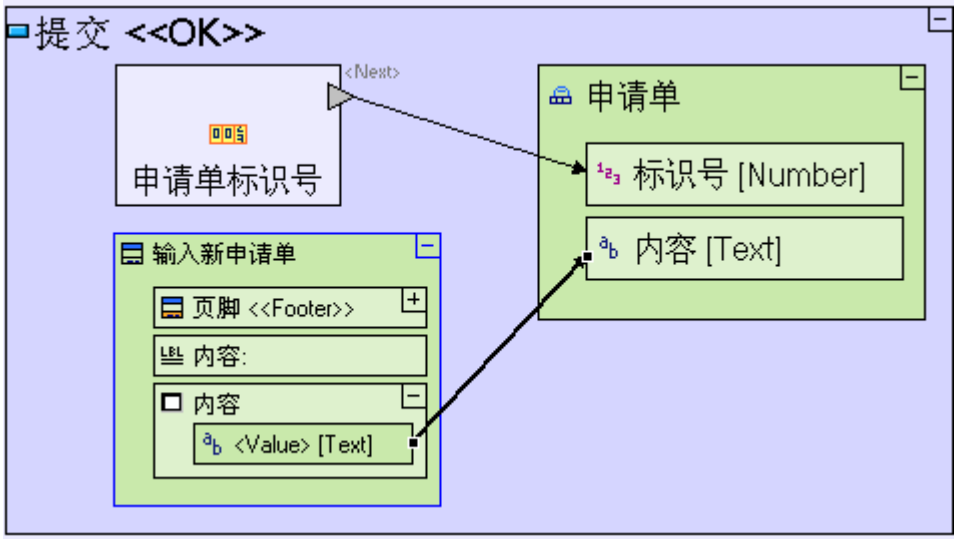
> 您可以看到插入的数据结构有一个蓝色的边框(其它普通的单元的边框为黑色)。边框为蓝色说明这个数据结构是另一个单元(“输入新申请单”弹出窗口)的参考数据，这样一个单元的数据有变化，另一个单元的数据也自动得到更新。

下一步您需要生成一个从“**输入新申请单**”内部的“**内容**”(Description)文本区到“**申请单**”数据结构中“**内容**”(Description)字段的流程。我们可在“**输入新申请单**”数据单元内部“**内容**”文本区中的<**数值**>“**<Value>**”数据单元中看到用户实际输入的文本：

使用“**流程**”(Flow)工具连接“**输入新申请单**”/“**内容**”/“**<数值>**”和“**申请单**”/“**内容**”。

> 大部分的显示单元都包含可处理的缺省数据单元。这些数据单元以一个放在括号“**<...>**”里面的描述性的名称(如“数值”)与其它数据单元区别开来。

现在“**提交**”模型的外观应该是这个样子：



我们总结一下这个模型到目前为止做了些什么：

“**Requisition Id**”分过程分成一个标识码，这个标识码被传送到“**申请单**”的“**Id**”字段，用户输入的“**内容**”文本区的内容被传送到“**申请单**”的“**内容**”字段。

使用插入模板（Insert Template）以将数据保存到数据库里面 Use the Insert Template (to Store data in the database)

现在我们已经生成了“**申请单**”的数据结构并填上相关数据，这些数据必须保存在数据库里面。要做到这一点，我们将用到另一个过程模板-“**插入**”（Insert）模板。

选择 “**数据库/插入**”(Database/Insert)模板并把它放在“**申请单**”数据模型旁边。

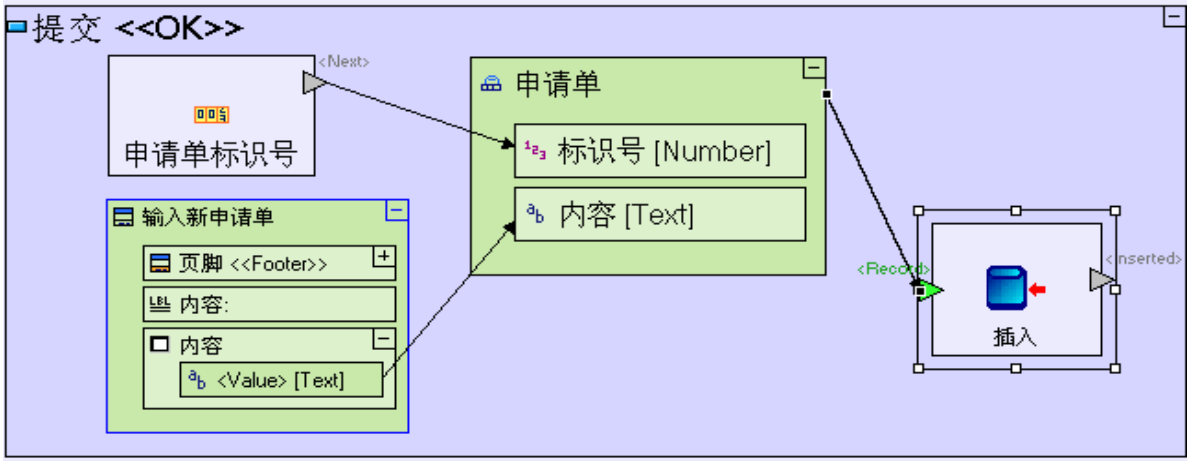
“**插入**”模板包含一个“**<Record>**”触发槽（trigger slot），“**插入**”模板通过这个触发槽接收保存在数据库中的数据结构。

当过程开始执行时，触发槽是过程接收输入数据的接入点。

要发送“**申请单**”的数据结构到“**插入**”，我们必须生成一个流程：

选择 “**流程**”(Flow)工具，接着点击“**申请单**”数据结构(在“**标识号**”(Id)”和“**内容**”(Description)字段以外的任何地方)，把它定义为流程的来源(source)，然后点击“**插入**”模型的“**<Record>**”触发器，把它定义为流程的目标(target)。

现在“**提交**”模型(Submit model)的外观应该是这样的：



最后一个修改的意义应该很清楚-“**申请单**”数据库记录数据结构作为数据库一个新的记录插入以数据库里面，插入的数据库表格名称与原来一样（**Requisition**）。如果表格在数据库中不存在，应用程序服务器将自动生成表格。

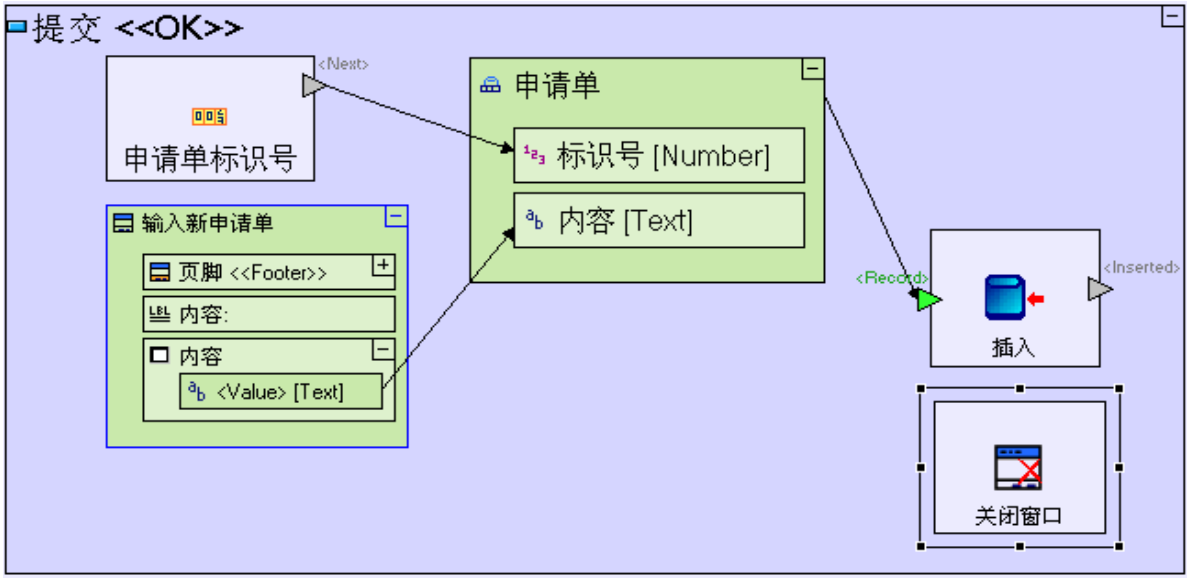
您可以看到，模板中许多槽名(slot names)（如以上屏幕快照中的“序列”（Sequence）和“插入”（Insert）都使用特定的命名规定，并且名称在括号“**<…>**”里面）。模板功能可能与一些保留名称绑定在一起。如果保留名称有所变化，模板可能失效或运作不正常。

使用关闭窗口模板（Close Window Template）和确保过程发生顺序正确 Use the Close Window Template (and make sure processes occur in the right order)

一旦申请单输入数据库中，弹出窗口就会关闭：

选择 **“显示动作/关闭窗口”**（Display Action/Close Windows）模板并把它放到**“提交”**（Submit）按钮里面。

现在**“提交”**模型外观应该是这样的：



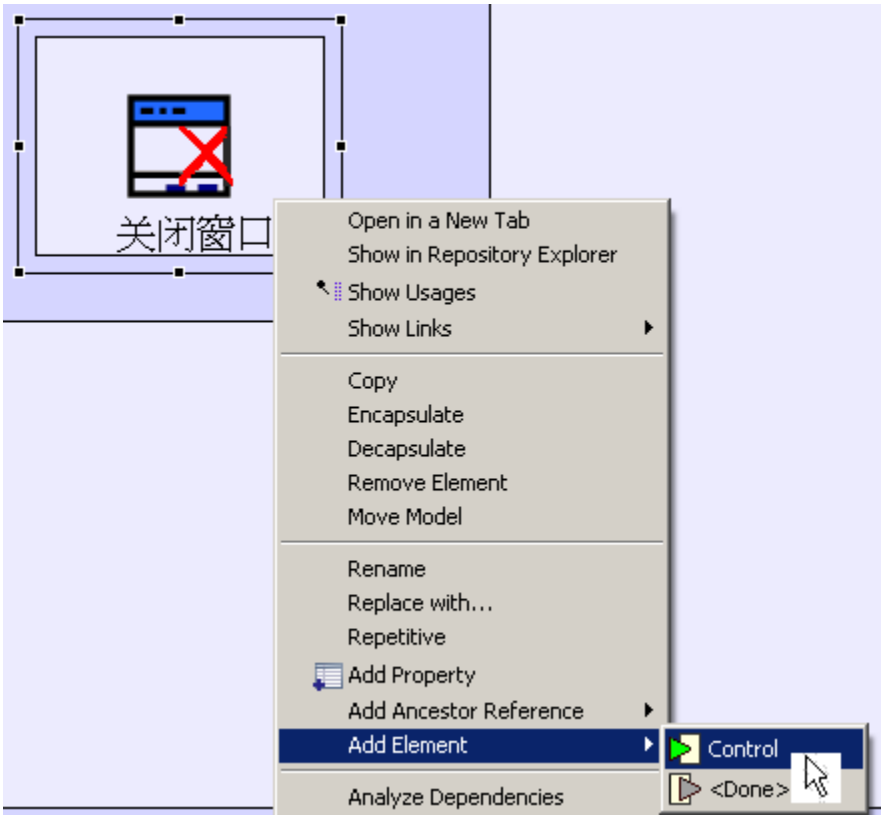
“提交”模型有两个部分，这两个部分发生的顺序没有规定：一个是**“关闭窗口”**(close window)过程，另一个是把申请单数据保存在数据库里面的一系列过程。

由于过程顺序是应用程序逻辑的一个基本的部分，有时必须指定哪个过程先发生。而在其它情况下，顺序并不重要，但确保过程发生以规定好的顺序发生仍是一种好的做法（虽然相对来说没那么重要）。

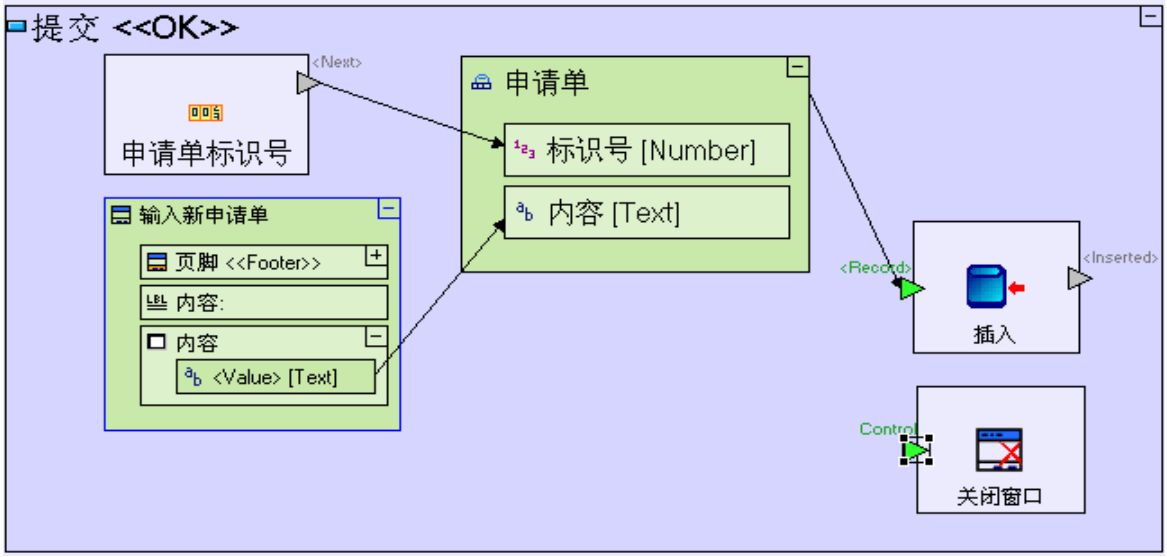
我们想确保只有保存数据之后才关闭**“输入新申请单”**(Enter new requisition)弹出窗口，所以我们应该定义这样一个过程：规定**“关闭窗口”**过程在**“插入”**过程后面发生并且**“关闭窗口”**过程发生的前提是**“插入”**成功。

为了做到这一点，我们必须首先添加一个**“关闭窗口”**的触发器。我们可以通过添加一个从选项板（palette）中选择的触发槽来做到这一点，我们也可以通过利用**“关闭窗口”**“隐藏”了一个预先定义好的、正好有这个作用的触发器来做到这一点：

右击**“关闭窗口”**单元，打开**“增加单元”**（Add Element）分菜单，然后选择 **“控制”**（Control）触发单元。



现在“**提交**”模型外观应该是这样的：



由于“**关闭窗口**”（Close Window）模板作为“**原型**”（Prototype）被执行，所以系统提供“隐藏的”“控制”（Control）触发器。

原型规定组成一个特定模型的单元和特定模型的哪些部分是必需的。在“**关闭窗口**”的情况下，不一定要有触发器“**关闭窗口**”才能起作用（正如弹出窗口中的“**取消**”(Cancel)按钮所展示的一样）；但在在部分使用情况下（如我们刚执行的“**关闭窗口**”模板），触发器是必需的，所以系统提供触发器，作为“**关闭窗口**”原型的可选单元。

应该指出的是，其它模板也作为原型被执行。例如：“**插入**”(Insert)的原型包括一个“隐藏的”出口（hidden Exit）和**<重复(Duplicate)>**（在插入的记录有一个重复的键时很有用，在这种情况下，也就是如果“**申请单**”数据结构含有一个已经在数据库中的“**ID**”）。但由于我们的模型不会发生这种情况（我们使用**序列号**（Sequence Number）模板生成独一无二的“**ID**”，所以**<Duplicate>**可能一直保持“隐藏”状态。

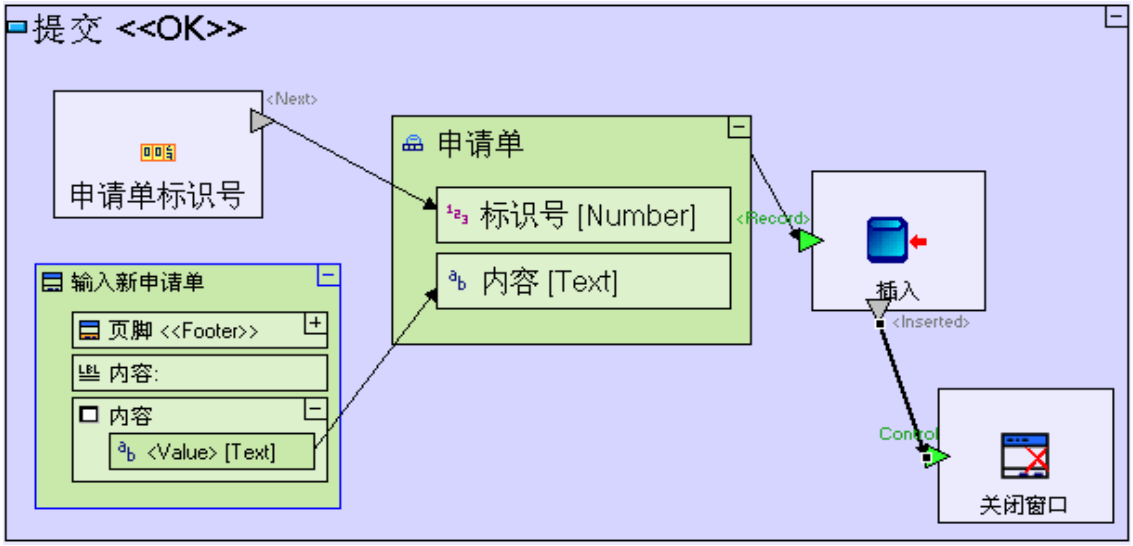
必须指出，“**增加单元**”（Add Element）分菜单中看到的**所有**单元仅用于“快捷方式”(shortcut)

的用途，这一点很重要。实际上每个单元都可以人工创建(manually)（在特殊情况下，可通过把选项板上的触发器拖到“**关闭窗口**”，然后把它命名为“**Control**”），并且每个单元都可以用。

我们希望记录插入到数据库里面之后才关闭窗口（如“插入”（Insert）成功然后从<**Inserted**>出口退出）：

选择“**流程**”（Flow）工具以连接**Insert**的<**Inserted**>出口和“**关闭窗口**”（Close Windows）的“**控制**”（Control）触发器。

现在“**提交**”模型外观应该是这样的：



您可能注意到前面的屏幕快照中的**Insert**/<**Inserted**>已从缺省位置删除。删除的原因是为了使模型对于建模者来说更具有可读性，删除之后对模型的功能没有什么影响。

保存您的工作成果。

想在浏览器中查看应用程序：

点击 **“启动应用程序”**(launch the application)工具栏按钮(toolbar button)。

尝试输入申请单。

在这个阶段，我们还看不到数据库的内容，所以不能确认申请单是否已经保存到数据库里面。我们在后面会建立这方面的模型。

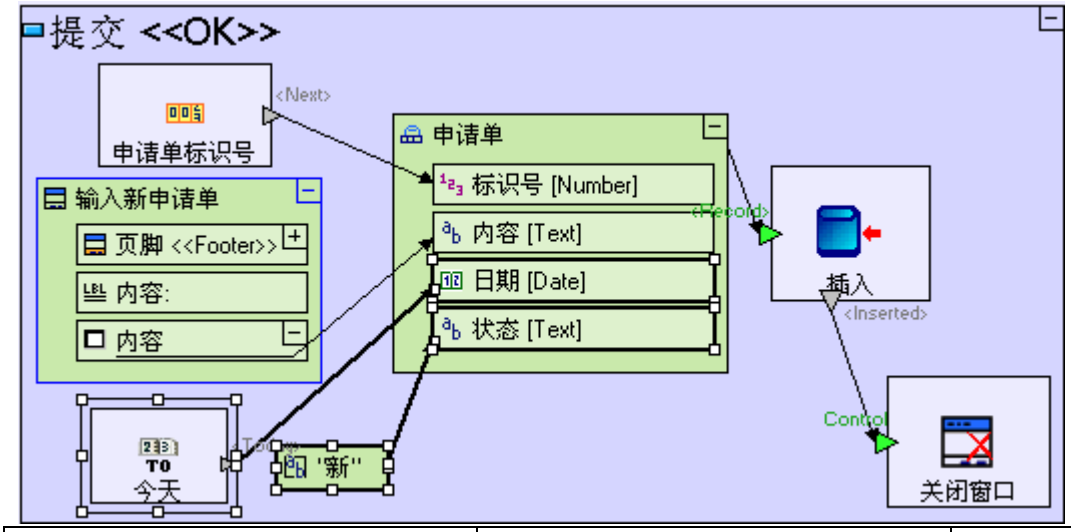
如果您想做到这一点的话，您可以使用外部工具（通常是DBMS供方提供的工具），以确定数据库里面是否已创建了申请单表格并查看申请单表格(requisition table)的内容。

完成阶段 3　Completing Stage 3

输入项目样本“**Tutorial 3-4**”并把它作为教程下阶段的基础。

想了解怎样输入项目样本，请参阅**阶段2**最后“**输入项目样本**”（Importing a Sample Project）部分。

项目样本还包括以下其它功能：



功能　　Functionality	怎样建模 How to Model	位置 Located in
添加字段到“**申请单**”表格里面以供将来使用	添加“**日期**”(Date)字段（拖住“**数据类型/日期**”（Data Types/Date）模板） 选择**日期->特性(properties)**， 点击  把它命名为 **“columnName”** 把 “columnName”命名为 **“date”** 添加“**状态**”(status)字段（拖住“**数据类型/文本**” 模板(Data Types/Text template） 选择**状态->特性(properties)**，点击  把它命名为 **“columnName”** 把 “columnName”命名为 **“Status”**	**申请单 Requisition** 数据结构 data structure

设定申请单“**日期**”字段的缺省值（当前日期）	拖住“**日期/今天**”(Dates/Today)模板 定义从“**今天/<Today>**”（Today/<Today>）到“**申请单/日期**”(Requisition/Date)的流程	“**提交**”（Submit）按钮
设定申请单状态字段的缺省值“**New**”	拖住“**常数/文本**”（Constants/Text）模板 把它命名为 “**新**”(**New**) 定义从“**New**”到“**申请单/状态**”(Requisition/Status)的流程	“**提交**”（Submit）按钮

我们已提取用户输入的资料并把它保存到数据库里面，从而完成了幕后运行逻辑的建模过程。

现在您可进入**阶段 4**，在**阶段 4** 里面，我们将运行保存在“**申请单**”表格中的所有申请单的表格显示（Table display）。

实例 See it live

点击此处在另一个窗口里面打开实际项目。

阶段 4-建立简单表格显示（Table Display）模型 Stage 4-Modeling a Simple Table Display

阶段目标 Stage Goals

覆盖的 Tersus 概念 Tersus Concepts Covered

完成本阶段后，您将会熟悉以下概念：
建模概念： 重新使用、过程
建模技术： 从数据库中提取数据，把数据显示在表格里面
有用的过程模板： 动作（Action）、查找（Find）
有用的显示模板： 简单表格

建立应用程序功能模型 Application Functionality Modeled

在这个阶段里面，您将为从数据库提取出来的一些申请单的显示建模。
建成的网页应用程序外观如下：

本阶段建模在上个阶段最后输入的项目“Tutorial 3-4”中进行。

建立数据表格显示（Tabular Display）模型 Model a Tabular Display

在之前几个阶段里面，我们学习了怎样显示用于输入数据的表单并把输入的数据保存在数据库里面。

下一步我们想以表格形式（一行一个记录one row per record）查看之前输入的数据。

表格将是我们之前创建的“**打开申请单**”（Open Requisitions）视图的一部分。

根据以下任一项操作进入“**打开申请单**”视图：

在编辑器里面双击“**打开申请单**”视图

或

在大纲（outline）中找到“**打开申请单**”视图并双击“**打开申请单**”视图

表格位于“**新申请单**”（New Requisition）按钮下面，确保“**打开申请单**”模型中有足够的空间可以放我们即将生成的新模型。

调整“**新申请单**”按钮大小并把它放在“**打开申请单**”视图上面。

生成表格显示 Create a Table Display

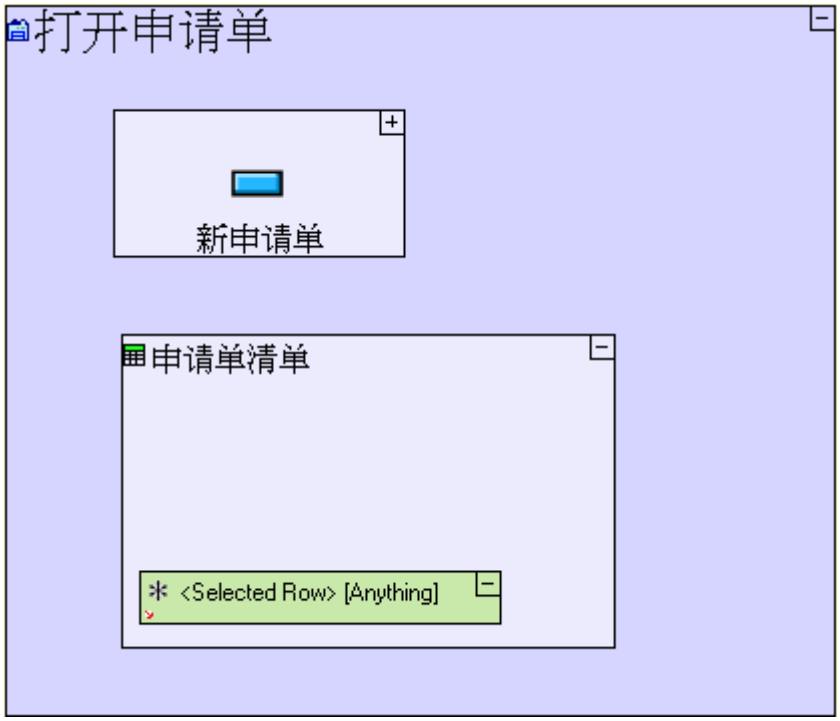
现在我们从向“**打开申请单**”（Open Requisitions）中添加一个“**表格显示**”（Table Display）开始。

从选项板中选择 “**显示/简单表格**”（Display/Simple Table）模板并把它插入到“**打开申请单**”（Open Requisitions）中。

把它命名为 “**申请单清单**”(**Requisition List**)。

> 在缺省条件下，“**简单表格**”(simple table)模板包括一个<**Select Row**>单元。我们现在先不理会这个单元，但我们在本教程**阶段6**中将会用到这个单元。

现在“**打开申请单**”视图模型外观应与下图类似：

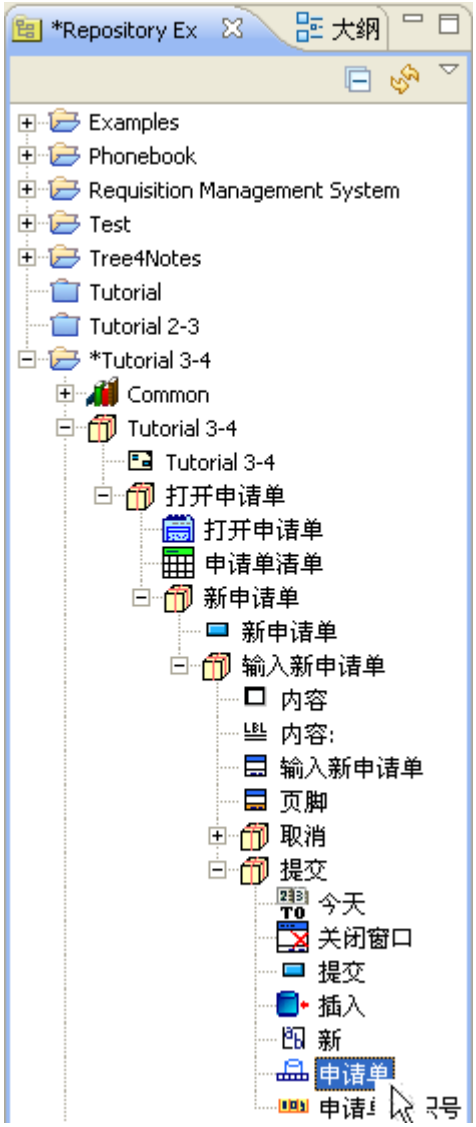


重新使用数据结构定义表格内容 Reuse a Data Structure to Define Table Contents

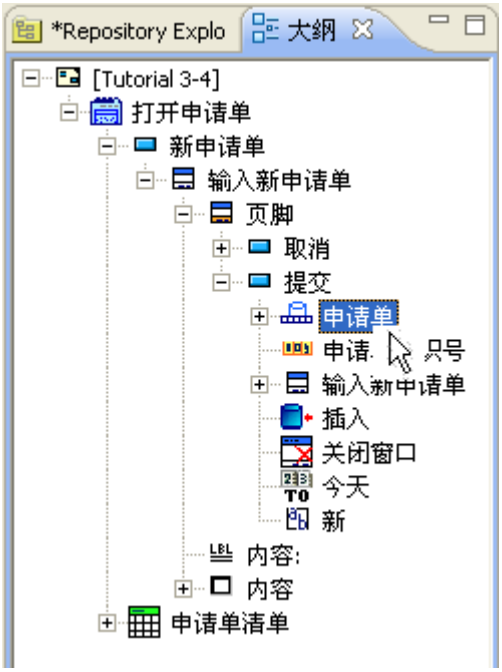
我们希望表格尽可能简单并显示“**申请单**”的所有字段（**Id, Description, Date, Status**），所以我们只**重新使用“申请单”**的数据结构：

> 重新使用之前定义好的模型是建模过程一个不可分割的部分，可大大提高开发速度和模型的清晰性。重新使用模型时，对模型所做的任何修改都会影响模型运行的动作，这样就确保了模型的一致性。您可以重新使用任何类型的模型（数据模型、显示模型和过程模型）。

在“**储存库**”(repository)视图中找到“**申请单**”的数据结构。

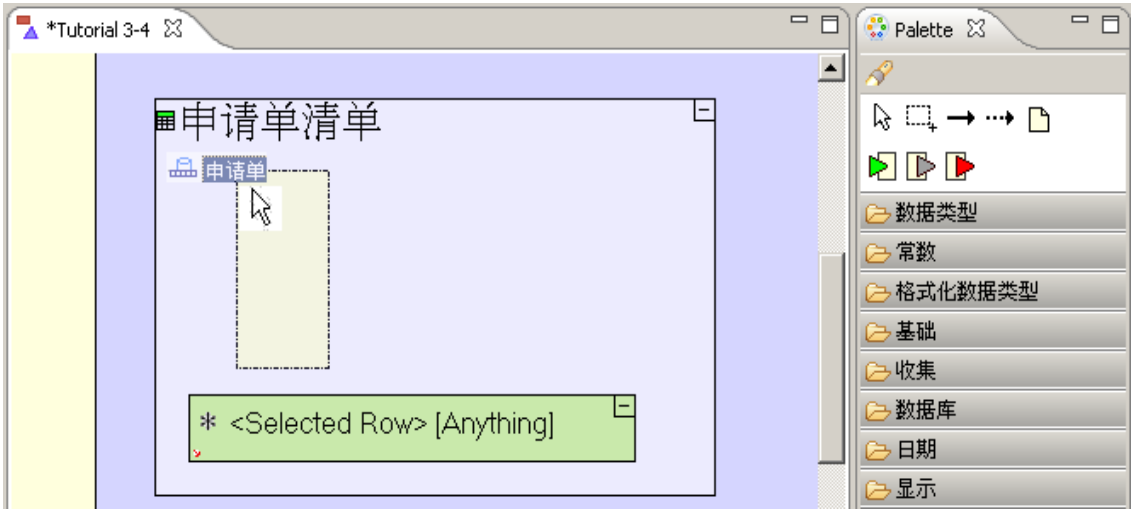


或在“**大纲**”(outline)视图里面

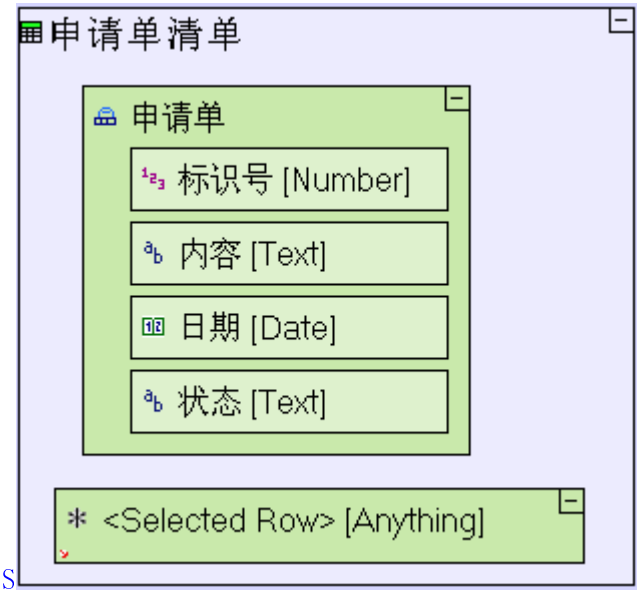


放大到“**申请单清单**”表格。

把“**申请单**”表格拖放到模型编辑器里面。



把“**申请单**”表格拖放到“**申请单清单**”单元里面。

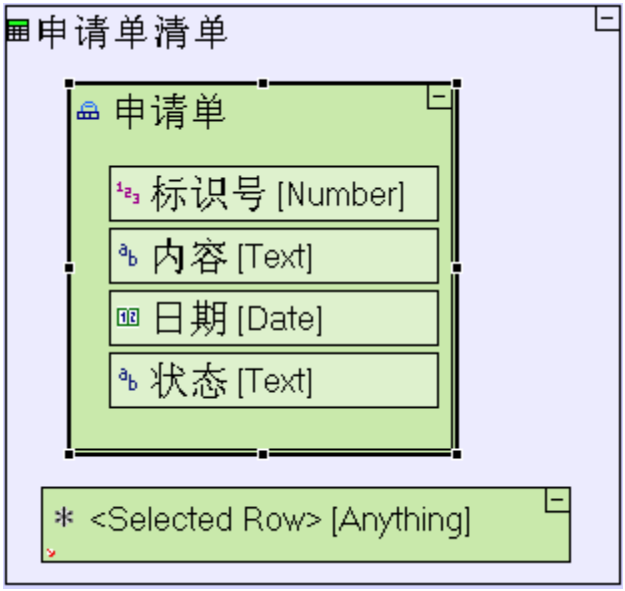


现在由于表格行数不止一行，每一行都有同样的数据结构（“**申请单**”（**Requisition**）），我们需要把数据结构定义为“重复”：

右击刚创建的“**申请单**”（Requisition）单元，从菜单中选择“**重复**”（repetitive）。

您可以通过进行以下操作，确认“**申请单**”单元是否已定义为“重复”：
1. 右击“申请单”单元，确认是否已勾选“重复”。
2. 勾选“属性”（Properties）窗格里的“重复”属性。
3. 在模型编辑器里面确认“申请单”单元是否重叠。（见以下屏幕快照）

“**申请单清单**”表格模型现在外观应与下图类似：



保存您的工作成果并在浏览器中查看应用程序。应用程序外观应与下图类似：

您可以看到“**申请单清单**”表格里面没有什么内容（只有每一列的标题，表格里面没有行列）。我们还需要建立从数据库中提取数据的模型以填写“**申请单清单**”表格的行列。

使用动作模型定义过程（以数据库数据填写显示表格） Use Action Model to Define a Process(populating the display table with data from the database)

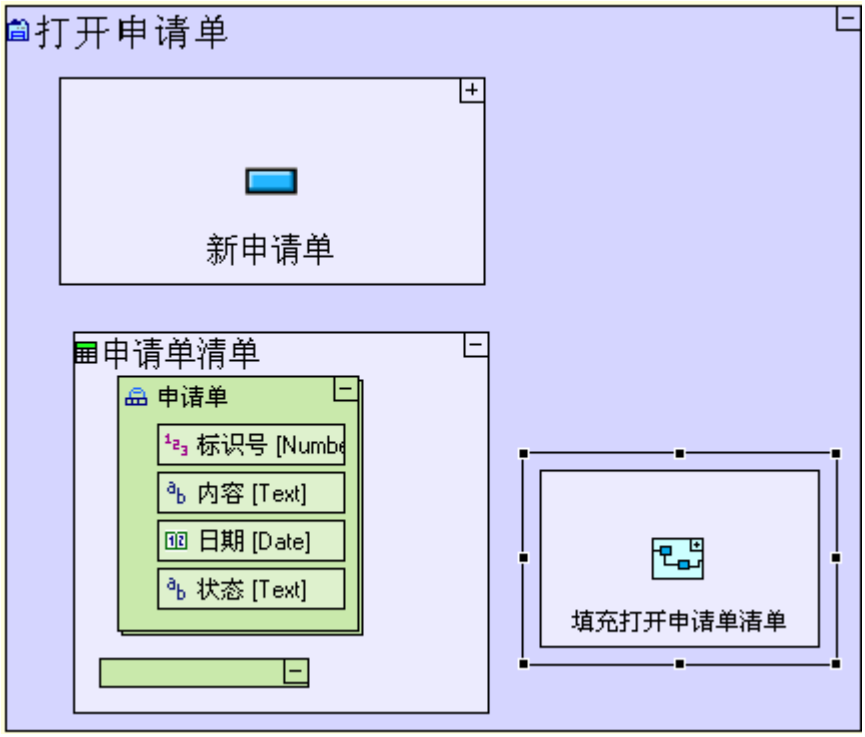
我们现在建立从数据库中提取数据的模型并用这些数据填写表格：

选择“基础/动作”（Basic/Action）模板（  ），把它拖放到“**打开申请单**”（Open Requisitions）模型里面。

把它命名为“**填充打开申请单清单**”（Populate open Requisition List）。

> “**动作**”（Action）模板是定义复合过程的容器。
> 由于“**填充打开申请单清单**”没有定义触发器，运行“**填充打开申请单清单**”的父项（“**打开申请单清单**”）时，“**填充打开申请单清单**”会自动运行。
> 选择“**填充打开申请单清单**”这个名字因为这个名字本身已经很清楚，不需要进行解释。“**填充打开申请单清单**”在运行时间没有什么作用，您也可以用其它名字，这样做对应用程序的运行没有什么影响。

“**打开申请单**”视图模型现在外观应该是这个样子：



您可以看到“**填充打开申请单清单**”不是一个显示模型，而是一个过程模型。它不是表格显示的一部分（这一点和“**申请单清单**”的“**申请单**”单元有所不同），而是为了使表格能够正确显示所进行的一些处理。
正如上面所解释的，它没有触发器，所以可自动运行，所以我们把它称为“初始化过程”。任何一个显示模型都可包含这种初始化过程。

“**填充打开申请单清单**”（Populate open Requisition List）将会生成“**申请单清单**”数据单元（补充从数据库读取的“**申请单**”），然后生成的“**申请单清单**”就会被发送到显示（display）。

定义子过程（在内存中生成表格数据单元）Define a Sub-Process(populating the display table with data from the database)

要生成“**申请单清单**”数据单元，我们必须给“**填充打开申请单清单**”过程生成一个子过程：

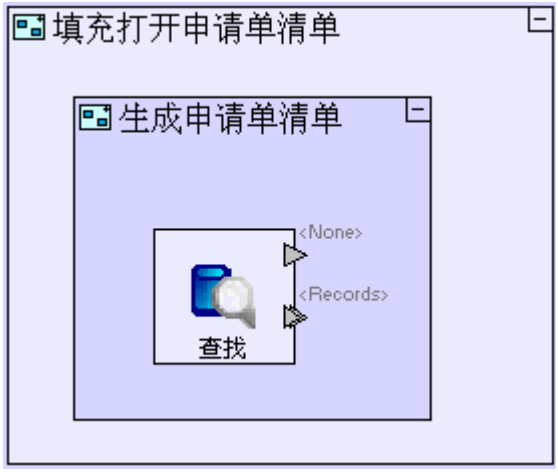
放大到“**填充打开申请单清单**”。
选择基础/动作（Basic/Action）模板（  ），然后把它拖放到“**填充打开申请单清单**”模型中。
把它命名为“**生成申请单清单**”。

使用“查找”模板（以定义数据类型和生成表格数据单元）Use the Find template (to retrieve data from the database)

“**生成申请单清单**”(**Generate Requisition List**)启动并从数据库提取数据，我们可通过使用“**查找**”(Find)模板运行“**生成申请单清单**”。

选择**数据库/查找**（Database/Find）模板（  ），然后把它插入到“**生成申请单清单**”里面。

“**填充打开申请单清单**”动作模型现在外观应该是这个样子：



“**查找**”模板包括两个出口：<**None**>和<**Record**>，根据“**查找**”(Find)是否找到任何记录，这两个出口可被“激活”(activated)（并显示输出结果）。如果没有找到任何记录，<**None**>出口就会激活。如果找到至少一个记录，就会通过<**Record**>出口把这些记录显示出来。

如果您仔细看一下<**Record**>出口，您就会看到<**Record**>出口被标识为“重复”（  ），这意味着“**查找**”可以输出多个记录。

我们还需要规定“**查找申请单**”向哪个数据库表格提取记录。我们可以通过规定<**Record**>出口的**数据类型**(Data Type)（即出口输出的数据结构）做到这一点。下一步我们就对此进行规定。

使用显示数据单元（以定义数据类型和生成表格数据单元）Use the Display Data Element(to define the data type & create the table data element)

“**查找**”提取的记录应该保存在正确的数据结构里面，在当前情况下，正确的数据结构是指代表我们之前创建的“**申请单清单**”显示。

从大纲中拖住“**申请单清单**”模型，把它放到“**查找申请单**”单元旁边。

这样就会生成代表“**申请单清单**”模型的“**申请单清单**”显示单元。

> 您可以看到，我们刚创建的“**显示数据单元**”(Display Data Element) 与我们通过“**添加原始参考数据**”（Add Ancestor Reference）功能创建的显示数据单元（见阶段3）类似，但不相同。
> 类似之处在于两者都是显示模型的数据表现形式。区别之处在于通过“**添加原始参考数据**”，您创建了实际运行时间显示的参考资料（从而使您能使用显示的数据），而在当前的情况下，我们在内存中创建一种新的与显示有同样的结构的显示数据单元，但这种显示数据单元不是实际运行时间显示；或者说，我们创建了一种逻辑（或在内存中in-memory）的显示数据单元。

下一步我们添加从“**查找申请单**”(Find Requisition）提供的记录中生成“**申请单清单**”表格的**流程**：

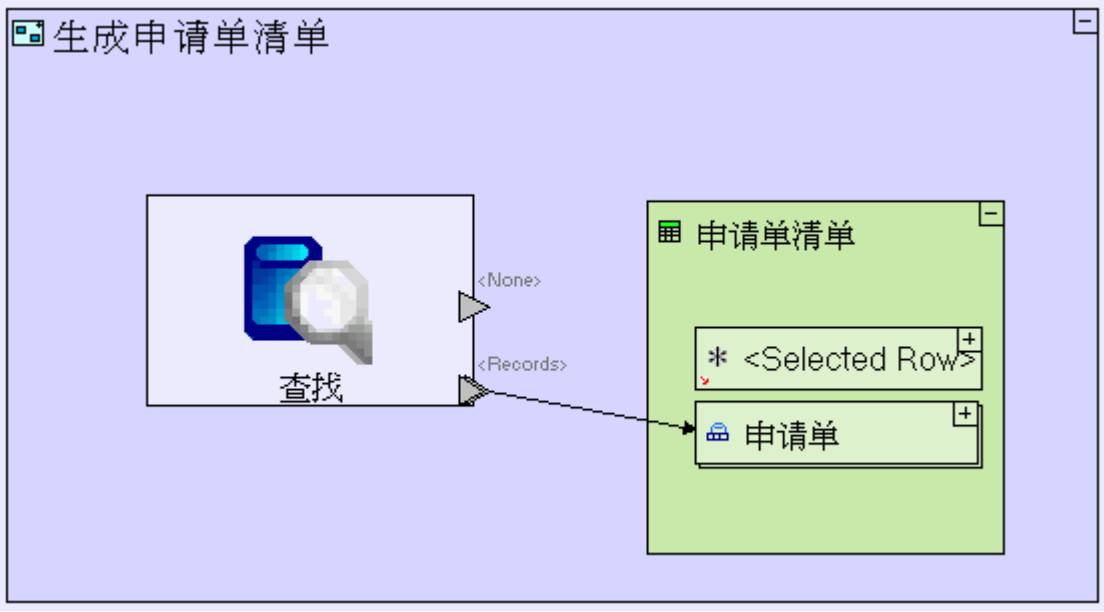
扩展“**申请单清单**”单元（通过点击“**申请单清单**”单元右上角的⊞）以显示其内容。

选择“**流程**”(Flow）工具以连接“**查找**”(Find）的<**Records**>触发器和重复的“**申请单清单**”中的“**申请单**”数据单元。

> 通过创建以上流程，您做了两件事情：

1. 清楚地规定了“查找”输出到哪里（“**申请单清单**”表格的“**申请单**”单元）。
2. 间接规定了“**查找**”<**Records**>出口的数据类型（如“申请单”）。由于“**申请单**”数据类型是建立在“**数据库记录**”模板的基础上，“**查找**”知道从哪个数据库表格进行查找（“**申请单**”表格）。

“**生成申请单清单**”（**Generate Requisition List**）动作模型现在外观应该是这个样子：



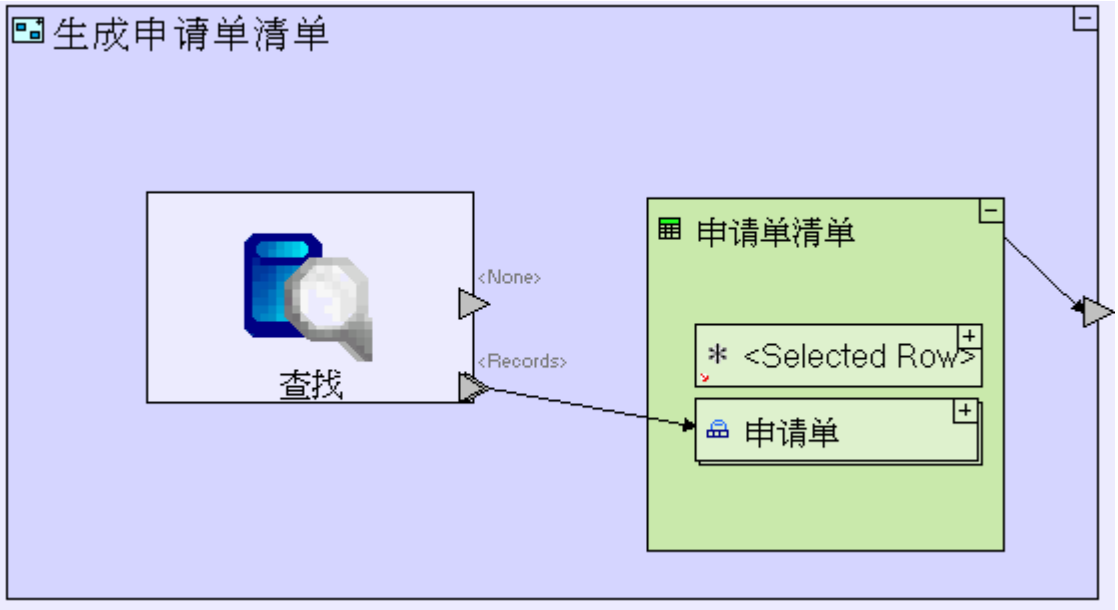
从子过程输出数据 Output Data from the Sub-process

创建“**生成申请单清单**”子过程的目的是用来输出数据（“**申请单清单**”显示数据单元），所以我们需要规定输出什么数据和输出的资料保存在什么地方。

从选项板（pallet）选择“**出口**”（Exit）槽（  ），点击“**生成申请单清单**”（Generate Requisition List）单元的边框。

选择“**流程**”（Flow）工具，把“**申请单清单**”（Requisition List）显示数据单元和我们添加的“**出口**”（Exit）槽连接起来。

“**生成申请单清单**”子过程现在的外观应该是这个样子：



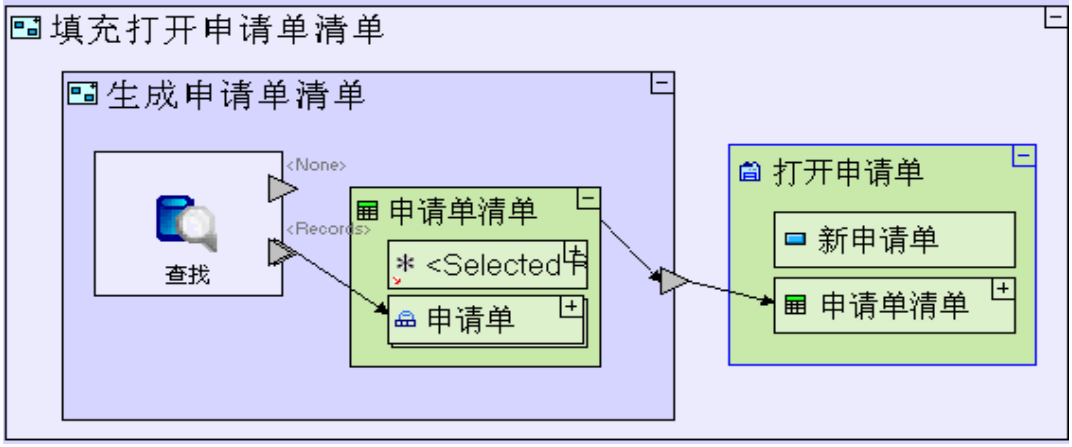
使用显示数据单元（把数据输出到显示）Use a Display Data Element (to output data to the display)

有上个阶段里面，我们学习了怎样使用“**原始参考数据**”（Ancestor Reference）显示数据单元从显示中提取用户输入的资料。我们现在使用同一种技术进行相反的操作;如把数据输出到显示。

进入“**填充打开申请单清单**”(Populate open Requisition List）单元。
右击“**填充打开申请单清单**”单元，从菜单中选择“**添加原始参考数据**”（Add Ancestor Reference）,然后选择“**打开申请单**”（Open Requisition）。

现在我们可以使用我们创建的“**打开申请单**”原始参考数据作为“**生成申请单清单**”发送输出表格的目标。
双击“**打开申请单**”数据单元以放大到**生成申请单清单**”(Generate Requisition）。
扩展“**申请单清单**”(Requisition List）单元（通过点击单元右上角的标记）以显示其内容。
选择“**流程**”(Flow）工具，把“**生成申请单清单**”(Generate Requisition）出口和“**打开申请单/申请单清单**”(Open Requisitions/Requisition List）连接起来。

“**填充打开申请单清单**”模型现在外观应该是这个样子：



保存您的工作成果并在浏览器中查看应用程序。应用程序外观应与下图类似：

表格里面显示的数据是您之前输入的数据（或与 **Tutorial 3-4** 应用程序样本捆绑在一起的数据）。
现在尝试输入一个新的申请单：

点击“**新申请单**”（New Requisition）按钮。

在“**输入新申请单**”（Enter New Requisition）弹出窗口中输入“**内容**”（Description），然后点击“**提交**”（Submit）。

您可以看到，虽然数据已经保存，但还没有更新。您可以通过刷新浏览器进行确认。

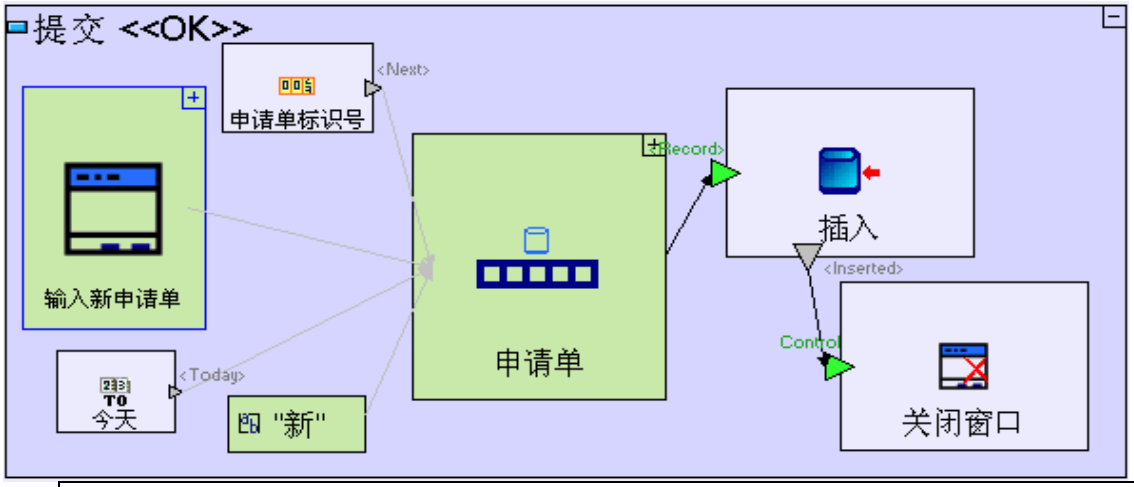
重新使用动作过程（以刷新表格显示）Reuse the Action Process (to refresh the table display)

我们不能期望用户人工刷新浏览器显示；“新申请单”提交之后，表格必须自动刷新。
刷新显示意味着以最新数据填写“**新申请单清单**”，这正是我们刚建立的模型所能做到的，这样我们就能够重新使用“**填充打开申请单清单**”（Populate open Requisition List）过程。

由于“新申请单”提交之后必须自动进行刷新，我们必须在“**提交申请单**”（Submit Requisition）按钮内部建立模型：

进入上阶段我们创建的“**提交**”（Submit）按钮。

“**提交**”模型现在外观应与下图类似：



上图的屏幕快照出现一些浅灰色的箭头(指向“**申请单**”(Requisition)),这表示流程的来源(source)或目标(target)没有显示出来(因为它们的父项已崩溃)。

成功插入申请单之后,我们给按钮添加一个新的动作过程,以进行所需的刷新:

选择“**基础/动作**”(Basic/Action)模板(),然后把它拖放到“**提交申请单**”(Submit Requisition)里面。

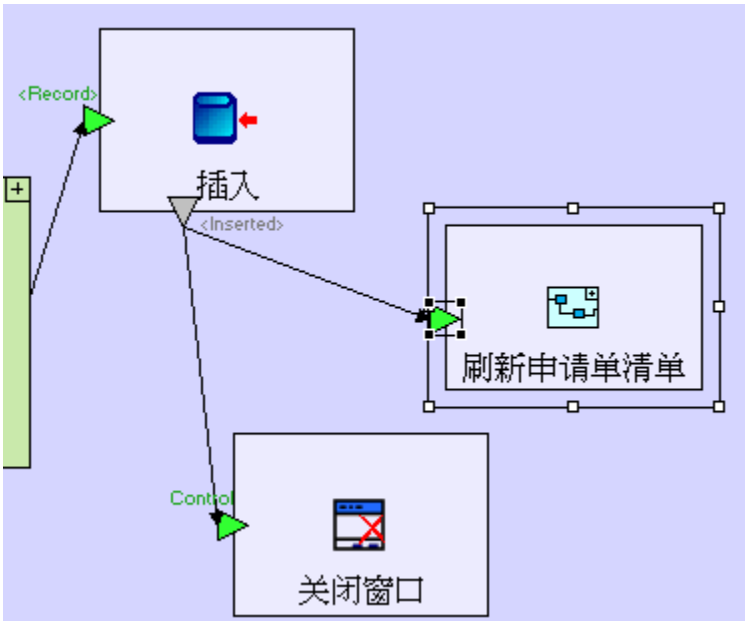
把它命名为“***刷新申请单清单***” ***Refresh Requisition List***

从选项板(pallet)选择“**触发**”槽(Trigger slot)(),点击“**刷新申请单清单**”单元的边框。

选择“**流程**”(Flow)工具,把“**插入**”(Insert)的**<Inserted>**槽和“**刷新申请单清单**”的“**触发**”槽连接起来。

您可以看到,向“**刷新申请单清单**”添加触发器是添加控制过程发生顺序的触发器的一个可选的方法。我们也在添加“**关闭窗口**”单元时(见**阶段3**)介绍了其它方法(使用“**添加单元**”情景菜单选项)。这也说明“**控制**”(Control)这个名字本身对触发器起作用的方式没有什么影响。它只是使模型更具有可读性(通过名字说明触发器的作用)。

模型现在外观应与下图类似:

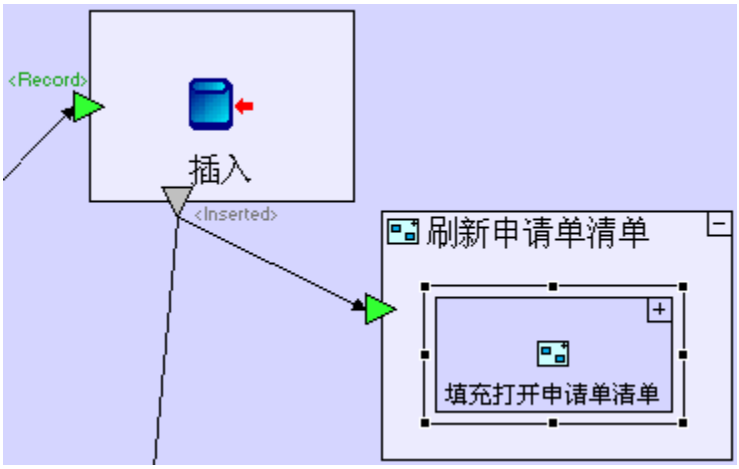


注意：槽可作为多个流程定义的来源/目标。在当前这种情况下，成功退出“插入”模型时，就会同时运行“**关闭窗口**”(close window) 和“**刷新申请单清单**”(refresh requisition list)。

现在我们**重新使用“填充打开申请单清单”**(Populate open Requisition List)过程：

从储存库（Repository）或大纲（outline）中拖住**“填充打开申请单清单”**并把它放到**“刷新申请单清单”**里面。

“刷新申请单清单”模型现在的外观应下图类似：



您可能会问为什么在另一个过程里面（**“刷新申请单清单”**）“包装wrapped”“ **填充打开申请单清单**”，而不是直接把它放在**“提交”**(Submit)按钮模型里面然后再从“插入”（Insert）创建指向它的流程。原因是如果是那样的话，我们就需要添加一个**“填充...”**（Populate）过程的触发器，同时由于过程在**“打开申请单”**视图里面重新使用，触发器也必须出现在**“打开申请单”**视图里面，那样的话触发器就不能起作用，因为在那种情况下触发器接收不到任何流量。

保存您的工作成果并在浏览器里面查看应用程序。

尝试输入新的申请单看它们有没有出现在清单里面。

完成阶段 4 Completing Stage 4

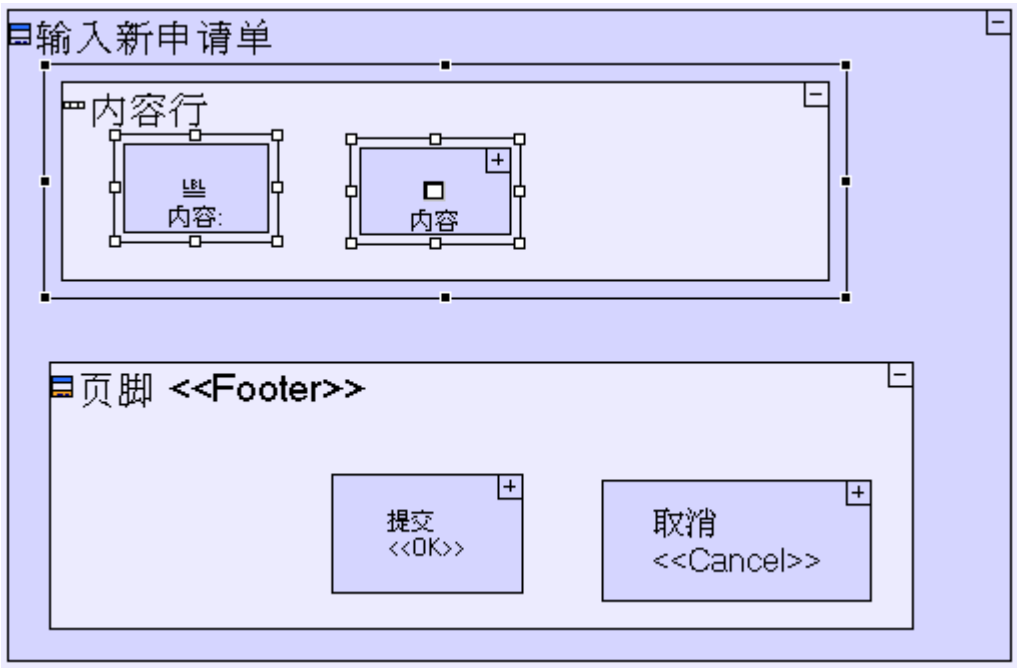
输入项目样本“**Tutorial 4-5**”，把它作为教程下阶段的基础。

> 想再了解如何输入项目样本，请参阅**阶段2**最后的“**输入项目样本**”（Importing a Sample Project）部分。

这个样本项目包含了到目前为止建立的模型的所有的功能。

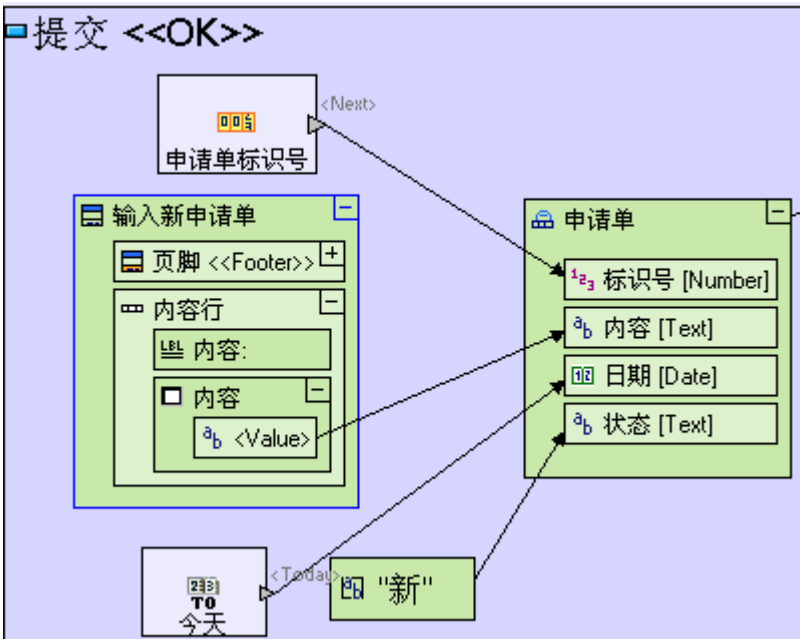
这个样本项目还包含以下其它功能：

1. 把“**内容**”（Description）文本区和标签放到行单元（row element）里面以改善弹出窗口的格式。



怎样建模 How to Model	位置 Located in
添加“**显示/行**”（Display/Row）。把它命名为“**内容行**”（Description Row）。	“**输入新申请单**”（Enter New Requisition）弹出窗口
从储存库（Repository）/大纲（outline）中拖住“**内容**”（Description）文本区。 从储存库（Repository）/大纲（outline）中拖住“**内容**”:（Description:）标签。	“**内容行**”（Description Row）
删除“**内容**”（Description）文本区。 删除“**内容**”:（Description:）标签。	“**输入新申请单**”（Enter New Requisition）弹出窗口

2. 把“流程”（flow）指向“**提交申请单**”（Submit Requisition）模型里面



怎样建模	位置
点击红色的流动箭头（red flow arrow）。把来源（source）拖到“**输入新申请单/内容行/内容/<数值>**”（Enter New Requisition/Description Row/Description/<Value>	“**提交申请单**”（Submit Requisition）按钮

您现在可以开始学习**阶段 5**。在**阶段 5**，我们将向申请单添加一个支持性的“紧迫性”字段。

实例 See it Live

点击此处以在单独的窗口里面打开实际项目。

阶段 5-建立选取器模型和使用常数 Stage 5 - Modeling Choosers and Using Constants

阶段目标 Stage Goals

覆盖的 Tersus 概念 Tersus Concepts Covered

在本阶段完成时，你将会熟悉以下概念：

建模概念： 常数（constant）

建模技术： 创建一个下拉式选单（drop-down list）

更新表格结构（updating a table structure）

有用的显示模版： 行（Row）、选取器(Chooser)

建立应用程序功能模型 Application Functionality Modeled

在这一阶段您将为添加字段到现有的数据结构（和相应的数据库表格）里面建模。

新字段会有一些预先定义好的数值,用户可以从下拉式选单中自行选择。

本阶段建模在上个阶段最后输入的“Tutorial 4-5”项目中进行。

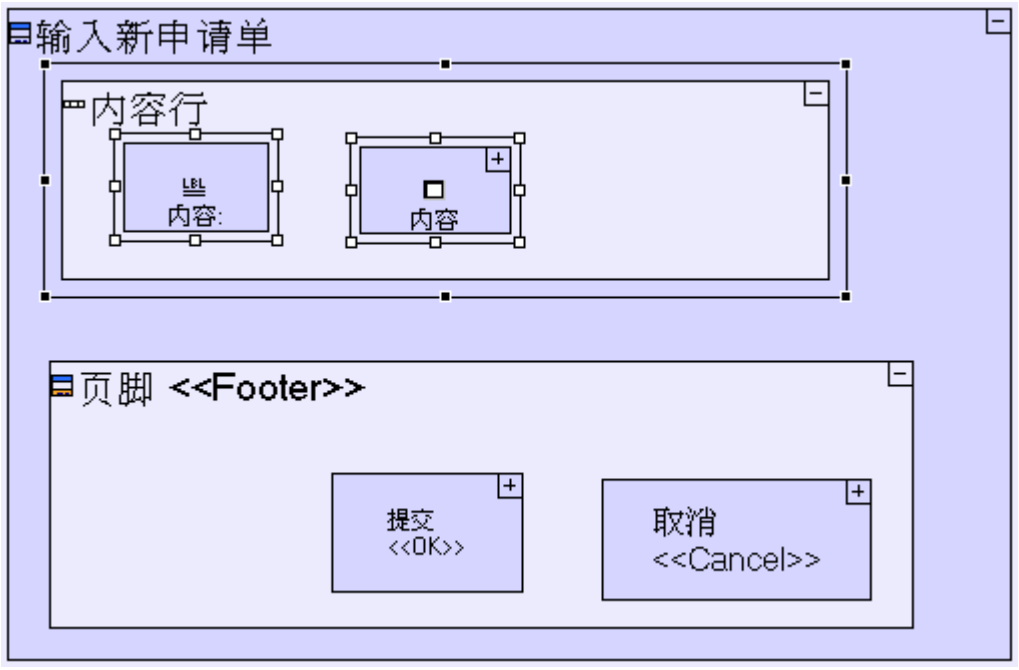
建立选取器模型 Model a Chooser

我们想让员工能够区分一般的申请单和“**紧急申请单**”（**urgent requisitions**）。

要这样做，我们可以在申请单中添加一个“**紧急性**”（**Urgency**）字段，“**紧急性**”（**Urgency**）字段有两种数值：“**普通**”（**Regular**）和“**紧急**”（**Urgent**）。我们应用了“**选取器**”（**Chooser**）显示元素来做到这一点，使用户能够从预先定义好的数值清单中进行选择。

使用行单元以改善（弹出窗口显示)格式 Use a Row Element for Better Formatting(of the popup display)

我们看一下 “**输入新申请单**”（**Enter New Requisition**）弹出窗口，在上阶段完成时导入的模型中，该弹出窗口已经完成了更新：



我们可以看到“**内容**”（**Description**）文本区和标签（label）已经被放进一个叫“**内容行**”（**Description Row**）的“**行**”（**Row**）显示单元中。这样确保了在行里面的单元永远显示在同一行，从而使弹出窗口看起来更简洁。

我们希望把即将添加的“**紧急性**”（**Urgency**）选取器放在它自己的行（Row）中：

放大到“**输入新申请单**”（**Enter New Requisition**）弹出窗口。

选择“**显示/行**”（**Display/Row**）模版(▭)，并把它拖到“**输入新申请单**”（**Enter New Requisition**）里面，放在“**内容行**”（**Description Row**）下面。

将它命名为“***紧急性行***”***Urgency Row***

添加选取器显示(到弹出窗口里面) Add a Chooser Display (to the popup)

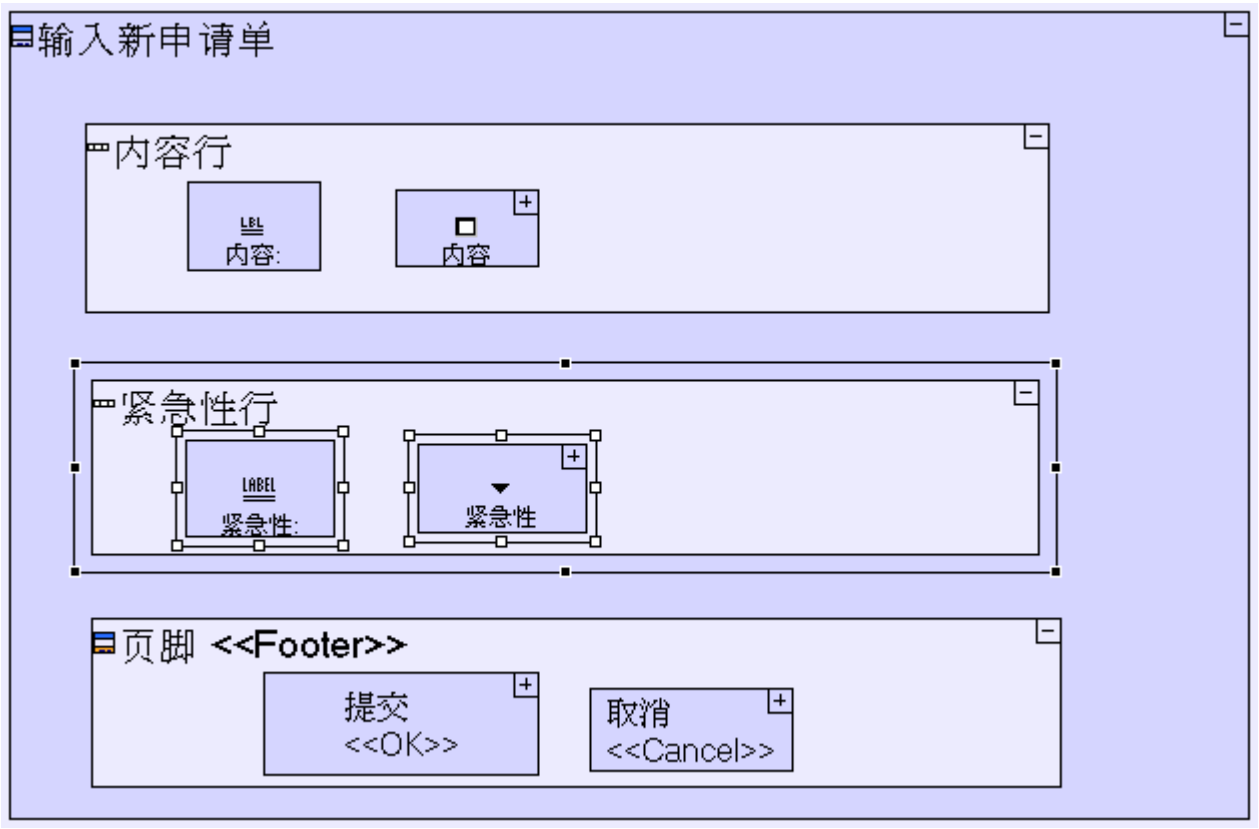
下一步，我们将添加选取器本身（和一个标签）：

放大到“**紧急性行**”（**Urgency Row**）。

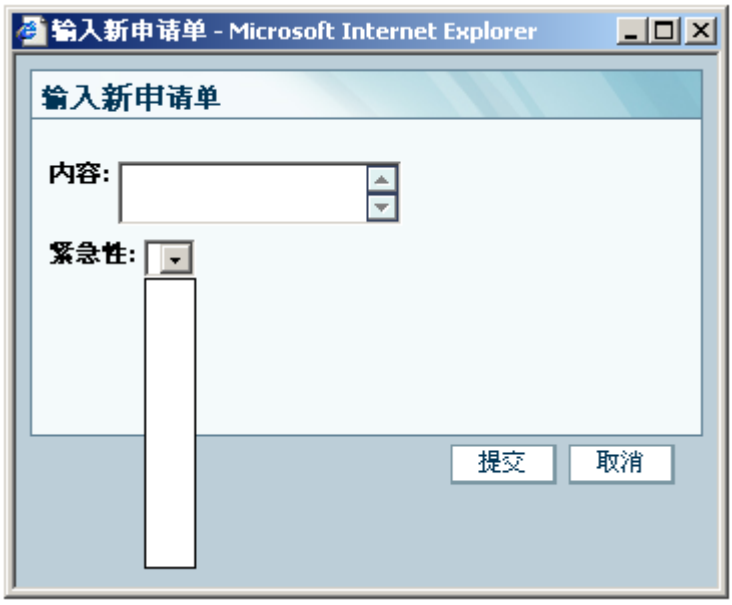
选择“**显示/标签**”（**Display/ Label**）模版(▭)，并把它拖入“**紧急性行**”（**Urgency Row**）。 命名为 “***紧急性:***”***(Urgency:)***

选择“**显示/选取器**”（**Display/ Chooser**）模版（▾），并把它拖入“**紧急性行**”（**Urgency Row**）。 命名为 “***紧急性***”***(Urgency)***

现在模型外观应与下图类似：



如果现在保存你的工作成果，运行应用程序并点击“**新申请单”（New Requisition）**按钮，正如以下的屏幕一样，你就可以看到刚刚建模的行里面含有一个空白的下拉式选单（选取器（chooser））：



生成初始化过程(填入选取器数值) Create an Initialization Process (that populates the chooser with values)

为了指定紧急性（**Urgency**）的数值，“**输入新申请单（Enter New Requisition）**”弹出窗口里面需要有一个初始化过程。每次“**输入新申请单**”弹出窗口打开时，该初始化过程都会运行，而且会给 **Urgency** 的下拉式选单填入可能的数值以供选择：

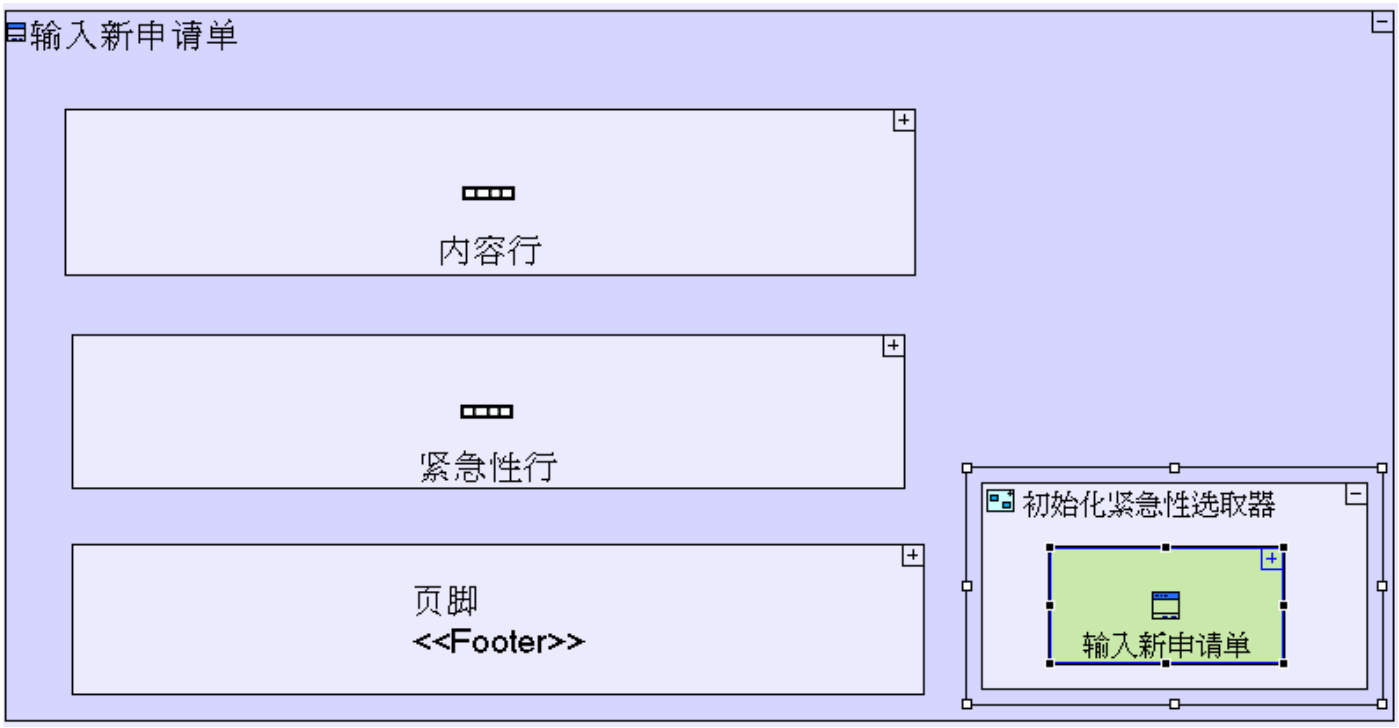
选择“**基本/动作**”模板（Basic/Action template）（），把它拖入“**输入新申请单**”（Enter New Requisition）里面； 命名为 “***初始化紧急性选取器”（Initialize Urgency Chooser）***

放大到“**初始化紧急性选取器**”（Initialize Urgency Chooser）。

这个过程必须包括**紧急性**选取器显示单元的索引，这个索引是“**输入新申请单**”弹出窗口的一部分：

右击“**初始化紧急性选取器**”过程，从菜单中选择“**添加原始参考数据**”（**Add Ancestor Reference**），并选择 “**输入新申请单”（Enter New Requisition）。**

现在模型外观应如下图所示：



使用常数(定义选取器显示的数值) Use Constants(to define the values displayed in the chooser)

鉴于我们只有两个想用的预先定义好的“**紧急性数值”（urgency values）**，我们可以使用常数（是预先定义好的固定数值的数据单元）对数值进行定义。

选择“**常数/文本”**模板（**Constants/Text** template）(),并把它拖入“**初始化紧急性选取器**”(Initialize Urgency Chooser)里面；

把它命名为 **“正常”(*Regular*)**。

注意：我们刚添加的 **Regular** 数据单元(regular data element)显示为**“Regular”**，而不是 **Regular**。这说明它是个**常数**(Constant)。
“Regular” 是一个文本常数（一串字符）。常数还包括数字常数、日期常数等等。

选取器的数值不一定需要使用常数来定义。数值可能从任何数据来源中得来，比如之前过程的结果、数据库表格或电子表格。

现在需要流程（Flow）来指定应该填入选取器的常数：

创建一个从“Regular”常数到**输入新申请单/紧急性行/<选则>的流程 (Enter New Requisition/Urgency Row/Urgency/<Options>)。**

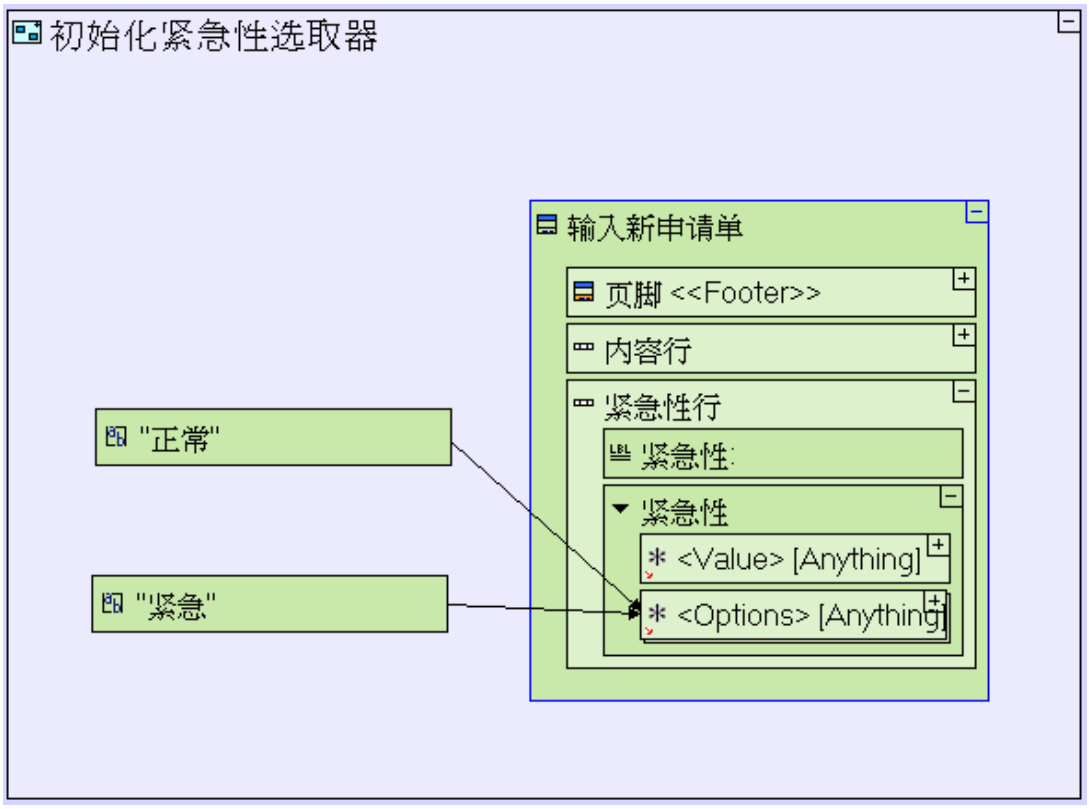
要指定选取器**紧急性(Urgency)**第二个可能的数据:

选择**常数/文本**模板(**Constants/Text** template)  , 并把它拖放到**“初始化紧急性选取器”“Initialize Urgency Chooser”里面**。

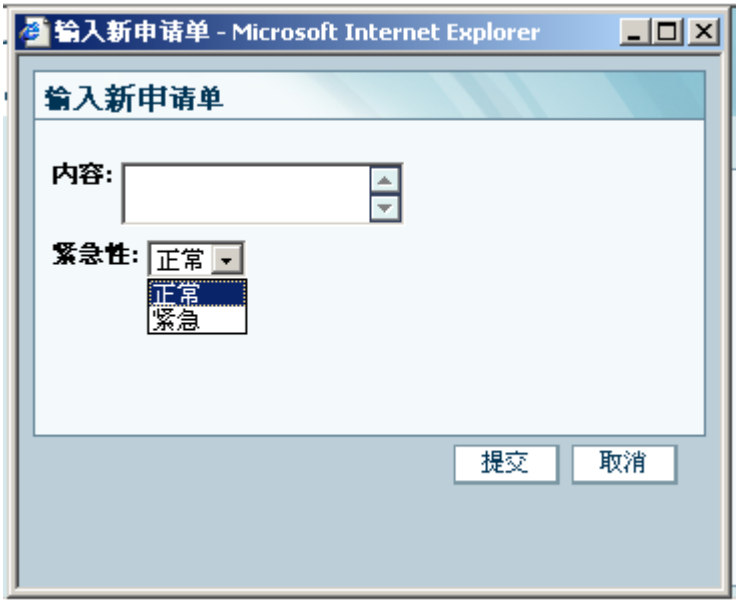
把它命名为 **“紧急”(*Urgent*)**。

创建一个从“Urgent”常数到**输入新申请单/紧急性行/<选则>。**

Initialize Urgency Chooser 模型应与下图类似:



如果你现在保存你的工作并运行应用程序，正如以下屏幕快照所显示的一样，“**输入新申请单**”（**Enter New Requisition**）弹出窗口中的下拉式选单应该允许您从“**Regular**”和“**Urgent**”这两个可能的数值中选择：



添加字段到数据结构中 Add a Field to the Data Structure

Urgency 选取器现在出现在显示列表中，但是输入的数值并没有和申请单一起保存。我们还必须建立向“**申请单**”（**Requisition**）数据库记录中添加“**Urgency**”字段的模型。

进入“**页脚/提交**”（**Footer/Submit**）中的“**申请单**”数据库记录。

选择“**数据类型/文本**”模版（**Data Types/Text** template）（  ），把它拖放到“**申请单**”中。命名为 “**紧急性**”（***Urgency***）

注意：由于申请单数据库记录在两个模型（**提交**和**申请单列表**（**Submit** and **Requisition List**））中重新使用，实际上你可以在**申请单列表/申请单**中操作以上步骤，这样也会产生同样的效果。在任何情况下，所做的改变对两者都会产生影响。

下一步我们要添加一个以用户在选取器中选择的数值填充“**申请单**”数据结构的“**紧急性**”字段的流程（Flow：

进入“**页脚**”（**Footer**）中的“**提交**”（**Submit**）按钮。

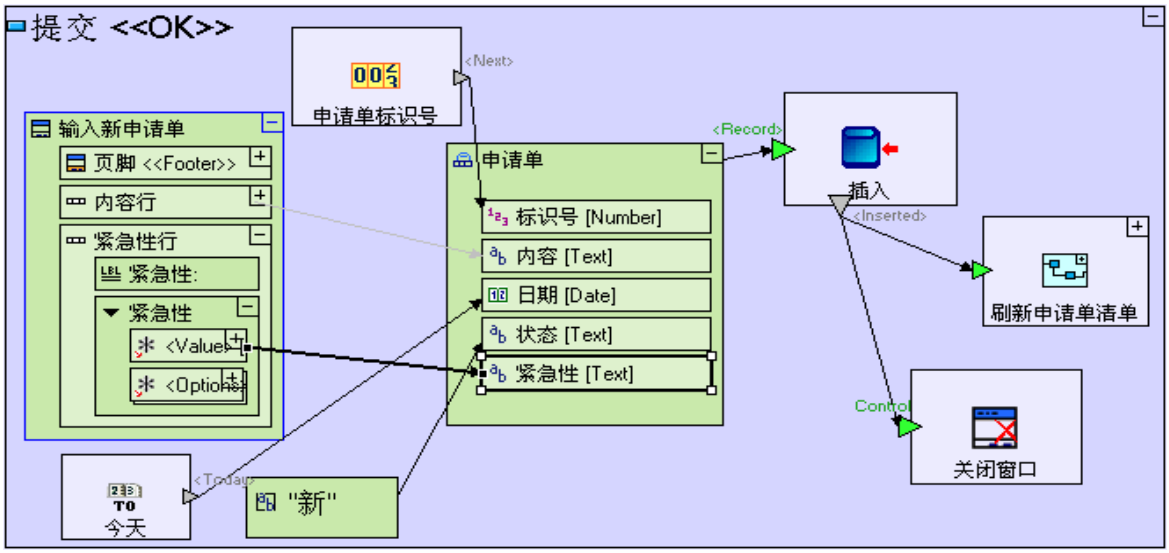
创建一个从**输入新申请单/紧急性行/紧急性/<数值>**（**Enter New Requisition/Urgency Row/Urgency/<Value>**）到**申请单/紧急性**流程（**Requisition/Urgency**）。

选择**紧急性->特性**（**properties**），点击  把它命名为 “**columnName**”

把 “columnName”命名为 “**urgency**”

选取器（Chooser）有 2 个预先定义好的数据单元—“<**选则**>”和“<**数值**>”。 “<**选则**>” 包含选取器中显示的所有数值（所以选取器是一个重复性的单元）。“<**数值**>”则包含目前已选择的值。

我们在上一步刚刚建立起的模型外观应与下图相似：



重新使用意味着建模速度更快(显示和数据库结构自动更新) Reuse Means Faster Modeling(display &database structure are automatically updated)

如果您现在保存工作成果并在浏览器中启动应用程序，结果应该如下图所示：

我们没有在“**申请单清单**”（**Requisition List**）表格模型上建立任何额外的模型，但现在表格已经有“**Urgency**”这一列。这是因为“**申请单 Requisition**”数据库记录在**申请单清单**”（**Requisition List**）模型中重新使用。 同时由于“**Requisition**”数据库记录定义数据库表格的实际结构，数据库中“**Requisitions**”表格的结构也以新的字段“**Urgency**”进行更新。

在添加“**Urgency**”字段之前，数据库中的记录的数值为“空”（NULL ）并正如我们从以上屏幕快照所看到的一样，显示的数值为空白。

完成阶段 5 Completing Stage 5

现在我们输入项目样本“**Tutorial 5-6**”并把它作为本教程下一个阶段的基础。

> 想重新了解如何输入一个项目样本，请参阅**阶段 2** 最后的“**输入项目样本**”(Importing a Sample project)的部分。

这个项目样本具备到目前为止我们建立的模型的所有功能。

现在，您可以开始学习**阶段 6**，在**阶段 6** 里面，我们将建立申请单生命周期（requisition's lifecycle）下一阶段（经理批准雇员的申请单）的模型。

实例 See It live

点击此处在另一个独立窗口中打开实际项目。

阶段 6 - 建立附加视图模型和更新数据 Stage 6 - Modeling an Additional View and Updating Data

阶段目标 Stage Goals

覆盖的 Tersus 概念 Tersus Concepts Covered

完成本阶段后，您将会熟悉以下概念：

建模概念:槽名（slot names）(保留槽名 reserved 和用户自定义槽名 user-defined)

建模技术: 给应用程序添加视图 (adding views)

处理显示表单（display table）中选定的行

更新数据库中储存的数据

有用的过程模板: 更新（Update）

有用的显示模板:〈Selected Row〉(在简单表格里面 in Simple Table)

建立应用程序功能模型 Application Functionality Modeled

在这个阶段，您将给应用程序添加一个新的视图。最初的视图用于输入申请表，经理用新视图批准申请单。

本阶段建模在上个阶段最后输入的项目“**申请单管理系统(5-6)**”（**Requisition Management system**）中进行。

用户建模 User Modeling

给模型添加一个视图 Add a View to the Model

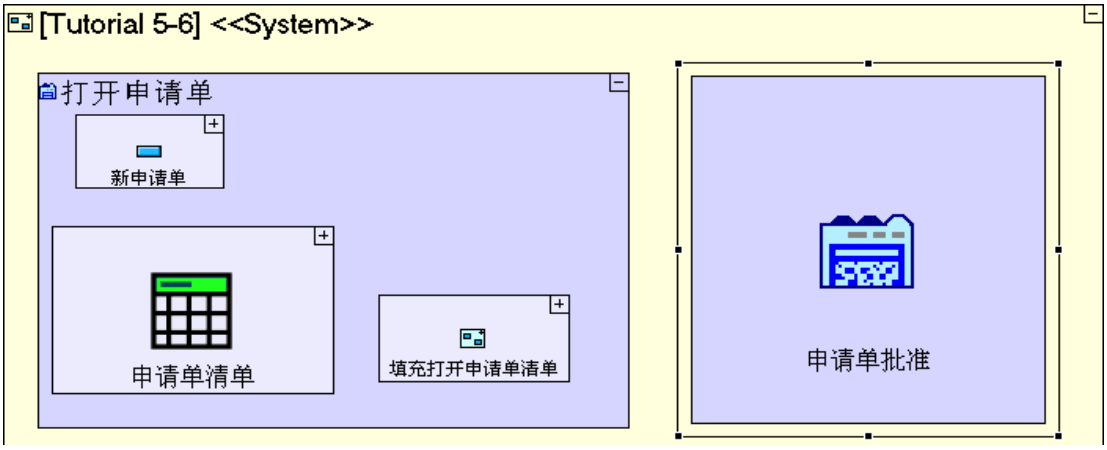
我们现在给我们的应用程序“**申请单的批准**”（Requisition Approval ）添加第二个视图。经理将用这个视图查看雇员输入的申请单并决定是否批准申请单（或者退回申请单-下个阶段）。

进入“根模型”（root model）(**Requisition Management System**).

在选项板（Palette）中选择“**显示/视图**”模板（ **Display/View** template）( )，并把它放到“**打开申请表**”（**Open Requisitions**）视图旁边。

把它命名为 “***申请单批准” (Requisition Approval)***

现在的模型外观应与下图类似：



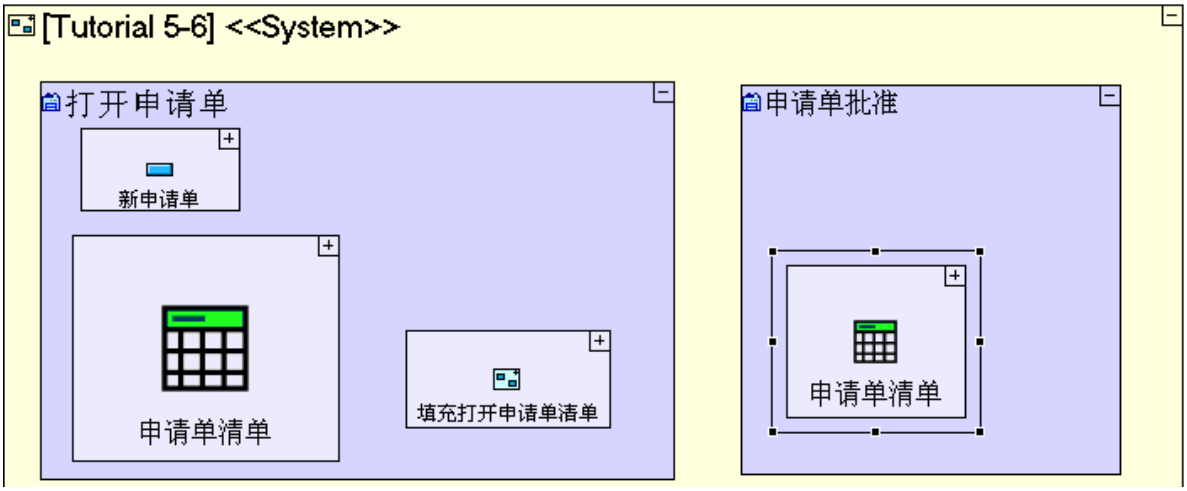
“**申请单批准**”（Requisition Approval）视图包含打开申请单清单和批准/驳回申请单的按钮。

重新使用显示单元（申请单清单）Reuse a Display Element (Requisition List)

我们在前个阶段里面已为打开的申请单（open requisitions）清单建立了模型，现在我们重新使用这个清单：

从储存器（Repository）（或大纲（Outline））中把“**申请单清单**”（**Requisition List**）拖放到“***申请单批准***”（**Requisition Approval**）视图中。

现在的模型外观应与下图类似：



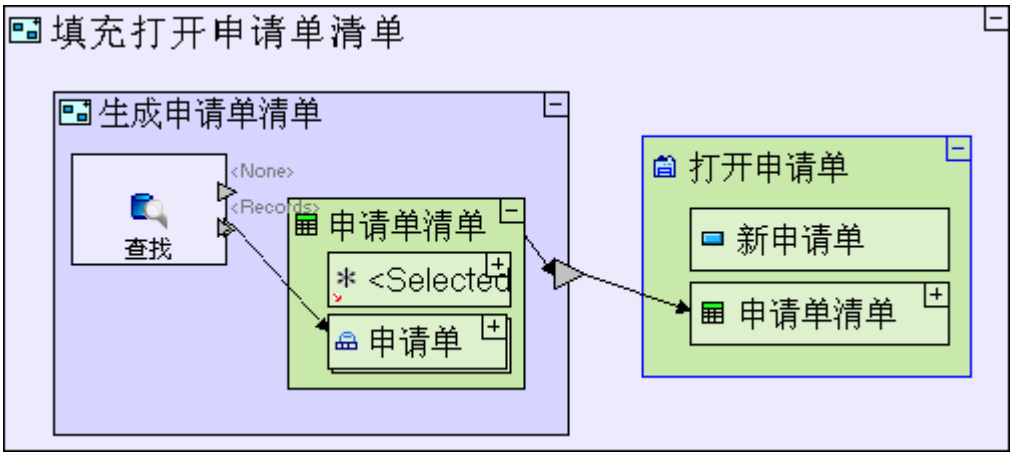
保存您的工作成果并在浏览器中查看应用程序。可以看到“**打开申请单**”（**Open Requisitions** ）视图的右边多了一个“**申请单批准**”（**Requisition Approval**）的标签。

如果你点击“**申请单批准**”（**Requisition Approval**），就会看到下图所示情况（“**申请单清单**”表格显示出来了，但仍是空的）：

在无法完全重新使用时重新创建一个过程(以填充申请单清单) Recreate a Process when it cannot be Reused in full (to populate the requisition list)

“**申请单清单**”（Requisition List）是空的，因为我们没有在“**打开申请单**”视图中重新使用原来填充清单的过程-“**填充打开申请单清单**”（Populate Open Requisition List）。

现在我们再看一下这个过程，回忆一下它的作用：



“**填充打开申请单清单**”（ Populate Open Requisitions List）的过程包括一个“**生成申请单清单**”（Generate Requisition List）的过程，这个过程创建了一个填充数据库“**申请单**”的“**申请单清单**”（Requisition List）数据单元，然后系统就会通过“**打开申请单**”（Open Requisitions）上级索引发送生成的“申请单清单”以填充表单显示。

这跟我们在这里想要做的几乎是一样的，只是这里的目标表单显示是在“**申请单批准**”（Requisition Approval）视图中(我们刚刚定义的新视图)，而不是在“**打开申请单**”视图中(原来的视图)。因此,我们应该重新使用“**生成申请单清单**”（Generate Requisition List）过程，从而在当前的视图中建立一个新的“**填充申请单批准清单**”（Populate Requisition Approval List）过程。

放大到“**申请单批准**”（Requisition Approval）。

在选项板中选择**基本/动作（Basic/Action）**并把它放到“**申请单批准**”中。

把它命名为 “***填充申请单批准清单***” ***(Populate Requisition Approval List)***

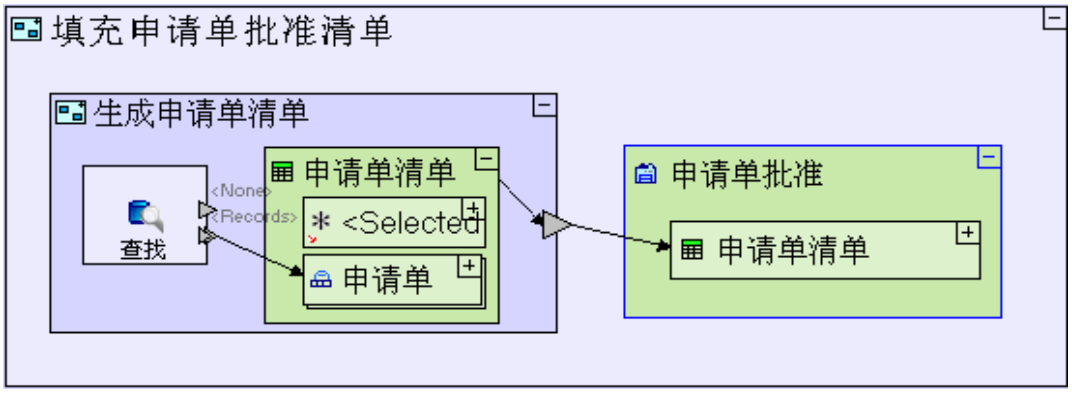
把“**生成申请单清单**”从大纲中拖出来并放到“**填充申请单批准清单 Populate Requisition Approval List**”里面以要重新使用“**生成申请单清单**”。

右击“**填充申请单批准清单**”单元,从菜单中选择“**添加上级索引**”（Add Ancestor Reference），再选

择 **“申请单批准”**。

使用**流程**工具（**Flow tool**）把**“生成申请单清单”**出口和**“申请单批准/申请单清单”**连接起来。

现在模型外观应该跟下图类似:



保存你的工作成果并在浏览器中查看应用程序。

点击**“申请单批准”Requisition Approval**”标签，您就可以看到下图：

您可以确定一下两个视图中是否都有申请单清单，并且在**“打开申请单”(Open Requisitions)**视图中输入新的申请单时，申请单清单是否会自动更新。

添加一个更新现有记录并将记录标识为“已批准”的按钮 Add a Button(that updates an existing record, marking it Approved)

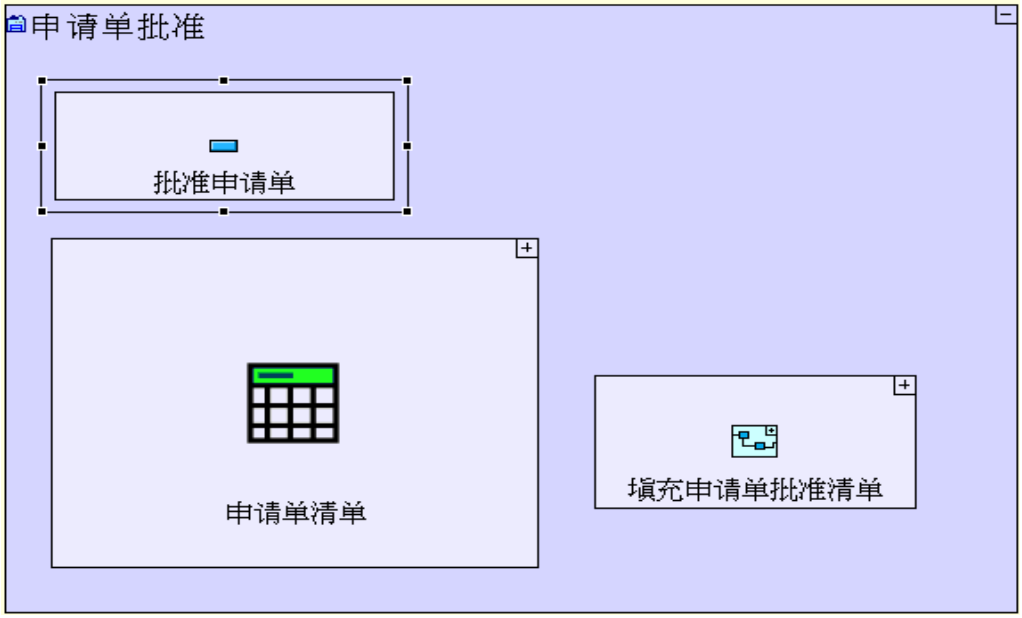
申请单批准视图（**Requisition Approval** view）应该让经理能够在表格里面选取一行，将其标识为“已批准”（Approved）（或者“退回”（Rejected））。

我们先从生成一个按钮自身开始。为了使这个视图和**“打开申请单”**(Open Requisitions)视图保持一致，我们把按钮放在表格上面：

在选项板中选择**“显示/按钮”(Display/Button)**，把它拖到**“申请单批准”（Requisition Approval）**里面（在申请单清单上面）。

把它命名为 ***“批准申请单”Approve Requisition***

申请单批准模型（**Requisition Approval** model）外观应该和下图类似:



现在，我们为按钮逻辑建模，建模过程应包括以下步骤：

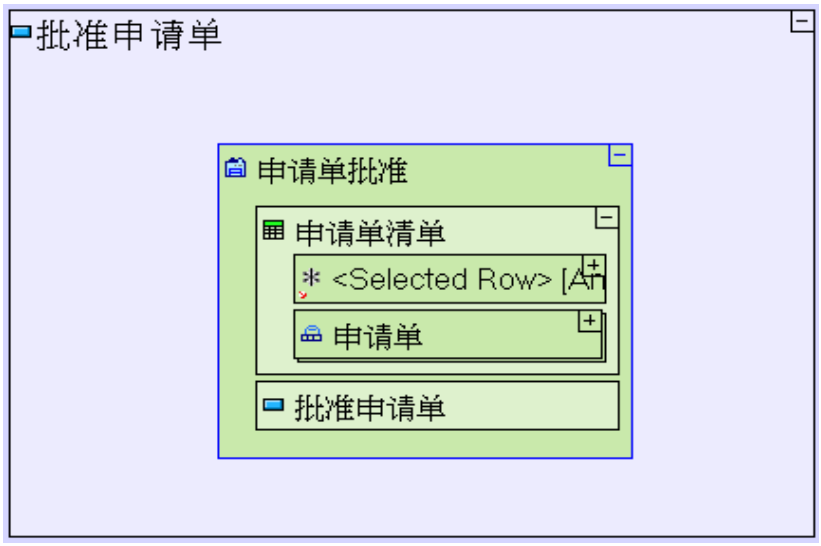
1. 从表格中提取选定的行
2. 把所选的行的“**状态**”（**Status**）字段数值改为“**批准**”（**Approved**）。
3. 更新数据库里对应的记录
4. 刷新“**申请单清单**”(**Requisition List**)表格以显示更新后的记录

从表格显示中提取已选定行 Retrieve the Selected row from a Table Display

要从表格显示中提取已选定行,首先我们必须添加显示的索引。

放大到“**批准申请单**”（**Approve Requisition**）。
右击**批准申请单**,从菜单中选择“**添加上级索引**”(**Add Ancestor Reference**),
再选择 **申请单批准**(**Requisition Approval**)

“批准申请单”模型外观应该与下图类似：



在缺省情况下，任何基于“**简单表单”（Simple Table）**（或**表格**）模板的显示单元都包括**〈选定行〉（Selected Row）**数据单元。任何时候只要用户点击表格中的一行（运行时间），**〈选定行〉（Selected Row）**单元就会填上那一行的数据。

由于<Selected Row>单元基于“任何”(Anything)数据类型,<Selected Row>单元能够支持任何母模型（表格）定义的行结构。

改变字段数值 Change a field’s Value

<Selected Row>单元以“**申请单”**的数据结构返回选定行。我们现在必须创建一个过程，把选定申请单的**状态（Status）**改成“**已批准”（Approved）**。

在选项板中选择**“基本/动作”（Basic/Action）**，把它拖到**“申请单批准”（Requisition Approval）**上级

索引旁边。

把它命名为 ***“修改申请单状态” (Change Requisition Status)***

我们正在创建的“**修改申请单状态**”过程应该把选定的**申请单**作为输入数据：

在选项板中选择“**触发槽”（Trigger slot）**（  ），把它拖到“**修改申请单状态**”边框上。

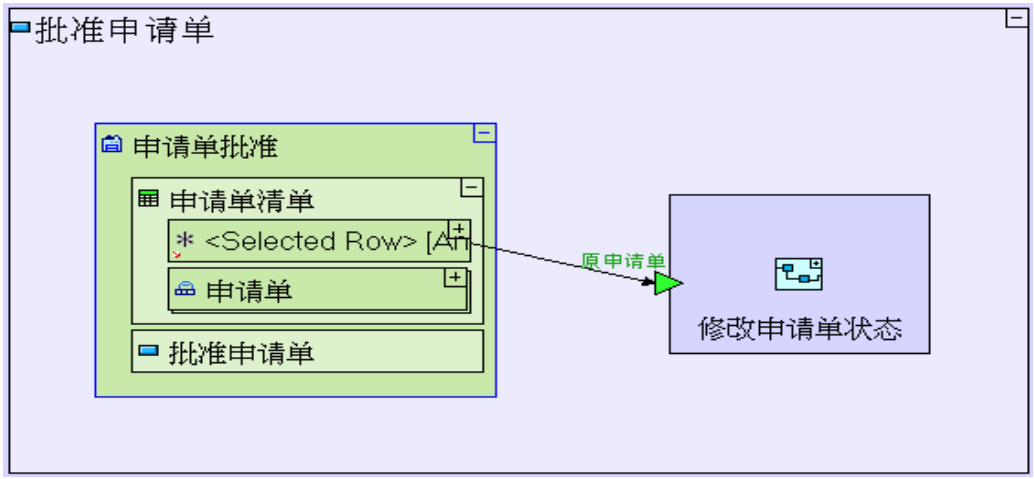
添加把**申请单批准/申请单清单/选定行(Requisition Approval/Requisition List/<Selected Row>)**和刚刚添加到“**修改申请单状态**”中的“**触发器**”（trigger）连接起来的“**流程**”（Flow）。

一会我们还要添加更多的槽（slots），所以我们最好现在就命名刚刚创建的槽，使它的用途（或内容）让任何看模型的人都一目了然。

选择我们刚添加的触发器（trigger）然后按**[F2]**键。

在出现的编辑框中，输入“**原申请单”**(*Original Requisition*) 作为 trigger 的名称。

“批准申请单”（Approve Requisition）模型外观应该与下图类似：



另外输入**“修改申请单状态”（Change Requisition Status）**中的数值必须是设定**“状态”（Status）**的新数值：

在选项板中选择 **“常数/文本”（Constant/Text）**并将它拖到新触发器**（trigger）**旁边（在**“修改申请单状态”**外部）。把它命名为 ***“批准”（Approved）***

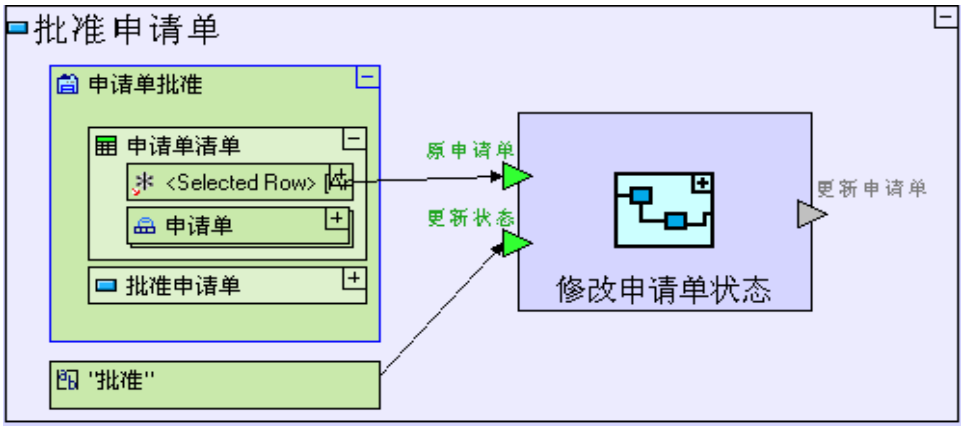
添加另外一个**触发器（trigger）**到**“修改申请单状态”（Change Requisition Status）**，选择**触发器**并按[F2]键，将其命名为 ***“更新状态”（Updated Status）***。

添加连接***“批准”（Approved）***常数和**“修改申请单状态/更新状态”**触发器**（Change Requisition Status/Updated Status）**的**流程（Flow）**。

最后，这个过程应该输出已更新的**“申请单”（Requisition）**：

给**“修改申请单状态”**边框架中添加一个**“退出”**槽（Exit slot）（  ），选择**“退出”**槽并按[F2]键，将其命名为 ***“更新申请单”（Updated Requisition）***

这个**“批准申请单”（Approve Requisition）**模型外观应该和下图相似：



现在我们进入**“修改申请单状态”**内部，并为**申请单**的**“状态”（Status）**字段更新建模。

放大到**“修改申请单状态”（Change Requisition Status）**。

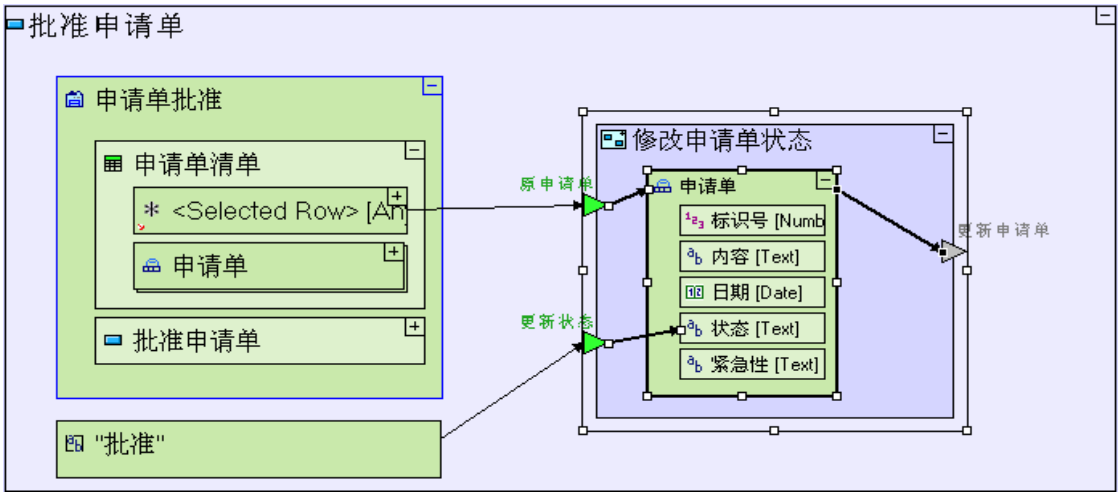
将“**申请单**”数据库记录拖入“**修改申请单状态**”。

添加连接“**原申请单**”（**Original Requisition** ）触发器和“**申请单**”（**Requisition**）数据结构的“**流程**”（**Flow**）。

添加连接“**更新状态**”触发器（**Updated Status** trigger）和“**申请单/状态**”（**Requisition/Status**）字段的“**流程**”（**Flow**）。

添加连接“**申请单**”（**Requisition**）数据结构和“**更新申请单**”（**Updated Requisition**）出口的“**流程**”（**Flow**）。

这个“**修改申请单状态**”（**Change Requisition Status**）模型外观应该与下图类似：



提交更新记录到数据库和刷新表格显示 Commit the Updated Record to the Database and Refresh the Table Display

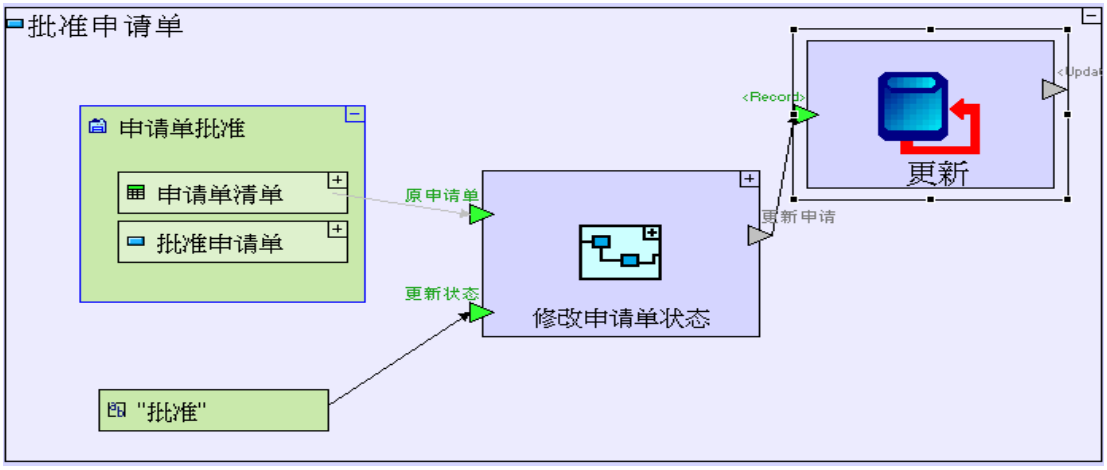
“**更新申请单**”（**Updated Requisition**）退出槽（exit slot）会返回一个已批准的申请单，我们必须在数据库里更新这个申请单。我们使用“**更新**”（**Update**）模板来做到这一点。

缩小当前页面并转到“**批准申请单**”（**Approve Requisition**）按钮模型。

选择“**数据库/更新**”（**Database/Update**）模板（  ）并把它放在“**修改申请单状态**”（**Change Requisition Status**）过程旁边。

创建连接“**修改申请单状态/更新申请单**”(**Change Requisition Status/Updated Requisition** ）出口和“**更新/<记录>**”（**Update/<Record>**）触发器的**流程**（**Flow**）。

“批准申请单”（Approve Requisition）模型应该与下图类似：



如果更新（Update）成功，我们就必须确保屏幕上的表格已被刷新，这样数据显示才能和数据库同步。成功更新后，**“Update”**会通过它的<**Updated**>出口退出：

把一个**“基本/动作”（Basic/Action）**模板拖到**“Update”**旁边，
命名为 ***“刷新申请单清单” (Refresh Requisition List)***

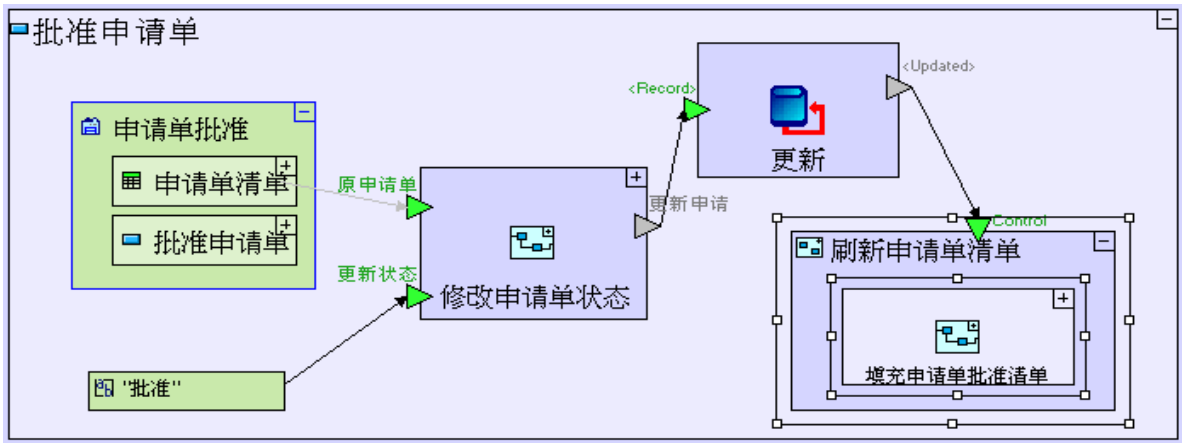
> 我们正在创建一个新的过程模型，这个过程模型的名字和一个现有过程的名字相同（**“提交申请单”(Submit Requisition)**里面有一个“更新申请单清单”(**Refresh Requisition List**)过程）。当这两个模型在储存库不同的“数据包”（packages）中时，这种情况是可能出现的。但在同一数据包中以现有模型的名字给新模型命名是不允许的。

右击**“刷新申请单清单”（Refresh Requisition List）**并打开**“添加单元”（Add Element）**子菜单以选择**“控制”（Control）**触发器。
创建连接**“更新/<更新>”（Update/<Updated>）**出口和 **“刷新申请单清单”（Refresh Requisition List）**触发器的**“流程”（Flow）**。

最后，重新使用我们在这一阶段初期创建的**“填充申请单批准清单”(Populate Requisition Approval List)**过程：

把**“填充申请单批准清单”(Populate Requisition Approval List)**过程从大纲（outline）中拖到**“刷新申请单清单”（Refresh Requisition List）**过程中。

“批准申请单”（Approve Requisition）模型应该和下图相似：



保存您的工作成果并在浏览器中查看应用程序。

点击“**申请单批准**”（**Requisition Approval**）标签（tab）。
选择一个申请单前，先点击“**批准申请单**”(**Approve Requisition**)按钮。

由于没有选择任何申请单，因此应该不会发生什么变化。

单击清单中的一个新申请单。

选定行（selected row）用稍深点的背景色表示，如下面屏幕快照所示（第 3 行）：

点击“**批准申请单**”(**Approve Requisition**)按钮。

正如下面屏幕快照所示一样,这个表格就会被刷新，选定的申请单的状态就会改为“**批准**”(**Approved**)：


Tutorial 5-6
Welcome Guest
TERSUS.
Tutorial 5-6
打开申请单
申请单批准
批准申请单
申请单清单
标识号 内容 日期 状态 紧急性
1 一支笔 2010-02-19 新 正常
2 两支铅笔 2010-02-19 新 正常
3 四把椅子和一张桌子 2010-02-19 批准 正常
4 大比萨饼 + 可乐 2010-02-19 新 紧急

完成阶段 6 Completing Stage 6

现在我们输入项目样本“**Tutorial 6-7**”，把它作为本教程下一个阶段的基础。

> 如果想再了解如何输入一个项目样本，请参阅**阶段 2** 最后“**输入项目样本**”(Importing a Sample Project)部分。

这个项目样本具备到目前为止我们建立的模型的所有功能。

现在，您可以开始学习**阶段 7**，在**阶段 7** 里面,我们将通过分解（修改）建模的各个部分和把它重新用于取消和退回申请单,从而建立与本阶段所建“批准申请单”（Approve Requisition）按钮模型类似的按钮模型。

实例 See It Live

点击此处在另一个窗口里面打开实际项目。

阶段 7--重新分解 - 改变过程以提高可重用性 Stage 7 - Re-factoring - Changing a process to Enhance Reusability

阶段目标 Stage Goals

覆盖的 Tersus 概念 Tersus Concepts Covered

本阶段完成时，您将会熟悉以下概念：
建模概念：重新分解（Re-factoring）
建模技术：删除模型单元（model elements）

建立应用程序功能模型 Application Functionality Modeled

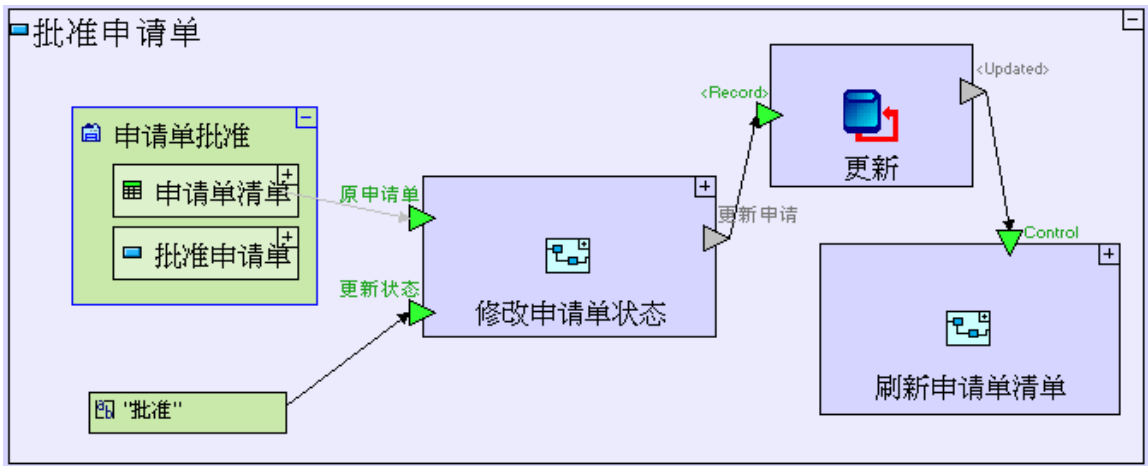
在这一阶段您将在“**打开申请单**”（**Open Requisition**）视图中添加一个**“取消申请单”（Cancel Requisition）**按钮。要取消一个申请单，只需要把它的**“状态”（Status**）字段设置成“**Canceled**”（取消）。

然后您将在“**申请单批准**”（**Requisition Approval**）视图中添加一个**“退回申请单”（Reject Requisition）**按钮，把“**状态**”（**Status**）字段设置为“**Rejected**”（**退回**）。

> 本阶段建模在上个阶段最后输入的“**Tutorial 6-7**”项目中进行。

重新分解现有模型 Re-factor an Existing Model

“取消申请单”（Cancel Requisition）和**“退回申请单”（Reject Requisition）**模型和我们在上一阶段创建的**“批准申请单”（Approve Requisition）**模型非常相似。现在我们回忆一下“**批准申请单**”模型是怎样建立的：



当然，我们不能重新使用现有**“批准申请单”（Approve Requisition）**模型，因为这个模型取决于一些它所独有的单元，正如下表所示一样，这些单元可能不同于**“取消申请单”（Cancel Requisition）**和**“退回申请单”（Reject Requisition）**：

模型单元	**批准申请单（Approve Requisition）**	**取消申请单（Cancel Requisition）**	**退回申请单（Reject Requisition）**
视图（上级索引）	**申请单批准（Requisition Approval）**	**打开申请单（Open Requisitions）**	**申请单批准（Requisition Approval）**
状态常数（Status constant）	**批准“Approved”**	**取消“Canceled”**	**退回“Rejected”**
“更新”（Update）以后执行的刷新过程	**刷新申请单清单（Refresh Requisition List）**（从**申请单批准**数据包中刷新）	**刷新申请单清单（Refresh Requisition List）**（从**打开申请单**数据包中刷新）	**刷新申请单清单（Refresh Requisition List）**（从**申请单批准**数据包中刷新）

除了这些单元之外，模型的其他单元全部是相同的，因此在一个独立的过程中，通过把所有能够重新利用的功能分组来改变模型，这个办法是行得通的。这种建模技术（用不同的方式重新排列模型）被称为**“重新分解”（Re-factoring）**。

过程可重用单元分组 Group Reusable Elements in a Process

现在我们通过创建一个可重用的过程，开始重新分解**“批准申请单”（Approve Requisition）**按钮：

进入**“批准申请单”（Approve Requisition）。**

把“**基本/动作”（Basic/Action）**模板拖放以**“批准申请单”（Approve Requisition）**。

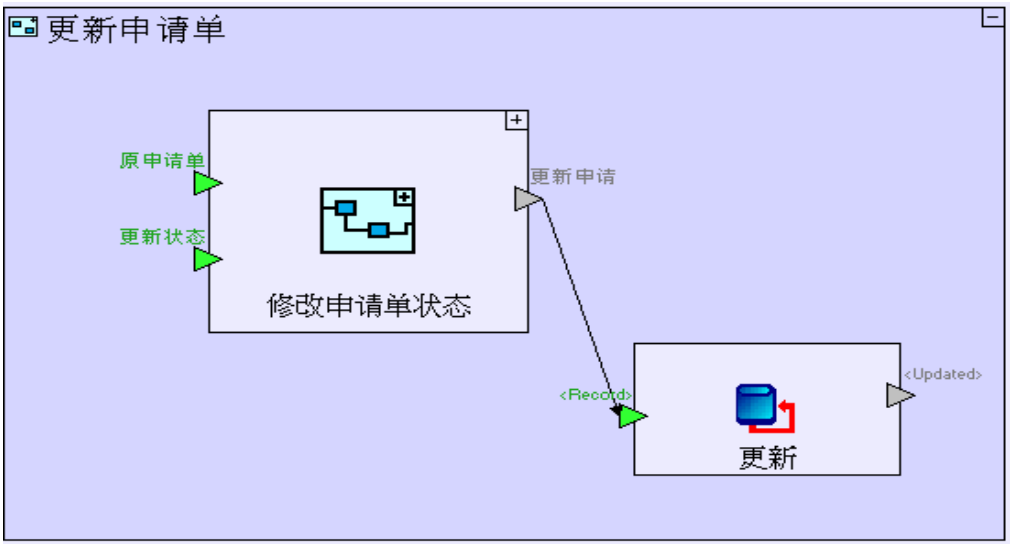
把它命名为 ***“更新申请单”（Update Requisition）*** 并放大到**“批准申请单”（Approve Requisition）**。

下面我们把上述可以重新利用的功能复制到新建的过程中：

从储存库中（或大纲）中拖放以下过程：**“修改申请单状态”（Change Requisition Status）**和**“更新”（Update）**。

创建连接**“修改申请单状态/更新申请单”（Change Requisition Status/Update Requisition）**出口和**“更新/<记录>”（Update/<Record>）**触发器的**流程（Flow）**。

现在**“更新申请单”（Update Requisition）**过程外观应该与下图类似：



有些槽（slots）没有流程（flow）连接，这是我们下一步的工作。回忆一下，所有这些没有流程连接的槽（例如**“原申请单”（Original Requisition）**触发器）原来都是连接我们不能重新使用的单元（element）的流程的目标(target)或来源(source)。我们现在给正在建模的**“更新申请单”（Update Requisition）**添加相应的槽（slots），这样就可以在可重新使用过程（reusable process）以外指定非可重新使用单元。

在**“更新申请单”（Update Requisition）**的左上角边框中放置一个**“触发器”（Trigger）**。(用[F2]键)把它命名为 ***“选定申请单”（Selected Requisition）***

创建连接这个触发器(trigger)和**“修改申请单状态/原申请单”(Change Requisition Status/Original Requisition)的流程（Flow）**。

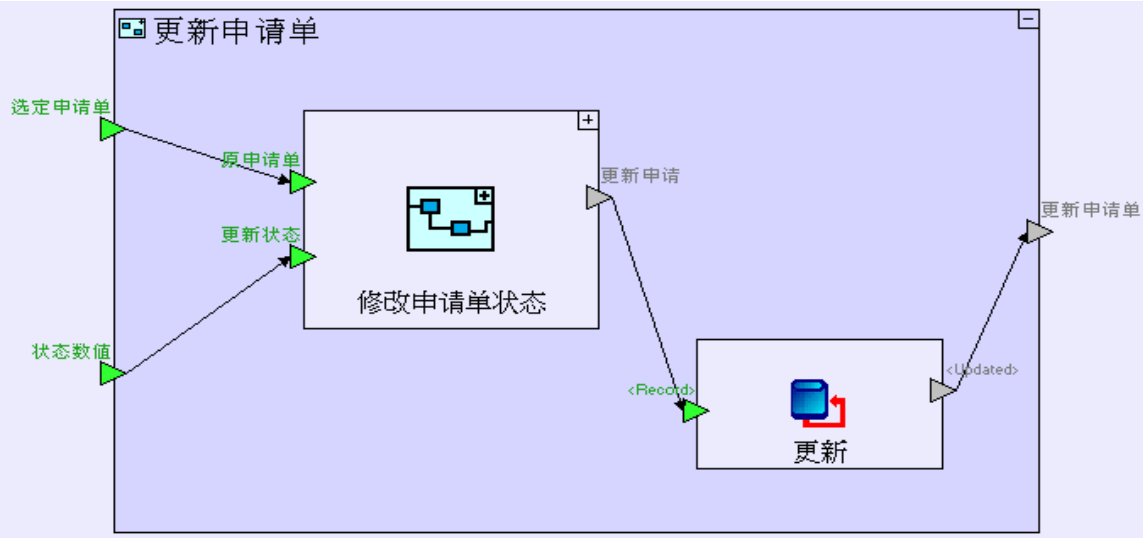
在**“更新申请单”（Update Requisition）**的左边框上放置第二个**“触发器”（Trigger）**。把它命名为 ***“状态数值”（Status Value）***

创建连接这个触发器(trigger)和**“修改申请单状态/更新状态”(Change Requisition Status/Updated Status)**的**“流程”（Flow）**。

在**“更新申请单”（Update Requisition）**的右边框放置一个**“退出”（Exit）**。

创建连接“**更新/<更新>**”（**Update/<Updated>**）出口和这个出口（Exit）“**流程**”（**Flow**）。

“**更新申请单**”（**Update Requisition**）过程的外观应该与下图类似：



完成重新分解(以新过程取代单元和流程) Finish Re-factoring (replacing elements and flow with the new process)

为了完成重新分解，我们需要重新定义“**更新申请单**”（**Update Requisition**）过程流入/流出的流程（flow）和清除多余的单元，从而清理“**批准申请单**”（**Approve Requisition**）。

（通过点击“**更新申请单**”（**Update Requisition**）过程右上角的[-]）缩小“**更新申请单**”（**Update Requisition**）过程（以尽量减少混淆或错误）。

缩小当前窗口，转到“**批准申请单**”（**Approve Requisition**）。

重新定义 3 个“**流程**”（**Flows**）：

点击选择连接“**申请单批准**”（**Requisition Approval**）显示数据单元和“**修改申请单状态/原申请单**”(**Change Requisition Status/Original Requisition**)触发器的**流程**（**Flow**）。

点击“**流程**”（**Flow**）的目标锚点（target anchor）（ 

）并把它拖放到“**更新申请单/选定申请单**”(**Update Requisition/Selected Requisition**)触发器上面。

选择连接“**批准**”（**Approved**）常数和“**修改申请单/更新状态**”(**Change Requisition/Updated Status**)触发器的**流程**（**Flow**）。

点击“**流程**”的目标锚点（ 

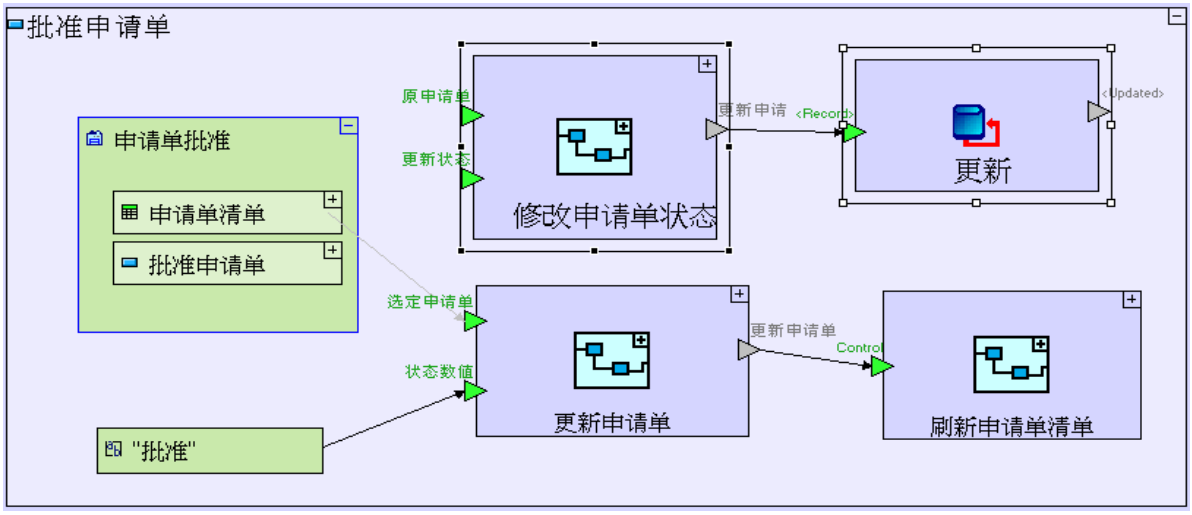
）并把它拖放到“**更新申请单/状态数值**”（**Update Requisition/Status Value**）触发器上面。

选择连接**“更新”（Update）**过程和 “**刷新申请单清单**”（**Refresh Requisition List**）过程的**流程**（**Flow**）。

点击“**流程**”的源锚点 （source anchor） 

，把它拖放“**更新申请单**”（**Update Requisition** ）出口上面。

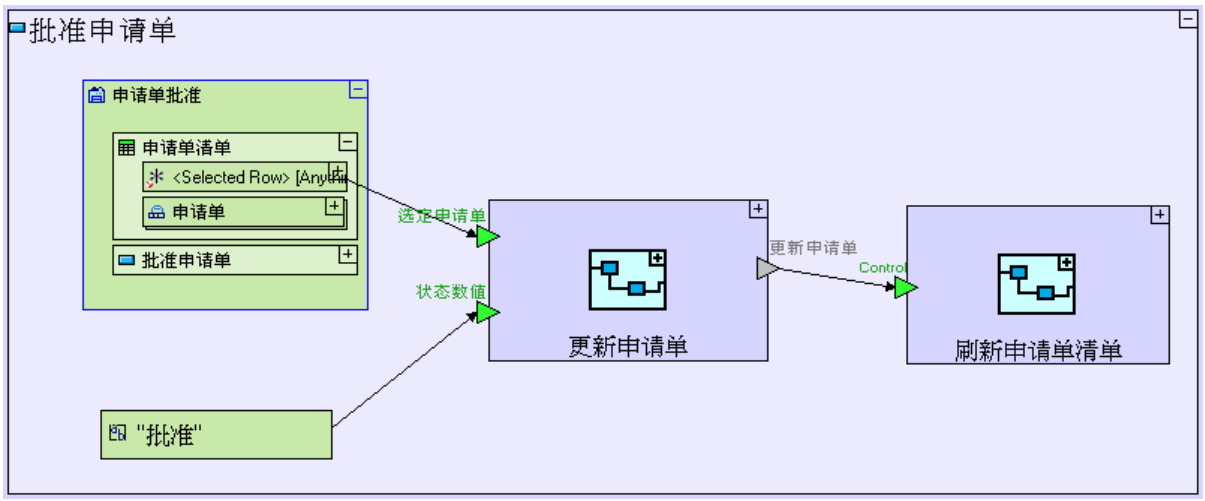
这个模型现在的外观应该与下图类似：



现在我们必须删除屏幕快照顶端出现的多余单元。

选择“**修改申请单状态**”（**Change Requisition Status**）和“**更新**”（**Update**）这两个多余单元和连接这两个单元的流程（flows），再按[**Del**]删除这两个多余单元。

现在“**批准申请单**”（**Approve Requisition**）的重新分解就完成了，模型的外观应该与下图类似：



保存您的工作成果并在浏览器中查看应用程序。它的外观和功能应该不会有什么不同。

确保“**批准申请单**”（**Approve Requisition**）按钮仍能正常发挥作用。

重新使用部分重新分解模型(取消申请单) Reuse part of the Re-factored Model (Cancel Requisition)

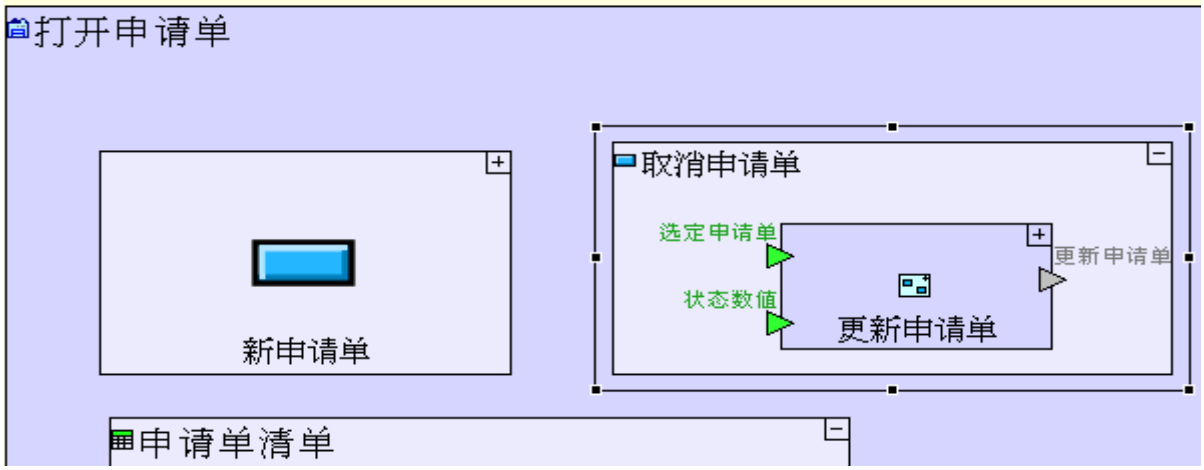
现在我们在“**打开申请单**”（Open Requisitions）视图中创建“**取消申请单**”（Cancel Requisition）按钮，该按钮应该和“**批准申请单**”（Approve Requisition）按钮类似。

打开“**打开申请单**”（Open Requisition）**视图。**
把“**显示/按钮**”（Display/Button）模板拖放到“**新申请单**”（New Requisition）按钮旁边。
把它命名为 “***取消申请单***”（*Cancel Requisition*）
放大到“**取消申请单**”（Cancel Requisition）。

现在重复使用“**更新申请单**”（Update Requisition）过程以建立一个与“**批准申请单**”（Approve Requisition）类似的模型（用前一个屏幕快照作为参考）：

将我们刚才创建的“**更新申请单**”（Update Requisition）过程从大纲（Outline）中拖放到“**取消申请单**”（Cancel Requisition）。

“**打开申请单**”（Open Requisitions）视图模型现在外观和下图类似：

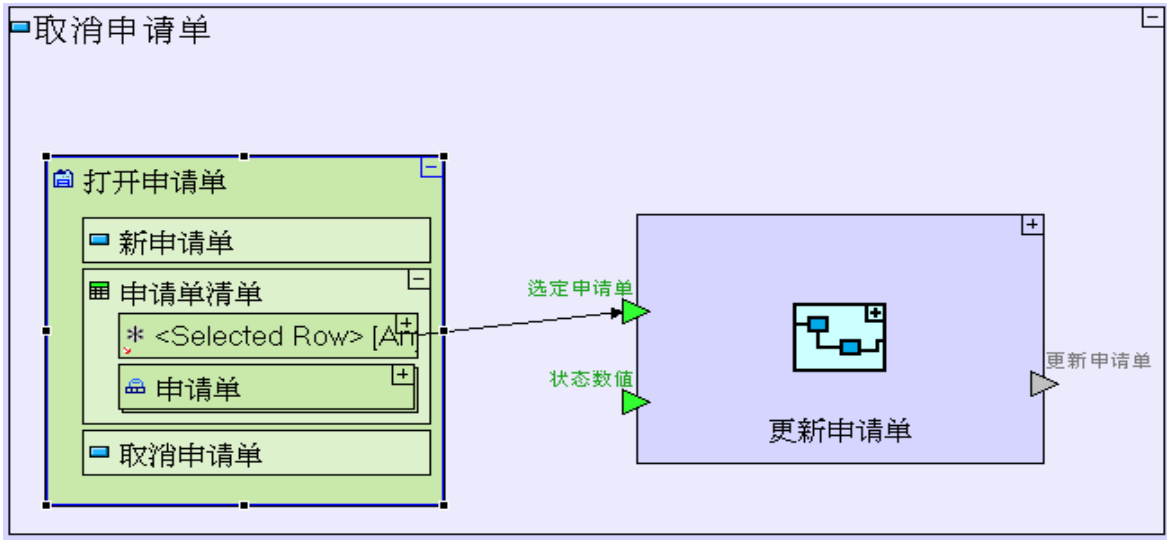


给这个视图创建一个提取“**申请单清单**”（Requisition List）的索引：

右击“**取消申请单**”（Cancel Requisition），选择“**添加上级索引**”（Add Ancestor Reference），并选择“**打开申请单**”（Open Requisitions）。

创建一个连接“**打开申请单**”（Open Requisitions）上级索引（ancestor reference）中的“**申请单清单/<选定行>**”（Requisition List/<Selected Row>）和“**更新申请单/选定申请单**”（Update Requisition/Selected Requisition）触发器的“**流程**”（Flow）。

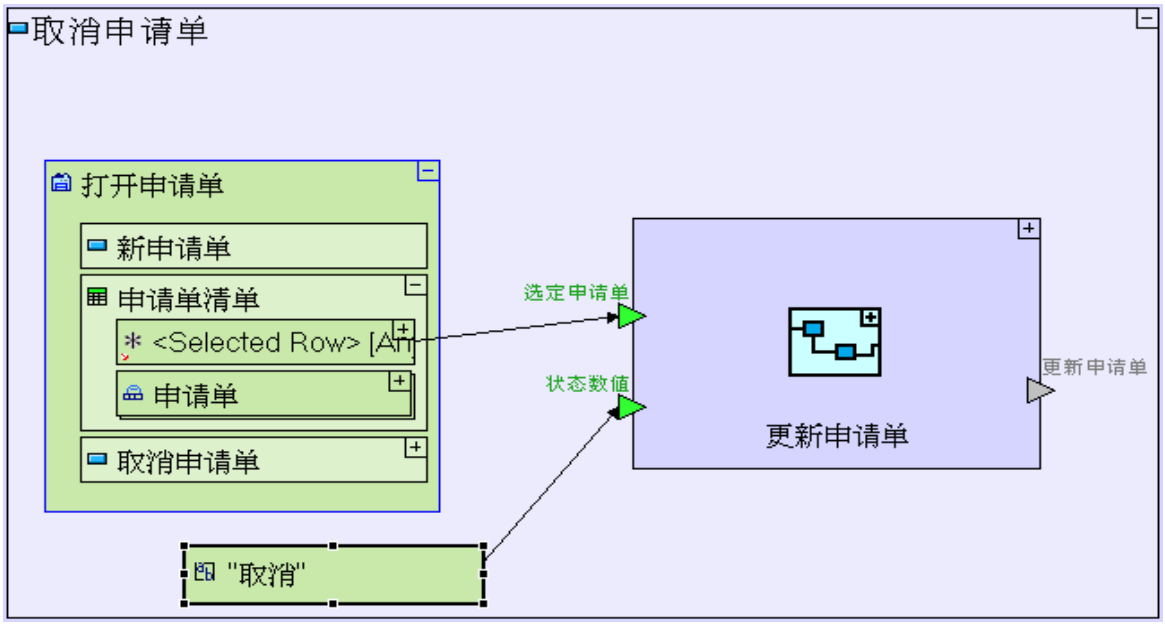
“取消申请单”（Cancel Requisition）按钮模型现在外观与下图类似：



定义新常数“取消”（**Canceled**）：

从选项板（palette）中添加一个**“常数/文本”（Constant/Text）**。把它命名为 ***“取消”*** *(Canceled)* 创建连接**“取消”**和**“更新申请单/状态数值”（Update Requisition/Status Value）**的流程（Flow）。

模型外观现在与下图类似：

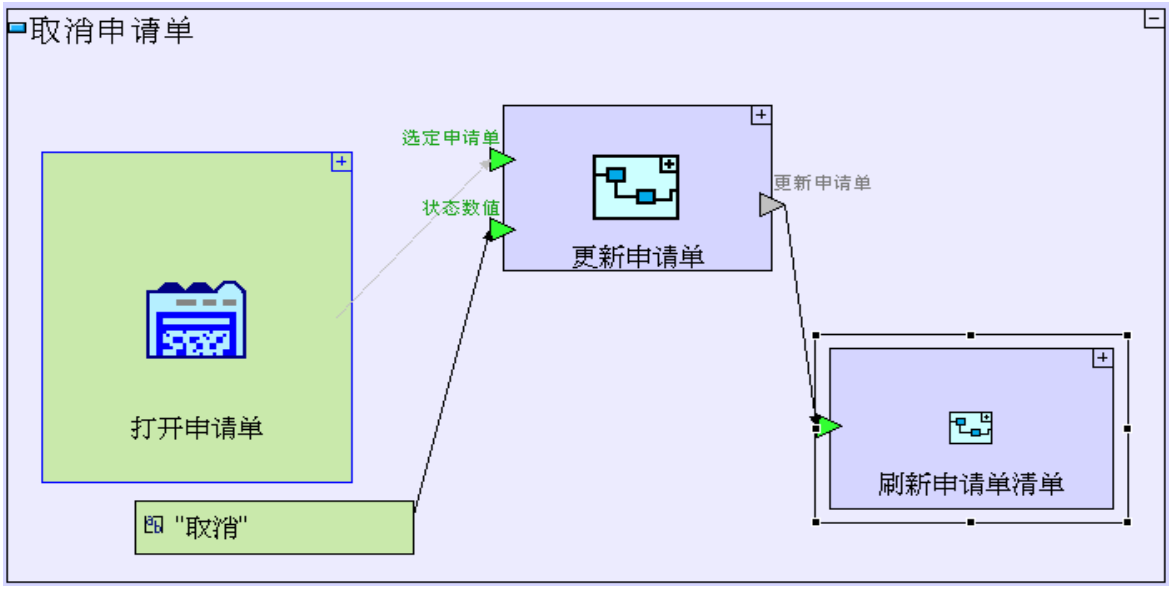


要完成这一过程，可在状态（status）更新后用现有的刷新过程刷新视图：

把**“刷新申请单清单”（Refresh Requisition List）**(曾用于**“输入新申请单”（Enter New Requisition）**弹出窗口的**“提交”（Submit）**按钮中)从储存库(repository)(或大纲)中拖放到**“取消申请单”（Cancel Requisition）**。

创建连接**“更新申请单”（Update Requisition）**的**“出口”（Exit）**和**“刷新申请单清单”（Refresh Requisition List）**触发器（trigger）的**“流程”（Flow）**。

“取消申请单”（Cancel Requisition）按钮现在完成了，外观应该与下图类似：



现在保存您的工作成果并在浏览器中查看应用程序，应用程序的外观应该与下图类似：

确保（**“打开申请单”（Open Requisition）**视图中）的**“取消申请单”（Cancel Requisition）**按钮仍能正常发挥作用。

完成阶段 7 Completing Stage 7

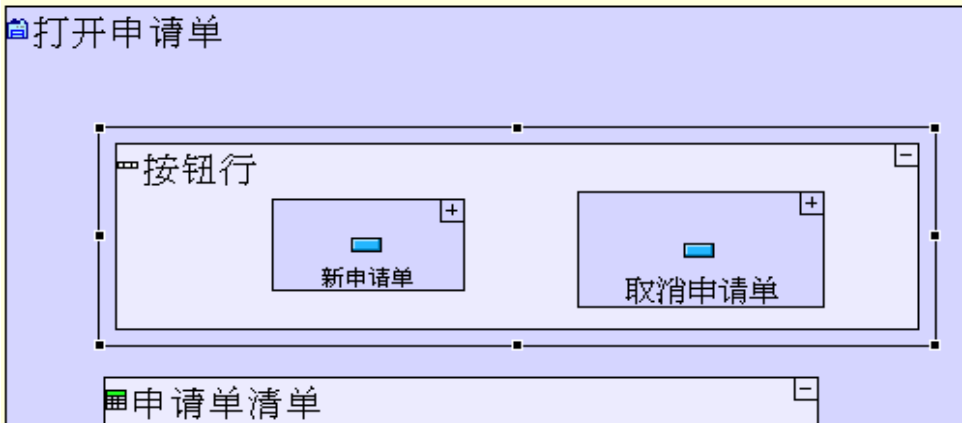
现在我们输入项目样本“**Tutorial 7-8**”，把它作为本教程下一个阶段的基础。

> 想再了解如何输入项目样本，请参见**阶段 2** 最后“**输入项目样本**”部分。

这个项目样本具备到目前为止我们建立的模型的所有功能。

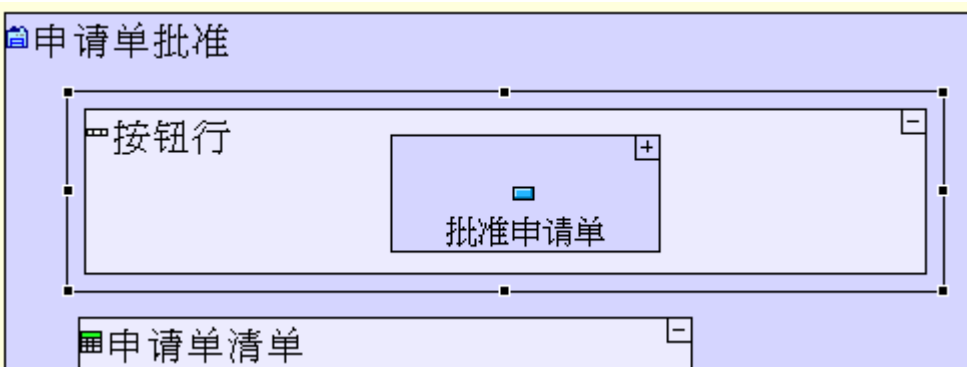
这个项目样本还具有如下功能：

1. 给“**打开申请单**”（**Open Requisitions**）视图添加一个“**按钮行**”（**Button Row**）并把按钮放进去，使显示更为整洁。



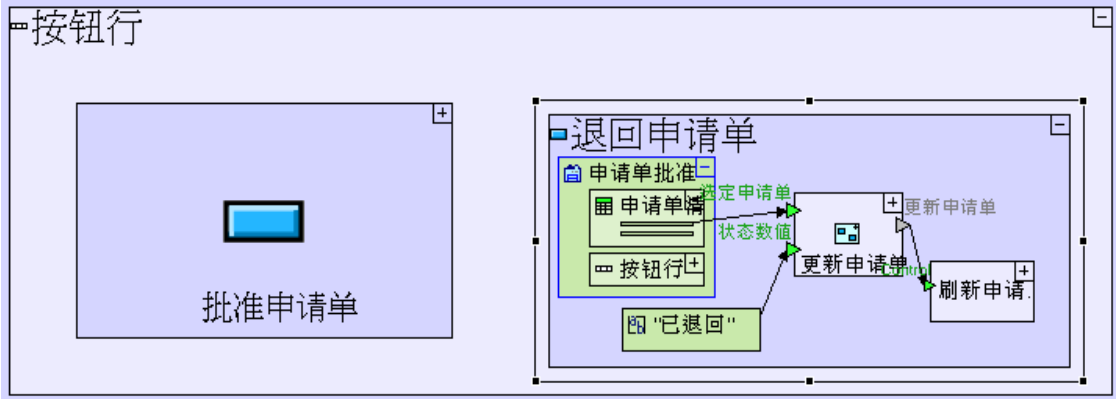
如何建模	位置
拖拽一个“**显示/行**”（Display/Row）模板。 把它命名为 “***按钮行***” ***(Button Row)***	**打开申请单**视图(Open Requisition)
从大纲/储存库中拖拽**“新申请单”**（New Requisition）和**“取消申请单”**（Cancel Requisition）	**按钮行** (Button Row)
删除**“新申请单”**（New Requisition）和**“取消申请单”**（Cancel Requisition）按钮	**打开申请单**视图(Open Requisition)

2. 给“**申请单批准**”（Requisition Approval）视图添加一个“**按钮行**”（Button Row），使显示更为整洁。



如何建模	位置
拖拽一个“**显示/行**”（Display/Row）模板。 把它命名为 “***按钮行***” ***(Button Row)***	**申请单批准**视图 (Requisition Approval)
从大纲/储存库中拖拽**“批准申请单”**（Approve Requisition）	**按钮行** (Button Row)
删除**“批准申请单”**（Approve Requisition）	**申请单批准**视图 (Requisition Approval)

3. 给“**申请单批准**”（Requisition Approval）视图添加一个“**退回申请单**”（Rejected Requisition）按钮，重新创建为**“批准申请单”**（Approve Requisition）和**“取消申请单”**（Cancel Requisition）建立的模型。



如何建模	位置
拖拽一个“**显示/按钮**”（**Display/Button**）模板。把它命名为 “***退回申请单*”** ***(Reject Requisition)***	**申请单批准/按钮行**（Requisition Approval/Button Row）
重新使用“**更新申请单**”（**Update Requisition**）过程。 给“**申请单批准**”（**Requisition Approval**）视图添加一个上级索引。 创建一个新的文本常数 “**退回**”（**Rejected**）。 （从“**申请单批准**”（**Requisition Approval**）数据包中）重新使用“**刷新申请单清单**”（**Refresh Requisition List**）过程。 创建与 **“批准申请单”（Approve Requisition）**相同的流程（flows）。	**“退回申请单”**（Reject Requisition）按钮

现在，您可以开始学习**阶段 8**，在**阶段 8** 里面,我们将对两个视图申请单的提取进行微调，确保每个视图只显示相关的申请单。

实例 See It Live

点击此处在另一个窗口里面打开实际项目。

阶段 8 - 过滤提取的数据 Stage 8 - Filtering Retrieved Data

阶段目标 Stage Goals

覆盖的 Tersus 概念 Tersus Concepts Covered

本阶段完成时，您将会熟悉以下概念：
建模概念：流程错误（Flow errors），删除流程(Remove Flow)
建模技术：过滤提取的数据，清空表格显示，在包(package)之间移动模型
有用的过程模板：（有条件）查找(Find)，高级查找(Advanced Find)

建立应用程序功能建模 Application Functionality Modeled

这一阶段您将微调已经创建的两个视图（**“打开申请单”(Open Requisitions)**和**“申请单批准”(Requisition Approval)**）。

现在两个视图都显示了所有的申请单，这与它们目标用户（intended user）的需求不符（而且，随着越来越多申请单输入到系统里面，这两个视图会变得难以使用）。在理想情况下，每个视图应该对申请单进行过滤并只显示与该视图相关的申请单：

- **打开申请单”（Open Requisitions**）视图应该显示已取消申请单之外的所有申请单。（也就是按以下条件过滤申请单：**状态 Status <>取消 Canceled**）
- **“申请单批准”（Requisition Approval**）视图应只显示新的申请单（**状态 Status=新 New**）

> 本阶段建模在上个阶段最后输入的**“Tutorial 7-8”**项目中进行。

用户建模 User Modeling

两个视图都包含一个能够把申请单填充申请单清单的过程。
我们从**“填充申请单批准清单”（Populate Requisition Approval List**）过程开始。
“填充申请单批准清单”（Populate Requisition Approval List）会在下面的位置重复使用：

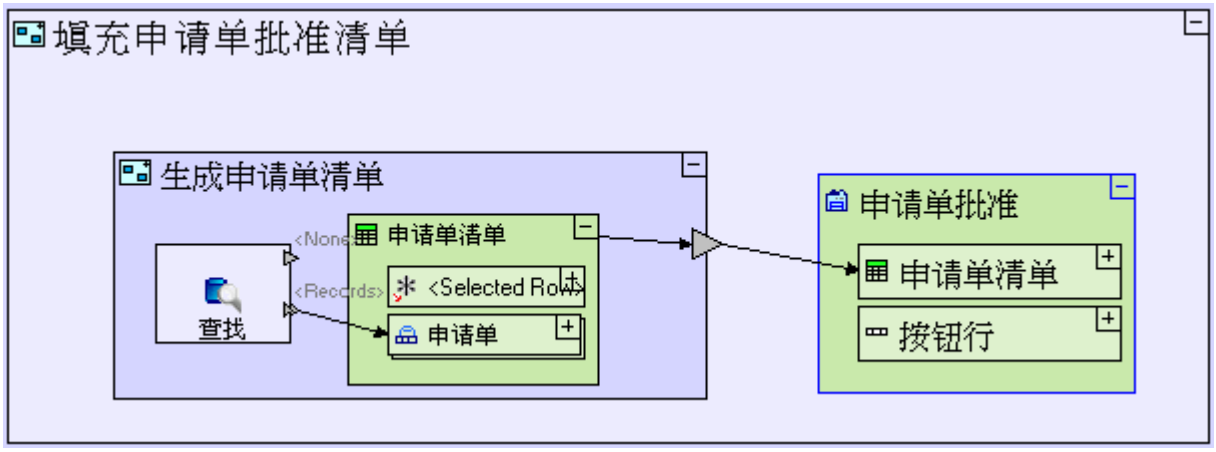
1. **“申请单批准”（Requisition Approval**）视图
2. **“申请单批准”（Requisition Approval/**）/**“批准申请单”（Approve Requisition**）按钮

3. “**申请单批准**”（**Requisition Approval/**）/“**退回申请单**”（**Reject Requisition**）按扭

因为重复使用,我们只需在这些位置的某一处编辑或修改“**填充申请单批准清单**”（**Populate Requisition Approval List**）。

在查找过程中添加一个触发器（以规定过滤的数值）Add a Trigger to the Find Process (to specify a value by which to filter)

“**填充申请单批准清单**”（**Populate Requisition Approval List**）现在应与下图类似:



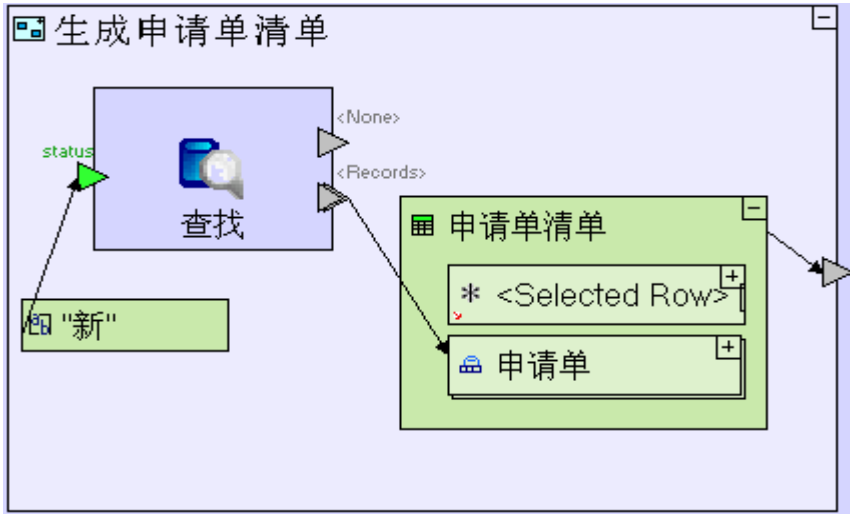
我们想让“**生成申请单清单**”（**Generate Requisition List**）子过程返回“**状态**”（**Status**）为“**新**”（**New**）的“**申请单**”（**Requisitions**）:

进入“**生成申请单清单**”（**Generate Requisition List**）
给“**查找**”（**Find**）添加一个“**触发器**”（**Trigger**）。（按[F2]）把它命名为 ***status***
从储存库（repository）（或大纲（outline））中拖拽一个“**新**”（**New**）常数，放到“**状态**”（**Status**）触发器旁边。

我们想在“**输入新申请单**”（**Enter New Requisition**）弹出窗口上的“**提交**”（**Submit**）按钮中，重复使用有的“**新**”（**New**）常数

创建流程（Flow），连接“**新**”（**New**）常数和“**查找/状态**”（**Find/Status**）触发器。

“生成申请单清单”（Generate Requisition List）过程应该与下图类似：



如果我们给“查找”（Find）过程添加一个触发器，并且触发器的名字与提取记录的表格中的字段（在这里，即“Status”字段）相匹配，那么“查找”(Find)就会只提取字段中包含提供给触发器的数值（在这里，即“New”数值）的记录。

> 更多“查找”（Find）相关信息，请参阅在线文档的“模板”（Templates）部分。

保存您的工作成果并在浏览器中查看应用程序。

切换到“申请单批准”（Requisition Approval）视图。

确认“申请单清单”（Requisition list）现在已被过滤为只显示“状态”(Status)为“新”(New)的申请单。

批准或退回一个申请单，并确认申请单已经从“申请单清单”（Requisition list）删除。

继续批准或退回申请单，直到只剩下一个申请单为止。

使用删除流程（清空表格）Use Remove Flow (to clear the table)

如果您现在批准或退回最后一个申请单，您就会看到这个申请单没有从显示中删除（尽管它在数据库中的状态已经改变了。您刷新一下浏览器显示就可以看到表格是空的）。

这是因为当我们为了刷新申请单清单运行“生成申请单清单”（Generate Requisition List）时，最后一个申请单被批准或退回之后，“查找”（Find）过程不能找到任何“状态=新”（Status=New）的记录，因此它就通过“<无>”(<None>)触发器（而不是通过通常的“<记录>”<Record>触发器）退出。<None>触发器没有从自身定义的流程（flow），因此过程就终止了，这也就意味着显示没有刷新。

在这种情况下，要刷新显示，我们需要定义退出<None>的流程并使用一种清除表格显示的特殊类型的流程“删除”(Remove)：

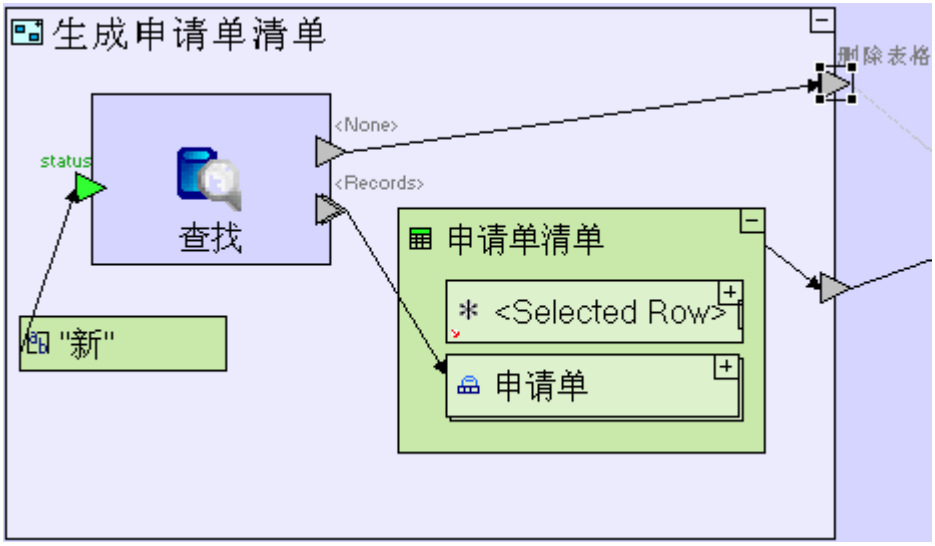
进入“**生成申请单清单**”（**Generate Requisition List**）。

给“**生成申请单清单**”（**Generate Requisition List**）添加一个退出槽（exit slot）。

把它命名为 “***删除表格***” ***(Clear Table)***

创建连接“**查找/<无>**”(**Find/<None>**)出口和“**生成申请单清单/删除表格**”(**Generate Requisition List/Clear Table**）出口的流程。

“**生成申请单清单**”（**Generate Requisition List**）模型应该与下图类似：

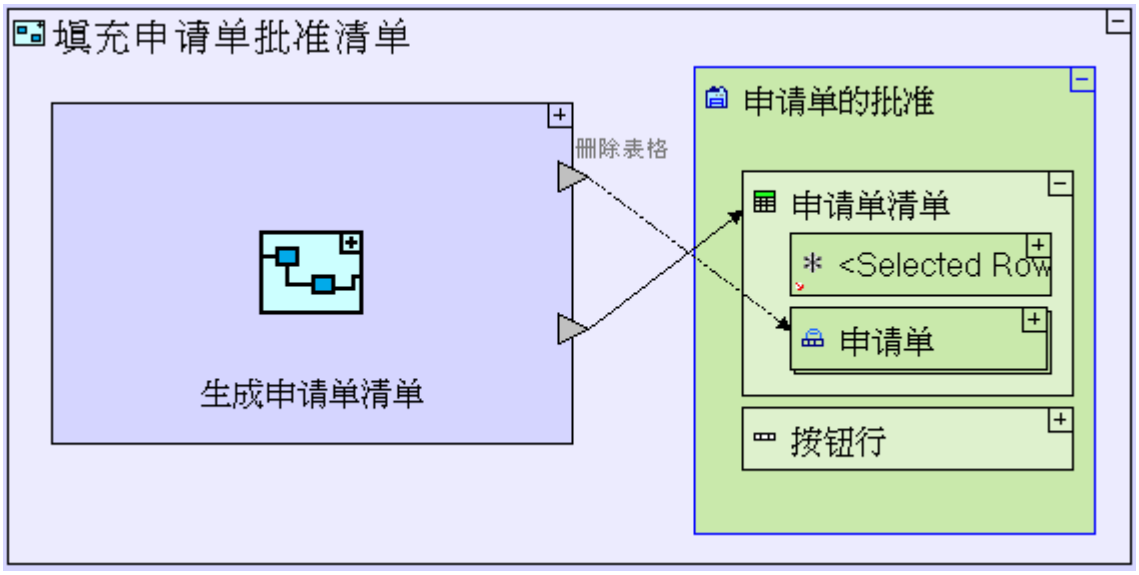


要清空显示，还必须添加“**删除**”（**Remove**）流程工具（flow tool），流程工具明确规定流程的目标必须清除现有单元（在这种情况下(在运行时)，现有单元就是我们刚刚批准或退回的那个申请单）。

进入“**填充申请单批准清单**”（**Populate Requisition Approval List**）。

选择“**删除**”（**Remove**）流程工具()。点击“**生成申请单清单/清空表格**（**Generate Requisition List/Clear Table**）出口以定义来源（source）。点击“**申请单批准/申请单清单/申请单**”（**Requisition Approval/Requisition List/Requisition**）数据库记录以定义目标。

“**填充申请单批准清单**”（**Populate Requisition Approval List**）模型应该与下图类似：



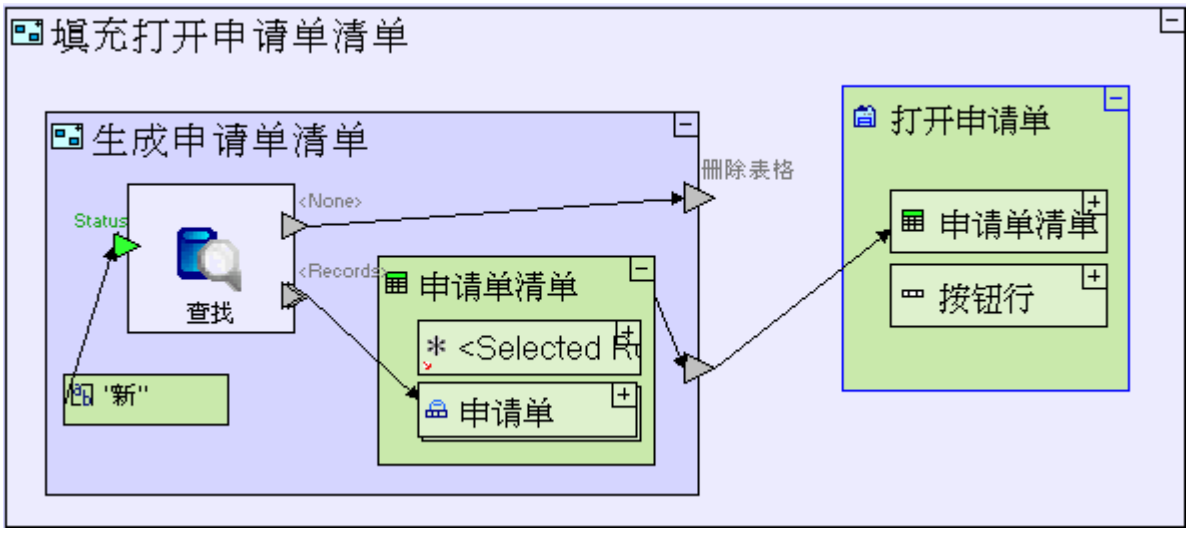
保存您的工作成果并在浏览器中查看应用程序。确保在表格中只有一个申请单的情况下，批准/退回申请单都会清空表格(如下面屏幕快照所示)：

注意：因为“**打开申请单**”（**Open Requisitions**）也重复使用“**生成申请单清单**”（**Generate Requisition List**），该表格会显示与状态为“**新**”（**New**）的申请单清单相同的申请单清单。我们接下来对此进行处理。

在另一个编辑器窗口中打开模型 Opening Models in a separate Editor Window

“**打开申请单**”（**Open Requisitions**）视图要求我们执行另一个不同的过滤器：“**状态<>取消**”（**Status<>Canceled**）。

“**填充打开申请单清单**”（**Populate Open Requisitions List**）现在外观应该如下图所示：



我们需要更新“**填充打开申请单清单**”（**Populate Open Requisitions List**）过程来执行那个不同的过滤器；由于实际过滤是在“**生成申请单清单**”（**Generate Requisition List**）子过程中进行的，（我们绝不能改变这一点，因为“**申请单批准**”（**Requisition Approval**）也使用这个过滤器），所以我们必须用一个新的（独立的）“**生成申请单清单**”（**Generate Requisition List**）来取代现有过程。

因为我们将要（部分地）重新创建已经在“**填充申请单批准清单**”（**Populate Requisitions Approval List**）中建模的功能，我们可以在一个独立的模型编辑窗口打开这个清单。这也就允许我们把清单作为以后建模的参考，在两个窗口中来回转换。

要在一个独立的窗口中打开“**填充申请单批准清单**”（**Populate Requisitions Approval List**），您可以：

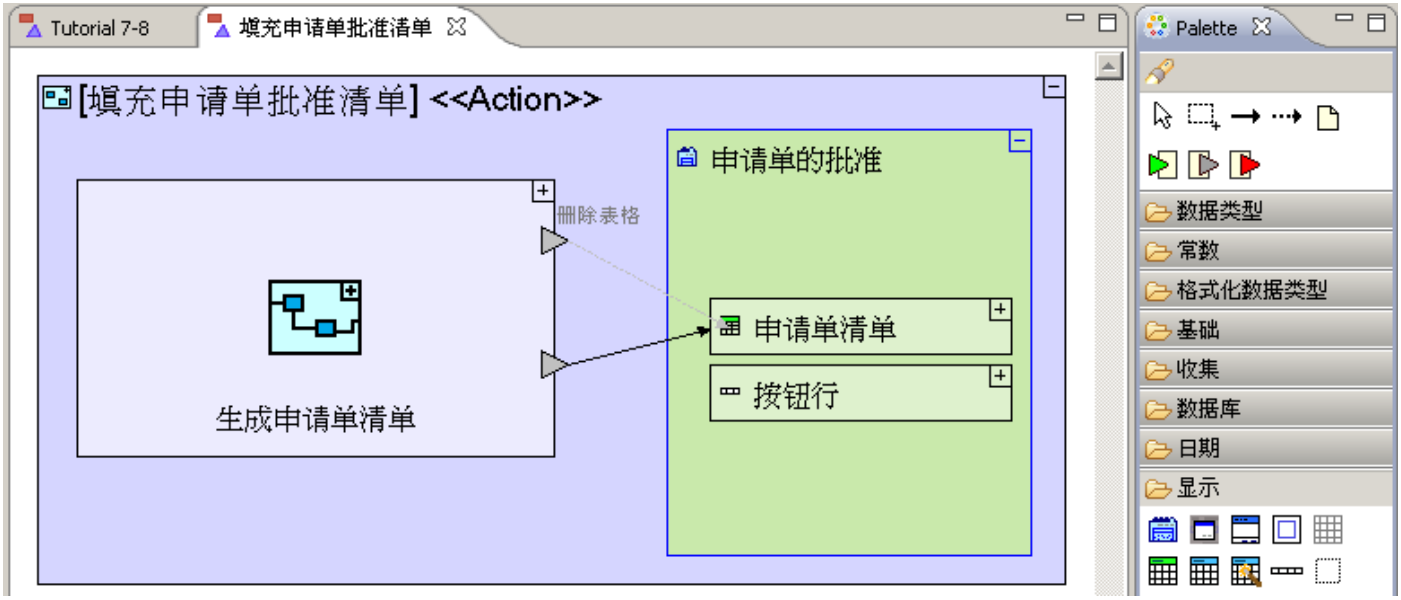
把“**填充申请单批准清单**”（**Populate Requisitions Approval List**）放进“**储存库管理器**”（**Repository Explorer**），并双击。

或者：

把“**填充申请单批准清单**”（**Populate Requisitions Approval List**）放进“**模型编辑器**”（**Model Editor**），右击，并选择“**在新标签中打开**”（**Open in a New Tab**）。

> 要把一个模型放入储存器，最快的方法是在编辑器中右击模型并选定模型。

您的模型编辑器现在应该显示两个编辑器窗口：一个是“**Tutorial 7-8**”根模型的编辑器窗口，另一个是“**填充申请单批准清单**”（**Populate Requisitions Approval List**）的编辑器窗口，如下面的屏幕快照所示：



从模型中删除单元 Remove an Element from the Model

返回继续编辑完整模型：

点击“Tutorial 7-8”编辑器标签。

把“**填充打开申请单清单**”（Populate Open Requisitions List）放到“**打开申请单**”（Open Requisitions）中，开始为所做的修改建模：

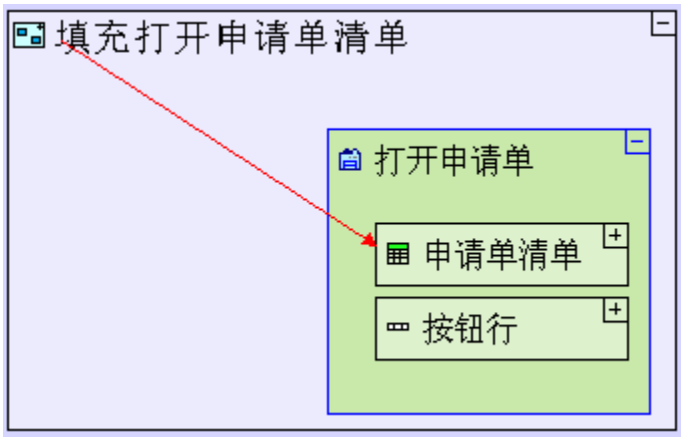
进入“**填充打开申请单清单**”（Populate Open Requisitions List）。

正如前面所解释的那样，任何一种模型都可以（只要它们是“**申请单批准**”（Requisition Approval）版本的模型）。

现在删除废弃的过程：

选择并删除“**生成申请单清单**”（Generate Requisition List）。

模型外观应该与下图类似：



发生建模错误时，连接“**生成申请单清单**”（Generate Requisition List）和“**打开申请单**”（Open Requisitions）显示数据单元的流程箭头（flow arrow）就会变成红红色。我们一会就会对此进行处理。

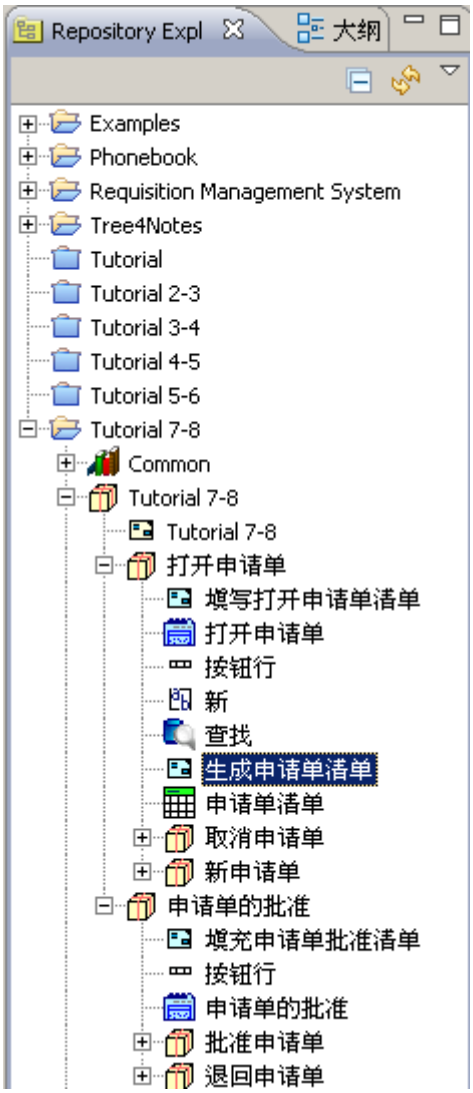
了解模型包装和命名 Understanding Model Packaging and Naming

尽管我们已经把**“生成申请单清单”**（Generate Requisition List）过程从**“打开申请单”**（Open Requisitions）视图中删除，也就是说它现在只在**“申请单批准”**（Requisition Approval）视图中使用，但是原来创建它的数据包（package）(亦即**“打开申请单”**（Open Requisitions）包)中还有**“生成申请单清单”**（Generate Requisition List）模型（我们可以在**“模型储存器”**（Model Repository）中查看确认）。这对运行时间的应用程序的运行没有任何影响，因为包装只是为了方便开发者，而且任何一个数据包中的模型都可能包含任何其他数据包的单元。

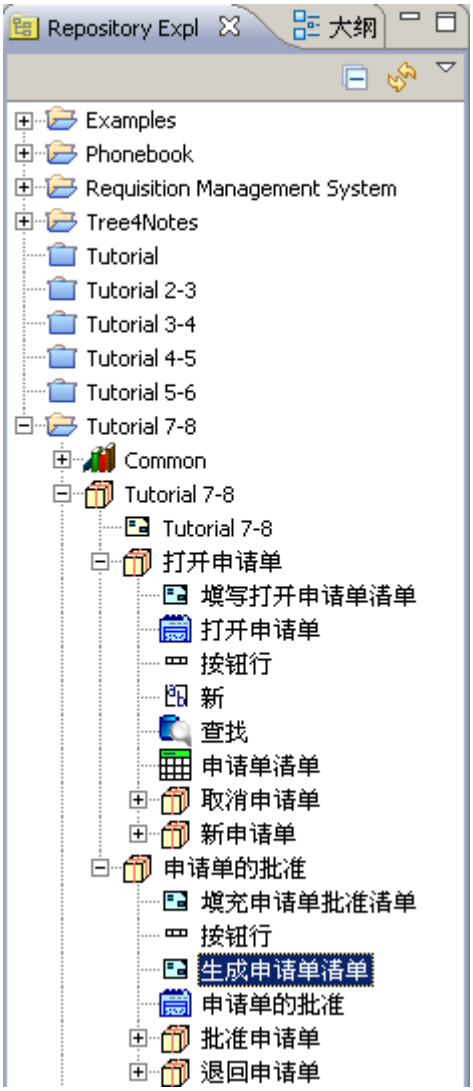
但这对我们后续的建模却有个很有趣的副作用：下来我们计划在**“填充打开申请单清单”**（Populated Open Requisitions List）中创建一个和现有模型相似的新**“生成申请单清单”**（Generate Requisition List）模型—这意味着在缺省情况下,这个模型创建的位置也是在**“打开申请单”**（Open Requisitions）数据包中（也就是现有**“生成申请单清单”**（Generate Requisition List）模型的同一个数据包），这样就有了一个叫**“生成申请单清单 2”**（Generate Requisition List 2）的新模型，这个新模型的作用没有不同，但为了保持整洁和整齐，我们想避免这种情况发生。

要解决这种情况，就要把现有的**“生成申请单清单”**（Generate Requisition List）模型移到**“申请单批准”**（Requisition Approval）数据包中:

把**“生成申请单清单”**（Generate Requisition List）放入**“打开申请单”**（Open Requisitions）数据包中（如下面屏幕快照所示）。



把**“生成申请单清单”**（Generate Requisition List）拖放到**“申请单批准”**（Requisition Approval）包上。



现在创建可选**“生成申请单清单”**（Generate Requisition List）过程：

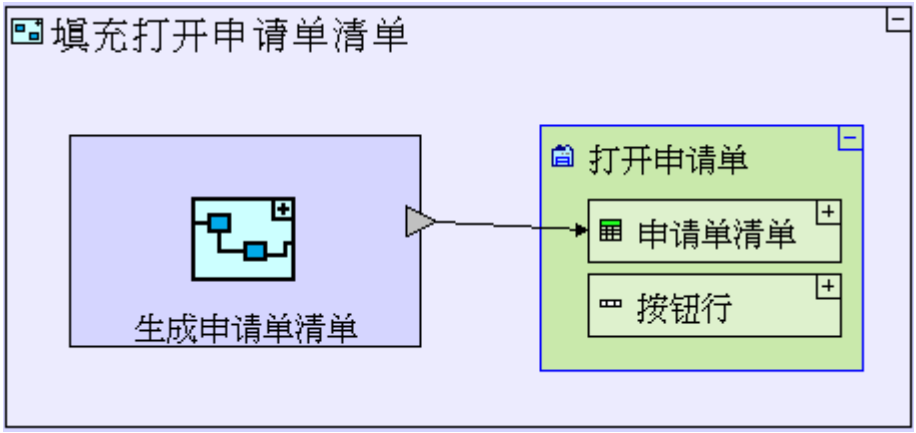
把“**基本/动作**”（Basic/Action）模板拖入“**填充打开申请单清单**”（Populate Open Requisition List）。把它命名为 “***生成申请单清单***” *Generate Requisition List*

给新过程添加一个“**退出**槽”（**Exit** slot）。

选择红色流程箭头，将它的“源锚点”拖进新出口。

如果您所有的名称全部输入正确，红色箭头会自动重新连接，就不需要上面的最后一步。

“填充打开申请单清单”（Populate Open Requisition List）外观应该与下图类似：

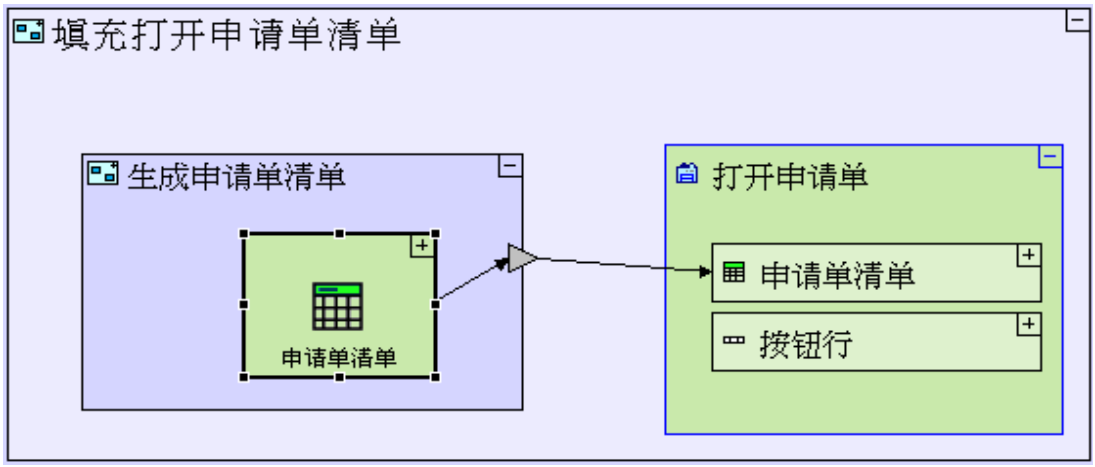


现在，我们再看一下现有的“**生成申请单清单**”（**Generate Requisition List**）（在我们在一个独立的模型编辑器窗口中打开的“**填充申请单批准清单**”（**Populate Requisition Approval List**）过程中），从而为新的“**生成申请单清单**”（**Generate Requisition List**）建模。

我们需要添加一个“**申请单清单**”（**Requisition List**）显示数据单元：

从储存器（repository）（或大纲（outline））中拖放“**申请单清单**”（**Requisition List**）到“**生成申请单清单**”（**Generate Requisition List**）上。
添加连接“**申请单清单**”（**Requisition List**）和“**生成申请单清单**”（**Generate Requisition List**）出口（exit）的流程（flow）。

“**填充打开申请单清单**”（**Populate Open Requisition List**）外观应该与下图类似：



使用高级查找模板（使用复杂规则过滤记录） Use the Advanced Find template (to filter records using a complex criteria)

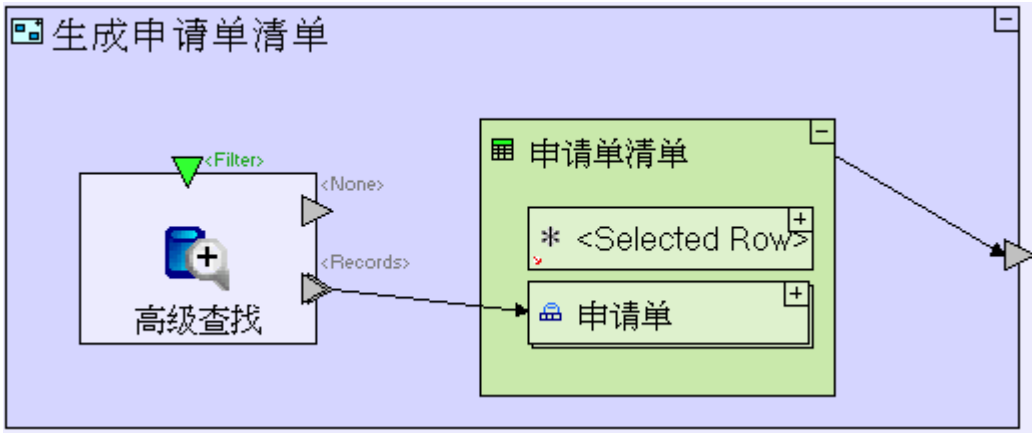
如果我们再看一下“**填充打开申请单清单**”（**Populate Open Requisition List**）窗口，就会发现这里缺失了提取数据的“**查找**”（**Find**）过程，但正如之前解释的那样，我们需要执行一个“不对等”的规则，“**查找**”（**Find**）模板又不支持这个规则。因而我们要使用“**高级查找**”（**Advanced Find**）模板：

选择“**数据库/高级查找**”（**Database/Advanced Find**）模板（），把它拖放到“**生成申请单清单**”（**Generate Requisition List**）中。

“**高级查找**”（**Advanced Find**）和“**查找**”（**Find**）的共同点是它也有两个出口（exits）—“**<无>**”（**<None>**）和“**<记录>**”（**<Records >**），因此：

创建连接“**高级查找/< 记录>**”（**Advanced Find/< Records >**）出口和“**申请单清单**”（**Requisition List**）中的“**申请单**”（**Requisition**）数据结构的流程（Flow）。

现在“**生成申请单清单**”（**Generate Requisition List**）的外观应该是这样的：



我们想要过滤掉“**状态**”（**Status**）为“**取消**”(**Canceled**)的记录。

作为定义表格中一个或多个字段的限制条件的文本串（text string），过滤器是通过“**高级查找**”（**Advanced Find**）的“**<过滤器>**”(**<Filter>**)触发器提供的。在此处的例子中，限制条件应为“**状态<>‘取消’**”（**Status<>‘Canceled’**）。

简单的“**高级查找**”（**Advanced Find**）仅依赖缺省的“**<过滤器>**”(**<Filter>**)触发器，但为了提供更多的灵活性，我们可以添加一些为过滤器本身提供参数的触发器：

给“**高级查找**”（**Advanced Find**）添加一个“**触发器**”（**Trigger**）。

把它命名为 ***status***（按[F2]键）

现在我们可以把过滤器定义为”“**Status<>${Status}**”，在运行时间，“**${Status}**”会被传到“**状态**”（**Status**）触发器的数值所替换。

> 注意：我们使用**$**符号和{curly}大括号是为了表示这是一个参数。

定义过滤条件本身：

从选项板（palette）中添加一个“**常数/文本**”（**Constant/Text**）。

命名为 ***status<>${status}***。

把它连接到“**高级查找/<过滤器>**”（**Advanced Find/<Filter >**）触发器。

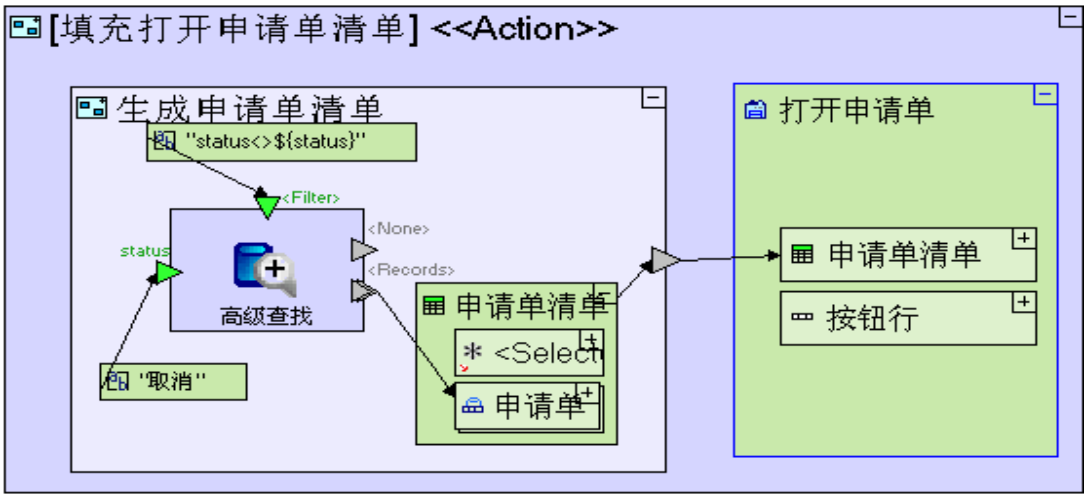
现在设置过滤器的参数“**状态**”（**Status**）：

把“**取消**”（**Canceled**）常数(从“**取消申请单**”（**Cancel Requisition**）按钮中)拖入储存器（或大纲）中。

把它与“**高级查找/< 状态>**”（**Advanced Find/<Status >**）触发器连接起来。

> 更多“**高级查找**”（**Advanced Find**）的相关信息，请参阅在线文档中的“**模板**”（**Template**）部分。

“**填充打开申请单清单**”（**Populate Open Requisition List**）模型外观应该与下图类似：



保存您的工作成果并在浏览器中查看应用程序。

确保“**打开申请单**”（**Open Requisitions**）中的“**申请单清单**”(**Requisition List**)不显示任何状态为“**取消**”（**Canceled**）的申请单。

从清单中选择一个申请单并按住“**取消申请单**”（**Cancel Requisition**）。“**申请单**”(**Requisition**)应该会从清单中消失。

这再一次展示了重新使用是怎样起作用的：在一个地方（在“**打开申请单**”（**Open Requisitions**）中）对“**填充打开申请单清单**”（**Populate Open Requisition List**）模型所做的修改，也会应用于被重新使用的其他地方（在此处，即“**打开申请单/取消申请单**”）。

完成阶段 8 Completing Stage 8

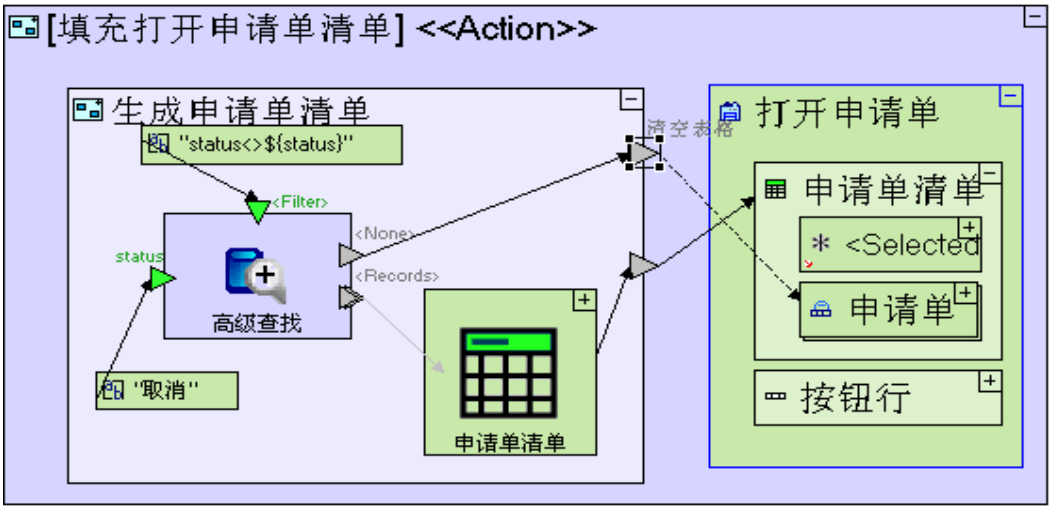
现在我们输入项目样本“**Tutorial 8-9**”并把它作为本教程下一个阶段的基础。

> 想再了解如何输入一个项目样本，请参见**阶段 2** 最后“**输入项目样本**”的部分。

这个项目样本具备到目前为止我们建立的模型的所有功能。

这个项目样本也包括以下一些其他的功能：

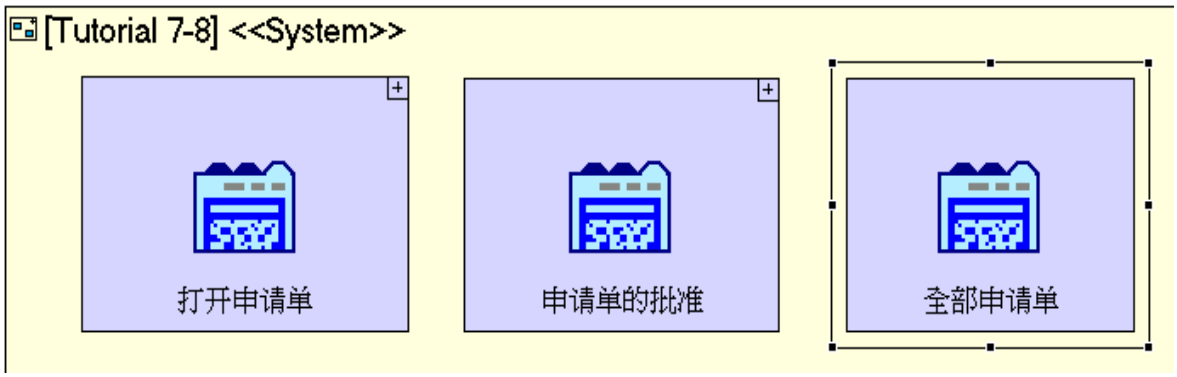
1. 在“**填充打开申请单清单/生成申请单清单**”（Populate Open Requisition List/Generate Requisition List）没有找到申请单时，清空“**申请单清单**”（Requisition List）显示。



如何建模	位置
给“**生成申请单清单**”（Generate Requisition List）添加“**出口**槽”（**exit** slot），把它命名为 “***清空表格”(Clear Table)*** 连接“**高级查找/<无>**”（Advanced Find/<None>）和“**生成申请单清单/清空表格**” （Generate Requisition List/Clear Table）。	“**填充打开申请单清单**”（Populate Open Requisition List）/“**生成申请单清单**”（Generate Requisition List）过程

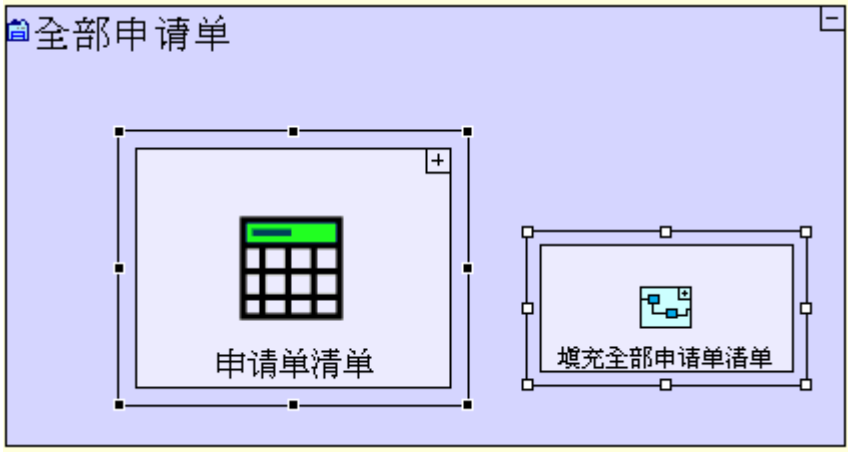
创建连接“**生成申请单清单/清空表格**”（**Generate Requisition List/Clear Table**）和“**打开申请单/申请单清单/申请单**”（**Open Requisitions/Requisition List/Requisition**）的“删除”流程。	“**填充打开申请单清单**”（**Populate Open Requisition List**）过程

2. 添加另一个显示系统中的所有申请单的视图“**全部申请单**”（**All Requisitions**），不论申请单的状态（Status）如何。



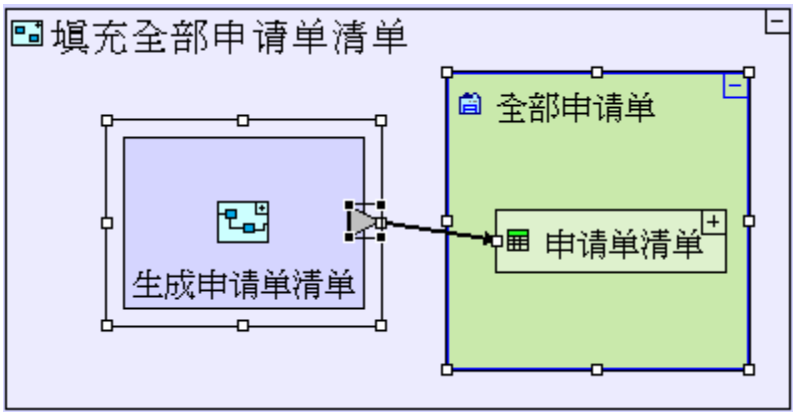
如何建模	位置
添加“**显示/视图**”（Display/View）。把它命名为 “***全部申请单”（All Requisitions）***	**Tutorial 7-8** 根（root）

3.“**全部申请单**”（**All Requisitions**）视图建模（继续）。



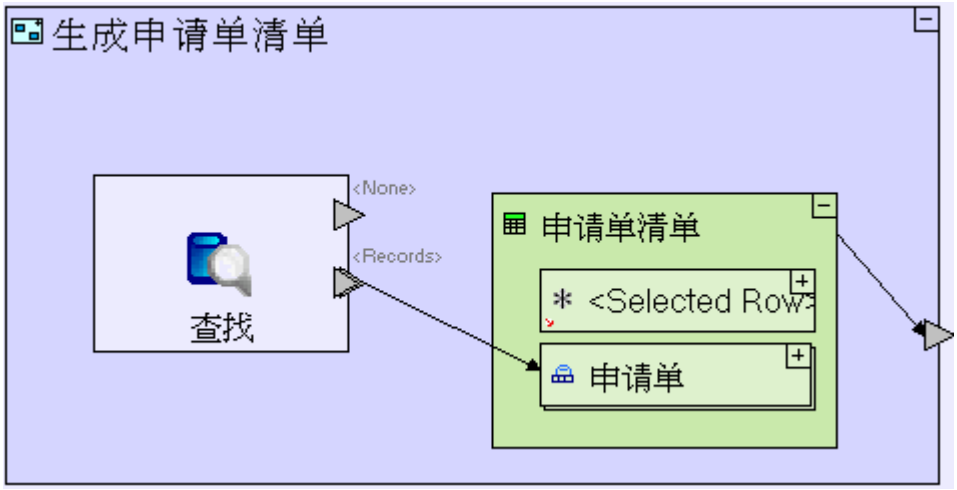
如何建模	位置
从大纲（outline）/储存器(repository)中拖拽“**申请单清单**”（**Requisition List**）简单表格。 添加“**基本/动作**”（**Basic/Action**）。把它命名为 “***填充全部申请单清单”(Populate All Requisitions List)***。	“**全部申请单**”（**All Requisitions**）视图

4. “**全部申请单**”（**All Requisitions**）/“**填充全部申请单清单**”（**Populate All Requisitions List**）过程建模。



如何建模	位置
添加一个“**全部申请单**”（**All Requisitions**）视图的上级索引。 添加“**基本/动作**”（**Basic/Action**）。把它命名为 “***生成申请单清单”(Generate Requisition List)*** 给“**生成申请单清单**”(**Generate Requisition List**)过程 添加一个“**出口**”（**Exit**）。 连接这个出口和“**全部申请单/申请单清单**”(**All Requisitions/Requisition List**）数据结构。	“**填充全部申请单清单**”（**Populate All Requisitions List**）过程

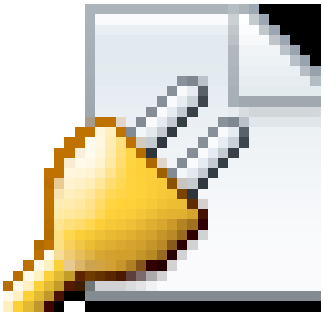
5. “**全部申请单/填充全部申请单清单/生成申请单清单**”(**All Requisitions/Populate All Requisitions List/Generate Requisition List**）过程建模。



如何建模	位置
添加“**数据库/查找**”（**Database/Find**）。 从大纲（outline）/储存器(repository)中拖拽“**申请单清单**”（**Requisition List**）。 连接“**查找/<记录>**”（**Find/<Records>**）和“**申请单清单/申请单**”（**Requisition List/ Requisition**）。 连接“**申请单清单**”（**Requisition List**）和“**生成申请单清单**”（**Generate Requisition List**）出口。	“**生成申请单清单**”（**Generate Requisition List**）过程

您现在可以开始学习**阶段 9**，在第 9 阶段，我们将从视觉效果上重新排列目前已经创建的全部视图，根据系统不同的目标用户对视图进行分组。

实例 See It Live



点击此处在另一个窗口里面打开实际项目。

阶段 9--在透视图中排列视图 Stage 9-Arranging Views into Perspectives

阶段目标 Stage Goals

覆盖的 Tersus 概念 Tersus Concepts Covered

完成本阶段后，您将会熟悉以下概念：

建模概念：透视图（Perspectives）
建模技术：视图分组（Grouping views together）
有用的显示模板：系统（Systems）（用于定义透视图）

建立应用程序功能模型 Application Functionality Modeled

除了已经建模的三个视图外，在后面的几个阶段我们将为更多的视图建立模型。采购人员和船运文员将会使用这些附加的模型。根据使用者不用的职能（如雇员、经理和采购人员）对视图进行组织和分组是一个好主意。

本阶段建模在上个阶段最后输入的项目“**Tutorial 8-9**”中进行。

用户建模 User modeling

应用程序现在外观应该是这样的：

“**打开申请单**”（Open Requisitions）、“**申请单批准**”（Requisition Approval）和“**全部申请单**”（All Requisitions）三个视图被分在同一组。我们想对这三个视图重新分组，使全部雇员使用的“**打开申请单**”（Open Requisitions）和“**全部申请单**”（All Requisitions）与经理使用的“**申请单批准**”（Requisition Approval）分离开来，如下面屏幕快照所示：

这两个屏幕快照主要的视觉差异在于多个“**透视图**”（Perspectives）的使用，以竖着排列的标签（tabs）的形式显示于屏幕左侧（在这个例子中，标签有两个：**雇员**（Employers）和**经理**（Manager））。

透视图用于对多个视图进行分组。对于没有在透视图中定义的视图，默认动作是把所有的视图都集合在同一个透视图中，透视图的名称即为根模型的名称。

（正如第一个屏幕快照所展示的一样，在这个例子中，根模型的名称是“Tutorial 8-9”）。

添加一个透视图（雇员）Adda perspective (Employee)

按下列步骤定义“**雇员”（Employee）**透视图:

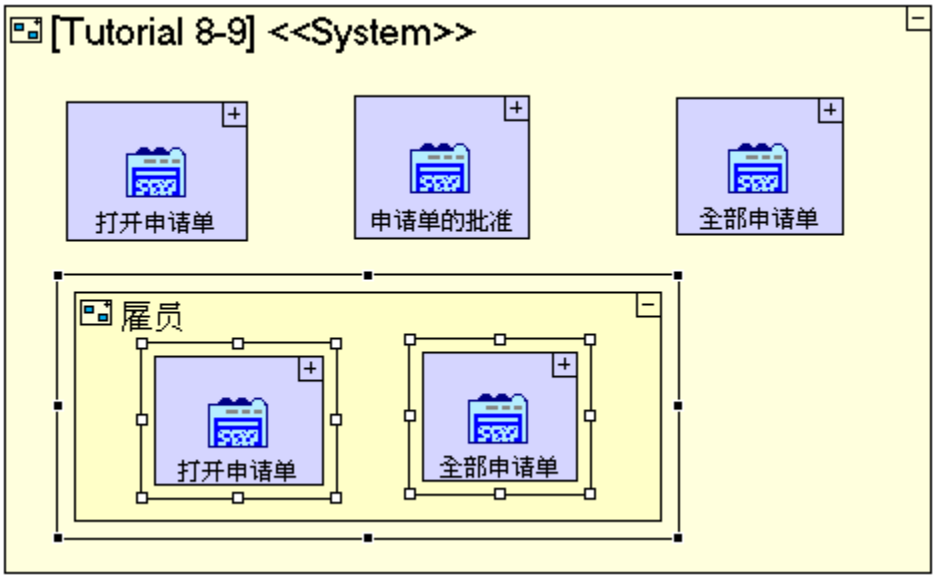
进入“**Tutorial 8-9**”根模型。

添加一个“**基本/系统**”（**Basic/System**）模板（）。把它命名为 “***雇员” (Employee)***

放大到“**雇员”（Employee）**系统。

从储存器（repository）（或大纲（outline））中把“**打开申请单”（Open Requisitions**）和“**全部申请单”（All Requisitions**）拖放到“**雇员”（Employee）**系统里面。

模型外观应该与下图类似:



保存您的工作成果并在浏览器中查看应用情况。应用程序外观应该是这样的:

我们没有从根模型中删除那三个视图；因此它们仍出现在以根模型命名的缺省

透视图中。

现在，点击新“**雇员**”（**Employee**）透视图。

您看到的应该是这个样子：

我们已正确定义了“**雇员**”（**Employee**）透视图。

> 现在这样定义的应用模型体现了这样一个事实，即视图和其他模型类型一样，都可在一个应用程序中被重新使用（就视图而言，也就是在不同透视图中被重新使用）。

删除缺省透视图 Remove the Default Perspective

要删除缺省透视图“**Tutorial 8-9**”，按以下步骤操作：

进入“**Tutorial 8-9**”根模型。

逐一选择“**打开申请单**”（**Open Requisitions**）、“**申请单批准**”（**Requisition Approval**）和“**全部申请单**”（**All Requisitions**）三个视图，并将它们从视图中删除。

> 虽然我们已经从其母模型中删除了“**申请单批准**”（**Requisition Approval**）视图。但这并不是个问题，因为“**申请单批准**”（**Requisition Approval**）还在储存器里面。

添加附加透视图（经理）Add an additional Perspective (Manager)

最后，我们来创建“**经理**”（**Manager**）透视图。

请按以下步骤定义“**经理**”（**Manager**）透视图：

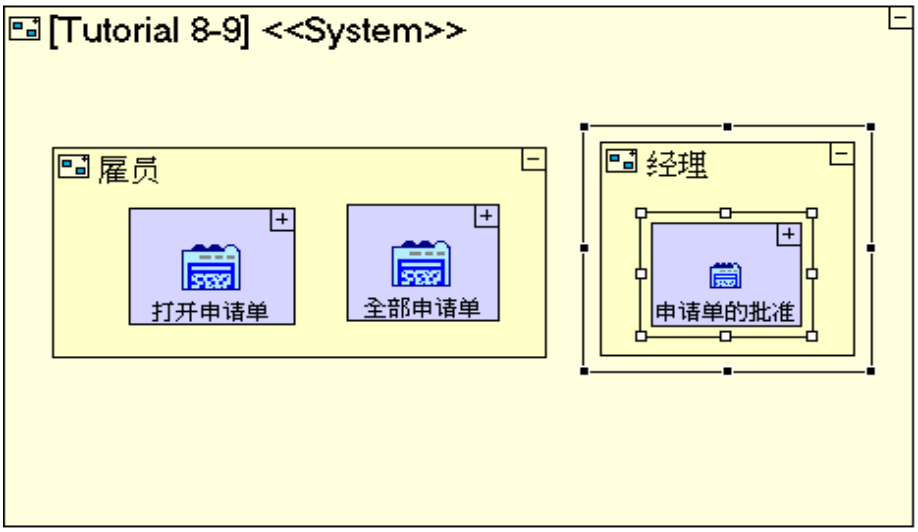
进入“**Tutorial 8-9**”根模型。

添加一个“**基本/系统**”（**Basic/System**）模板。把它命名为 “**经理**”（***Manager***）。

进入“**经理**”（**Manager**）系统。

从储存器（repository）中把“**申请单批准**”（**Requisition Approval**）视图拖入“**经理**”（**Manager**）系统中。

模型外观应该如下图所示：



保存您的工作成果并在浏览器中查看应用程序。应用程序外观应该如下图所示：

我们在后面会使用把视图分组（组成透视图）技术和“用户/权限”（User/Permissions）系统的内置支持，以规定用户登录系统之后显示哪些透视图。

完成阶段 9 Completing Stage 9

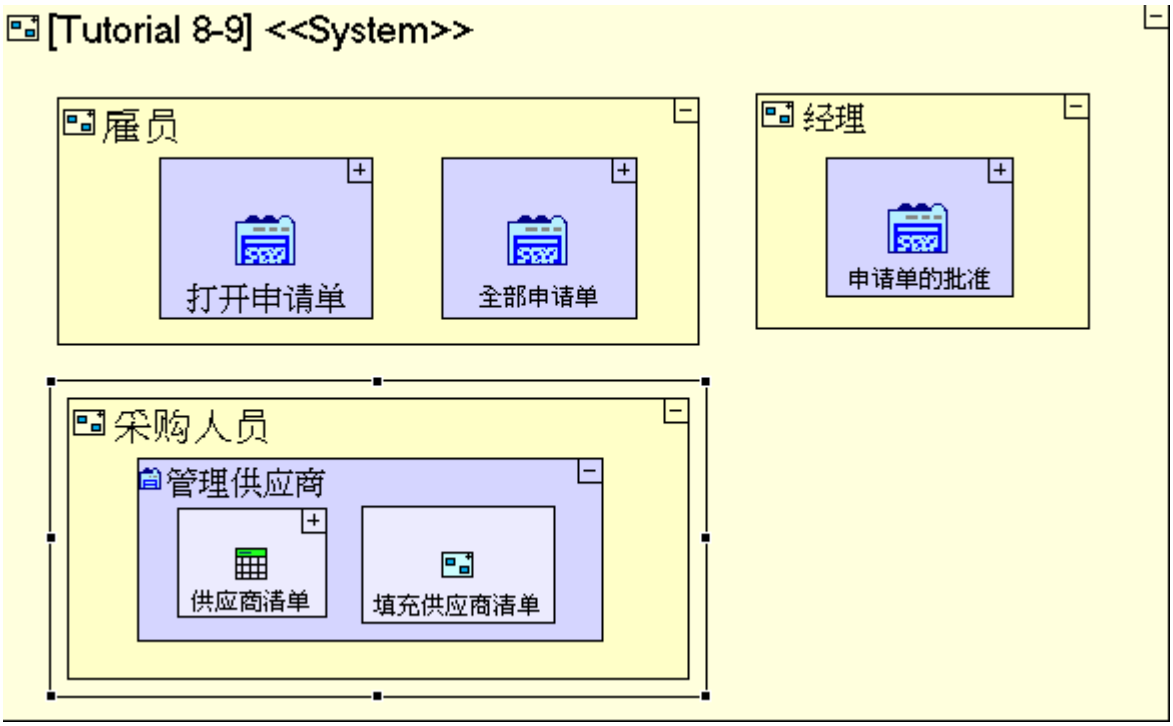
现在我们输入项目样本“**Tutorial 9-10**”并把它作为“Tutorial 8-9”下一个阶段

的基础。

> 想再了解如何输入一个项目样本，请参见**阶段 2** 最后“**输入项目样本**”部分。

这个项目样本具备到目前为止我们建立的模型的所有功能。
这个项目样本也包括其他以下的功能：

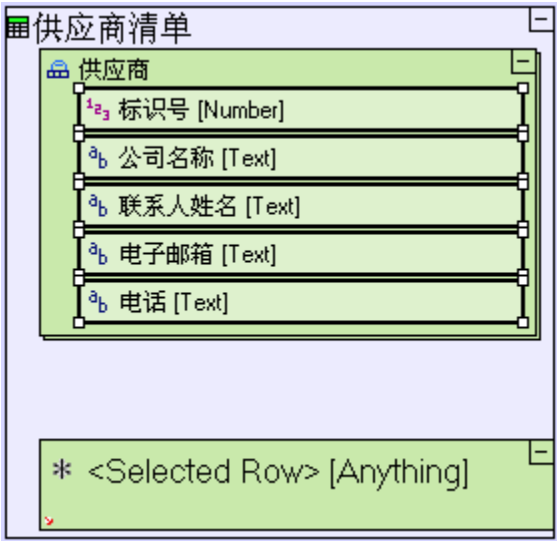
1. 添加一个新的“**采购人员**”（**Purchaser**）透视图和一个“**管理供应商**”（**Manage Suppliers**）视图，包括一个表格和一个“填充”（“Populate”）过程。



如何建模	位置
添加一个“**基本/系统**”（**Basic/System**）。把它命名为 ***“采购人员”（Purchaser）***	“Tutorial 8-9”根

添加一个“**显示/视图**”（**Display/View**）。把它命名为 ***“管理供应商”（Manage Suppliers）***	“**采购人员**”（**Purchaser**）系统
添加一个“**显示/简单表格**”（**Display/Simple Table**）。把它命名为 ***“供应商清单”（Supplier List）*** 添加一个“**基本/动作**”（**Basic/Action**）。把它命名为 ***“填充供应商清单”（Populate Supplier List）***	“**管理供应商**”（**Manage Suppliers**）视图

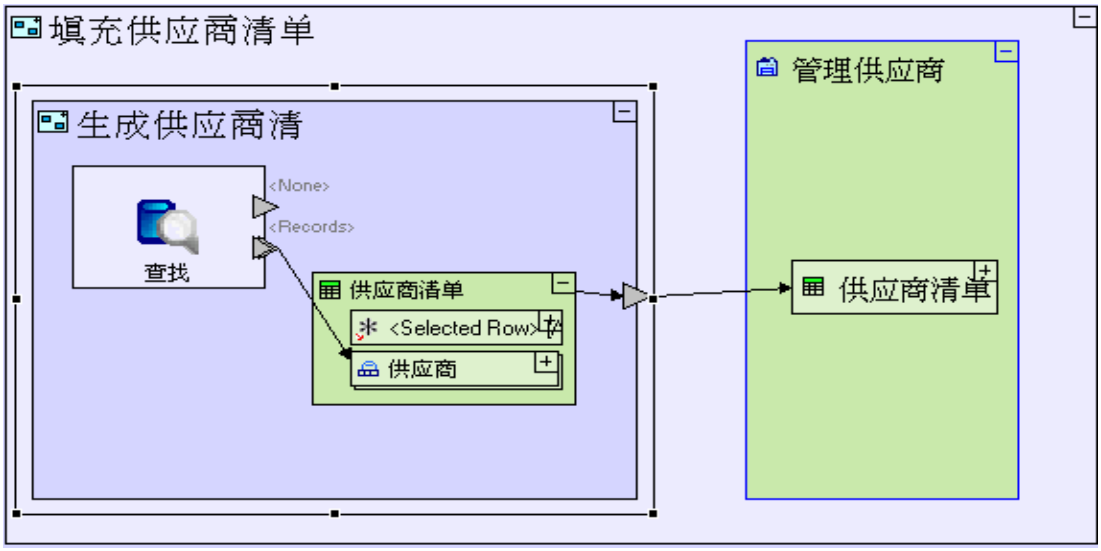
2. 定义一个新的数据库表格“**供应商**”（**Supplier**）。



如何建模	**位置**
添加一个“**数据类型/数据库记录**”（**Data Types/Database Record**）。把它命名为***供应商 Supplier***并把它设置为“**可重复**”（**repetitive**）。 选择***供应商*->特性**,把 tablename 命名 ***“供应商”***(Supplier)	“**供应商清单**”（**Supplier List**）表格
添加下列字段（圆括号内为数据类型） **“标识号(数字)”（*Id*（Number））** **“公司名称（文本）”（*Company Name*（Text））** **“联系人姓名（文本）”（*Contact Name*（Text））** **“电子邮箱（文本）”（*Email*（Text））** **“电话（文本）”（*Phone*（Text））** 选择**标识号->特性(properties)**，点击  把它命名为	“**供应商**”（**Supplier**）数据结构

“columnName” 把 “columnName”命名为 **“Id”** 选择**公司名称->特性(properties)**，点击  把它命名为 **“columnName”** 把 “columnName”命名为 **“company_name”** 选择**联系人姓名->特性(properties)**，点击  把它命名为 **“columnName”** 把 “columnName”命名为 **“contact_name”** 选择**电子邮箱->特性(properties)**，点击  把它命名为 **“columnName”** 把 “columnName”命名为 **“email”** 选择**电话->特性(properties)**，点击  把它命名为 **“columnName”** 把 “columnName”命名为 **“phone”**	

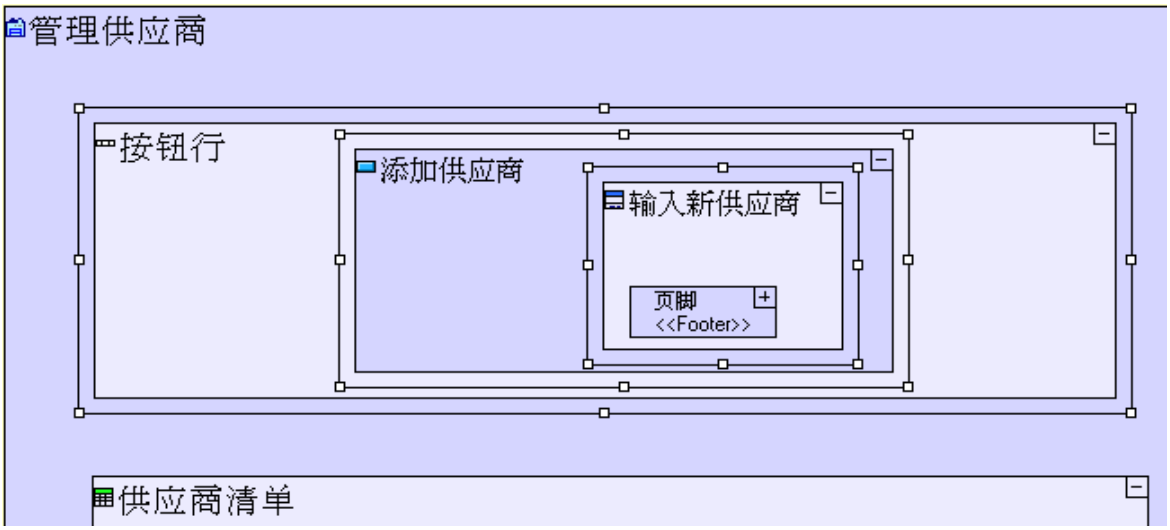
3.给“**填充供应商清单**”（**Populate Supplier List**）过程建模以启动供应商清单显示。



如何建模	位置
添加一个“**基本/动作**”（**Basic/Action**）。把它命名为“***生成供应商清单***”（***Generate Supplier List***）并给	“**填充供应商清单**”（**Populate Supplier**

它添加一个出口（exit）。 添加一个“**管理供应商**”（**Manage Suppliers**）视图的上级索引。 连接“**生成供应商清单**”（**Generate Supplier List**）出口和“**管理供应商/供应商清单**”（**Manage Suppliers/Supplier List**）。	**List**）过程
添加“**数据库/查找**”（**Database/Find**）。 从储存器/大纲中拖拽“**供应商清单**”（**Supplier List**）显示，以把“**供应商清单**”（**Supplier List**）显示作为一个数据结构重新使用。 连接“**查找/<记录>**”（**Find/<Records>**）和“**供应商清单/供应商**”（**Supplier List/Supplier**）。 连接“**供应商清单**”（**Supplier List**）和“**生成供应商清单**”（**Generate Supplier List**）出口。	“**生成供应商清单**”（**Generate Supplier List**）

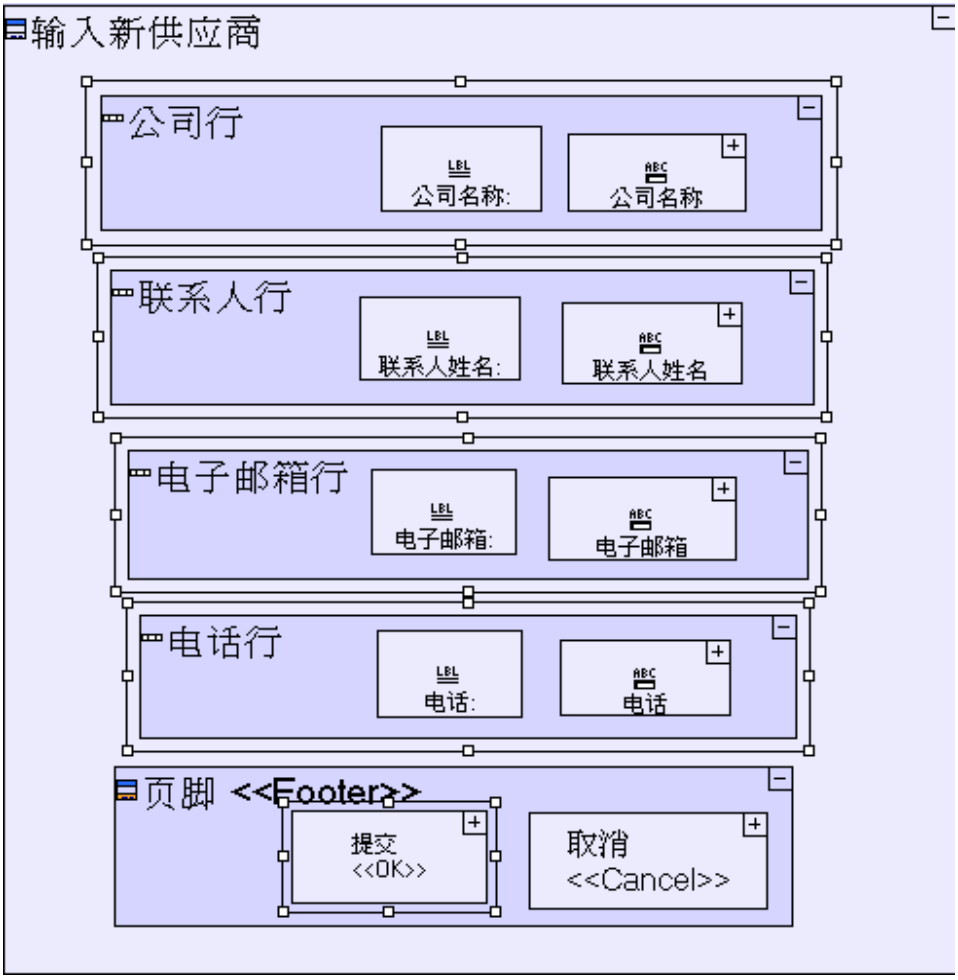
4. 给“**管理供应商**”（**Manage Suppliers**）视图添加一个有“**添加供应商**”（**Add Supplier**）按钮的“**按钮行**”（**Button Row**），“**添加供应商**”按钮含有“**输入新供应商**”（**Enter New Supplier**）弹出窗口。



如何建模	位置
添加一个“**显示/行**”（**Display/Row**）。 把它命名为 “***按钮行***” ***(Button Row)***	“**管理供应商**”（**Manage Suppliers**）视图
添加一个“**显示/按钮**”（**Display/Button**）。 把它命名为 “***添加供应商***” ***(Add Supplier)***	“**按钮行**”（**Button Row**）
添加一个“**显示/弹出窗口**”（**Display/Popup**）。	“**添加供应商**”（**Add**

把它命名为 *“输入新供应商”（Enter New Supplier）*	**Supplier**）按钮

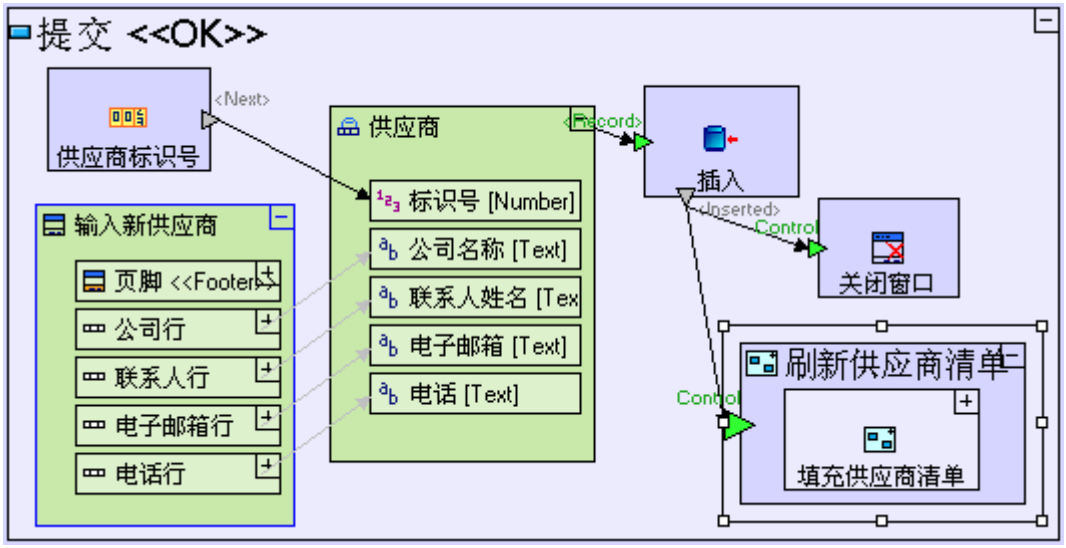
5. 给“**输入新供应商**”（**Enter New Supplier**）弹出窗口添加显示单元。



如何建模	位置
添加以下显示单元（圆括号中的模板）： **“公司行（行）”（*Company Row*（Row））** **“公司名称:（标签）”（*Company Name:*（Label））** **“公司名称:(文本输入字段)”（*Company Name*(Text Input Field））** **“联系人行（行）”（*Contact Row*（Row））** **“联系人姓名:（标签）”（*Contact Name:*（Label））** **“联系人姓名:（文本输入字段）”（*Contact Name*（Text Input Field））**	**“输入新供应商”**（Enter New Supplier）弹出窗口

“电子邮箱行（行）”（*Email Row*（Row）） “电子邮箱:（标签）”（*Email:*（Label）） “电子邮箱:（文本输入字段）”（*Email*（Text Input Field）） “电话行（行）”（*Phone Row*（Row）） “电话:（标签）”（*Phone:*（Label）） “电话:（文本输入字段）”（*Phone*（Text Input Field））	
将“OK”按钮重新命名为“提交”（Submit）	“页脚”（footer）页脚（footer）

6. **操作“输入新供应商/提交”**（Enter New Supplier/Submit）按钮（与我们在教程开始的时候创建的**“输入新申请单/提交”**（Enter New Requisition/Submit）按钮类似）。



如何建模	位置
添加一个“输入新供应商”（Enter New Supplier）弹出窗口的上级参考。 添加一个**“数据库/序列号”**（Database/Sequence Number）。把它命名为 ***“供应商标识号”(Suppliers Id)***	“提交”（Submit）按钮

重新使用储存器/大纲中的**“供应商”（Suppliers）**数据库记录。 连接**“供应商 Id/<下一个>”（Supplier Id/<Next>）**和**“供应商/Id”（Supplier/Id）。** 添加以下流程（flows）（来源在**“输入新供应商/供应商清单”（Enter New Supplier/Supplier List）**上级参考中，目标在**“供应商”（Suppliers）**数据结构中）： • **“公司名称/.值”（Company Name/.value）**到**公司名称”（Company Name）** • **“联系人姓名/.值”（Contact Name/.value）**到**“联系人姓名”（Contact Name）** • **“电话号码/.值”（Phone Number /.value）**到**“电话”（Phone）** • **“电子邮箱地址/.值”（Email Address/.value）**到**“邮箱”（Email）** 添加**“数据库/插入”（Database/Insert）**。 连接**“供应商”（Supplier）**数据结构和**“插入/<记录>”（Insert/<Record>）**。 添加**“基本/动作”（Basic/Action）**。把它命名为***“刷新供应商清单”（Refresh Supplier List）***。给它添加一个**“控制”（Control）**触发器。 连接**“插入/<插入>”（Insert/<Inserted>）**和**“刷新供应商清单/控制”（Refresh Supplier List/Control）**。 添加**“显示动作/关闭窗口”（Display Actions/Close Window）**。给它添加一个**“控制”（Control）**触发器。 连接**“插入/<插入>”（Insert/<Inserted>）**和**“关闭窗口/控制”（Close Window/Control）**。	
重新使用从大纲/储存器的**“填充供应商清单”（Populate Supplier List）。**	**“刷新供应商清单”（Refresh Supplier List）**过程。

您现在可以开始学习**阶段 10**。在**阶段 10** 里面，我们将会为一个把 excel 工作表中的供应商清单输入到数据库的过程建立模型。

实例 See It Live

点击此处在另一个窗口里面打开实际项目。

阶段 10 从 Excel 输入数据 Stage 10 - importing Data from Excel
阶段目标 Stage Goals

覆盖的 Tersus 概念 Tersus Concepts Covered

完成本阶段后，您将会熟悉以下概念：
建模技术：输入数据，关联文本数值（Concatenating text value）
有用的过程模板：服务(Service)，加载 Excel 表格（Load Excel Table），关联（（Concatenate）
有用的显示模板：文件输入字段（File Input Field）

建立应用程序功能建模 Application Functiionality Modeled

在这一阶段，我们将接着处理应用程序的“采购人员”（Purchaser）视图。

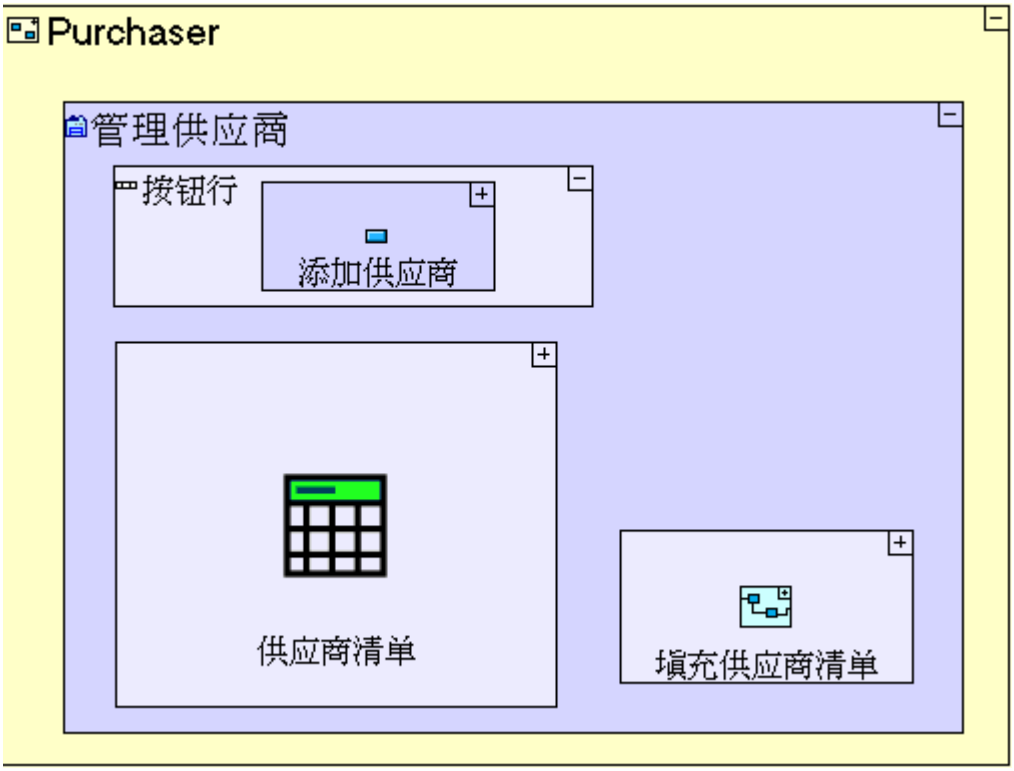
采购人员的工作是审阅申请单，并为申请单中指定的物品获取报价并发送“采购订单”（Purchase Orders（POs））。

采购人员管理供应商清单，他们把“采购订单”发送供应商清单里面的供应商。

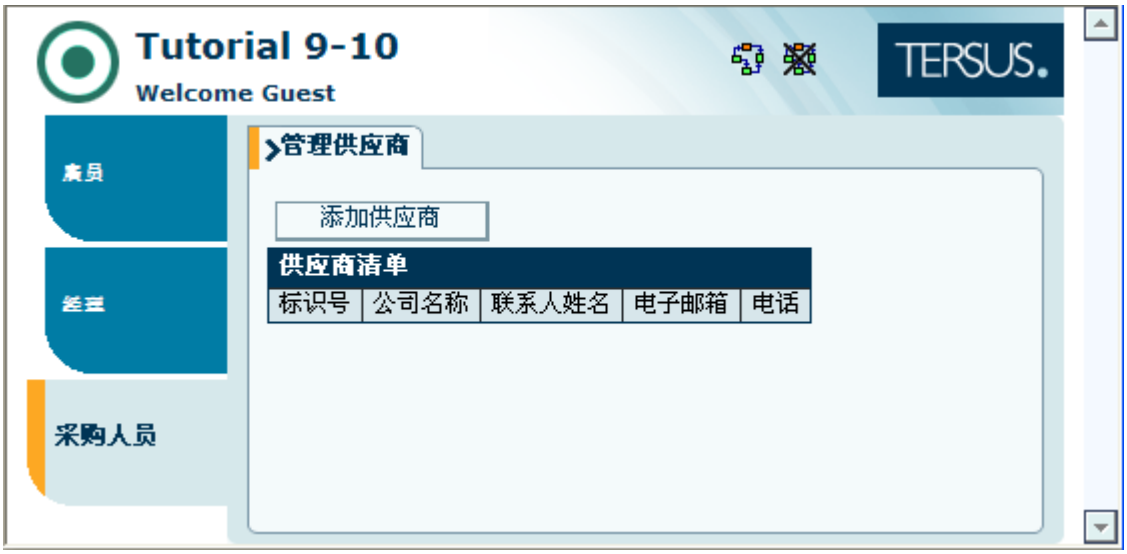
要支持这项工作，我们就要建立以下模型：
1. 管理已批准供应商清单—包括从 Excel 电子表格（spreadsheet）输入供应商清单。
2. 管理“采购订单”清单。对于每一份申请单，采购人员可能要发送多份“采购订单”给一个或多个供应商。

完成上一阶段时，我们在项目样本中建立了一个有基本的“**管理供应商**”（**Manage Supplier**）视图的新“**采购人员**”（**Purchaser**）透视图，这个透视图管理着一个“**供应商**”（**Suppliers**）数据库表格，并且已经运行了表格显示和添加新供应商的任务。

所提供的模型外观应该如下图所示：



如果我们在浏览器查看这个模型，它的外观应该如下图所示：



本阶段建模在上个阶段最后输入的项目“Tutorial 9-10”中进行。

用户建模 User Modeling

在大多数机构中，部分数据由特定的工作人员使用诸如 Excel 的电子表格程序(或能输出/导出 Excel 文件的应用程序)在他们的个人电脑上进行维护。

在此教程中，我们假定我们机构中的采购人员一直保存着一份储存在电子表格中的已批准的供应商清单。[tersus root]/workspace/Tutorial 9-10 中提供了电子表格样本“Supplier.xls”。

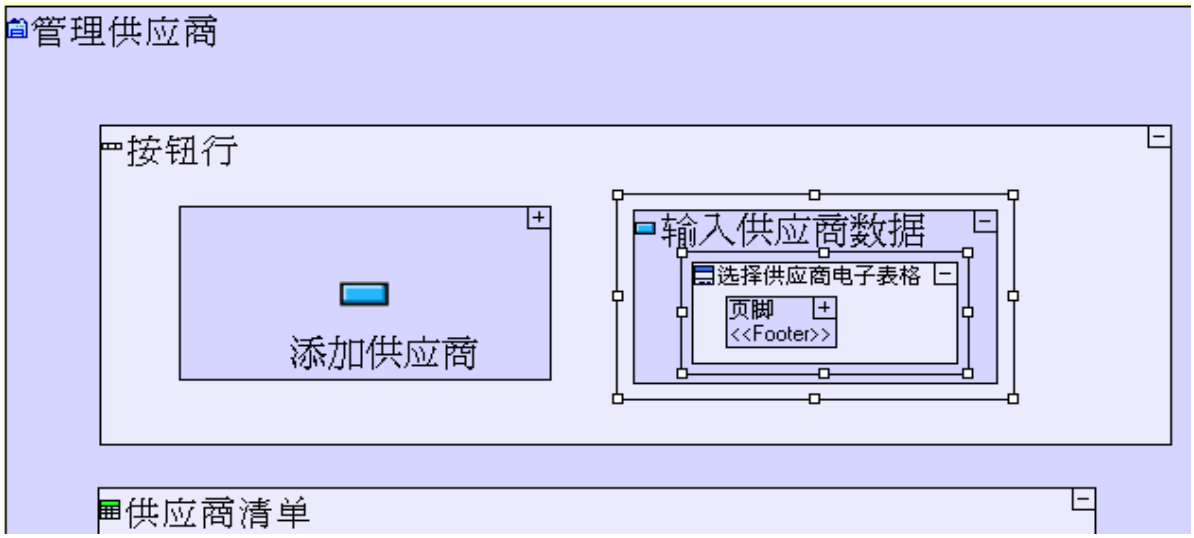
电子表格里面的数据应输入到已经建模的“**供应商**”（Suppliers）数据库表格中。

使用文件输入字段（以选择电子表格文件）Use a file Input Field (to select the spreadsheet file)

我们将添加一个允许用户选择并最终输入数据的按钮和弹出窗口：

进入“**管理供应商**”（Manage Suppliers）视图。
在“**添加供应商**”（Add Supplier）旁边添加“**显示/按钮**”(Display/Button)。
把它命名为 “***输入供应商数据”(Import Suppliers Data)*** 并放大到“**显示/按钮**”(Display/Button)。
添加“**显示/弹出窗口**”(Display/Popup)。把它命名为 “***选择供应商电子表格”(Select Suppliers Spreadsheet)***

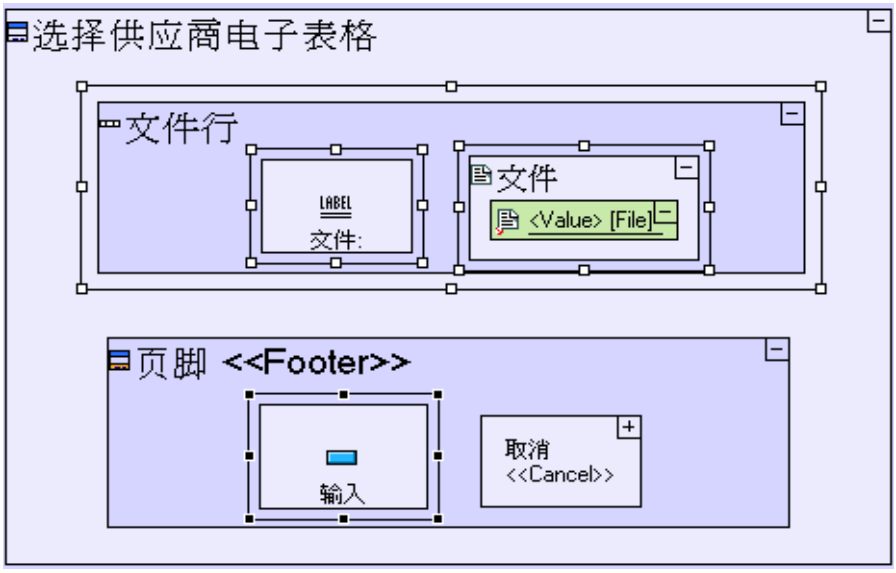
模型外观应该如下图所示：



进入“**选择供应商电子表格**”（Select Suppliers Spreadsheet）。
添加“**显示/行**”(Display/Row)。把它命名为“**文件行**”（File Row）并放大到“**选择供应商电子表格**”（Select Suppliers Spreadsheet）。
添加“**显示/标签**”(Display/Label)。把它命名为 “**文件:**”(File:)。
添加“**显示/文件输入字段**”(Display/File Input Field)。把它命名为“**文件**”（File）。
进入“**选择供应商电子表格/页脚**”（Select Suppliers Spreadsheet/Footer）。

将“OK”按钮重新命名为“**输入**”（Import）。

“**选择供应商电子表格**”（Select Suppliers Spreadsheet）弹出窗口模型外观应该如下图所示：



“**文件输入字段**”(File Input Field)包含一个缺省的（“**文件**”（File）类型的）“〈**数值**〉”（〈Value〉）数据单元，操作和提取文件内容要用到这个数据单元。

保存您的工作成果并在浏览器中查看应用程序。

点击“**采购人员**”（Purchaser）透视图标签（tab）.

按“**输入供应商数据**”（Import Suppliers Data）按钮。

我们刚创建的弹出窗口外观应该如下图所示：

我们可以看到“**文件输入字段**”(File Input Field)在浏览器中运行时实际上分成两个部分：一部分是含有可人工输入文件名称的“**文本编辑**”（text edit），另一部分是允许用户浏览其电脑里面的文件系统并选择

要输入文件的“**浏览…**”（**Browse…**）按钮。

使用载入 Excel 表格模板（提取数据行）Use a Load Excel table template (to extract data rows)

由于现在我们能够选择输入的电子表格文件，我们可以开始为实际的输入过程建模。

我们从提取用户在“**选择供应商电子表格**”（**Select Suppliers Spreadsheet**）弹出窗口中选择的文件开始：

进入“**输入**”（**Import**）按钮。

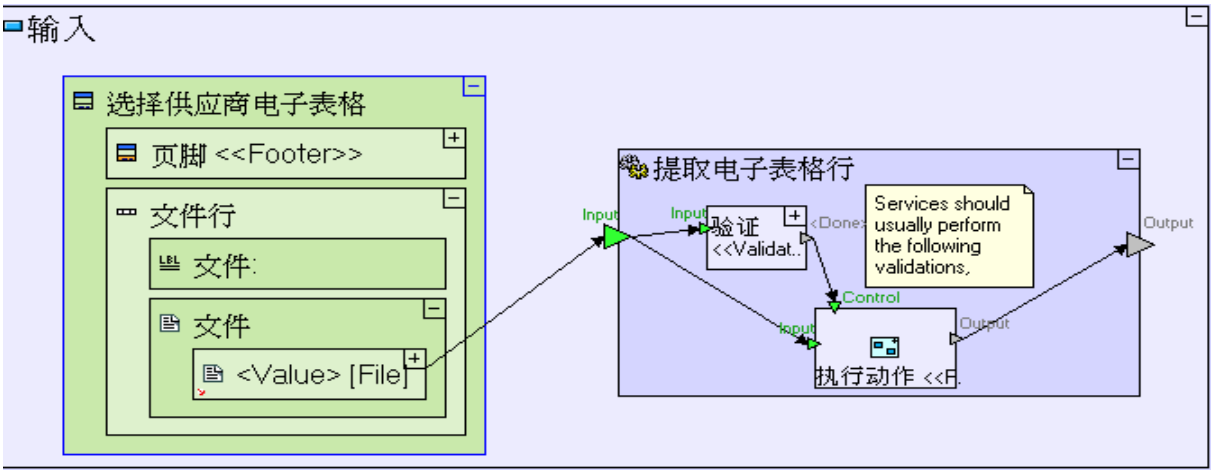
添加一个“**选择供应商电子表格**”（**Select Suppliers Spreadsheet**）弹出窗口的上级索引。

添加一个“**基本/安全服务**”（**Basic/Secure Service**）（）。

把它命名为 “***提取电子表格行***” ***(Extract Spreadsheet Rows)***

添加连接“**选择供应商电子表格/行/文件/<数值>**”（**Select Suppliers Spreadsheet/Row/File/<Value>**）和“**提取电子表格行/输入**”（**Extract Spreadsheet Rows/Input**）的流程（flow）。

“**输入**”（**Import**）按钮模型外观应该如下图所示：



注意：我们对“**提取电子表格行**”（**Extract Spreadsheet Rows**）使用了一个新类型的过程模板“**基本/服务**”（**Basic/Service**）（），而不是常用的“**基本/动作**”（**Basic/Action**）。

“**服务**”（**Service**）和“**动作**”（**Action**）根本的区别在于，“**服务**”内部建模总是在服务器端进行，而“**动作**”模型会根据模型中使用的不同模板的要求，在“**客户机**”（**Client**）和“服务器”（**Server**）间不断（透明地）进行转换。

还有某些建模方案（比如我们这里即将建立的模型）不能在客户端运行，而只能明确地规定在服务器端运行。更多信息，请参阅**阶段 13**。

“**服务**”（**Service**）是一个“**复合**”（**composite**）模板，这意味着它包含了预定义的建模—“**验证**”（**Validate**）和“**执行动作**”（**Perform Action**）动作和相关的流程（flows）。预定义建模不是强制性的，而只是鉴于使用服务器端资源的 Tersus 过程的结构的一种建议。更多详情，请参阅模型中的“**注意事项**”（**Notes**）（黄色字体）。

“**文件**输入字段”（**File** Input Field）的“**<数值>**”(**<Value>**)数据单元已经被传送到“**提取电子表格行**”（**Extract Spreadsheet Rows**）过程和另一个“**文件**”（File）数据单元：

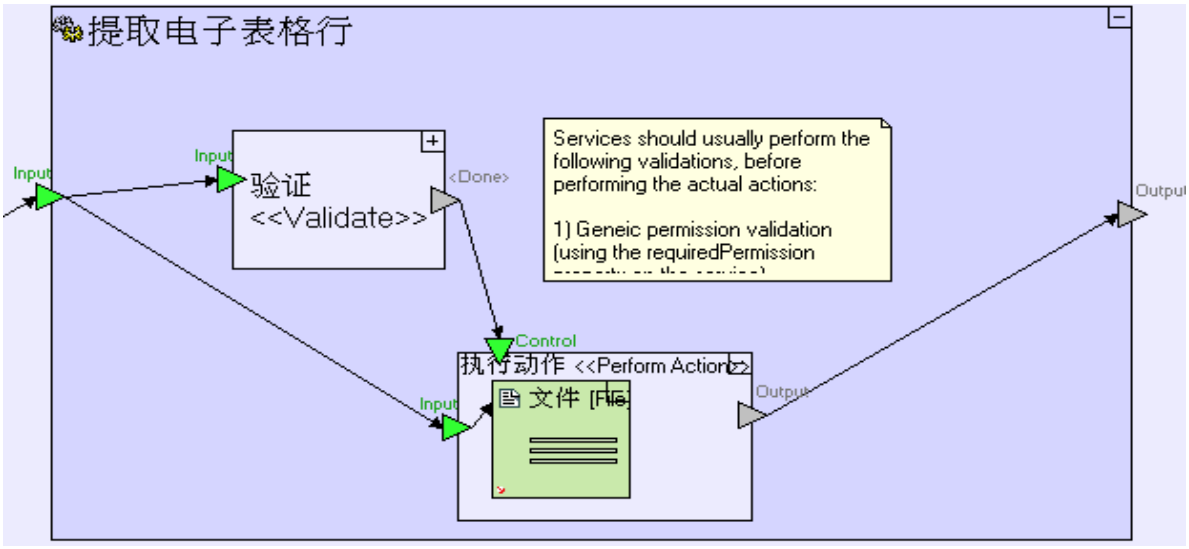
放大到“**提取电子表格行**”（**Extract Spreadsheet Rows**）。

放大到“**执行动作**”（**Perform Action**）。

添加一个“**数据类型/文件**”(**Data Types/File**)数据结构（）。

添加连接“**执行动作/输入**”（**Perform Action/Input**）触发器和“**文件**”(**File**)数据结构的流程（flows）。

“**提取电子表格行**”（**Extract Spreadsheet Rows**）服务模型外观应该如下图所示：

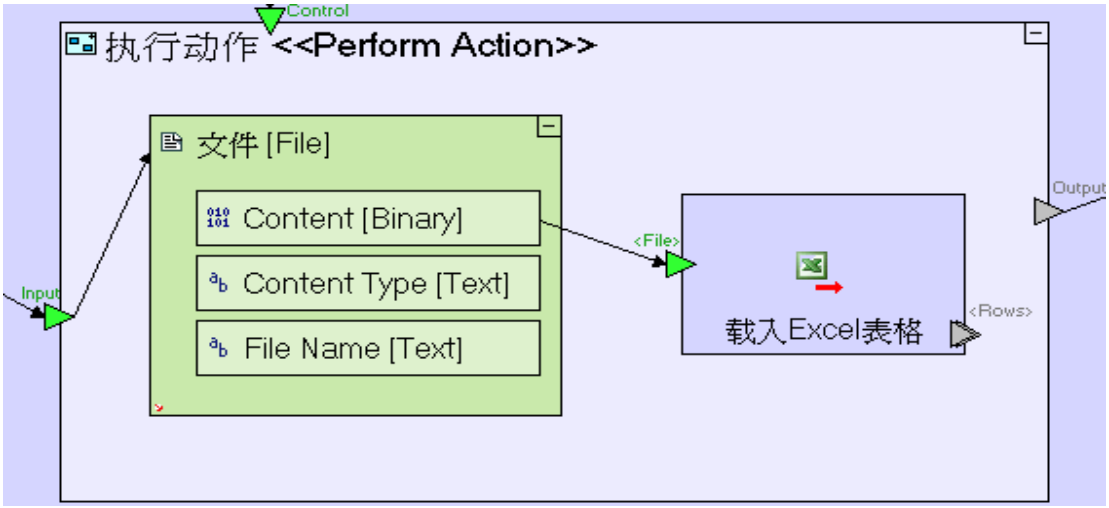


“**文件**”(**File**)数据结构的“**内容**”（**Content**）单元包含实际的 Excel 数据：

添加一个“**混合/载入 Excel 表格**”(**Miscellaneous /Load Excel Table**)模板()。

添加连接“**文件/内容**（**File/Content**）和“**载入 Excel 表格/<文件>**”（**Load Excel Table/<File>**）的流程。

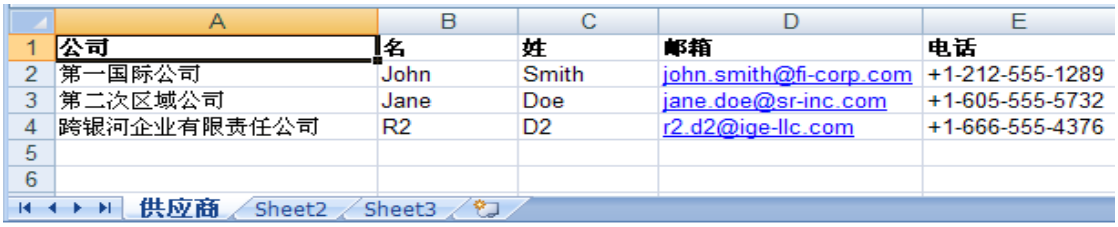
“**提取电子表格行**”（**Extract Spreadsheet Rows**）服务模板外观应该如下图所示：



您可能在想，我们目前在**“输入”**（**Import**）按钮中建立的模型太复杂了，可以通过直接传送**“选择供应商电子表格/行/文件/<数值>/内容”**（**Select Suppliers Spreadsheet/Row/File/<Value>/Content**）到**“载入 Excel 表格/<文件>”**（**Load Excel Table/<File>**）来简化。

我们之所以按目前所述的方法建模是因为浏览器有诸多限制，不允许直接读取**“内容”**（**Content**）。为了使 Tersus 能够从浏览器中提取二进制的内容（binary content），**“文件”**（**File**）数据必须原封不动地传送到服务器。

现在我们停一分钟，看一看样本电子表格文件 **“供应商.xls”**（**Suppliers.xls**），我们计划从这个文件中输入数据：



	A	B	C	D	E
1	**公司**	**名**	**姓**	**邮箱**	**电话**
2	第一国际公司	John	Smith	john.smith@fi-corp.com	+1-212-555-1289
3	第二次区域公司	Jane	Doe	jane.doe@sr-inc.com	+1-605-555-5732
4	跨银河企业有限责任公司	R2	D2	r2.d2@ige-llc.com	+1-666-555-4376
5					
6					

供应商 Sheet2 Sheet3

数据出现在**“供应商”**（**Suppliers**）工作表中并编排成表格的格式，表格中的第一行定义了每一列的名称（column names）**“公司，名字，姓， 邮箱和电话”**（**Company, First Name, Last Name, Email,** and **Telephone**），其它各行则包含了供应商信息（一行一个信息）。

在缺省情况下，**“载入 Excel 表格”**（**Load Excel Table**）会从第一个工作表搜索数据。如果要求的工作表不是文件的第一个工作表，您就可以使用可选的**“<工作表名称>”**（**<Sheet Name>**）触发器（通过右击**“添加单元”**（Add Element））。

定义从电子表格中提取的行的数据结构 Define the Data Structure of Rows extracted from the Spreadsheet

“载入 Excel 表格”（**Load Excel Table**）过程还需要工作表中相关数据组织方式的定义。该定义**“查找”**（**Find**)模板类似的方式提供(在之前的阶段讨论过)—从它的出口目标(target of its exit)**“<行>”**(**<Rows>**)来推导出数据结构,几分钟后我们就会看到。

一旦行（rows）被提取，数据集会被复制到**“供应商”**（**Supplier**）数据库记录中，该记录被用来在数据库中存储存供应商:

进入**“提取电子表格行”**（**Extract Spreadsheet Rows**）。

添加一个**“基本/动作”**（**Basic/Action**）。命名为 ***“写入供应商记录” (Write Supplier Record)***。

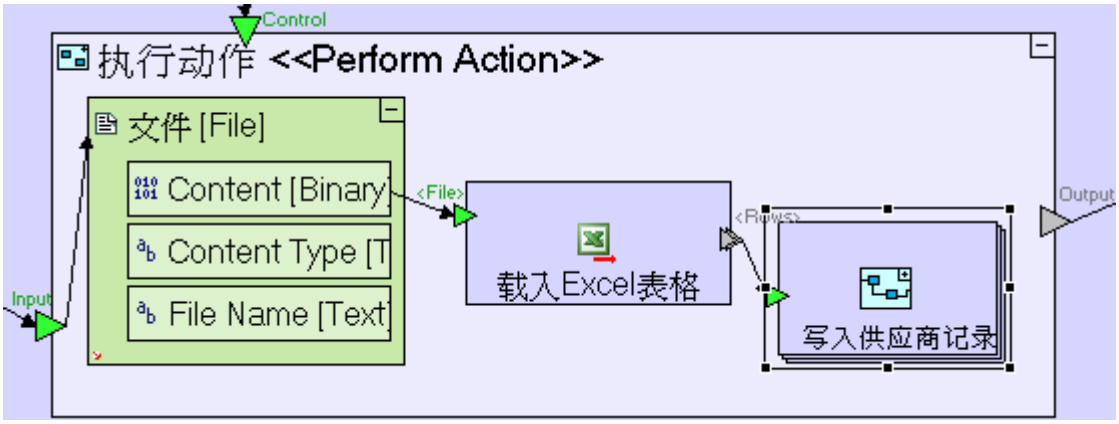
给它添加一个（trigger)。

添加一个连接**“载入 Excel 表格/<行>”**（**Load Excel Table/<Rows**）和**“写入供应商记录”**(**Write Supplier Record**) 触发器。

因为一般会从电子表格中提取多个行（这也解释了为什么**“<行>”<Row>**出口是**“重复性”**（**repetitive**）的），因此**“写入供应商记录”**(**Write Supplier Record**)过程应该也被标记为重复性的一一意思是说，每次从**“载入 Excel 表格”**（**Load Excel Table**）中提取一行，都会提取这个过程一次:

右击**“写入供应商记录”**(**Write Supplier Record**)过程，确认**“重复性”**（**Repetitive**）选择。

“提取电子表格行”（**Extract Spreadsheet Rows**）过程外观应该与下图类似:



输入给**“写入供应商记录”**(**Write Supplier Record**)触发器的是从电子表格中提取的唯一行，因此就要定义它的类型:

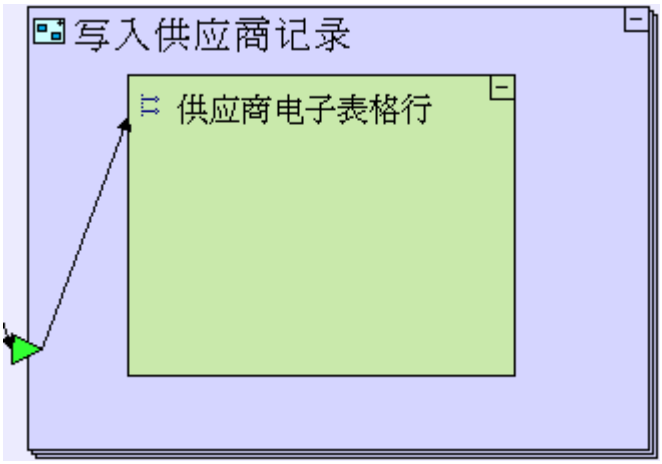
放大到**“写入供应商记录”**(**Write Supplier Record**)。

添加一个**“数据类型/数据结构”**（**Data Types/Data Structure**）（ ）。

命名为 ***“供应商电子表格行” (Supplier Spreadsheet Row)***

添加流程，连接**“写入供应商记录”**(**Write Supplier Record**)触发器和**“供应商电子表格行”**（**Supplier Spreadsheet Row**）。

“**写入供应商记录**”(**Write Supplier Record**)动作过程外观应该与下图类似：



> 回忆中之前的阶段，我们提到“**数据结构**”（Data Structure）和“**数据库记录**”（Database Record）几乎是相同的，唯一的不同是后者会自动地嵌进数据库中的表格里。

数据结构中的字段应该准确与电子表格的名称栏（columns names）(第一行）相匹配（参见上面的屏幕快照)：

放大到“**供应商电子表格行**”（Supplier Spreadsheet Row）。

添加一个“**数据类型/文本**”（Data Types/Text）。命名为 “***公司***” (*Company*)

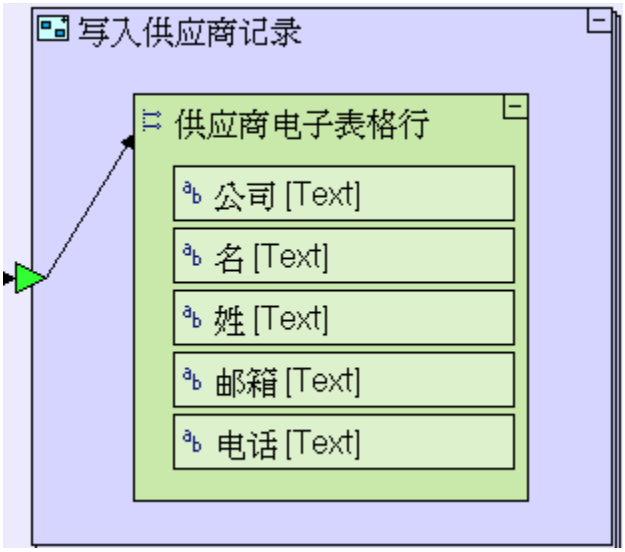
添加一个“**数据类型/文本**”（Data Types/Text）。命名为 “***名***” (*First name*)

添加一个“**数据类型/文本**”（Data Types/Text）。命名为 “***姓***” (*Last name*)

添加一个“**数据类型/文本**”（Data Types/Text）。命名为 “***邮箱***” (*Email*)

添加一个“**数据类型/文本**”（Data Types/Text）。命名为 “***电话***” (*Telephone*)

“**写入供应商记录**”(**Write Supplier Record**)过程现在外观应该如下图所示：



每个“**供应商电子表格行**”（Supplier Spreadsheet Row）中的数据都应该被复制到相应的“**供应商**”（Supplier）记录中。

进入“**写入供应商记录**”(**Write Supplier Record**)。

从储存器/大纲（在“**管理供应商**视图/**供应商清单**表格”（**Manage Suppliers** view/**Supplier List** table）中查找到）中重新使用“**供应商**”（**Supplier**） 数据结构。

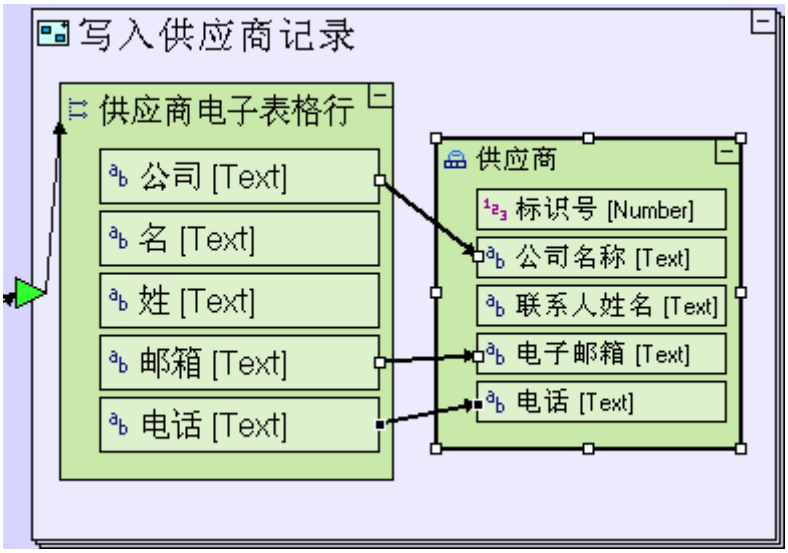
创建连接以下几项的流程：

“**供应商电子表格行/公司**”（**Supplier Spreadsheet Row/Company**）和“**供应商/公司名称**”（**Supplier/Company Name**）。

“**供应商电子表格行/邮箱**”（**Supplier Spreadsheet Row/Email**）和“**供应商/电子邮箱**”（**Supplier/Email**）。

“**供应商电子表格行/电话**”（**Supplier Spreadsheet Row/Company**）和“**供应商/电话**”（**Supplier/Phone**）。

“**写入供应商记录**”(**Write Supplier Record**)过程现在外观应该如下图所示：



还应该给“**供应商**”（**Supplier**）多填充两个字段：

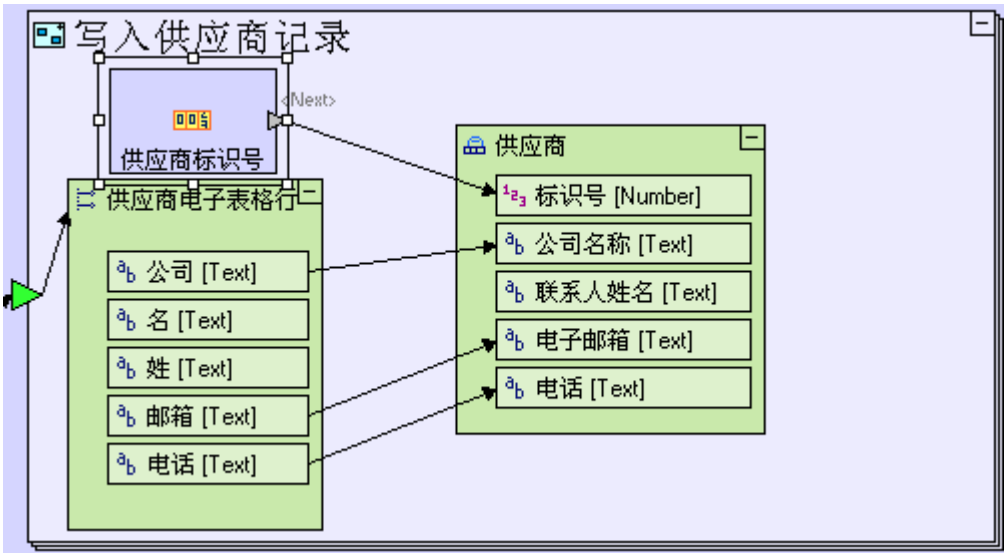
- **标识号 (Id)**——每个供应商专属的身份证(identifier)。
- “**联系人姓名**”(**Contact Name**)---- “**供应商**”（**Supplier**）记录只有一个字段来储存联系人的姓名，而电子表格有两个字段“**名**”（**First Name**）和“**姓**”（**Last Name**）。因此，“**联系人姓名**”(**Contact Name**)应该是把两个字段合二为一储存。

将使用“**序列号**”（**Sequence Number**）动作来填充**Id**。这个动作和“**添加供应商**按键/**输入新供应商**弹出窗口/**提交**按键”（**Add Supplier** button/**Enter New Supplier** popup/**Submit** button）中使用的是同一个：

通过从储存器/大纲（repository/outline）中拖拽，来重新使用“**供应商标识号**”（**Supplier Id**）。

添加流程，连接“**供应商标识号/<下一个>**”（**Supplier Id/<Next>**）和“**供应商/标识号**”（**Supplier/Id**）。

“写入供应商记录”(Write Supplier Record)过程现在外观应该如下图所示：



使用文本控制模板（连接文本数值）Using a Text Manipulation Template (to concatenate text values)

要把“名”（First Name）和“姓”（Last Name）合成一个单独的文本数值（text value）（请记住在中间添加空格，以把它们分隔开来），按照如下步骤操作：

添加一个“文本/关联”(Text/Concatenate)模板（abcd）。

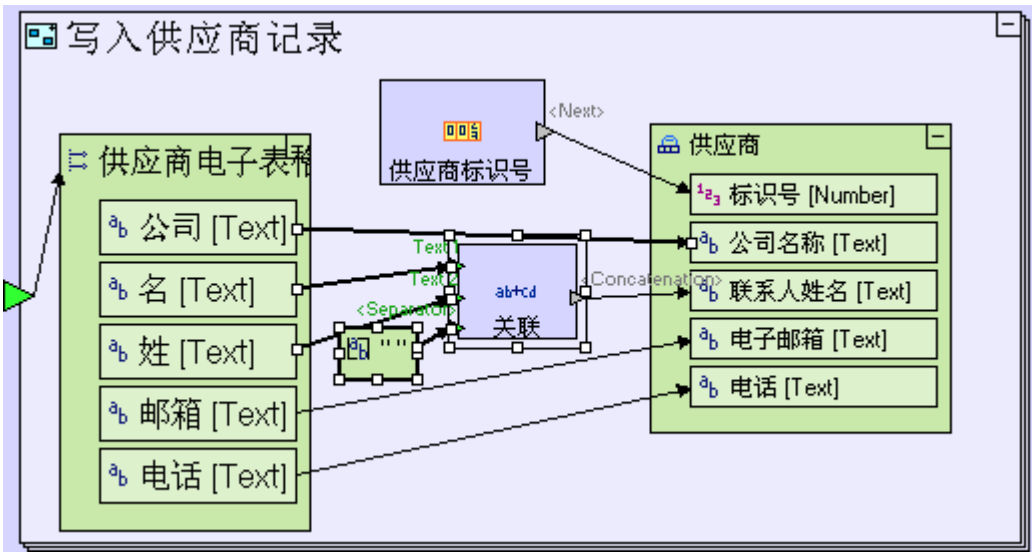
创建一个流程，连接“供应商电子表格行/名”（Supplier Spreadsheet Row/First Name） 和“关联/文本1” （Concatenate/Text 1）。

创建一个流程，连接“供应商电子表格行/姓”（Supplier Spreadsheet Row/Last Name） 和“关联/文本2” （Concatenate/Text 2）。

添加一个“常数/文本”（ Constants/Text）。按一次“空格键”（[Space]）来创建“ “常数，并创建一个流程来把它连接到“ “关联/<分隔符>”(Concatenate<Separator>)。

创建一个流程，连接“关联/<关联>”（ Concatenate/<Concatenation> ）和 “供应商/联系人姓名”（Supplier/Contact Name）。

“写入供应商记录”(Write Supplier Record)过程现在外观应该如下图所示：



完成输入过程 (Completing the import process)

要完成输入过程，我们需要注意以下几步：

1. 给数据库插入“**供应商**”（**Supplier**）记录。
2. 关闭“**选择供应商电子表格**”（**Select Suppliers Spreadsheet**）弹出窗口。
3. 刷新“**管理供应商**”(**Manage Suppliers**)视图中的“**供应商清单**”(**Supplier List**)表格。

要给数据库添加“**供应商**”（**Supplier**）记录：

进入“**写入供应商记录**”(**Write Supplier Record**)。

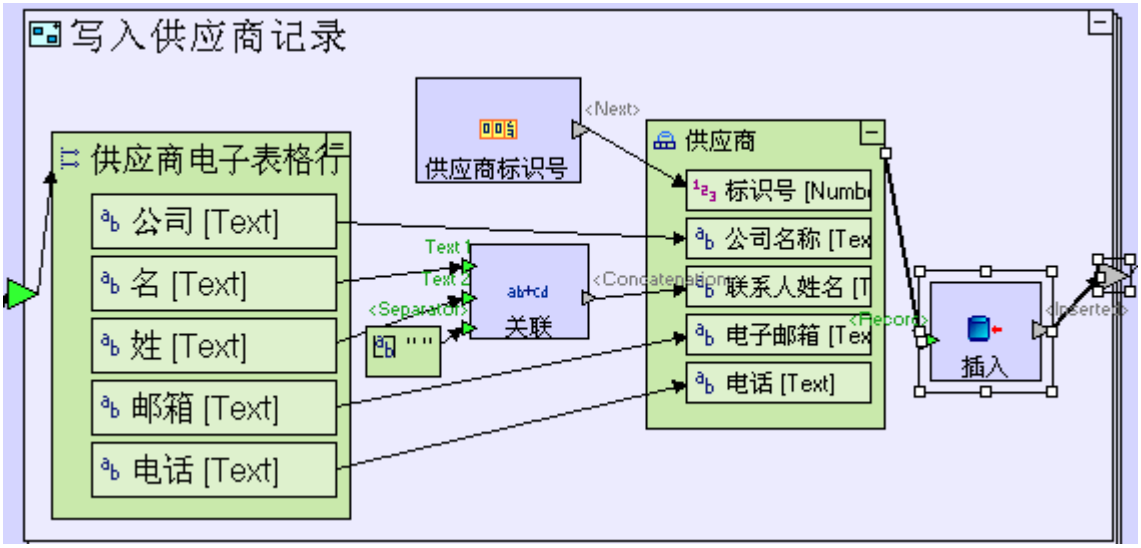
添加“**数据库/插入**”（**Database/Insert**）。

创建一个流程，连接“**供应商**”（**Supplier**）和“**插入/<记录>**”(**Insert/<Record>**)。

给“**写入供应商记录**”(**Write Supplier Record**)添加一个 “**出口**”（**Exit**）。

创建流程，连接“**插入/<插入>**”（**Insert/<Inserted>**）出口和“**写入供应商记录**”(**Write Supplier Record**)出口。

“写入供应商记录”(Write Supplier Record)过程现在外观应该如下图所示：

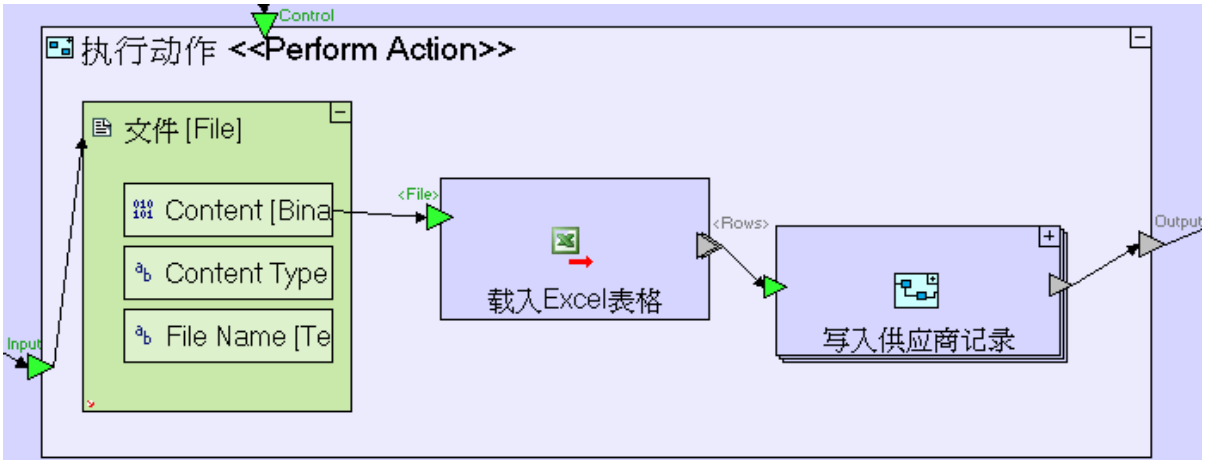


要结束输入过程，关闭弹出窗口并刷新“**供应商清单**”（**Supplier List**,），还要定义另外的流程：

缩小并转到“**执行动作**”（**Perform Action**）。

创建一个流程，连接“**写入供应商记录**”(**Write Supplier Record**)出口和“**执行动作/输出**”（**Perform Action/Output**）出口。

“**执行动作**”（**Perform Action**）现在外观应该如下图所示：



缩小并转到“**提取电子表格行**”（**Extract Spreadsheet Rows**）服务。

缩小并转到“**输入**”（**Import**）按键。

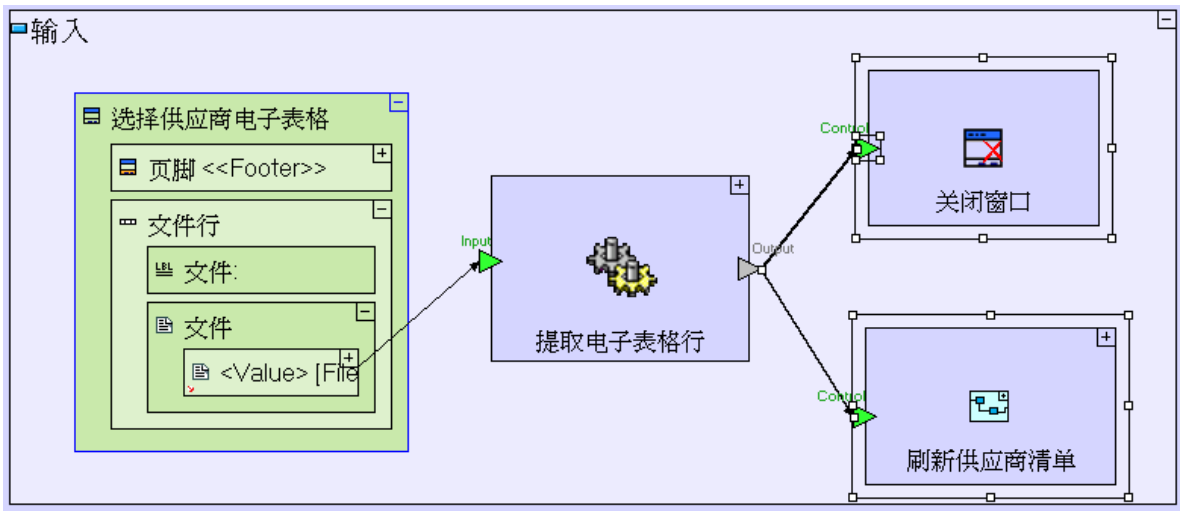
从储存器/大纲（repository/outline）中拖拽“**刷新供应商清单**”(**Refresh Supplier List**)过程（用于“**添加供应商**”按钮/“**输入新供应商清单**”弹出窗口/“**提交**”按钮”（**Add Supplier** button/**Enter New Supplier** popup/**Submit** button）中），以重新使用“**刷新供应商清单**”(**Refresh Supplier List**)过程。

连接“**提取电子表格行**”（**Extract Spreadsheet Rows**）出口和“**刷新供应商清单**”（**Refresh Supplier List**）触发器。

添加一个“**显示动作/关闭窗口**”（**Display Actions/Close Window**）模板。添加一个“**控制**”（**Control**）

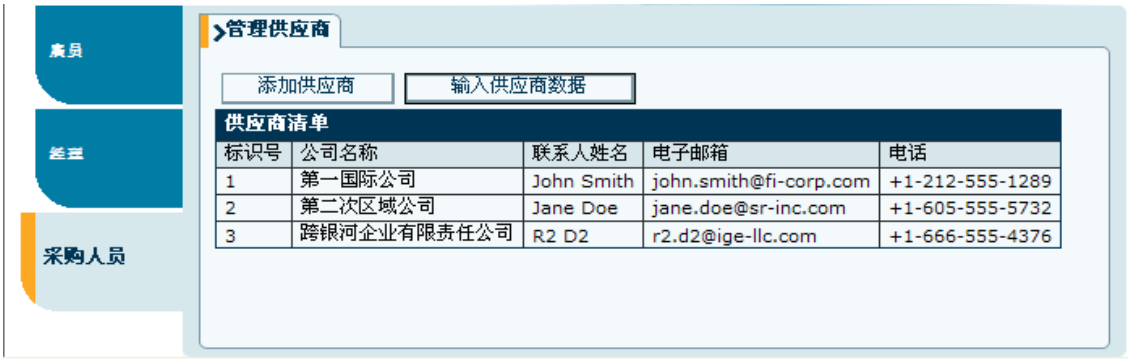
触发器（通过右击->“**添加单元**”（**Add Element**)或简单地添加一个触发器）。
连接“**提取电子表格行**”（**Extract Spreadsheet Rows**）出口和“**关闭窗口**”（**Close Window**）触发器。

“**输入**”（**Import**）按键现在外观应该如下图所示：



保存您的工作，并在浏览器中查看应用情况。
点击“**采购员**”（**Purchaser**）透视图标签（tab）。
按住“**输入供应商数据**”（**Import Suppliers Data**）按钮。
按住“**选择供应商电子表格**”（**Select Suppliers Spreadsheet**）弹出窗口中的“**浏览…**”（**Browse…**）按钮。
转到“**[tersus root]/workspace/Tutorial 9-10**”，并选择“**供应商.xls**”。
按住“输入”(Import)按钮。

结果出来的“**供应商清单**”（**Supplier List**）外观应该与下图类似：



完成阶段 10 Completing Stage 10

现在我们输入项目样本“**Tutorial 10-11**”，使用它作为本教程下一个阶段的基础。

> 如果要重新参考如何输入一个项目样本，请参见阶段 2 结束时“**输入项目样本**”（**Importing a Sample Project**）的部分。

这个项目样本具备到目前为止我们建立的模型的所有功能。

您现在可以进入**阶段 11** 了，第 11 阶段我们将用一种不同的技术为一个表格显示建模，这种技术可以对内容提供更好的控制。

实例 See It Live

点击此处在另一个窗口里面打开实际项目。

阶段 11 -- 控制表格显示 Stage 11-Controlling Table Display

阶段目标 Stage Goals

覆盖的 Tersus 概念 Tersus Concepts Covered

完成本阶段后，您将会熟悉以下概念：
建模技术：控制显示中出现的列
有用的显示模板：表格（Table），数字显示（Number Display），文本显示（Text Display）

建立应用程序功能模型 Application Functionality Modeled

现在我们开始为采购员发出的**“采购订单”（Purchase Orders(POs)）**建立模型。我们将从已获经理批准的申请单表格式清单显示开始。这个表格不同于先前建模的表格 – 我们将规定显示哪些列和显示的顺序，而不是依赖于缺省设置。

在后面的阶段中我们将添加：

- 一个提示（alert）（如果有任何申请单被标识为紧急，提示就会显示出来以提示采购员对这些申请单进行处理）。
- 一份当前已选定的申请单的已发出采购订单清单。
- 一个给当前已选定申请单的供应商发送采购订单的**“订购”（Order）**按钮。
- 发送采购订单后，申请单就会被标记为（部分或全部）已订购（ordered）。

> 本阶段建模在上个阶段最后输入的“Tutorial 10-11”项目中进行。

用户建模 User Modeling

我们从添加一个新的视图开始：

进入到**“采购员”**（Purchaser）
添加**“显示/视图”**（Display/View）。
把它命名为 ***“发送采购订单”（Issue Purchase Orders）***

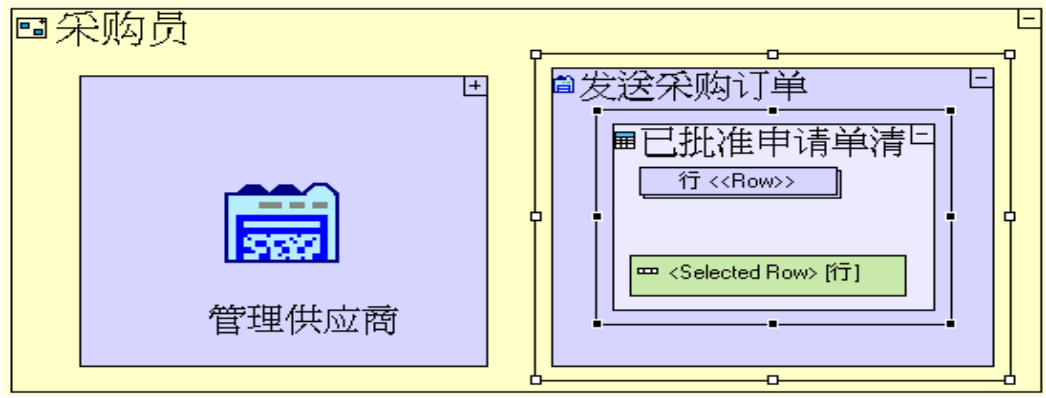
使用表格模板 Use a Table Template

到目前为止，我们想显示数据的表格式清单的时候，我们都使用“**简单表格”（Simple Table）**模板。使用此模板的好处是建模很快，不好之处是你无法控制显示哪些列和显示顺序。我们在后面的阶段还需要用到更多的功能，这些功能无法用“**简单表格”（Simple Table）**实现，而要用“**表格”（Table）**模板：

进入到“**发送采购订单”（Issue Purchase Orders）**。

添加一个“**显示/表格”（Display/Table）**模板（  ）。把它命名为 “***已批准申请单清单”（Approved Requisitions List）*** 并放大到“**发送采购订单”（Issue Purchase Orders）**。

“**采购员”（Purchaser）**模型外观应该如下图所示：



> **“表格”（Table）**模板是预先配置好的，带有一个**“行”（Row）**显示单元和一个（**“行”（Row）**的）“〈**选择行”**〉（**〈Selected Row〉**）显示数据单元显示，我们将在后面的步骤中使用这些设计和控制表格显示。

使用数字/文本/日期显示模板 Use Number/Text / Date /Display Templates

通过在行里插入“**文本/数字/日期显示”（Text/Number/Date Display）**模板定义这些列，每个模板定义行里的一个单元格（cell）。

表格显示时，行里面每一个单元的名称将用作列的标题（column heading）。

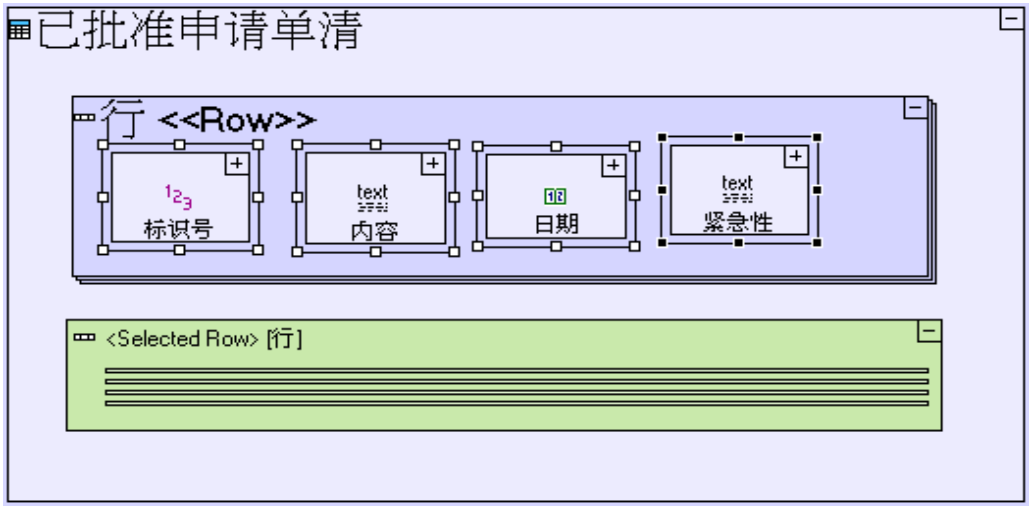
添加一个“**显示/数字显示”（Display/Number Display）**（  ）。把它命名为 “**标识号”（*Id*）**。

添加一个“**显示/文本显示”（Display/Text Display）**（  ）。把它命名为 “**内容”（*Description*）**。

添加一个“**显示/日期显示”（Display/Date Display）**（  ）。把它命名为 “**日期”（*Date*）**。

添加一个“**显示/文本显示”（Display/Text Display）**（  ）。把它命名为 “**紧急性”（*Urgency*）**。

“已批准申请单清单”（Approved Requisition List）表格模型外观应该与下图类似：



> **“数字/文本/日期显示”（Number/Text/Date Display）**模板与我们之前使用的**“文本区”（Text Area）**输入字段类似：每个模板包含模板**“<数值>”（<Value>）**单元并用作与此数值有关的流程的来源/目标（source/target）。
> 两者的不同在于用户界面方面：输入字段使用户能够改变数值，而显示则不能。

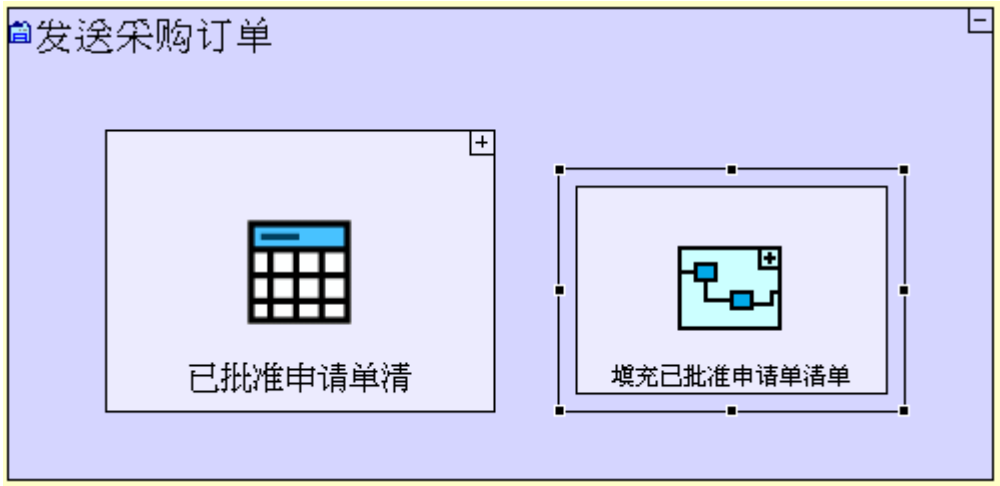
使用**“表格”（Table）**模板的灵活性现在开始发挥作用了。我们可以控制显示数据库的哪些字段（在这种情况下**“状态”（Status）**不显示），另外我们还可以明确定义各列（columns）的显示顺序。

填充表格 （以申请表）Populate the table (with requisitions)

对于**“简单表格”**（Simple Table）而言，我们应该创建一个以数据库中的数值填写表格的过程：

缩小并转到**“发送采购订单”（Issue Purchase Orders）**视图。
添加一个**“基本/动作”(Basic/Action)**。
把它命名为 ***“填充已批准申请单清单” (Populate Approved Requisitions List)***

“发送采购订单”（Issue Purchase Orders）视图模板外观应该与下图类似：



正象前面的例子一样，“**填充…**”(Populate…)的过程会有一个“**生成…**”（Generate…）的过程，“**生成…**”过程创建一个代表表格的显示数据单元并把它输出给显示的上级索引：

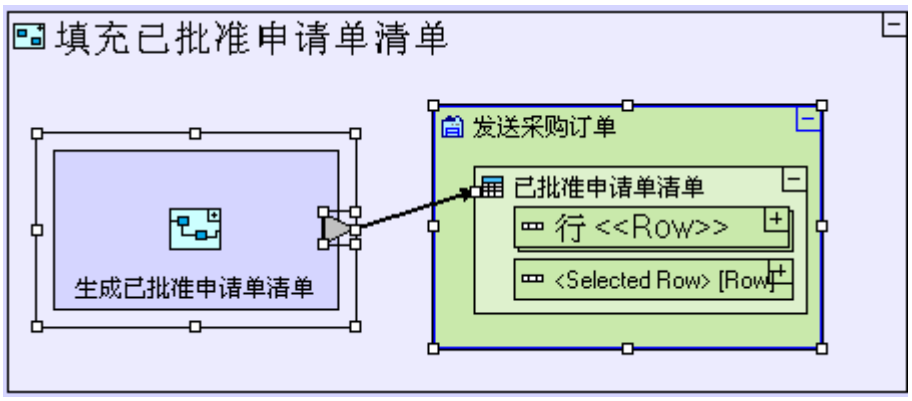
进入“**填充已批准申请单清单**”（Populate Approved Requisitions List）。

添加“**基本/动作**”(Basic/Action)。把它命名为 “***生成已批准申请单清单***” ***(Generate Approved Requisitions List)***。给它添加一个出口。

添加一个“**发送采购订单**”（Issue Purchase Orders）视图的上级索引。

创建一个流程，连接“**生成已批准申请单清单**”（Generate Approved Requisitions List）出口和“**发送采购订单/已批准申请单清单**”（Issue Purchase Orders /Approved Requisitions List）表格。

“**填充已批准申请单清单**”（Populate Approved Requisitions List）视图模板外观应该与下图类似：



“**生成已批准申请单**”（Generate Approved Requisitions List）过程提取已批准的所有申请单：

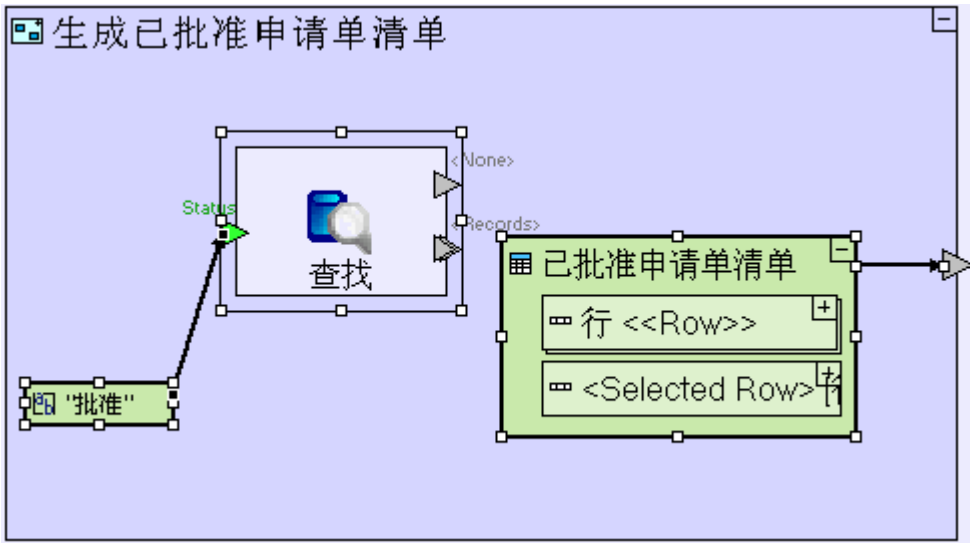
放大并进入“**生成已批准申请单**”（Generate Approved Requisitions List）。

添加**“数据库/查找”（Database/Find）**。给它添加一个触发器并把触发器命名为 “***Status”(状态)***。

重新使用“**批准**”（Approved）常数（“**经理**”（Manager）系统/“**申请单批准**”(Requisition Approval)视图/“**批准申请单**”（Approve Requisition）按钮中使用）。创建一个连接该常数和“**查找/状态**”（Find/Status）的流程。

从储存器/大纲中拖出“**已批准申请单**”（Approved Requisitions List）表格。创建从“**已批准申请单**”（Approved Requisitions List）表格到“**生成已批准申请单**”（Generate Approved Requisitions List）出口的流程。

“**生成已批准申请单清单**”（**Generate Approved Requisitions List**）视图模板外观应该与下图类似：



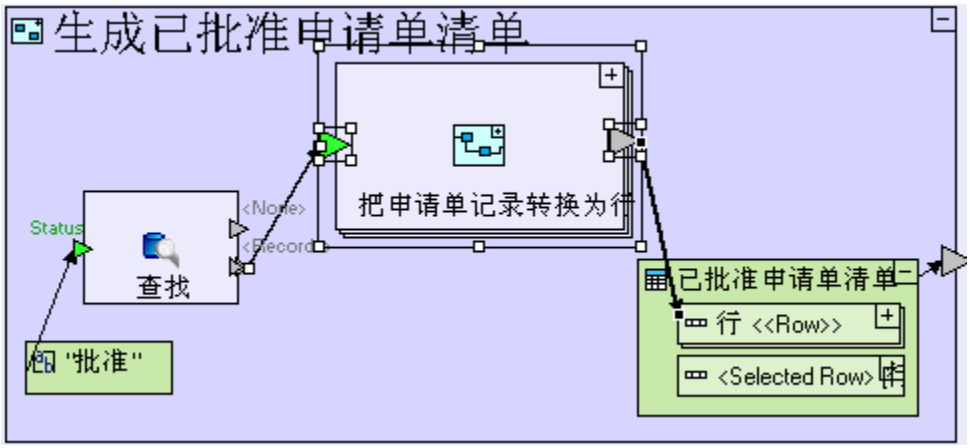
到目前为止，“**生成已批准申请单**”（**Generate Approved Requisitions List**）过程和我们建模的其它“**生成…**”（**Generate**）过程类似。但从这儿开始，建模会发生变化。由于表格不直接接受查找到的记录，我们必须创建一个过程，把“**申请单**”**(Requisition)**数据库记录的字段转化（converts）为(或映射(maps)到)表格的“**行**”**(Row)**显示数据单元：

添加一个“**基本/动作**”**(Basic/Action)**。把它命名为 “***把申请单记录转化为行***” ***(Convert Requisition to Row)*** 并把 它设置为重复性的(repetitive)。添加一个触发器和出口。

创建一个流程，连接“**查找/<记录>**”**(Find/<Records>)**和“**把申请单记录转化为行/记录**”**(Convert Requisition to Row/Record)**。

创建一个流程，连接“**把申请单记录转化为行/行**”（**Convert Requisition to Row/Row**）出口和“**已批准申请单清单/行**”（**Approved Requisitions List/Row**）。

“**生成已批准申请单**”（**Generate Approved Requisitions List**）视图模板外观应该与下图类似：



现在为从“**申请单**”（**Requisition**）到“**行**”（**Row**）的映射（mapping）建模：

放大到并进入“**把申请单记录转化为行**”（**Convert Requisition to Row**）。

从储存器/大纲(repository/outline)中拖出一个“**申请单”（Requisition）**数据库记录（它在多个位置重复使用，如“**打开申请单”（Open Requisitions）**视图/“**申请单清单”（Requisition List）**表格）。创建一个连接这个数据库记录和过程触发器（process trigger）的流程。

从储存器/大纲中拖出一“**行”（Row）**(即“**已批准申请单清单”（Approved Requisitions List）**表格中的一行)。创建一个连接这一行和过程出口的流程。

创建以下流程：

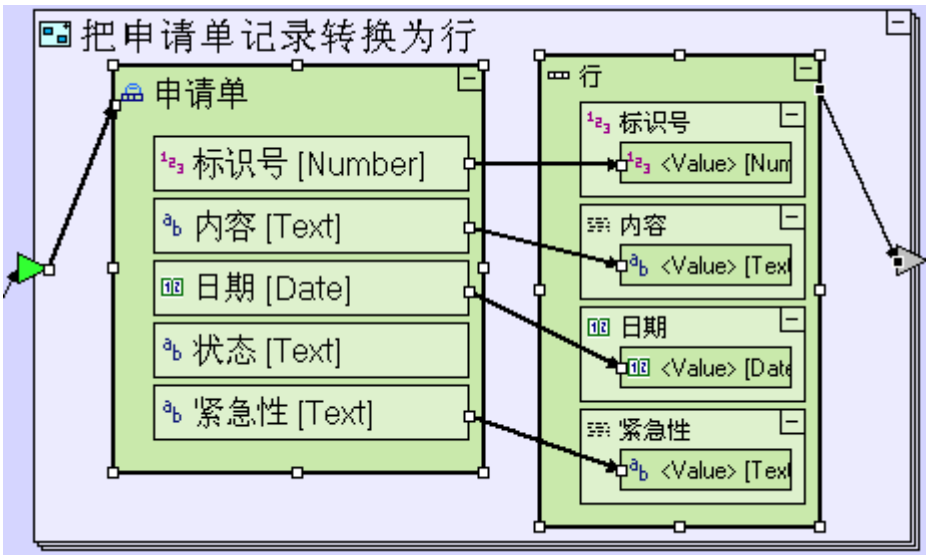
“申请单/标识号”（Requisition/Id）至“**申请行/标识号/<数值>”（Requisition Row/Id<Value>）**

“申请单/内容”（Requisition/Description）至“**申请单行/内容/<数值>”（Requisition Row/Description/<Value>）**

“申请单/日期”（Requisition/Date）至“**申请单行/日期/<数值>”（Requisition Row/Date /<Value>）**

“申请单/紧急性”（Requisition/Urgency）至“**申请单行/紧急性/<数值>”（Requisition Row/Urgency/<Value>）**

“把申请单记录转化为行”（Convert Requisition to Row）视图模板外观应该与下图类似：



保存您的工作成果并在浏览器中查看应用程序。

点击“**采购员”（Purchaser）**透视图标签（Tab）。

转换到“**发送采购订单”（Issue Purchase Orders）**视图。

您将看到一份与下图类似的（已批准的）申请单清单：

注意：所显示的申请单清单不同于之前建模的申请单清单，特别是它没有“**状态**”（**Status**）这一列。在后面的阶段里面，我们将使用“**表格**”（**Table**）显示的其它功能（如显示计算数值的功能）和选定某一行时执行一个动作的能力。

完成阶段 11　Completing Stage 11

现在我们输入项目样本“**Tutorial 11-12**”并把它作为本教程下一个阶段的基础。

> 想再了解如何输入一个项目样本，请参见**阶段 2** 最后的“**输入项目样本**”部分。

这个项目样本具备到目前为止我们建立的模型的所有功能。

现在您可以开始学习**阶段 12**，在**阶段 12**，我们将学习怎样有条件地执行动作。

实例　See It Live

点击此处在另一个窗口里面打开实际项目。

阶段 12—控制应用程序流程 Stage 12- Controlling Application Flow

阶段目标 Stage Goals

覆盖的 Tersus 概念 Tersus Concepts Covered

完成本阶段后，您将会熟悉以下概念：
建模概念：激活槽（Activated slots）,验证（Validation）
建模技术：控制过程流程（Controlling process flow），控制弹出窗口外观（Controlling popup appearance）
有用的过程模板：分支（Branch）
有用的显示模板：提示（Alert）

建立应用程序功能模型 Application Functionality Modeled

我们将通过添加以下功能，继续为采购员发送“**采购订单**”（**Purchase Orders**）（**POs**）建立模型：

- 当至少有一个申请单被标识为紧急（urgent）时，就会显示一个提示，提醒采购员进行处理。
- “**订购**”（**Order**）按钮用于把采购订单（purchase orders）发给供应商。

> 本阶段建模在上个阶段最后输入的“**Tutorial 11-12**”项目中进行。

用户建模 User Modeling

使用分支模板 Use a Branch Template

当显示已批准的申请单清单时，我们想要检查清单中是否有“**紧急性**”（**Urgency**）为“**紧急**”（**Urgent**）的申请单。

这种检查应该在已批准申请单刚从数据库中提取出来时立刻进行：

进入到“**生成已批准申请单清单**”（**Generate Approved Requisition List**）。

现在我们要添加一个能检查申请单状态是否为紧急（urgent）的过程。

添加一个“基本/动作”（**Basic/Action**）。把它命名为 ***“检查申请单是否紧急”（Check if Requisition is Urgent）***。给它添加一个“**触发器**”（**trigger**）和“**出口**”（**exit**）。将出口命名为 ***“是”（Yes）***。把过程标识为“重复性”（**repetitive**）的。

创建连接“**查找/<记录>**”（**Find/<Records>**）和“**检查申请单是否紧急**”（**Check if Requisition is Urgent**）的流程。

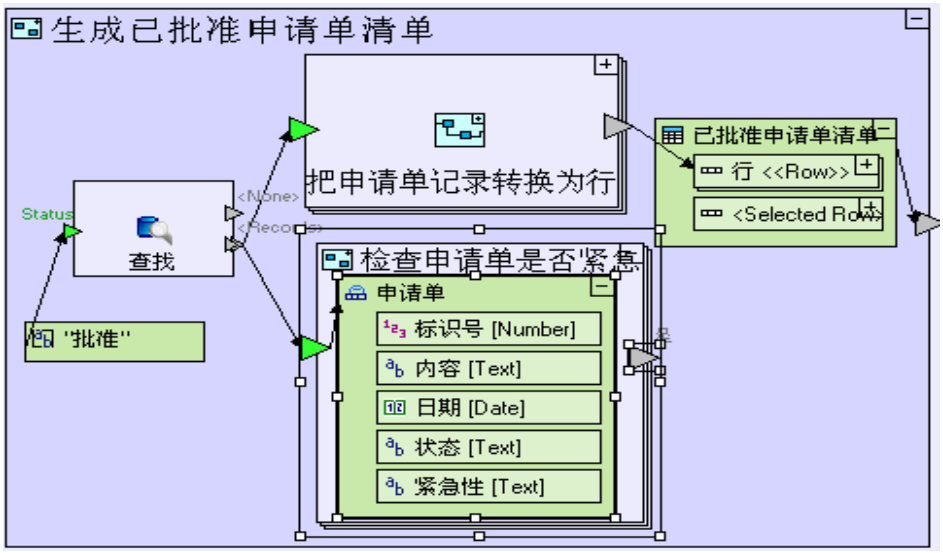
接下来我们为“实际紧急性检查”（actual urgency check）建立模型：

放大并进入“**检查申请单是否紧急**”（**Check if Requisition is Urgent**）。

拖拽一个“**申请单**”（**Requisition**）数据库记录。

创建一个连接“**检查申请单是否紧急**”（**Check if Requisition is Urgent**）的触发器和“**申请单**”（**Requisition**）的流程。

“**生成已批准申请单清单**”（**Generate Approved Requisition List**）模型外观应该与下图类似：



使用一个新模板“**分支**”（**Branch**）来进行检查，这个模板接收一个数值作为输入（input）并激活名称与数值相匹配的出口。

添加一个“**流程控制/分支**”（**Flow Control/Branch**）模板（  ）。

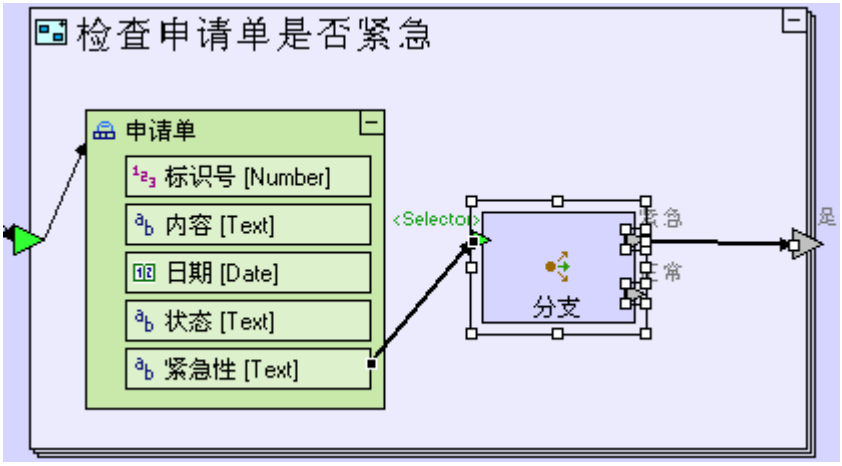
创建一个连接“**申请单/紧急性**”（**Requisition/Urgency**）和“**分支**”（**Branch**）的“**<选择器>**”（**<Selector>**）触发器的流程。

我们需要区分“**紧急**”（**Urgent**）和“**正常**”（**Normal**）数值。

（按**[F2]**键）将“**数值 1**”（Value 1）出口重新命名为 ***“紧急”（Urgent）***，把“**数值 2**”（Value 2）重新命名为 ***“正常”（Normal）***。

创建一个连接“**分支/紧急**”（**Branch/Urgent**）出口和“**是**”（**Yes**）出口的流程。

“**检查申请单是否紧急**”(**Check if Requisition is Urgent**)模型外观应该如下图所示：



当收到一个紧急申请单时，“**分支**”（**Branch**）会激活它的“**紧急**”（Urgent）出口并引发一个激活“**检查申请单是否紧急**”(**Check if Requisition is Urgent**)“**是**”（**Yes**）出口的应用程序流程。

> 激活一个出口，就是说无论出口是否输出数值，任何来自出口的流程都会被激活，因而如果出口和一个触发器连接，那个触发器也会被激活，也就意味着包含该触发器的过程也可被激活。

当提取一个非紧急申请单时，我们执行的功能就不需要进行运行任何动作；所以我们不定义来自“**分支/正常**”（**Branch/Normal**）出口的流程。实际上我们可以删除这个出口；我们留着这个出口只是为了改善模型的可读性（readability），将来也可能用这个出口增强模型的功能。

使用提示模板(向用户显示提示信息) Use an Alert Template (to display an alert to the user)

当“**检查申请单是否紧急/是**”（ **Check if Requisition is Urgent/Yes**）出口至少被激活一次时，我们想给用户显示一个提示。我们可以使用“**提示**”（**Alert**）模板来做到这一点。

在使用“提示”模板之前，我们必须先考虑要把模板放在模型中的什么地方。有 3 种选择：

1. “**检查申请单是否紧急**”(**Check if Requisition is Urgent**)--这种情况下，每有一个紧急申请单，就会反复显示提示（alert）。
2. “**生成已批准申请单清单**”（**Generate Approved Requisitions List**）--这种情况下，不论有多少紧急申请单，提示都只显示一次，但只在填充申请单清单（发生在此过程以外）之前显示。
3. “**填充已批准申请单清单**”（**Populate Approved Requisitions List**）--这种情况下，提示会在填充申请单清单时同时出现。

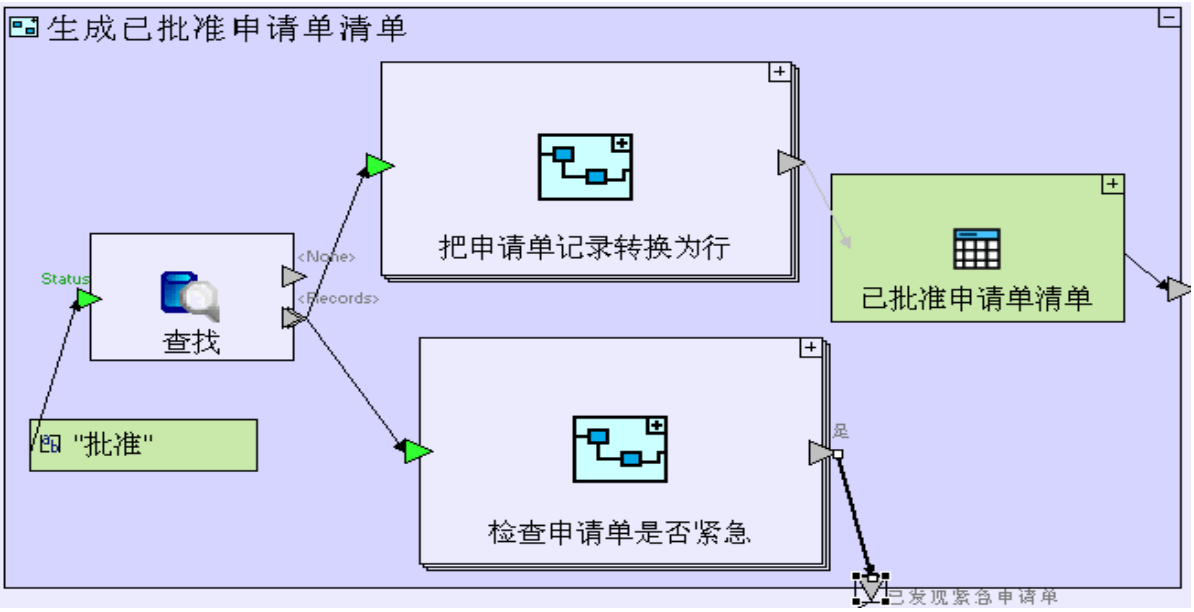
第三种选择似乎最为理想：

缩小当前页面并转到“**生成已批准申请单清单**”（Generate Approved Requisition List）。

给它添加一个出口。把出口命名为 “***已发现紧急申请单***” ***(Urgent Requisitions Found)***。

创建一个连接“**检查申请单是否紧急/是**”（ Check if Requisition is Urgent/Yes）出口和“**生成已批准申请单清单/已发现紧急申请单**”（Generate Approved Requisition List/Urgent Requisition Found）的流程。

“**生成已批准申请单清单**”（Generate Approved Requisition List）模型外观应该与下图类似：



我们使用“**提示**”（**Alert**）模板来显示提示（alert）。

缩小当前页面并转到“**填充已批准申请单清单**”（Populate Approved Requisition List）。

提示（alert）需要一个信息（message）来充盈（feed）它的“**<信息>**”（<**Message**>）触发器。

添加一个“**常数/文本**”（Constants/Text）。

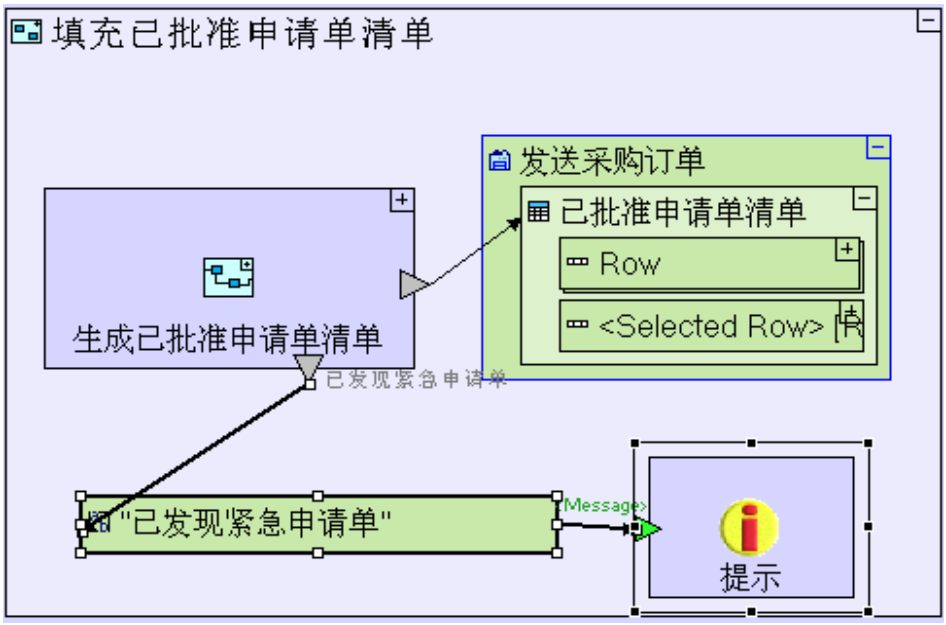
把它命名为 “***已发现紧急申请单***”***(Urgent requisitions are waiting to be processed***）。

创建一个连接常数和“**提示/<信息>**”（Alert/<Message>）的流程。

最后，激活含有“**提示**”（**Alert**）信息的常数时，就会激活“**提示**”（**Alert**）：

创建一个连接“**生成已批准申请单清单/已发现紧急申请单**”（Generate Approved Requisition List/ Urgent Requisition Found）和“**紧急申请单正在等待处理**”（“Urgent requisitions are waiting to be processed”）常数的流程。

“**填充已批准申请单清单**”（Populate Approved Requisition List）模型现在外观应该与下图类似：



保存您的工作成果并在浏览器中查看应用程序。

点击“**采购员**”（Purchaser）透视图标签（tab）。

点击“**发送采购订单**”（Issue Purchase Orders）视图标签（tab）。

您会看到一个提示（alert）弹出窗口，如下面的屏幕快照所示。

按条件显示弹出窗口 Display a Popup Conditionally

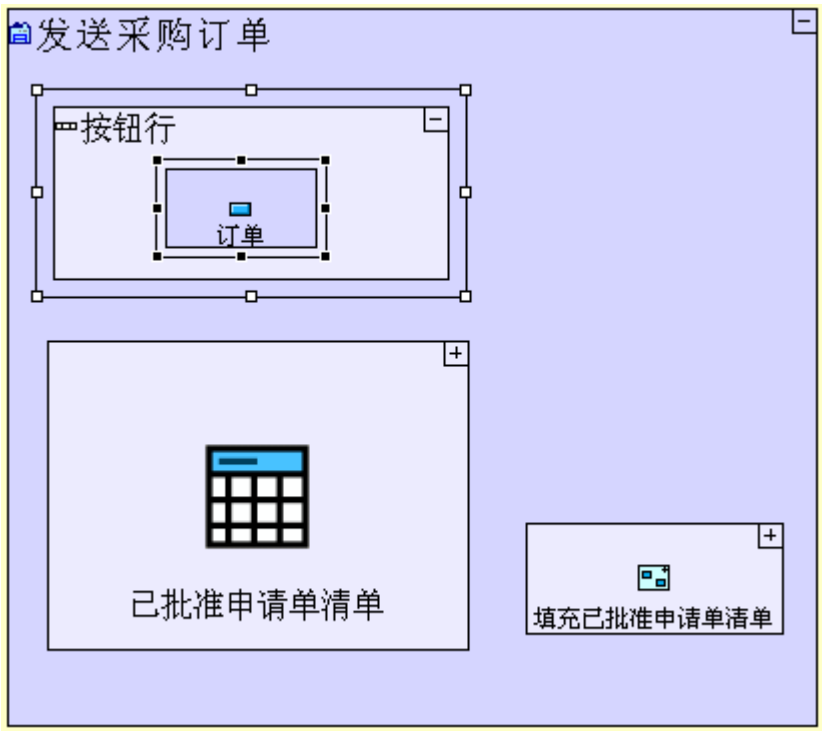
现在我们给视图添加一个“**订购**”（Order）按钮，后面的阶段里面将会用这个按钮来发送选定申请单的采购订单：

缩小页面并转到“**发送采购订单**”（Issue Purchase Orders）视图。

添加一个“**显示/行**”（**Display/Row**）。把它命名为 “***按钮行” (Button Row)*** 并放大到“**发送采购订单**”（**Issue Purchase Orders**）。

添加一个“**显示/按钮**”（**Display/Button**）。把它命名为 “***订单”(Order)***。

“**发送采购订单**”（**Issue Purchase Orders**）视图模型现在外观应该与下图类似：



这个按钮应该打开一个用来输入采购订单的弹出窗口，但是只有在已经从申请单清单中选定一个申请单的情况下才应该打开这个弹出窗口：

放大“**订单**”（**Order**）按钮。

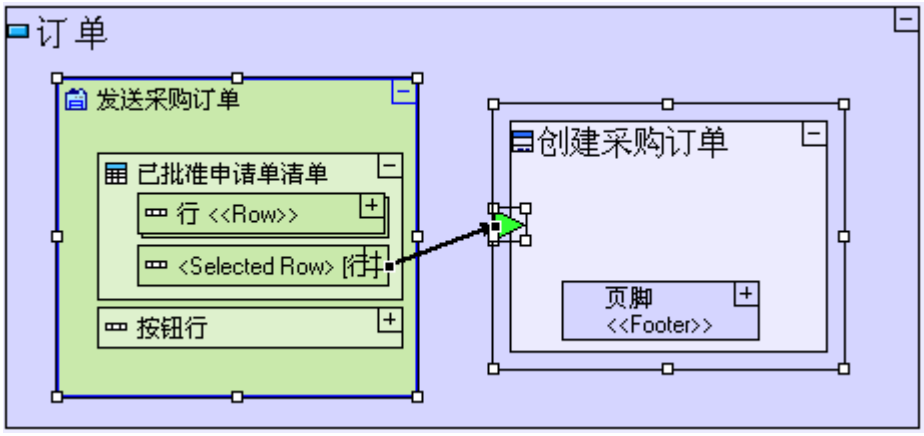
给“**发送采购订单**”（**Issue Purchase Orders**）视图添加一个上级索引。

添加一个“**显示/弹出窗口**”（**Display/Popup**）。

把它命名为 “***创建采购订单”(Create Purchase Order)***。给它添加一个“**触发器**”（**trigger**）。

创建一个连接“**发送采购订单/已批准申请单清单/选定行**”（**Issue Purchase Orders/Approved Requisitions List/Selected Row**）和“**创建采购订单**”（**Create Purchase Order**）的流程。

“**订购**”（Order）按钮模型现在外观应该与下图类似：



只有当“**获得表格选择**”（Get Table Selection）从“**<选定行>**”（**<Selected Row>**）出口退出时，“**创建采购订单**”（Create Purchase Order）弹出窗口才会打开，另外由于选定的“**申请单行**”（Requisition Row）数据结构也被传送到流程中，我们立即就能获取“**创建采购订单**”（Create Purchase Order）内选定的申请单的详情（无需再次提取）。

保存您的工作成果并在浏览器中查看应用程序。

点击“**采购员**”（Purchaser）透视图标签（tab）。

点击“**发送采购订单**”（Issue Purchase Orders）视图标签（tab）。

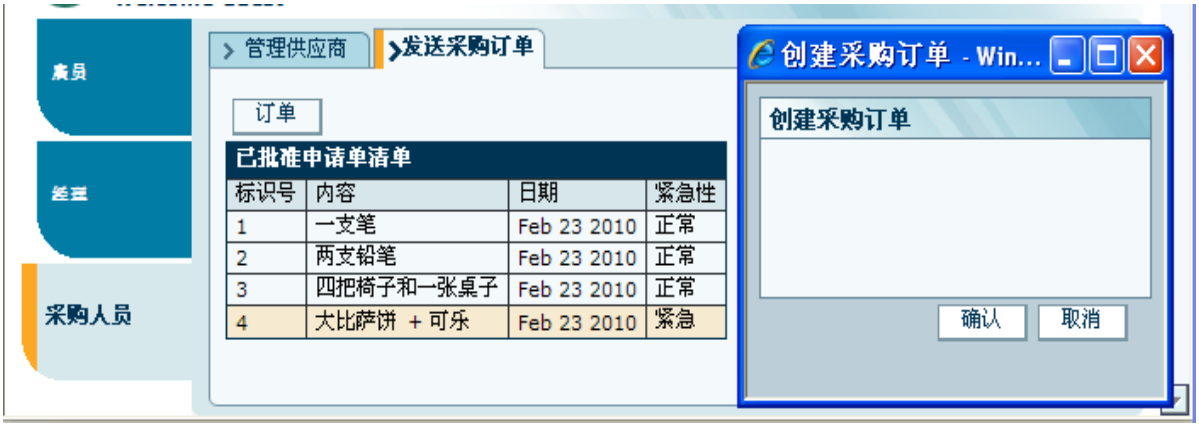
点击“**订购**”（Order）按钮（在选定某一行之前）。

由于还没有选定任何一行，所以不会有什么变化。

从“**申请单清单**”（Requisition List）中选择一个行。

点击“**订购**”（Order）按钮。

“**创建采购订单**”（Create Purchase Order）弹出窗口外观应该与下图类似：



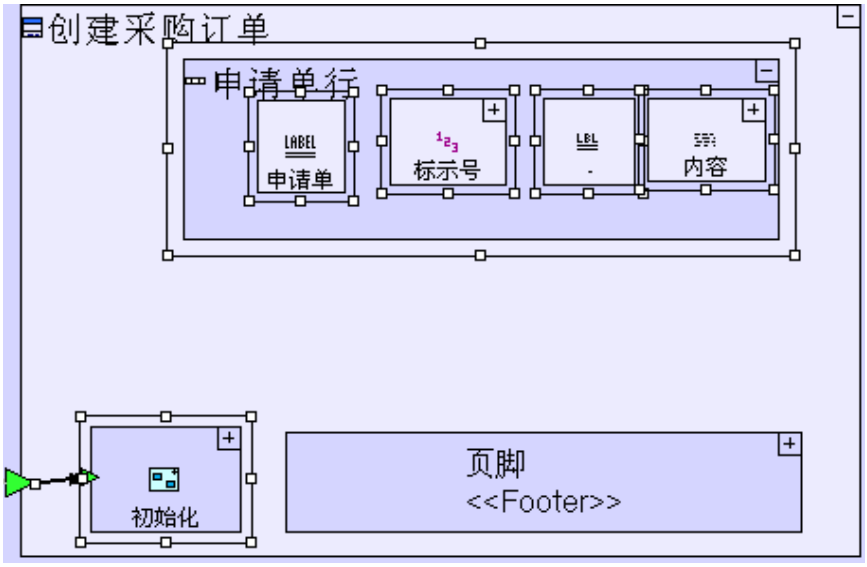
完成阶段 12 Completing stage 12

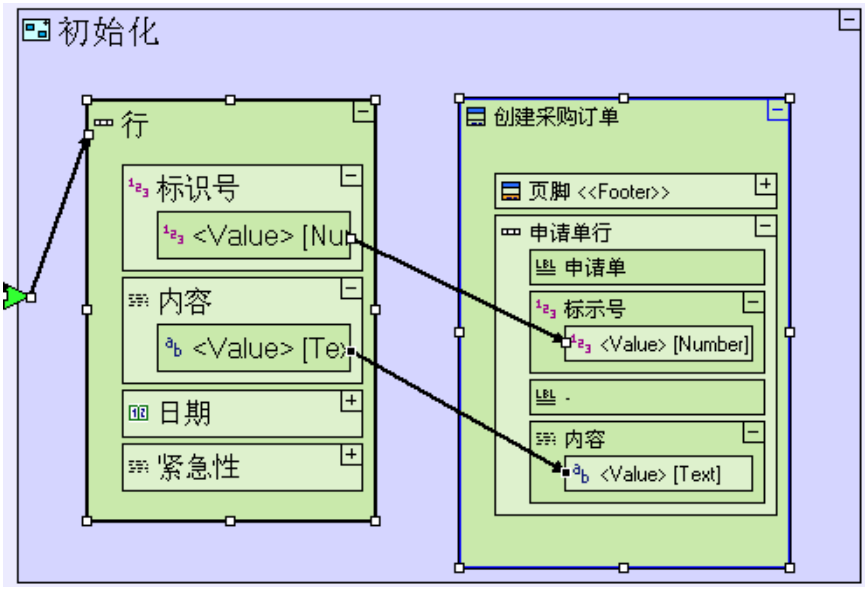
现在我们输入项目样本“**Tutorial 12-13**”并把它作为本教程下一个阶段的基础。

> 想再了解如何输入一个项目样本，请参见**阶段 2** 最后“**输入项目样本**”的部分。

这个项目样本具备到目前为止我们建立的模型的所有功能。
这个项目样本也包括以下一些其它的功能：

1. 添加显示单元，用于在“**创建采购订单**”（**Create Purchase Order**）弹出窗口中显示“**申请单**”（**Requisition**）详情。用选定行中的数据初始化（initialize）显示单元。

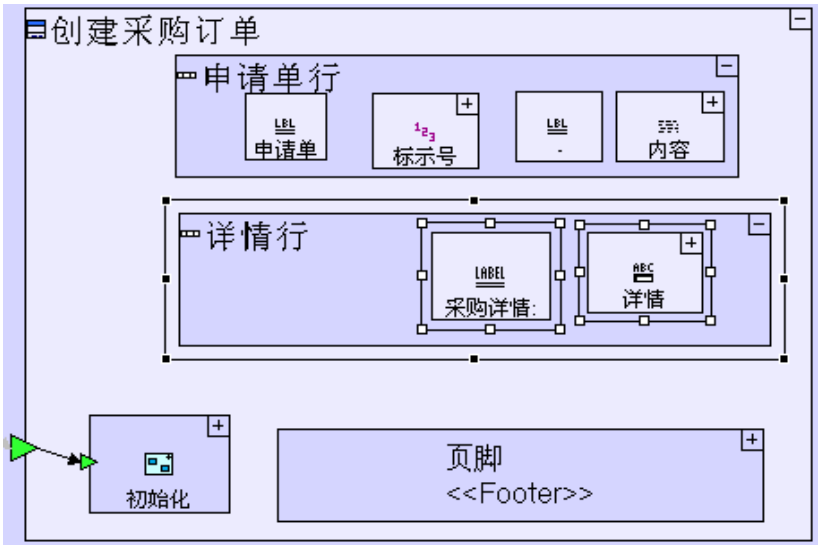


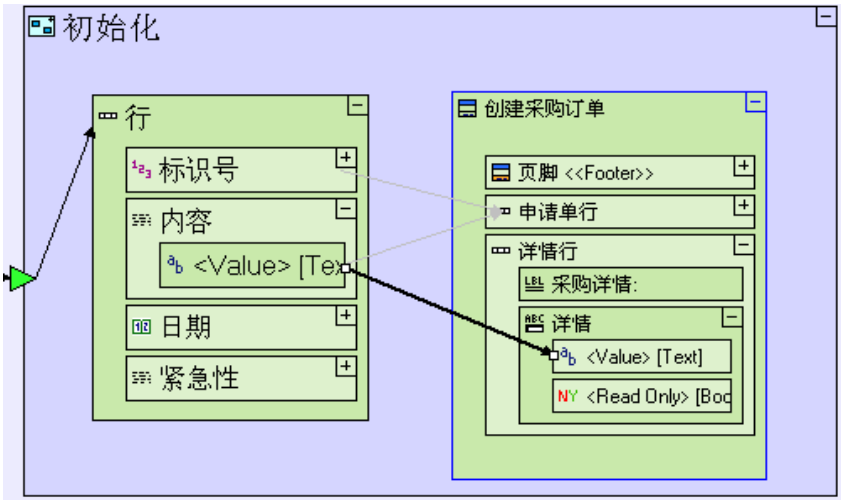


如何建模	位置
添加“**显示/行**”（**Display/Row**）。把它命名为 “***申请单行***”（***Requisition Row***）。 添加一个“**基本/动作**”（**Basic/Action**）。把它命名为“***初始化***”（***Initialize***）并给它添加一个触发器。	“**创建采购订单**”（**Create Purchase Order**）弹出窗口

创建一个连接“**创建采购订单**”（**Create Purchase Order**）触发器和“**初始化**”（**Initialize**）触发器的流程。	
添加一个“**显示/标签**”（**Display/Label**）。把它命名为***申请单 Requisition***”。 添加一个“**显示/数字显示**”（**Display/Number Display**）。把它命名为 “**标示号**”（***Id***）。 添加一个“**显示/标签**”（**Display/Label**）。把它命名为“**-**”（连字符）。 添加一个“**显示/文本显示**”（**Display/Text Display**）。把它命名为 “***内容***”（***Description***）。	“**申请单行**”（**Requisition Row**）
拖拽一个“**申请单行**”（**Requisition Row**）（以创建一个显示数据单元）。 把它和“**初始化**”（**Initialize**）的触发器连接起来。 给“**创建采购订单**”（**Create Purchase Order**）添加一个上级索引。 连接“**申请单行/标识号/<数值>**”（**Requisition Row/Id/<Value>**）和“**创建采购订单/申请单行/标识号/<数值>**”（**Create Purchase Order/Requisition Row/Id/<Value>**）。 连接“**申请单行/内容**”（**Requisition Row/Description**）和“**创建采购订单/申请单行/内容/<数值>**”（**Create Purchase Order/Requisition Row/Description/<Value>**）。	“**初始化**”（**Initialize**）过程

2.给“**创建采购订单**”（**Create Purchase Order**）弹出窗口添加一个“**详情**”（**Details**）输入字段。用“**申请单**”的“**内容**”（**Requisition's Description**）初始化“**详情**”（**Details**）字段。

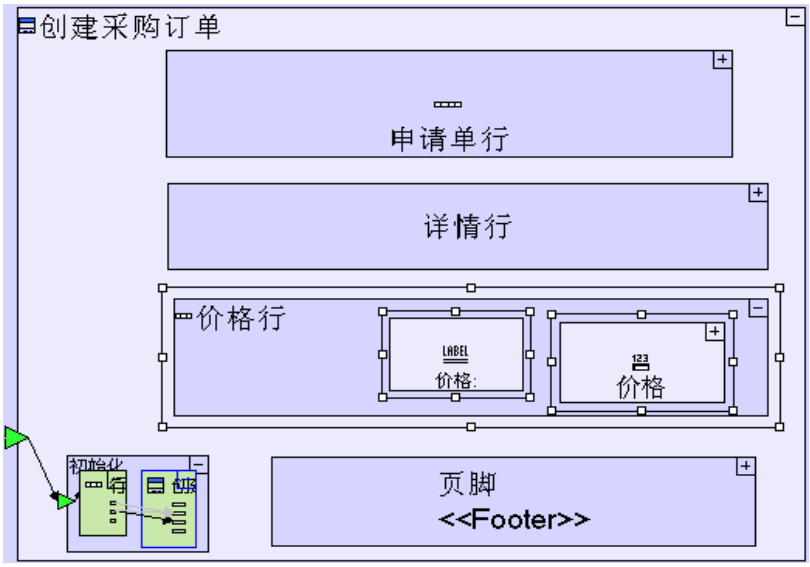


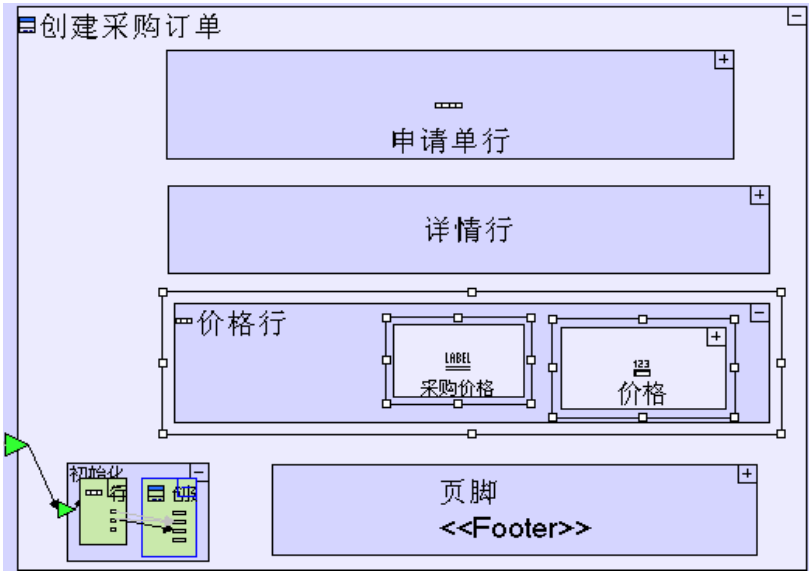


如何建模	位置
添加“**显示/行**”（Display/Row）。把它命名为“***详情行”(Details Row)***。	“**创建采购订单**”（Create Purchase Order）弹出窗口

添加“**显示/标签**”（Display/Label）。把它命名为 “***采购详情:”(Order Details)***。 添加一个“**显示/文本输入字段**”（Display/Text Input Field）。把这它命名为 “***详情”(Details)***。	“**详情行**”（Details Row）
连接“**申请单行/内容/<数值>**”(Requisition Row/Description/<Value>)和“**创建采购订单/详情行/详情/<数值>**”（Create Purchase Order/Details Row/Details/<Value>）。	“**初始化**”（Initialize）过程

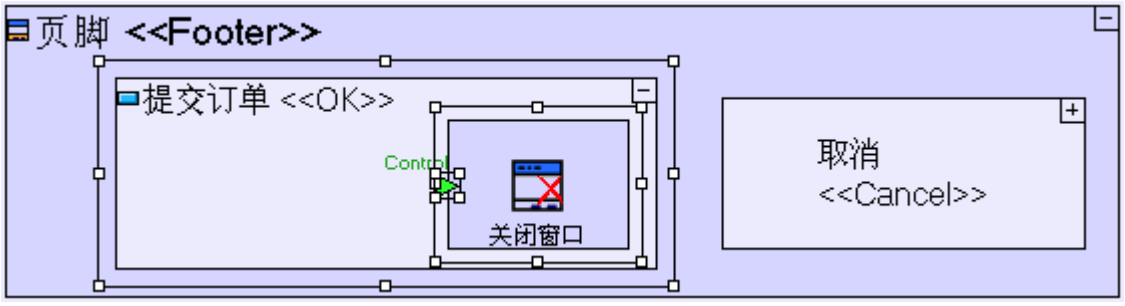
3.给“**创建采购订单**”（Create Purchase Order）弹出窗口添加一个“**价格**”(Price)输入字段。





如何建模	位置
添加“**显示/行**”（**Display/Row**）。把它命名为 “***价格行”(Price Row)***。	“**创建采购订单**”（**Create Purchase Order**）弹出窗口
添加“**显示/标签**”（**Display/Label**）。把它命名为 “***采购价格:”(Order Price:***）。 添加一个“**显示/数字输入字段**”（**Display/Number Input Field**）。把它命名为 “***价格”(Price)***。	“**价格行**”（**Price Row**）

4.创建一个“**提交订单**”（**Submit Order**）按钮，用于选择一个供应商后发送采购订单。



如何建模	位置
将“**确认**”（OK）按钮重新命名为 “***提交订单”(Submit Order)***。	“**创建采购订单/页脚**”（Create Purchase Order/Footer）
添加“**显示动作/关闭窗口**”（Display Actions/Close Window）。 添加一个“**控制**”（Control）触发器（通过“**添加单元**”（Add Element））。	“**提交订单**”（Submit Order）按钮

请注意：在提供的项目样本中，应用程序模型验证（一个内置的检查您的模型并通知您潜在的问题的程序）失败。

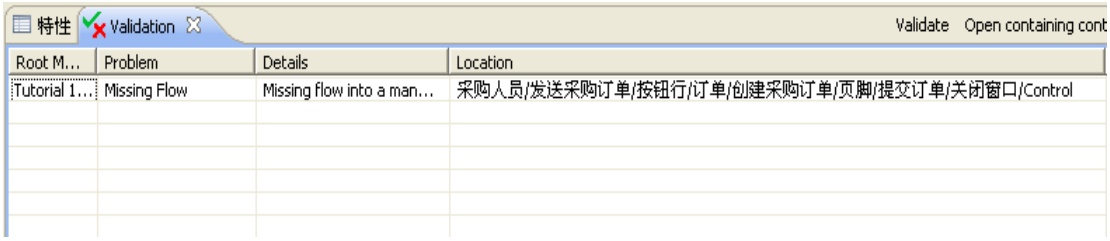
您可以通过以下操作看到这一点。

在“**储存器管理器**”（Repository Explorer）中，双击根模型(Tutorial 12-13/Tutorial 12-13)。

转换到“**验证**”（Validation）视图。如果当前的eclipse透视图（**Tersus Modeling**）中没有这个视图的话，请从eclipse菜单中选择“**窗口/展示视图/其他…**”(Window/Show View/Other...)，再从“**展示视图**”（Show View）对话框中选择“**Tensus/验证**”（Tersus/Validation)。

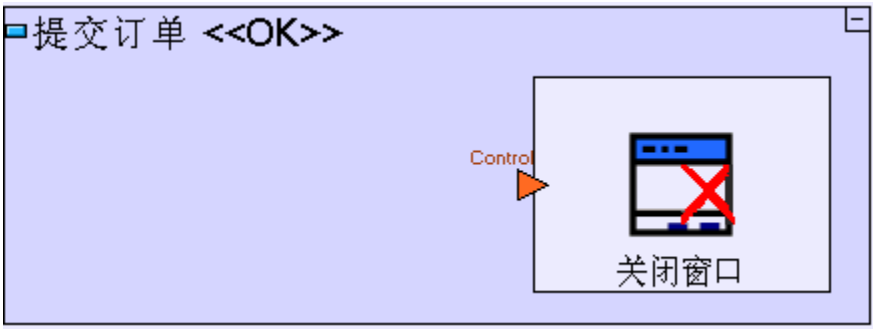
点击“**验证**”（Validation）视图上面的“**验证**”（Validate）按钮。

这样操作之后就会显示如下图所示的“**验证**”（Validation）视图：



双击验证行（validation row）的“**根目录**”(Root Model)那一列。

模型编辑器将转到“**关闭窗口**”（Close Window），“**关闭窗口**”（Close Window）的“**控制**”（Control）触发器以红色突显出来，以引起我们对验证问题（validation problem）的注意（如下面的屏幕快照所示）：



产生验证问题的原因是“**提交订单**”（**Submit Order**）按钮的“**关闭窗口**”（**Close Window**）子过程有一个强制性的触发器（“**Control**”），并且没有任何流程流入这个触发器。在下一个阶段我们对流程进行定义之后，这些问题就会解决。

您现在可以开始学习**阶段 13**，在**阶段 13** 里面，我们将学习管理关系数据的技术。

实例 See It Live

点击此处在另一个窗口里面打开实际项目。

阶段 13-关系数据建模 Stage 13 – Modeling Relational Data

阶段目标 Stage Goals

覆盖的 Tersus 概念 Tesus Concepts Covered

本阶段完成时，您将会熟悉以下概念：

建模概念：服务(Service)过程与动作(Action)过程，自动交易支持，〈已完成〉(〈Done〉)出口。

建模技术：相关数据建模，保持索引完整性(Maintaining referential integrity)，基于复合数据结构的选取器（Chooser based on a composite data structure）。

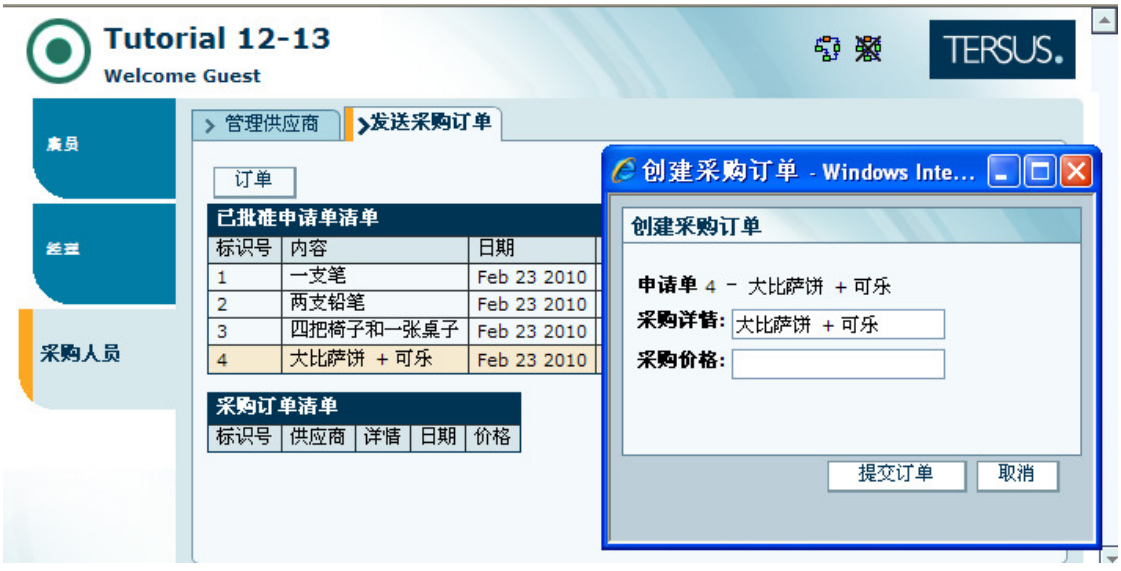
有用的过程模板：服务（Service）

建立应用程序功能模型 Application Functionality Modeled

您所载入的项目样本包含了“**创建采购订单**”（**Create Purchase Order**）弹出窗口用户界面建模的内容：

- 已选定申请单的“**标识号**”(**Id**)和“**内容**”（**Description**）。
- 一个“**详情**”（**Details**）输入字段。如果采购订单的内容与申请单相同，则以申请单的内容（description）初始化。采购员可更新采购申请单详情。
- “**价格**”(**Price**)输入字段（订单总价）。
- “**提交订单**”（**Submit Order**）按钮,此按钮包含了一个因为没有流程连接而“**未**”(**not**)运行的“**关闭窗口**”（**Close Window**）过程。

如果您在浏览器中查看应用程序，应用程序外观应该如下图所示：



“创建采购订单”(Create Purchase Order)弹出窗口需要进一步建模：

1. 一个“**供应商**”（**Supplier**）选取器，以显示从“**供应商**”（**Supplier**）数据库表格中提取的供应商清单。
2. “**提交订单**”（**Submit Order**）按钮需要另外建模以处理数据库操作：

保存用户输入的采购订单详情和更新申请单以表示已下订单。

本阶段建模在上个阶段最后输入的项目“Tutorial 12-13”中进行。

用户建模 User Modeling

根据数据结构使用选取器 Use a Chooser Based on a Data Structure

我们想添加一个显示供应商清单的下拉式选单：

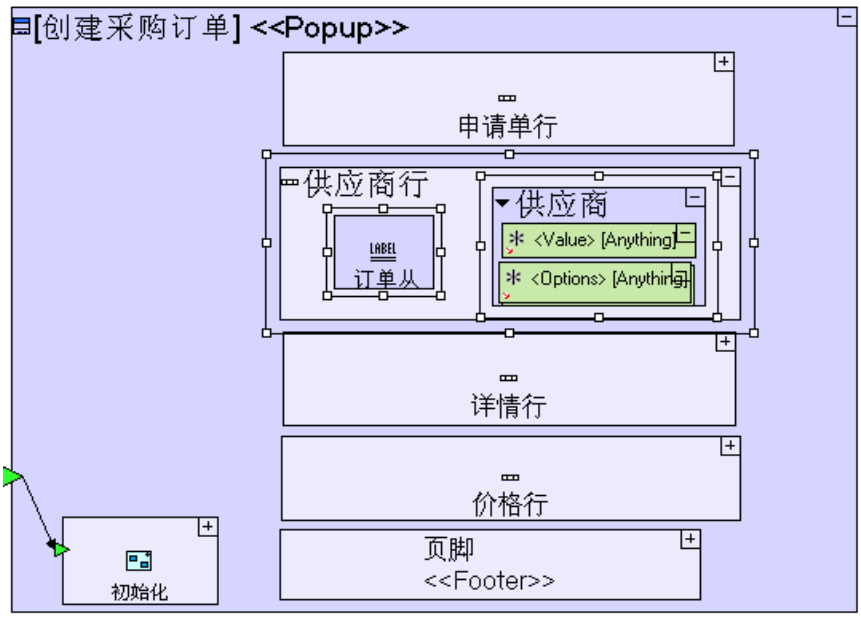
进入“**创建采购订单**”（Create Purchase Order）弹出窗口。

添加一个“**显示/行**”（Display/Row）。把它命名为 “**供应商行**”(*Supplier Row*)。放大到“**创建采购订单**”（Create Purchase Order）。

添加“**显示/标签**”（Display/Label）。把它命名为 “**订单从:**”(*Order From:*)。

添加“**显示/选取器**”（Display/Chooser）。把它命名为 “***供应商*”(*Supplier*)**。

“**创建采购订单**”（Create Purchase Order）弹出窗口模型外观应该与下图类似：



回忆一下，我们以前使用“**选取器**”（Chooser）显示(以指定申请单紧急性)时，用文本数值（“**正常**”（Regular）和“**紧急**”（Urgent），定义为常数）填充“**选取器**”（Chooser）显示。

这次选取器中要显示的数值（供应商名称）必须从“**供应商**”（Supplier）表格中提取。我们要用从数据库中提取的“**供应商**”（Supplier）记录来填充“**供应商/<选项>**”（Supplier/<Option>），并且由于“**公司名称**”（Company Name）是“**供应商**”（Supplier）数据结构里的第一个文本字段，选取器会自动显示每个供应商的公司名称：

进入弹出窗口的“**初始化**”（Initialize）过程。

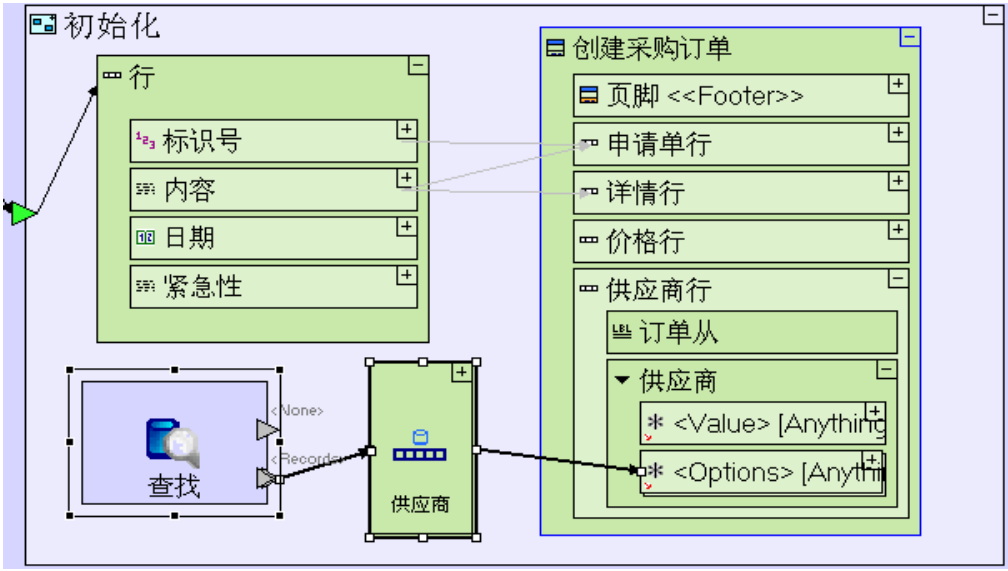
添加一个“**数据库/查找**”（Database/Find）.

从大纲/储存器（outline/repository）中拖出一个“**供应商”（Supplier）**数据库记录（“**管理供应商”（Manage Suppliers）**视图和“**供应商清单”（Supplier List）**表格中都有使用），把它设置为重复性(repetitive)的。

创建一个连接“**查找/<记录>”（Find/<Records>）**和“**供应商”（Supplier）**数据结构的流程。

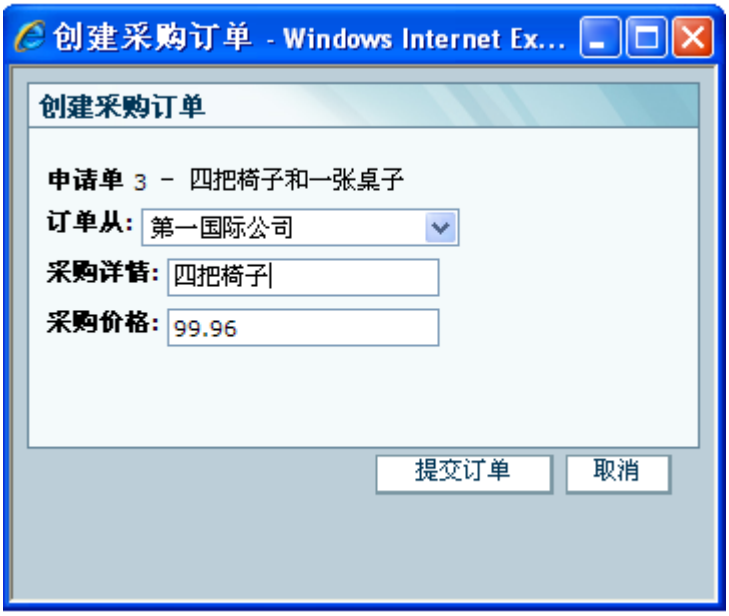
创建一个连接“**供应商”（Supplier）**数据结构和“**创建采购订单/供应商行/供应商/<选项>”（Create Purchase Order/Supplier Row/Supplier/<Options>）**的流程。

“初始化”(Initialize)模型外观应该与下图类似：



我们可以看到下拉式选单只显示每个供应商的公司名称，但一旦用户选择了一个供应商，**“供应商/<数值>”(Supplier/<Value>)**里面就会填充所选供应商的全部记录（后面我们会对**“供应商/<数值>/标识号”(Supplier/<Value>/Id)**感兴趣），而无需再次从数据库中提取详情。

保存您的工作成果并在浏览器中查看应用程序。应用程序外观应与下图类似：



使用服务模板(过程必须在服务器上运行时) Use a Service Template (when a process must run on the server)

现在我们转到完成“**提交订单”(Submit Order)**按钮建模的问题上。我们将给“**提交订单”(Submit Order)**按钮添加一个过程，以处理上面提到的数据库问题。

到目前为止，我们主要用“**动作”(Action)**模板作为建模的容器（container）。把“**动作”(Action)**作为容器可让 Tersus 运行时间引擎决定“**动作”**的每一个子模型在哪里运行 – 在客户机还是在服务器上。

绝大多数情况下，过程在哪里执行是很明显的。如果过程的用途是显示用户界面单元（或提供索引给用户界面单元）时，它就必须并且会自动在客户机（浏览器）上运行。各种数据库动作则与此相反，它们总是在服务器上运行。

但还有一些过程既可在客户机上又可在服务器上运行的情况。在这种情况下，Tersus 运行时间引擎会在客户机上运行过程，除非建模者明确规定这些过程在服务器上运行。建模者决定过程在哪里运行时，主要考虑因素是性能。一般而言，将控制许多数据的过程和读取集中管理数据的过程放在服务器是个好主意。如果一个过程有许多内部数据流程，并且它的一些子过程仅在服务器端，那么在服务器上运行整个过程也是行得通的，这样做可以避免服务器和客户机之间产生许多通信量（traffic）。

除了性能问题之外，还有其它一些虽然不是强制性但建议在服务器上运行模型的情况。数据库“**交易”(transaction)** 就是这些情况中的一种。当模型运行多个数据库操作（如更新数据库一个或一个以上表格中的多个记录，并且这些更新彼此相关时），所有更新必须作为一个整体进行，要么全部成功，要么全部失败。Tersus 提供了一个自动的交易机制。指定一个过程在服务器上运行可确保当从客户机过程呼叫这个过程时，就会打开单独一个数据库交易。这个交易仅在过程成功时才会进行（如果过程没有成功则会被退回）。

要定义在服务器上运行一个过程，请使用“**服务”(Service)**模板：

进入“**提交订单”(Submit Order)**。

添加“**基本/安全服务”(Basic/Secure Service)**。把它命名为 “**处理数据库”(*Handle Database*)**，放大并进入到”**提交订单”(Submit Order)**。

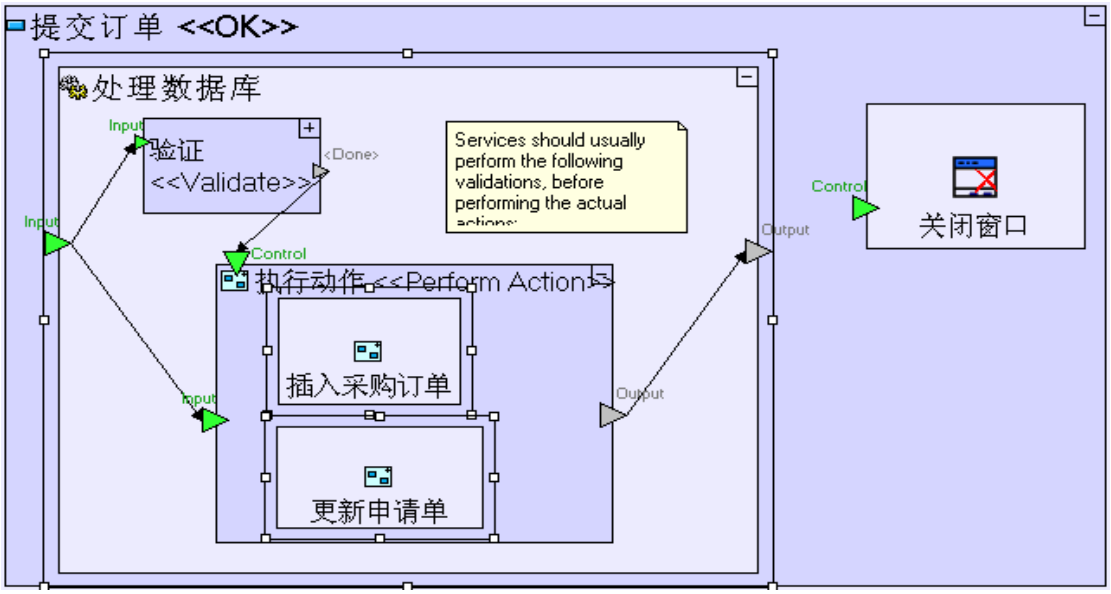
“服务”(Service)模板是一个“**复合”(composite)**模板，这就意味着它包含预定义建模 – “**验证”(Validate)**动作、“**执行动作”(Perform Action)**动作和相关的触发器、出口和流程。这种建模并非是强制性的，而只是对于读取服务器资源的 Tersus 过程结构的一种建议。更多详情请参阅模型中的“**注意事项”(Notes)**（黄色字体）。

“处理数据库”(Handle Database)过程包含 2 个子过程，一个用来处理采购订单，另一个用来处理申请单的状态更新：

添加一个“**基本/动作”**（Basic/Action）。把它命名为 “***插入采购订单”(Insert Purchase Order)***。

添加一个“**基本/动作”**（Basic/Action）。把它命名为 “***更新申请单”(Update Requisition)***。

“提交订单”(Submit Order)按钮模型外观应与下图类似：



> 注意：两个子过程是基于普通的**“动作”（Action）**模板而不是**“服务”（Service）**模板 - 它们会在服务器运行，因为它们包含在**“处理数据库”（Handle Database）**中，而**“处理数据库”（Handle Database）**是一种**“服务”（Service）**。这两个子过程都不会产生单独的交易。由于它们包含在母交易中，因此插入（insert）和更新（update）要么都成功，要么都失败。

定义（采购订单的）数据库记录 Define a Database Record for the purchase order)

我们从**“采购订单”(Purchase Order)**数据库记录的定义开始：

放大并进入**“插入采购订单”(Insert Purchase Order)**。

添加一个**“数据类型/数据库记录”（Data Types/Database Record)**。把它命名为 **“采购订单”(*Purchase Order*)** 并进入**“*Purchase Order*”**。

选择**采购订单->特性(properties),**把 **tablename** 命名 **“采购订单”(Purchase Order)**

每一个采购订单应该包含以下数据：

1. 一个唯一的标识号（identifier)：

   添加**“数据类型/数字”（Data Types/Number)**。把它命名为 **“*标识号*”(*Id*)**。

   选择**标识号->特性(properties),** 点击  把它命名为 **“columnName”**

   把 “columnName”命名为 **“Id”**

2. 发送采购订单的申请单。现在我们把标识发送采购订单的申请单的标识号保存在**“申请单”（Requisitions）**表格中：

添加“**数据类型/数字”（Data Types/Number）**。把它命名为 “**申请单标识号”（*Requisition Id*）**。

选择**申请单标识号->特性(properties)**，点击  把它命名为 **“columnName”**

把 “columnName”命名为 **“Requisition_Id”**

3.接收采购订单的供应商。我们把标识接收采购订单的供应商的标识号保存在“**供应商”（Suppliers）**表格中：

添加“**数据类型/数字”（Data Types/Number）**。把它命名为 “**供应商标识号”（*Supplier Id*）**。

选择**供应商标识号->特性(properties)**，点击  把它命名为 **“columnName”**

把 “columnName”命名为 **“Supplier_Id”**

我们保存的是“**申请单和供应商”(Requisition & Supplier)**的“**标识号”（Id)**，而不是给采购订单提供索引时显示给用户的实际数据（对于“**供应商”(Supplier)**表格，实际数据一般是“**公司名称”(Company Name)**或“**联系人姓名”(Contact Name)**），会没有关系数据模型经验的读者可能觉得奇怪。实际上，由于“**标识号”（Id）**永不会改变，但详情可能会频繁改变（“**联系人姓名”(Contact Name)**就是一个很好的例子），所以这是关系数据建模的正确方法。有了“**标识号”（Id)**，您就很容易查找到并显示所有的供应商详情。

4. 采购订单详情：

添加“**数据类型/文本”（Data Types/Text）**。把它命名为 “**详情”（*Details*）**。

选择**详情->特性(properties)**，点击  把它命名为 **“columnName”**

把 “columnName”命名为 **“Details”**

5. 发送订单的日期：

添加“**数据类型/日期”（Data Types/Date）**。把它命名为 “**日期”（*Date*）**。

选择**日期->特性(properties)**，点击  把它命名为 **“columnName”**

把 “columnName”命名为 **“date”**

6. 与供应商达成的价格：

添加“**数据类型/数字”（Data Types/Number）**。把它命名为 “**价格”（*Price*）**

选择**价格->特性(properties)**，点击  把它命名为 **“columnName”**

把 “columnName”命名为 **“price”**。

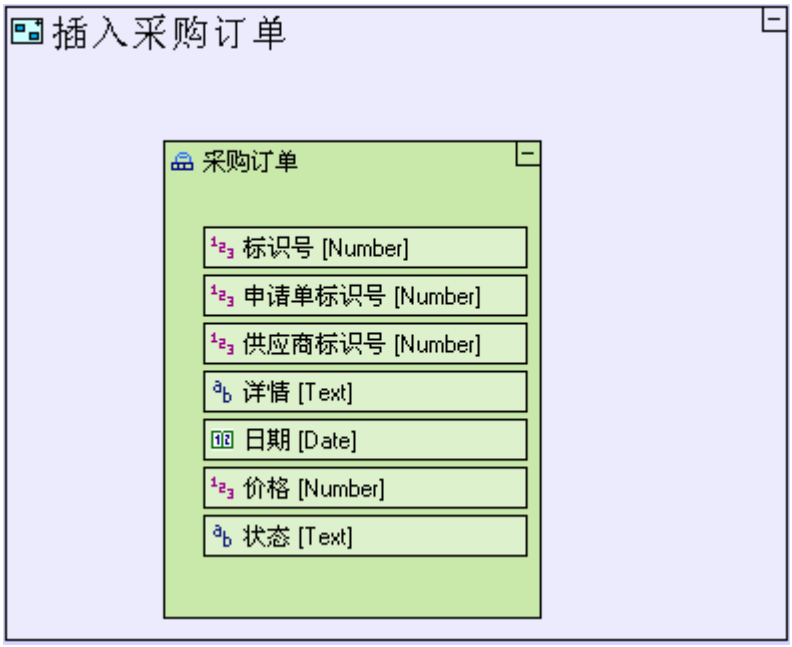
7. 订单的状态（已发送订单或已送货）：

添加“**数据类型/文本”（Data Types/Text）**。把它命名为 “**状态”（*Status*）**。

选择**状态->特性(properties)**，点击  把它命名为 **“columnName”**

把 “columnName”命名为 **“Status”**

“插入采购订单”（Insert Purchase Order）模型外观应该与下图类似：



正确放置上级索引（不是在服务中） Positioning an Ancestor Reference Correctly (not in a service)

现在我们已经定义了“**采购订单”（Purchase Order）**的数据结构，接着我们必须用关系数据（部分关系数据来自“**创建采购订单”（Create Purchase Order）**

弹出窗口）填充“**采购订单”（Purchase Order）**的数据结构，因此我们需要使用一个**“上级索引”(Ancestor Reference)**。

问题是这个数据填充过程应该放在模型的什么位置？

我们可能想把它放在“**插入采购订单”（Insert Purchase Order）**过程中，但由于这个过程是在服务器运行的，所以它不能直接给弹出窗口（一个客户机端单元）提供索引。解决的方法是把上级索引放在上级过程中，并把数据通过触发器传送到这个过程中。因为过程的上两级的过程（**“处理数据库”（Handle Database）**）是一个**“服务”（Service）**过程，所以我们将把上级索引放到上两级的过程中：

缩小当前页面并转到**“提交订单”（Submit Order）**。

给**“创建采购订单”（Create Purchase Order）**添加一个上级索引。

我们需要从显示中提取四个数据单元：申请单的标识号、选定供应商的标识号、采购订单详情和价格。我们将添加一个创建数据库记录并把数据库记录输送到**“处理数据库”（Handle Database）**和**“插入采购订单”(Insert Purchase Order)**的过程，因此我们必须定义流程和触发器：

添加一个**“基本/动作”（Basic/Action）**。把它命名为**“创建采购订单记录”*(Create Purchase Order***

Record）。

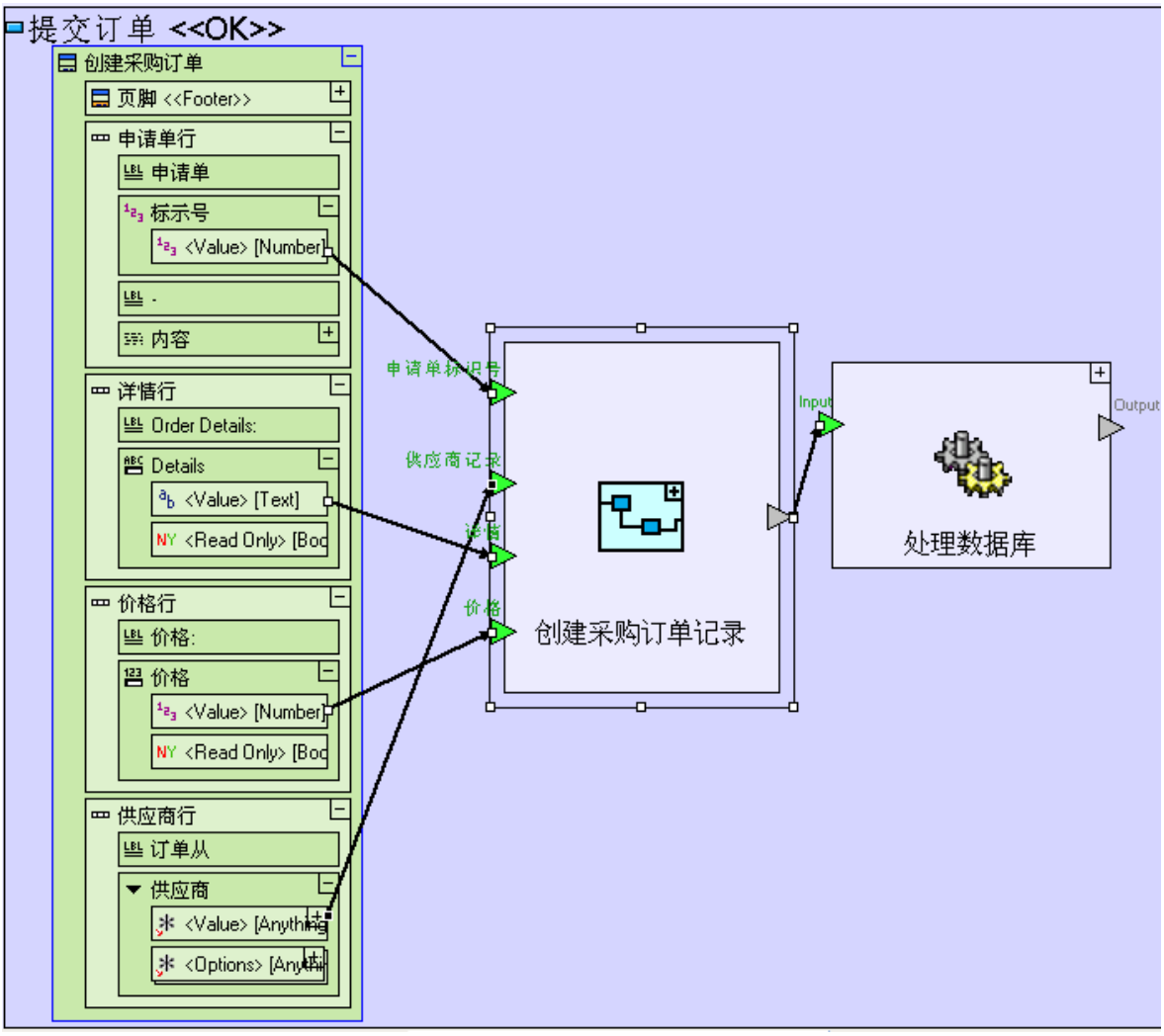
给“**创建采购订单记录**”（Create Purchase Order Record）添加 4 个触发器。把它们命名为“**申请单标识号**”（*Requisition Id*）、“**供应商记录**”（*Supplier Record*）、“**详情**”（*Details*）和“**价格**”（*Price*）。

创建以下 4 个流程：

来源（在“**创建采购订单**”（Create Purchase Order）中）	目标（“**创建采购订单记录**”（Create Purchase Order Record）触发器）
“**申请单行/标识号/<数值>**”（Requisition Row/Id/<Value>）	“**申请单标识号**”（Requisition Id）
“**供应商行/供应商/<数值>**”（Supplier Row/Supplier/<Value>）	“**供应商记录**”（Supplier Record）
“**详情行/详情/<数值>**”（Details Row/Details/<Value>）	“**详情**”（Details）
“**价格行/价格/<数值>**”（Price Row/Price/<Value>）	“**价格**”（Price）

添加一个“**创建采购订单记录**”（Create Purchase Order Record）出口。创建一个连接出口和“**处理数据库/输入**”（Handle Database/Input）触发器的流程。

现在模型外观应与下图类似：



填充采购订单记录（以采购订单详情） Populate the Purchase Order Record (with purchase order details)

放大并进入“**创建采购订单记录”（Create Purchase Order Record）**。
从储存器/大纲（repository/outline)中拖拽一个“**采购订单”（Purchase Order）**数据库记录。
创建一个连接“**采购订单”（Purchase Order）**和“**创建采购订单记录”（Create Purchase Order Record）**出口的流程。

1. “**申请单标识号”（Requisition Id）**应该设置为通过“**申请单标识号”（Requisition Id）**触发器接收到的“**申请单”（Requisition Id）**数据结构的“**标识号”（Id）**:
创建一个连接“**申请单标识号”（Requisition Id）**触发器和**“采购订单/申请单标识号”（Purchase Order/Requisition Id）**的流程。

2. “**供应商标识号**”（**Supplier Id**）应该设置为用户选定的供应商的“**标识号”（Id）**。在这个例子中，下拉式选单里面填充了“**供应商”(Supplier)**数据库记录，因此我们可以从选定的记录中提取“**标识号”(Id)**:
从储存器/大纲（repository/outline)中拖拽一个“**供应商”（Supplier）**数据结构。
创建一个连接“**供应商记录”（Supplier Record）**触发器和“**供应商”（Supplier）**数据结构的流程。
创建第二个流程，以连接“**供应商/标识号”（Supplier/Id）**和“**采购订单/供应商标识号”（Purchase Order/Supplier Id）**。

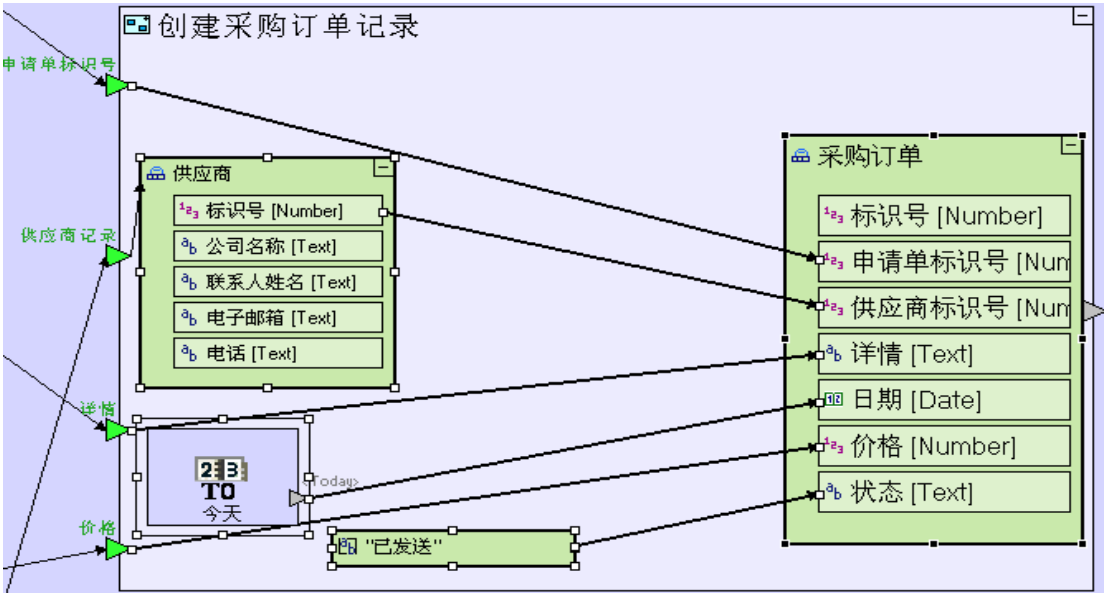
3. “**详情”(Details)**应该设置为用户在弹出窗口中输入的通过“**详情”（Details）**触发器传送的文本：
创建一个连接“**插入采购订单/详情”（Insert Purchase Order/Details）**触发器和“**采购订单/详情”（Purchase Order/Details）**的流程。

4. “**价格”（Price）**应该设置为用户在弹出窗口输入的通过“**价格”（Price）**触发器传送的数值：
创建一个连接“**插入采购订单/价格”（Insert Purchase Order/Price）**触发器和“**采购订单/价格”（Purchase Order/Price）**的流程。

5. “**日期**”（**Date**）字段应该设置为当天的日期：
添加一个“**日期/今天”（Dates/Today）**。
创建一个连接“**今天/<今天>”（Today/<Today>）**和“**采购订单/日期”（Purchase Order/Date）**的流程。

6. 采购订单的“**状态”（Status）**应该设置为已发送：
添加一个“**常数/文本”（Constants/Text）**。把它命名为 “**已发送”（*Issued*）**。
创建一个连接“**已发送”（“Issued”）**和“**采购订单/状态”（Purchase Order/ Status）**的流程。

现在“**创建采购订单记录**”（Create Purchase Order Record）过程模型外观应该与下图类似：



我们可以看到“**采购订单/标识号**”（Purchase Order/Id）字段还没有设置。这是因为“**采购订单/标识号**”应该是一个使用“**数据库/序列号**”（Database/Sequence Number）的唯一的标识号，因此应该是数据库交易的一部分，也就是说我们必须在“**处理数据库**”（Handle Database）服务中对它进行设置。

进入“**处理数据库**”（Handle Database）。

进入“**执行动作**”（Perform Action）。

给“**插入采购订单**”（Insert Purchase Order）添加一个触发器。

创建一个连接“**执行动作/输入**”（Perform Action/Input）触发器和“**插入采购订单**”（Insert Purchase Order）触发器的流程。

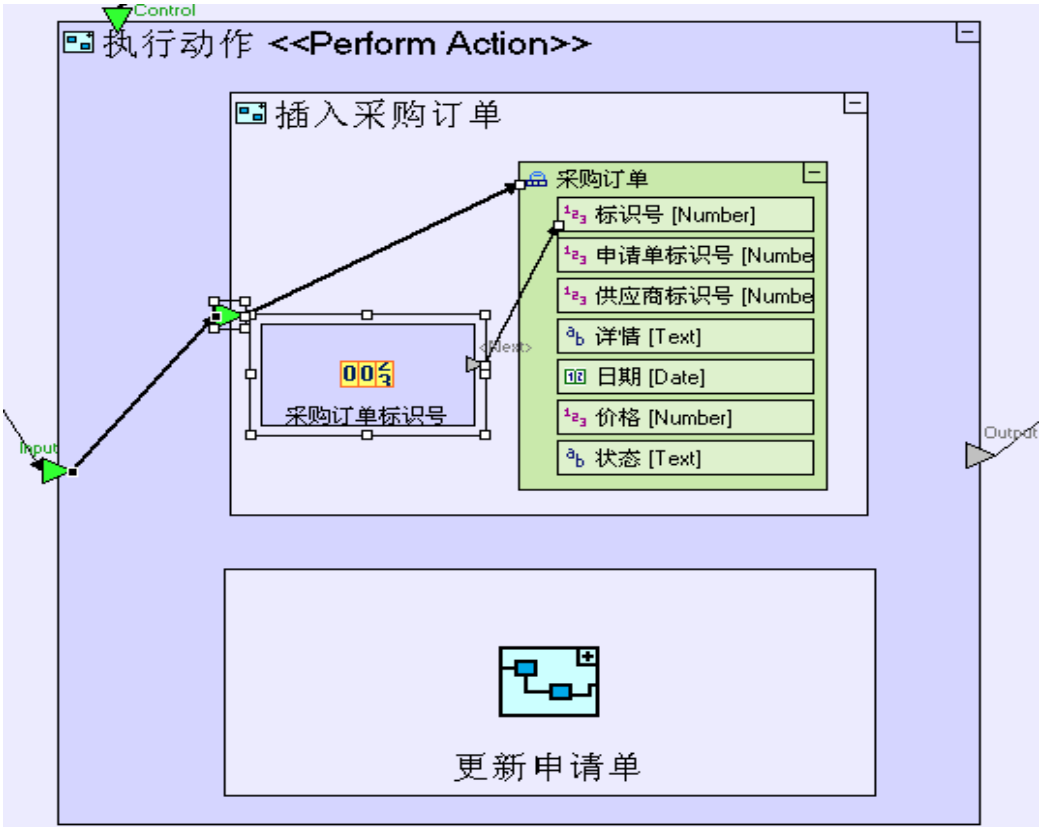
放大并进入“**插入采购订单**”（Insert Purchase Order）。

创建一个连接“**输入**”（Input）触发器和“**采购订单**”(Purchase Order）数据库记录的流程。

添加“**数据库/序列号**”（Database/Sequence Number）。把它命名为 “**采购订单标识号**”(***Purchase Order Id***)。

创建一个连接“**采购订单标识号/<下一个>**”（Purchase Order Id/<Next>）和“**采购订单/标识号**”（Purchase Order/Id）的流程。

现在“**插入采购订单**”（Insert Purchase Order）过程模型外观应与下图类似：



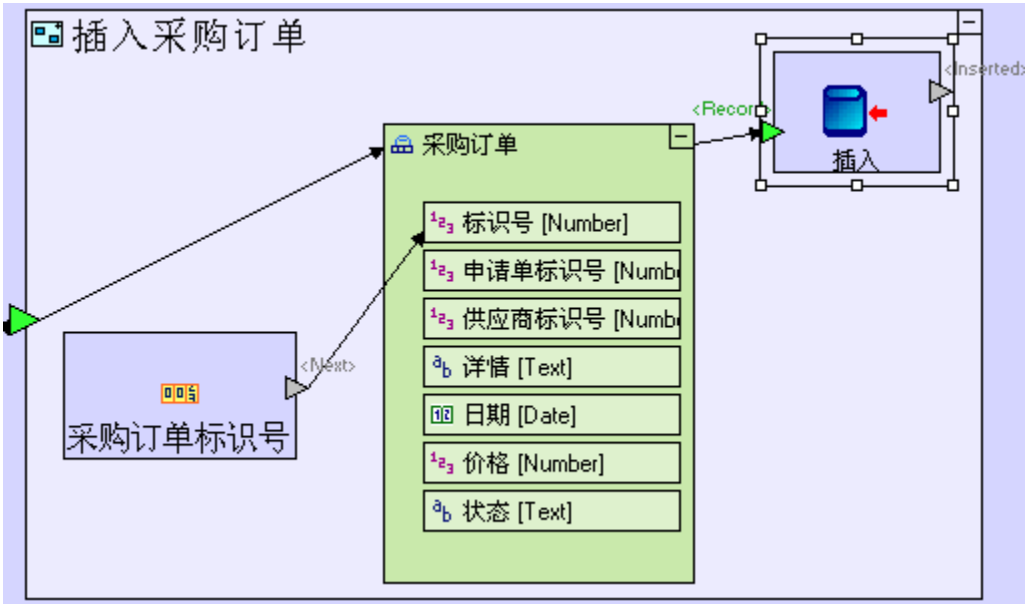
现在“**采购订单**”（Purchase Order）数据库记录已经完全填充了，我们可以把它添加到“**采购订单**”（Purchase Orders）表格中：

进入“**插入采购订单**”（Insert Purchase Order）。

添加一个“**数据库/插入**”（Database/Insert）。

创建一个连接“**采购订单**”（Purchase Order）数据结构和“**插入/<记录>**”（Insert/<Record>）的流程。

现在“**插入采购订单**”（Insert Purchase Order）过程外观应该与下图类似：



更新申请单 Update the Requisition

现在我们转到“**更新申请单**”（**Update Requisition**）过程。我们应该用一个新的状态更新申请单，以表示申请单的订单已经发送。在我们创建的“**采购订单**”（**Purchase Order**）记录中有申请单的标识号，因此：

进入“**执行动作**”（**Perform Action**）。

给“**更新申请单**”（**Update Requisition**）添加一个触发器。

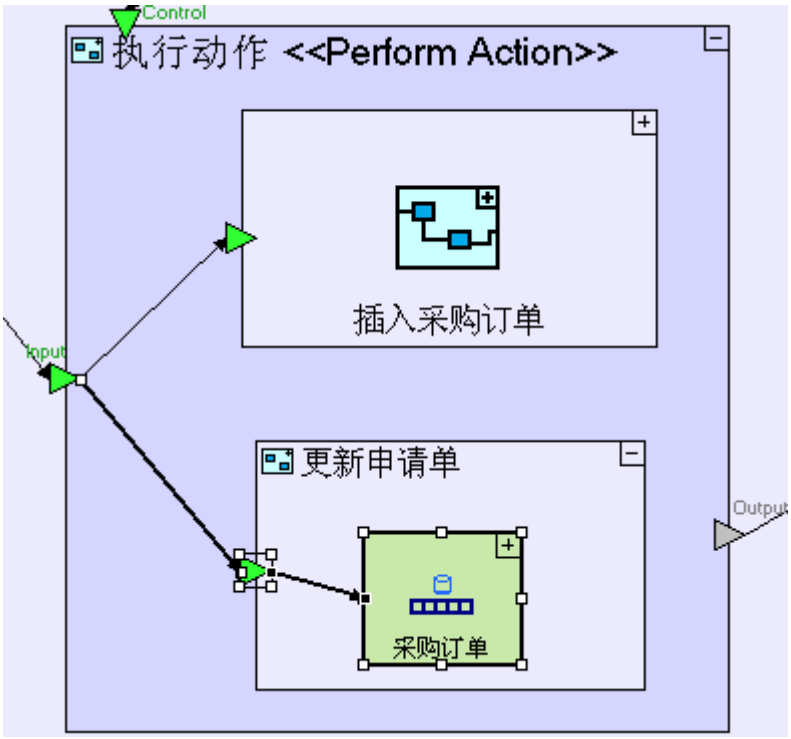
创建一个连接“**运行动作/输入**”（**Perform Action/Input**）触发器和“**更新申请单**”（**Update Requisition**）触发器的流程。

放大并进入“**更新申请单**”（**Update Requisition**）。

从储存器/大纲（repository/outline）中拖拽一个“**采购订单**”（**Purchase Order**）数据库记录。

创建一个连接“**更新申请单**”（**Update Requisition**）触发器和“**采购订单**”（**Purchase Order**）数据库记录的流程。

“**运行动作**”（**Perform Action**）过程模型应该与下图类似：



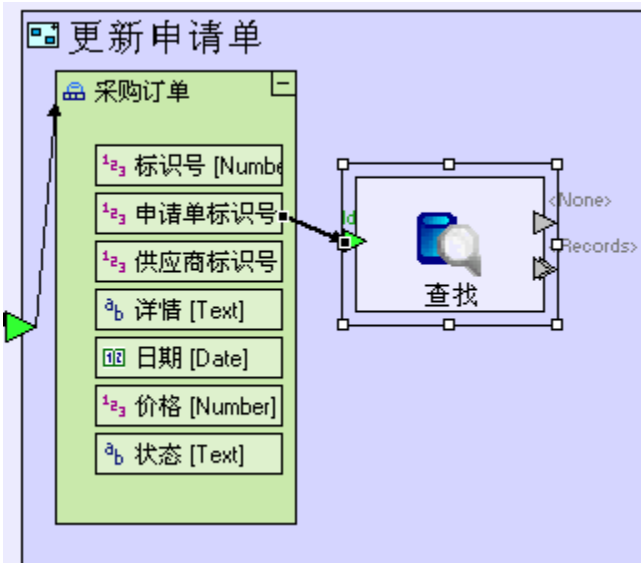
我们需要从数据库中提取申请单并更新申请单的状态：

进入“**更新申请单**”（**Update Requisition**）。

添加一个“**显示/查找**”（**Database/Find**）。添加一个新触发器。把它命名为 “*Id*”。

创建一个连接“**采购订单/申请单标识号**”（**Purchase Order/Requisition Id**）和“**查找/标识号**”（**Find/Id**）的流程。

现在“**更新申请单**”（**Update Requisition**）模型应该与下图类似：



要更新申请单状态，我们可以重新使用上个阶段我们建模的过程“**修改申请单状态**”（**Change Requisition Status**）（在“**经理**系统/**申请单批准**视图/**按钮行**/**批准申请单**按钮/**更新申请单**过程”（**Manager** system/**Requisition Approval** view/**Button Row**/**Approve Requisition** button/**Update Requisition** process）中）：

从储存器/大纲（repository/outline）中拖拽“**修改申请单状态**”（**Change Requisition Status**）。

创建一个连接“**查找/<记录>**”（**Find/<Records>**）和“**修改申请单状态/原申请单**”（**Change Requisition Status/Original Requisition**）的流程。

添加一个“**常数/文本**”（**Constant/Text**）。把它命名为 “**已发送订单**”（***Order Issued***）。

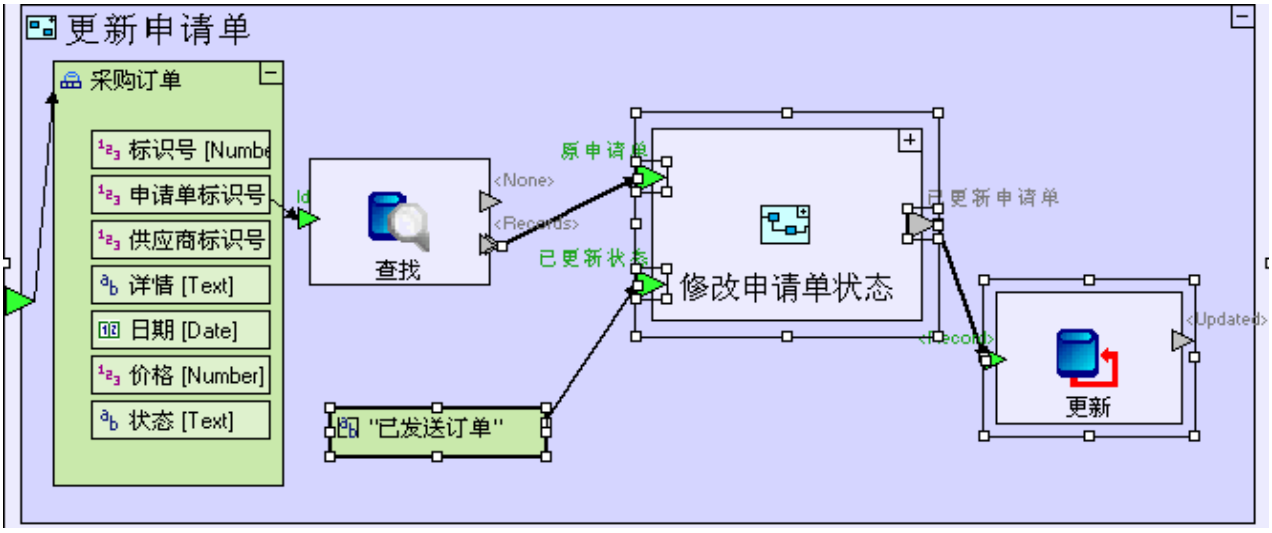
创建一个连接“**已发送订单**”（**Order Issued**）和“**修改申请单状态/已更新状态**”（**Change Requisition Status/Updated Status**）的流程。

最后，我们需要在数据库中进行更新：

添加一个“**数据库/更新**”（**Database/Update**）。

创建一个连接“**修改申请单状态/已更新申请单**”（**Change Requisition Status/Updated Requisition**）和“**更新/<记录>**”（**Update/<Record>**）的流程。

现在“**更新申请单**”（**Update Requisition**）过程应该与下图类似：



使用<已完成>（<Done>）出口（以规定运行顺序） Use a <Done> Exit (to specify the order of execution)

现在我们已经处理了数据库的问题，我们还必须处理最后一个问题—确保“**关闭窗口**”（**Close Window**）只在“**处理数据库**”（**Handle Database**）完成之后才运行：

缩小当前页面并转到“**提交订单**”（**Submit Order**）。

右击“**处理数据库**”（**Handle Database**）。从“**添加单元**”(**Add Element**)子菜单中选择“**<完成>**”（**<Done>**）出口。

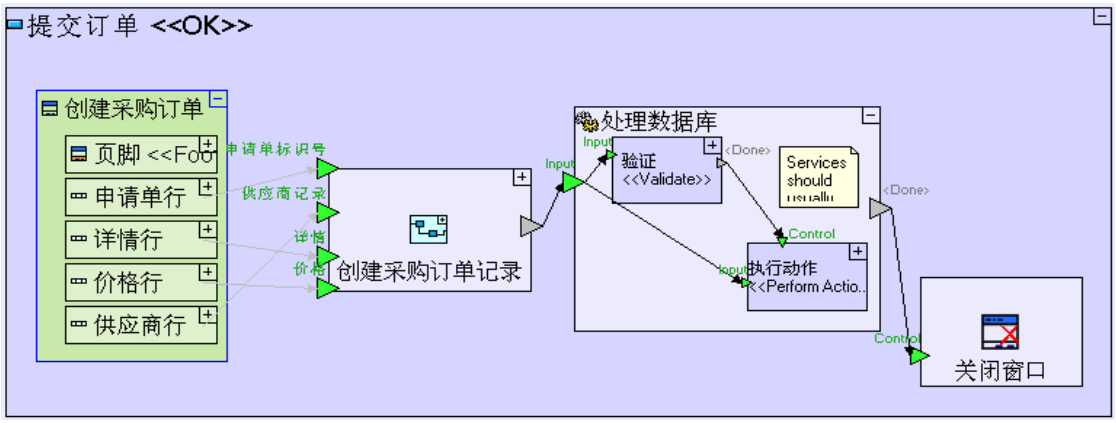
创建连接“**处理数据库/<完成>**”（**Handle Database/<Done>**）和“**关闭窗口**”（**Close Window**）触发器的流程。

扩大并进入“**处理数据库**”（**Handle Database**）。

选择连接“**运行动作/输出**”（**Perform Action/Output**）出口和“**处理数据库/输出**”（**Handle Database/Output**）出口的流程，并按“**删除键**”（**[Del]**）（或右击并选择“**删除单元**”（**Remove Element**））删除流程。

删除“**处理数据库/输出**”（**Handle Database/Output**）和“**运行/动作**”（**Perform/Action**）出口。

“**提交订单**”（**Submit Order**）过程应该与下图类似：



命名一个过程的出口（“**<完成>**”（**<Done>**））确保过程运行完成以后，即使（“**处理数据库**”（**Handle Database**）中）没有任何流程激活这个出口，出口也会被激活。如果出口被命名为除了“**<完成>**”（**<Done>**）以外的名字，它就不能被激活。

> 注意：
> 1. 创建“**<完成>**”（**<Done>**）的另一个可选方法是添加一个常规出口（regular exit）并把它命名为“**<完成>**”（**<Done>**）。然后您需要通过选择“**数据类型/无**”（**Data Types/Nothing**）并把它拖到出口上，从而将出口的类型设置为“**无**”（**Nothing**）。您重新命名出口（类型为“**模型名称**”（**Model Name**））时，会看到槽（slot）的类型。
> 2. 我们删除的“**输出**”（**Output**）出口和流程是原“**服务**”（**Service**）复合模板中的一部分，而“**处理数据库**”（**Handle Database**）是建立在原“**服务**”（**Service**）复合模板的基础上的，但由于我们除了从服务器中输出“**<完成>**”（**<Done>**）指示之外，不需要从服务器输出任何信息，因此我们可以删除这个模型。

保存您的工作成果并在浏览器中启动应用程序。

点击“**采购员**”（**Purchaser**）透视图标签（tab）。

点击“**发送采购订单**”（**Issue Purchase Orders**）视图标签（tab）。

从“**申请单清单**”（**Requisition List**）中选择一行（row）。

点击“**订购**”（**Order**）按钮。

选择一个“**供应商**”（**Supplier**）并输入“**价格**”（**Price**）。

点击“**提交订单**”（**Submit Order**）按钮。

在下一个阶段，我们将为与每个申请单相关的采购订单的显示建立模型。

完成阶段 13 Completing Stage 13

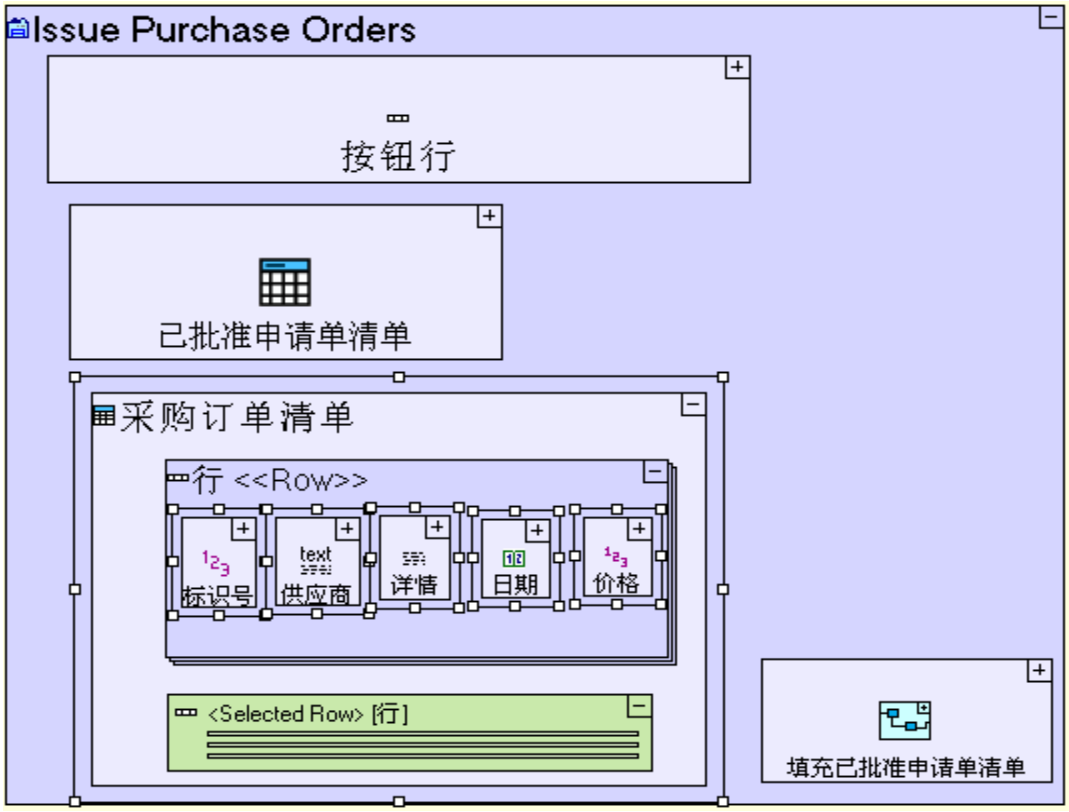
现在我们输入项目样本“**Tutorial 13-14**”，把它作为本教程下一个阶段的基础。

> 想再了解如何输入一个项目样本，请参见**阶段 2** 最后的“**输入项目样本**”部分。

这个项目样本具备到目前为止我们建立的模型的所有功能。

这个项目样本也包括以下一些其他的功能：

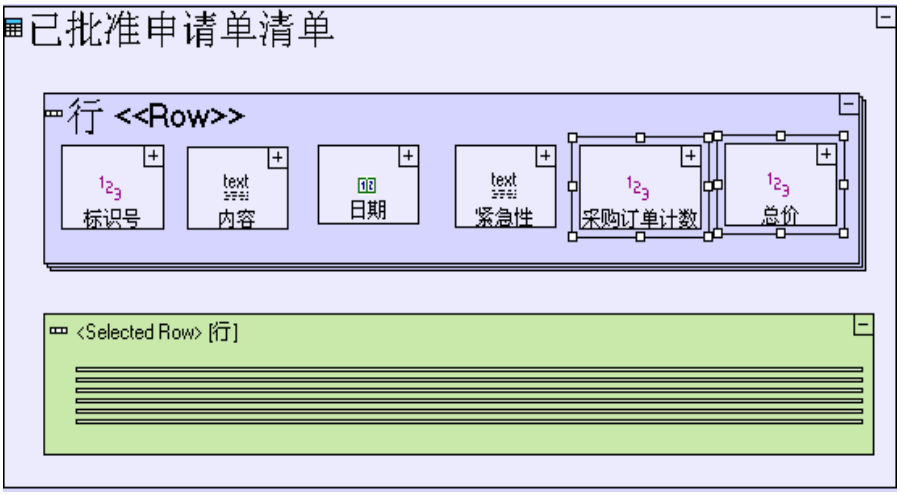
1. 添加第二个表格，在“**发送采购订单**”(Issue Purchase Orders)视图中显示采购订单。



如何建模	位置
添加“**显示/表格**”(Display/Table)。把它命名为 “***采购订单清单***” (*Purchase Order List*)。	“**发送采购订单**”(Issue Purchase Orders)视图

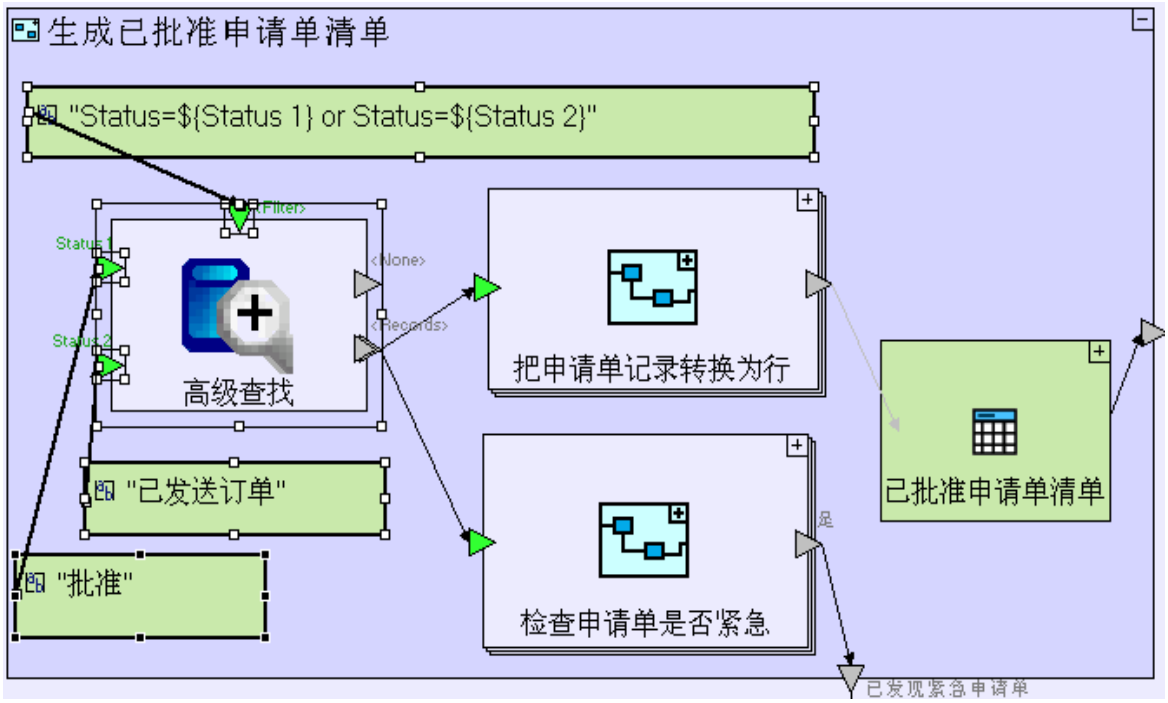
添加一个“**显示/数字显示**”（**Display/Number Display**）。把它命名为 “**标识号**”（***Id***）。 添加一个“**显示/文本显示**”（**Display/Text Display**）。把它命名为 “**供应商**”（***Supplier***）。 添加一个“**显示/文本显示**”（**Display/Text Display**）。把它命名为 “**详情**”（***Details***）。 添加一个“**显示/日期显示**”（**Display/Date Display**）。把它命名为 “**日期**”（***Date***）。 添加一个“**显示/数字显示**”（**Display/Number Display**）。把它命名为 “**价格**”（***Price***）。	“**采购订单清单**”表格/“**行**”（**Purchase Order List** table/**Row**）

2. 给“**已批准申请单清单**”（**Approved Requisitions List**）表格添加一些列（columns）。



如何建模	位置
添加一个“**显示/数字显示**”（**Display/Number Display**）。把它命名为 “***采购订单计数”(PO Count)***。 添加一个“**显示/数字显示**”（**Display/Number Display**）。把它命名为 “***总价”(Total Price)***。	“**已批准申请单清单**”表格/“**行**”（**Approved Requisitions List** table/**Row**）

3.更新“**生成已批准申请单清单**”（**Generate Approved Requisitions List**），使其包含状态为“**已发送订单**”（**Order Issued**）的申请单。



如何建模	位置
用“**数据库/高级查找**”（Database/Advanced Find）代替“**查找**”（Find）。 重新连接从“**高级查找/<记录>**”（Advanced Find/<Records>）出口流出的流程。 添加 2 个触发器；把它们命名为“***Status 1***”和“***Status 2***”。 连接“**批准**”（Approved）和“Status 1”。 拖拽“**已发送订单**”（Order Issued）。连接“**已发送订单**”（Order Issued）和“Status 2”。 添加一个“**常数/文本**”（Constants/Text）。把它命名为“***Status=${Status 1} or Status=${Status 2}***”。把它和“**<过滤器>**”（<Filter>）连接起来。	“**生成已批准申请单清单**”（Generate Approved Requisitions List）过程

现在您可以开始学习**阶段14**，在**阶段14**中，我们将利用“**采购订单清单**”（Purchase Order List）表格来显示连接到当前已选定申请单的采购订单，并为每个申请单计算采购订单的总金额。

实例 See It Live

击此处在另一个窗口里面打开实际项目。

阶段 14 -- 显示多个（连接）表格　Stage 14 – Displaying Multiple (Linked) Tables

阶段目标 Stage Goals

覆盖的 Tersus 概念 Tersus Concepts Covered

完成本阶段后，您将会熟悉以下概念：
建模概念：<点击>(<On Click>)过程
建模技术：显示一对多（one-to-many）关系,汇总（summing）
有用的过程模板：计数（Count），汇总（Sum）
有用的显示模板：刷新（Refresh）

建立应用程序功能模型 Application Functinality Modeled

我们将继续为“**发送采购订单**”（**Issue Purchase Order**）视图建模。

上个阶段我们为采购订单和申请单之间的关系建模。具体地说，所建立模型是一种一对多的关系，多个采购订单可能和同一个申请单相关，每个订单都含有一些申请的物品。

您载入的项目样本包括以下添加到“**发送采购订单**”（**Issue Purchase Order**）视图中的内容：

- 一份出现在申请单清单下的采购订单（PO）清单。
- 添加到申请单清单中的新建的 2 个列（columns）：“**采购订单计数**”和“**总价**”（**PO Count** & **Total Price**）。

添加到视图的这些内容还没有显示任何数据。我们需要为以下过程建模：

- 每次从申请单清单中选定一个申请单时，采购订单清单（PO list）都会以与选定申请单连接的采购订单进行更新。
- 为每个申请单计算“**采购订单计数**”（**PO Count**）和“**总价**”(**Total Price**)，计算连接到每个申请单的采购订单（POs）的数量并计算这些采购订单的总价。

本阶段建模在上个阶段最后输入的项目“**Tutorial 13-14**”中进行。

<点击>过程（点击某一行时运行一个过程）<On Click> Process (to execute a process when a row is clicked)

我们的第一个任务是创建一个过程；点击申请单清单中的一个申请单时，这个过程就会更新采购订单清单。

Tersus 通过一个带有保留名称“**<点击>**”(**<On Click>**)的过程做到这一点：

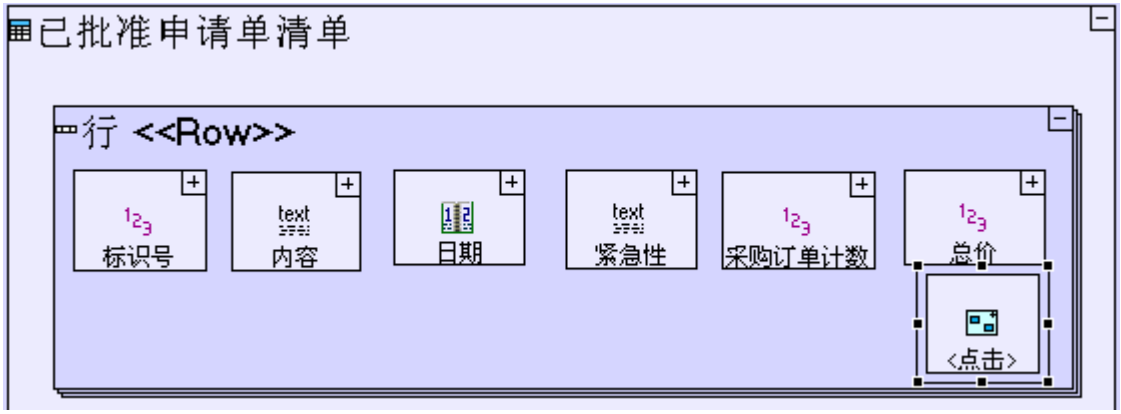
进入“**已批准申请单清单/行**”（**Approved Requisitions List/Row**）。

添加一个“**基本/动作**”（**Basic/Action**）。把它命名为 “***<点击>”（<On Click>）***。

> 您也可以通过打开（右击）“**行**”（**Row**）单元的情景菜单（context menu）并从“**添加单元**”(**Add Element**)子菜单中选择**<On Click>**，以添加“**<点击>**”（**<On Click>**）过程。

> 这是表格显示不能基于“**简单表格**”（**Simple Table**）模板的情况的另一个例子—因为“**<点击>**”(**<On Click>**)过程是“**行**”（**Row**）显示单元的一部分，而一个“**简单表格**”（**Simple Table**）只包含一个数据结构。

现在“**已批准申请单清单**”（**Approved Requisition List**）模型外观应该与下图类似：



“<点击>”(<On Click>)过程应该提取所有与点击的申请单连接的采购订单，因此这个过程首先需要获得这一行里显示的申请单的标识号。可以通过上级索引找到所点击的行的申请单标识号：

放大并进入“**<点击>**”(**<On Click>**)。

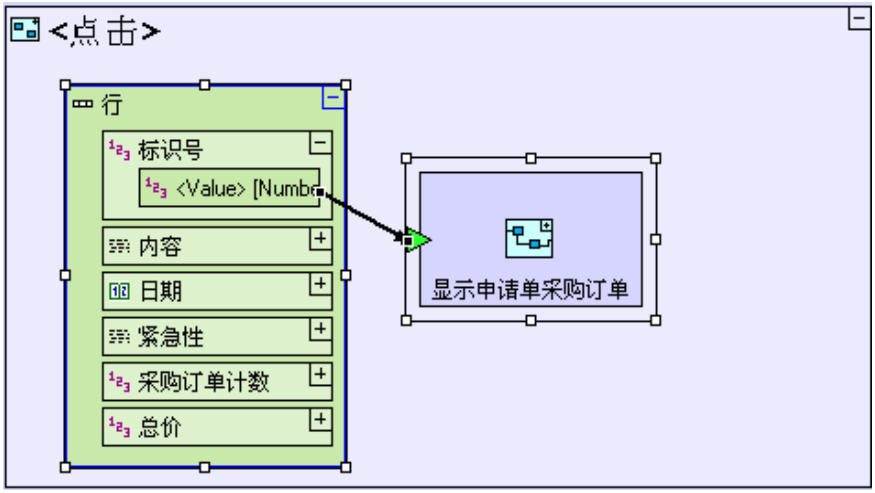
添加“**行**”（**Row**）的上级索引。

现在我们要创建一个接收申请单的“**标识号**”（**Id**）的子过程并以匹配的采购订单填充“**采购订单清单**”（**Purchase Order List**）：

添加一个“**基本/动作**”（**Basic/Action**）。把它命名为 “**显示申请单采购订单**”(***Display Purchase Orders for Requisition***)。

给它添加一个触发器，创建一个连接“**申请单行/标识号/<数值>**”（**Requisition Row/Id/<Value>**）和这个触发器的流程。

“**<点击>**”（**<On Click>**）模型外观应该与下图类似：



现在我们提取所有“**申请单标识号**”（**Requisition Id**）字段和当前申请单的“**标识号**”（**Id**）相匹配的采购订单：

放大并进入“**显示申请单采购订单**”（**Display Purchase Orders for Requisition**）。

添加一个“**数据库/查找**”（**Database/Find**）。

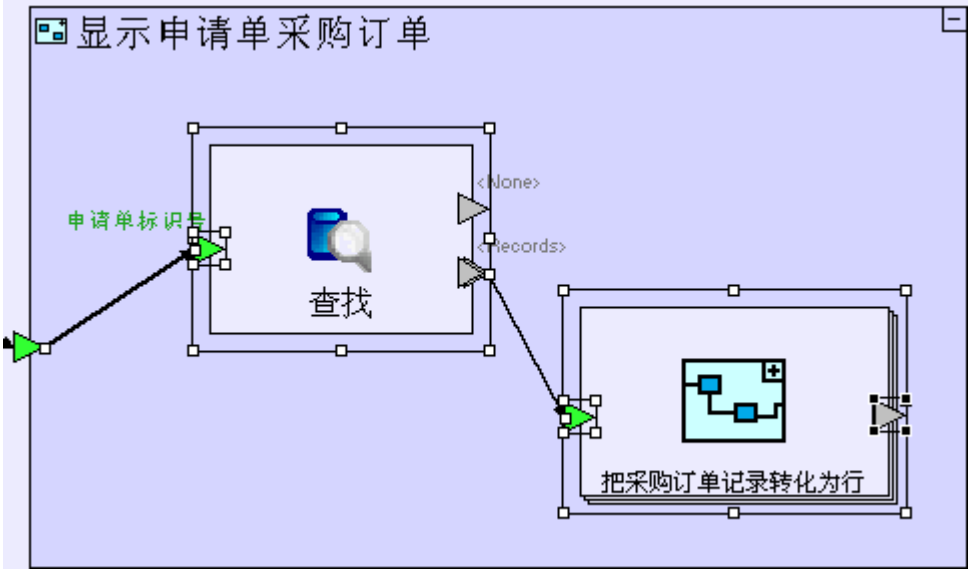
给它添加一个触发器。把它命名为“***申请单标识号***”（***Requisition Id***）。

创建一个连接“***申请单标识号***”（***Requisition Id***）和“**显示申请单采购订单**”（**Display Purchase Orders for Requisition**）触发器的流程。

提取的采购订单必须转化为“**采购订单清单**”（**Purchase Order List**）表格中的行：

添加一个“**基本/动作**”（**Basic/Action**）。把它命名为“**把采购订单记录转化为行**”（***Convert Purchase Order Record to Row***）并把它标记为重复性（repetitive）的。给它添加一个触发器和一个出口。创建一个连接“**查找/<记录>**”（**Find/<Records>**）和该触发器的流程。

“显示申请单采购订单”（Display Purchase Orders for Requisition）模型外观应该与下图类似：



放大并进入“**把采购订单记录转化为行**”（**Convert Purchase Order Record to Row**）。

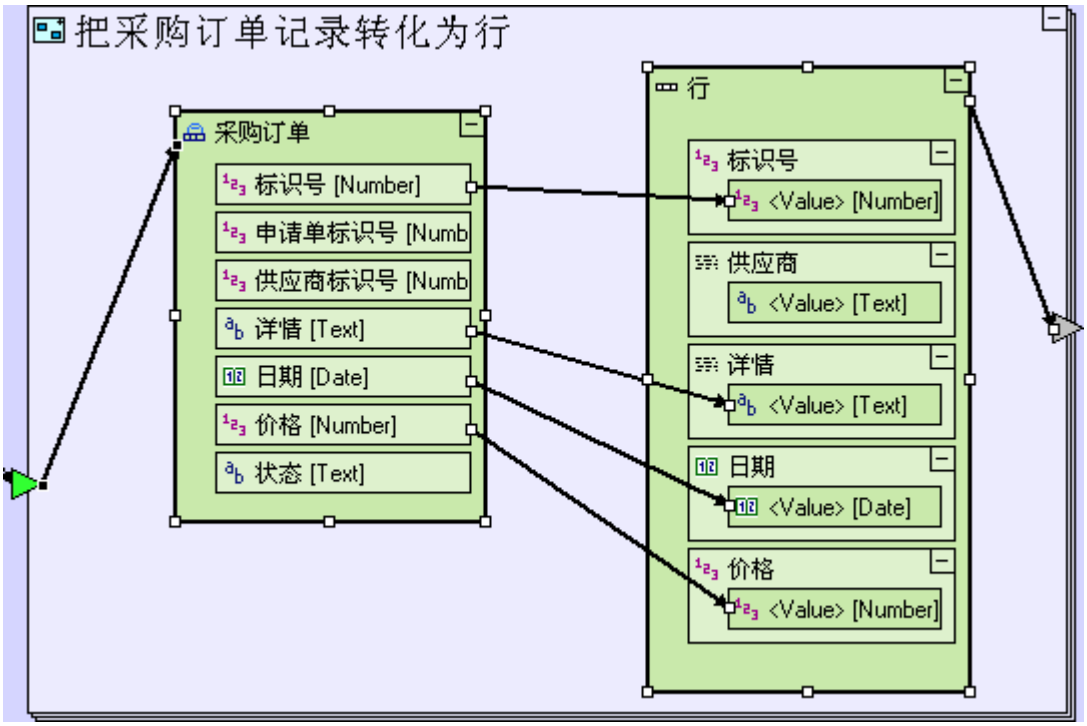
从储存器/大纲（repository/outline）中拖拽一个“**采购订单**”（**Purchase Order**）数据结构（可从“**订购**”按钮/“**创建采购订单**”弹出窗口/“**提交订单**”按钮”（**Order** button/**Create Purchase Order** popup/ **Submit Order** button）中找到）。添加一个连接“**把采购订单记录转化为行**”（**Convert Purchase Order Record to Row**）触发器和以上数据结构的流程。

从储存器/大纲（repository/outline）中拖拽“**采购订单清单/行**”（**Purchase Order List/Row**）（以创建一个显示数据单元）。添加一个连接“**采购订单清单/行**”（**Purchase Order List/Row**）和“**转化采购订单记录到行**”（**Convert Purchase Order Record to Row**）出口的流程。

创建以下流程，将采购订单数据库记录的字段映射（map）到显示表格行的字段里面：

来源（Source）	目标（Target）
“**采购订单/标识号**”（**Purchase Order /Id**）	“**行/标识号/<数值>**”（**Row/Id/<Value>**）
“**采购订单/日期**”（**Purchase Order /Date**）	“**行/日期/<数值>**”（**Row/Date/<Value>**）
“**采购订单/详情**”（**Purchase Order /Details**）	“**行/详情/<数值>**”（**Row/Details/<Value>**）
“**采购订单/价格**”（**Purchase Order /Price**）	“**行/价格/<数值>**”（**Row/Price/<Value>**）

现在“**把采购订单记录转化为行**”（Convert Purchase Order Record to Row）模型外观应该与下图类似：



“**行**”（Row）中还有一个我们尚未填充的字段：“**供应商**”（Supplier）字段。“**采购订单**”（Purchase Order）数据结构中有一个“**供应商标识号**”（Supplier Id）字段，这个“**供应商标识号**”（Supplier Id）字段包含供应商的数字标识号，但没有提供足够的信息。因而，我们从数据库中的“**供应商**”（Supplier）表格中显示供应商的公司名称：

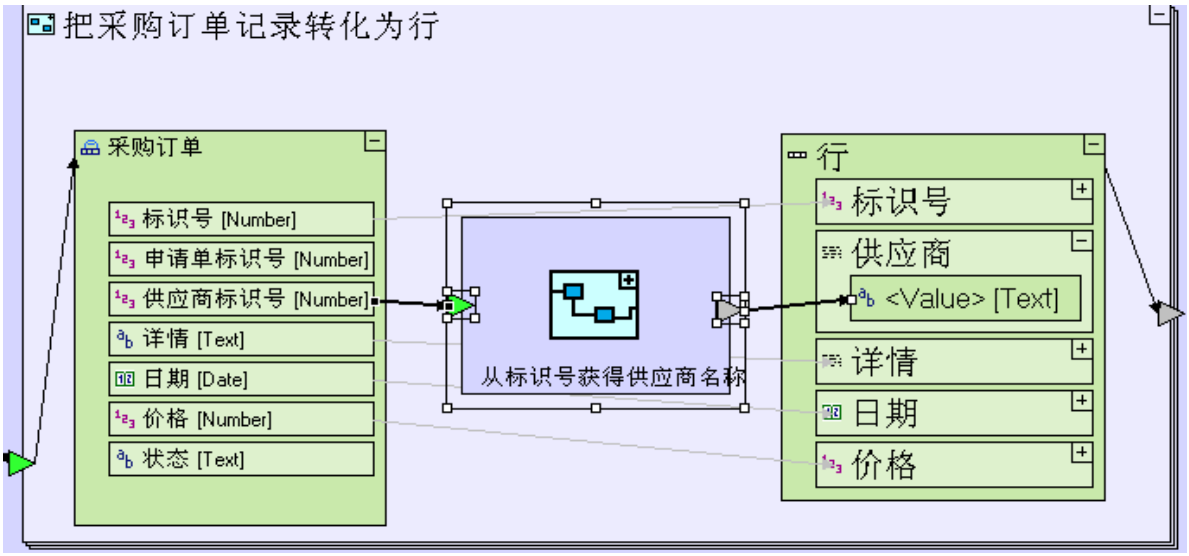
放大并进入“**把采购订单记录转化为行**”（Convert Purchase Order Record to Row）。

添加一个“**基本/动作**”（Basic/Action）。把它放在“**采购订单**”（Purchase Order）数据库记录和“**行**”（Row）显示数据单元之间。把它命名为 “**从标识号获得供应商名称**”（*Get Supplier Name from Id*）。给它添加一个触发器和一个出口。

创建一个连接“**采购订单/供应商标识号**”（Purchase Order/Supplier Id）和“**从标识号获得供应商名称**”（Get Supplier Name from Id）触发器的流程。

创建一个连接“**从标识号获得供应商名称**”（Get Supplier Name from Id）出口和“**行/供应商/<数值>**”（Row/Supplier/<Value>）的流程。

现在“**把采购订单记录转化为行**”（Convert Purchase Order Record to Row）模型外观应该与下图类似：



放大并进入“**从标识号获得供应商名称**”（Get Supplier Name from Id）。

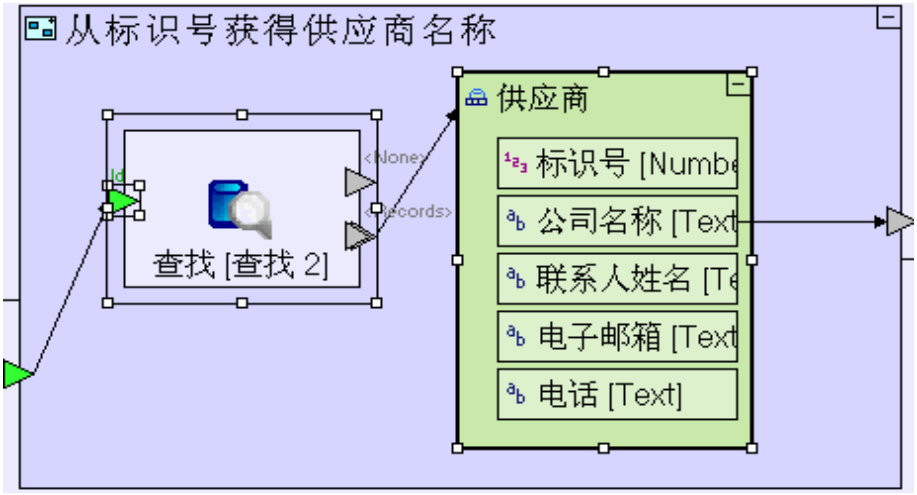
添加一个“**数据库/查找**”（Database/Find）。给它添加一个触发器并把它命名为“***Id***”。创建一个连接“**从标识号获得供应商名称**”（Get Supplier Name from Id）触发器和“**查找/标识号**”（Find/Id）的流程。

把一个“**供应商**”（Supplier）数据库记录（我们能够找到这个记录的其中一个位置是“**管理供应商**”视图/“**供应商清单**”表格”（Manage Suppliers view/Supplier List table））拖放到“**从标识号获得供应商名称**”（Get Supplier Name from Id）。创建一个连接“**查找/<记录>**”（Find/<Records>）和“**供应商**”（Supplier）的流程。

> 注意：尽管“**<记录>**”（<Records>）出口是一个重复性的出口，但它与一个非重复性的目标数据结构连接。但这不是个问题，因为我们使用“**标识号**”（Id）作为“**查找**”（Find）过程的提取规则，而“**标识号**”（Id）是唯一的，因此过程不会返回一个以上记录。

创建一个连接“**供应商/公司名称**”(Supplier/Company Name)和“**从标识号获得供应商名称**”（Get Supplier Name from Id）出口的流程。

“**从标识号获得供应商名称**”（**Get Supplier Name from Id**）模型外观应该与下图类似：



您可能已经注意到我们添加的“**查找**”（**Find**）单元和平常看起来略有不同，它显示了2个名称：“**查找[查找2]**”（**Find[Find 2]**）。

“**查找**”（**Find**），是“**单元名称**”（**Element Name**）。而显示在方括号中的“**[查找2]**”（**[Find 2]**）是“**模型名称**”（**Model Name**）。正如之前所解释的（见**阶段2**），单元既有一个“**模型名称**”（**Model Name**）又有一个“**单元名称**”（**Element Name**），这两个名称都是自动定义的—但由于它们一般是相同的，因此一般显示为一个名称。

给模型添加一个单元时，如果用户所提供的名称已经被占用，Tersus工作室（Tersus Studio）会通过添加一个索引号码（index number），自动修改这两个名称中的一个以保持唯一性（uniqueness）。

在当前的情况中，“**从标识号获得供应商名称**”（**Get Supplier Name from Id**）中的“**查找**”（**Find**）单元（在缺省情况下）会被保存在“**已批准申请单清单**”（**Approved Requisitions List**）数据包（package）中（我们可通过右击模型编辑器中的“**查找**”（**Find**）并选择“**显示在储存器管理器中**”（**Show in Repository Explorer**）来检查这一点）。这个数据包已经包含了一个名称为“**查找**”（**Find**）的模型（也就是我们之前在“**显示申请单采购订单**”（**Display Purchase Orders for Requisition**）中所创建的“**查找**”（**Find**）模型），因此后面的“**查找**”（**Find**）就必须有一个不同的“**模型名称**”（**Model Name**）。

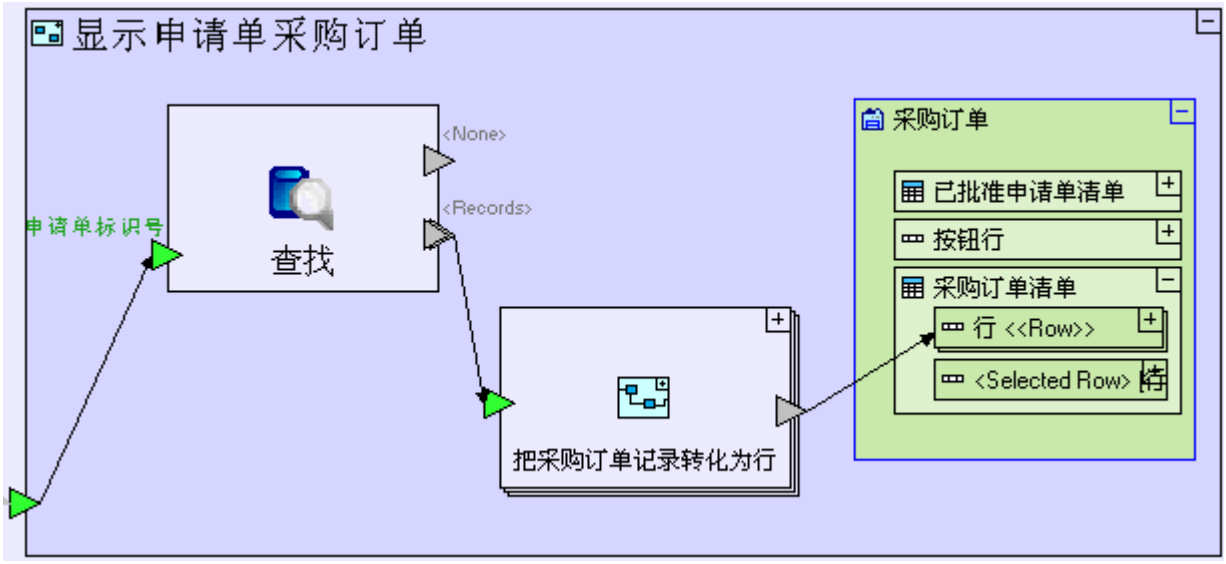
要完成“**显示申请单采购订单**”（**Display Purchase Orders for Requisition**建模，我们需要将创建的行（rows）发送到显示（display）：

缩小当前页面并转到“**显示申请单采购订单**”（**Display Purchase Orders for Requisition**）。

给“**发送采购订单**”（**Issue Purchase Orders**）视图添加一个上级索引。

创建一个连接“**把采购订单记录转化为行**”（**Convert Purchase Order Record to Row**）出口和“**发送采购订单/采购订单清单/行**”（**Issue Purchase Orders/Purchase Order List/Row**）的流程。

“显示申请单采购订单”（Display Purchase Orders for Requisition）模型外观应该与下图类似：



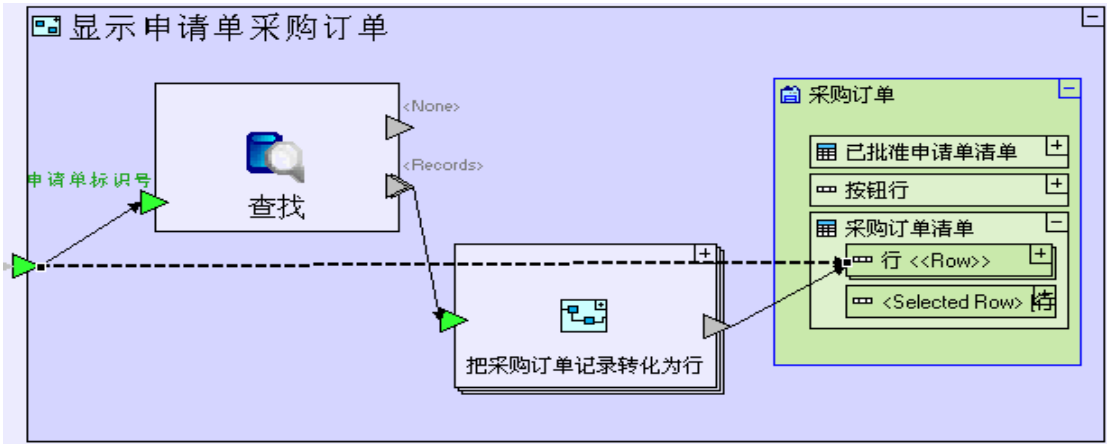
请注意：我们现在使用一种不同的方法以行填充表格（与前面的阶段我们为各种**“填充…”（Populate…）**过程建模不同）：我们不是把行（rows）发送给中转的（intermediate）**“采购订单清单”（Purchase Orders List）**数据单元，再由**“采购订单清单”（Purchase Orders List）**数据单元把行（rows）发送到显示上级索引，而是把行（rows）直接发送给显示。
这个模型比之前建模的**“填充…”（Populate…）**过程简单，但是有一个性能

（behavioral）区别需要注意：因为显示现在是在行级别（row level）而不是在表格显示级别（table display level）进行更新，重复性行单元会把行（rows）累积起来，这意味着如果在运行过程之前表格中的行就显示出来，那么过程创建的新的行就会添加到现有的行中，而不是替代这些行。

解决方法是确保在更新显示之前，表格已被清空：

> 创建一个连接**“显示申请单采购订单”（Display Purchase Orders for Requisition）**触发器和**“发送采购订单/采购订单清单/行”（Issue Purchase Orders/Purchase Order List/Row）**的**“删除”（Remove）**流程。

现在模型外观应该与下图类似：



> 您可能会想，我们本来可以从“**查找/<无>**”（**Find/<None>**）出口设置删除流程（remove flow）的来源（之前我们为各种“**生成…**”（**Generate…**）模型建模就是那样做的），但在这里这样做却是错误的，因为它只会在“**查找**”（**Find**）没有找到任何采购订单的情况下才会清空表格；如果“**查找**”（**Find**）找到记录，那么就不能清空表格，这样表格内容就会出现重复。
> 用我们现在的方法设置删除来源（remove source）可确保无论“**查找**”（**Find**）结果如何，都会清空表格；实际上在运行“**查找**”（**Find**）之前，表格就已被清空。

保存您的工作成果并在浏览器中查看应用程序。

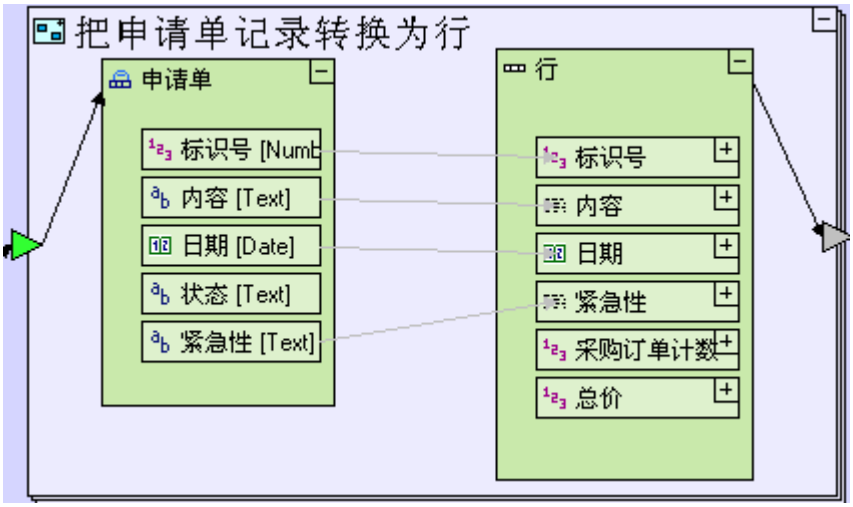
如果您点击一个已发送采购订单（issued purchase orders）的申请单，申请单应出现在较下的表格中（方式与下图类似）：

添加一个过程（以计算每个申请单的采购订单总金额）Add a process (to calculate PO aggregates for each requisition)

“已批准申请单清单”（Approved Requisitions List）有两列：**“采购订单计数”（PO Count）**和**“总价”（Total Price）**，这两列目前都是空的。我们现在要计算每个申请单的总金额。申请单清单中的所有申请单的总金额很快可以计算出来：

进入**“发送采购订单/填充已批准申请单清单/生成已批准申请单清单/把申请单记录转换为行”（Issue Purchase Orders/Populate Approved Requisitions List/Generate Approved Requisitions List/Convert Requisition Record to Row）**。

现在**“把申请单记录转换为行”（Convert Requisition Record to Row）**模型外观应该与下图类似：



现在我们给它添加一个计算某个申请单总金额的子过程：

添加一个**“基本/动作”（Basic/Action）**模板。把它命名为**“*计算申请单采购订单总金额“ (Calculate PO Aggregates for Requisition*)**。

给过程添加一个触发器。把它命名为 **“申请单标识号”（*Requisition Id*）**。创建一个连接它**“*Requisition Id*”**和**“申请单/标识号”(Requisition/Id)**的流程。

给过程添加一个出口。把它命名为 **“采购订单计数”（*PO Count*）**。创建一个连接**“*PO Count*”**和**“申请单行/采购订单计数/<数值>”（Requisition Row/PO Count/<Value>）**的流程。

给过程添加第二个出口。把它命名为 **“总价”（*Total Price*）**。创建一个连接它和**“申请单行/总价/<数值>”（Requisition Row/Total Price/<Value>）**的流程。

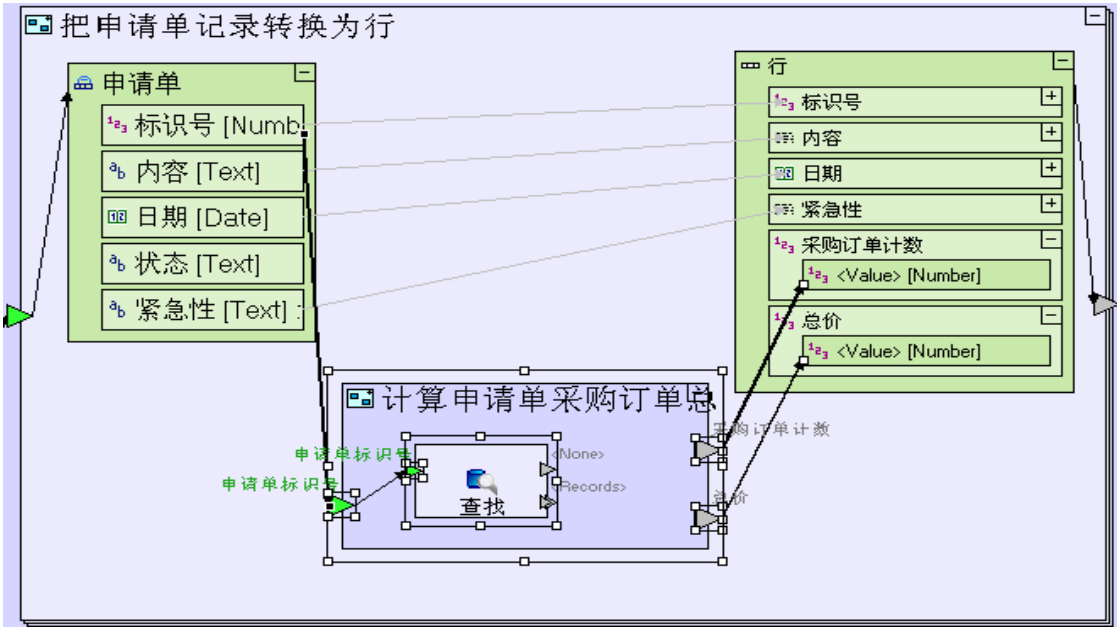
过程应该提取给定申请单的所有采购订单：

放大并进入**“计算申请单采购订单总金额”（Calculate PO Aggregates for Requisition）**。

添加一个**“数据库/查找”（Database/Find）**模板。

添加一个触发器，把它命名为**“申请单标识号”（*Requisition Id*）**。创建一个连接**“*Requisition Id*”**和**“计算申请单采购订单总金额/申请单标识号”（Calculate PO Aggregates for Requisition/Requisition Id）**触发器的流程。

“把申请单记录转换为行”（Convert Requisition Record to Row）模型现在外观应该与下图类似：



实际上我们对“**查找**”（**Find**）过程的两个单独结果很有兴趣：

1. 计算已返回“**采购订单**”（**Purchase Order**）记录的数目 - 这是“**采购订单计数**”（**PO Count**）。
2. 汇总每个已返回“**采购订单**”（**Purchase Order**）记录的“**价格**”（**Price**）- 这是“**总价**”（**Total Price**）。

使用计数模板(以计算“查找”命令找到的记录数量) Use a count Template to count the number of records found by Find)

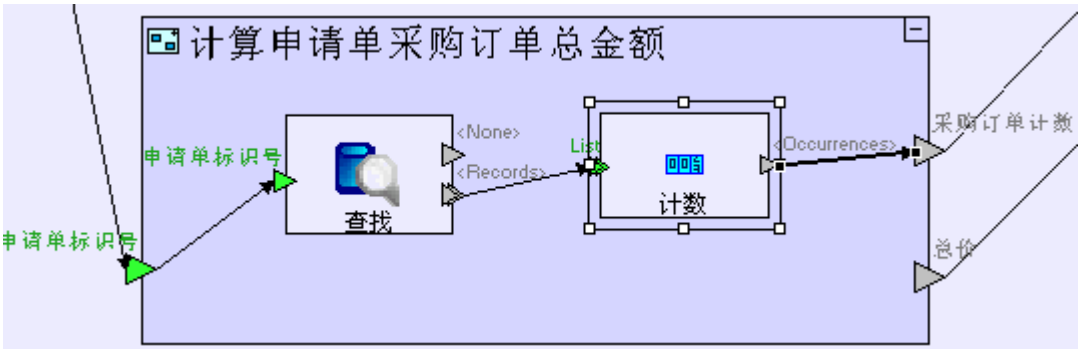
要计算“**查找**”（**Find**）所返回记录的数目，就得使用“**计数**”（**Count**）模板；它会对发送给它的触发器（或多个触发器）的对象（objects）的数目进行计算：

添加一个“**收集/计数**”（**Collections/Count**）（  ）。删除未命名的非重复性触发器，保留重复性“**清单**”（**List**）触发器。

创建一个连接“**查找/<记录>**”（**Find/<Records>**）和“**计数/清单**”（**Count/List**）的流程。

创建一个连接“**计数/<发生>**”（**Count/<Occurrences>**）和“**计算申请单的采购订单总金额/采购订单计数**”（**Calculate PO Aggregates for Requisition/PO Count**）出口的流程。

现在“**计算申请单的采购订单总金额**”（**Calculate PO Aggregates for Requisition**）模型外观应该与下图类似：



> “**计数**”（**Count**）所做的就是这些；它计算发送给它的所有对象（objects）的数量，当不再发送对象（objects）时，它就会输出对象总数。
> 注意：“**收集/计数**”（**Collections/Count**）的图标与“**数据库/序列号**”（**Database/Sequence Number**）的图标类似 – 请确保不会把这二者混淆起来。

使用汇总模板(以计算申请单总价) Use a Sum Template (to calculate the total price of a requisition)

要计算申请单总价，我们首先需要提取每个采购订单的价格：

从储存器/大纲（repository/outline）中拖拽一个“**采购订单**”（**Purchase Order**）数据结构。把它设置为重复性的（repetitive）。

创建一个连接“**采购订单**”（**Purchase Order**）数据结构和“**查找/<记录>**”（**Find/<Records>**）的流程。

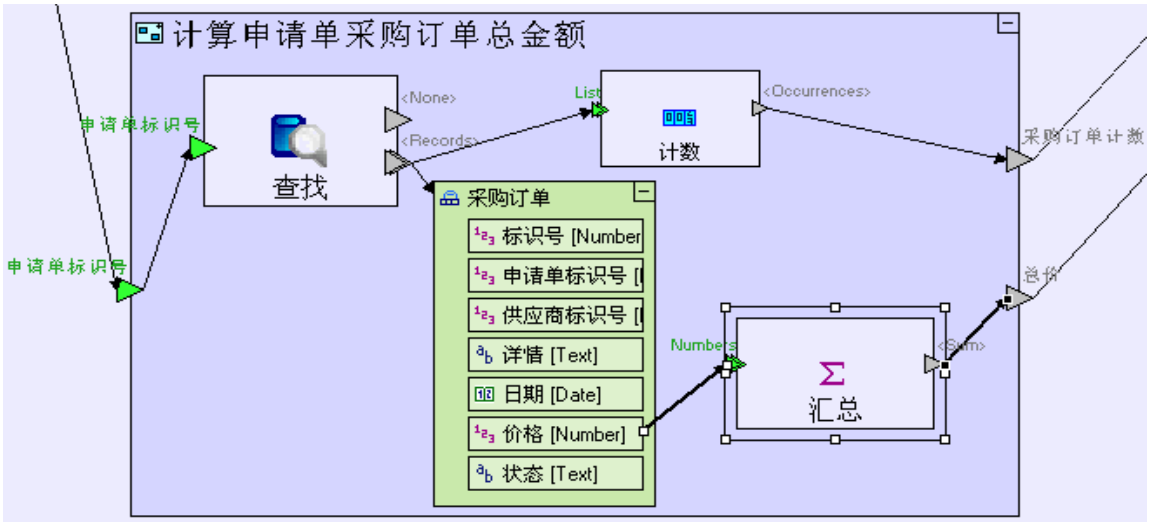
我们使用“**汇总**”（**Sum**）模板计算申请单的总价。

添加一个“**数学/汇总**”（**Math/Sum**）（  ）。

创建一个连接“**采购订单/价格**”（**Purchase Order/Price**）和“**汇总/数目**”（**Sum/Numbers**）的流程。

创建一个连接“**汇总/<汇总>**”（**Sum/<Sum>**）和“**计算申请单的采购订单总金额/总价**”（**Calculate PO Aggregates for Requisition/Total Price**）的流程。

现在“**计算申请单的采购订单总金额**”（**Calculate PO Aggregates for Requisition**）模型外观应该与下图类似：



保存您的工作成果并在浏览器中查看应用程序，应用程序应该会显示已发送订单的申请单的总金额（外观与下图类似）：

使用刷新模板(以刷新显示的数据) Use a Refresh template (to refresh data in your display)

打开“**发送采购订单**”（Issue Purchase Orders）视图并且“**已批准申请单清单**”（Approved Requisitions List）表格已填充时，就会正确显示总金额。

但发送新订单时，不会更新显示（包括总金额）。

在之前的阶段里面，我们已经重复使用“**填充…**”（**Populate...**）过程来更新表格，尽管我们在此也可以进行同样的操作，但这样做并不足够，因为这样操作不会清空表格（尽管这个可以解决）。还有“**采购订单清单**”（**Purchase Order List**）表格的问题；由于没有选定任何申请单，“**采购订单清单**”（**Purchase Order List**）表格应该被清空。

另一个可以采取的策略就是强迫视图完全刷新，操作步骤如下：

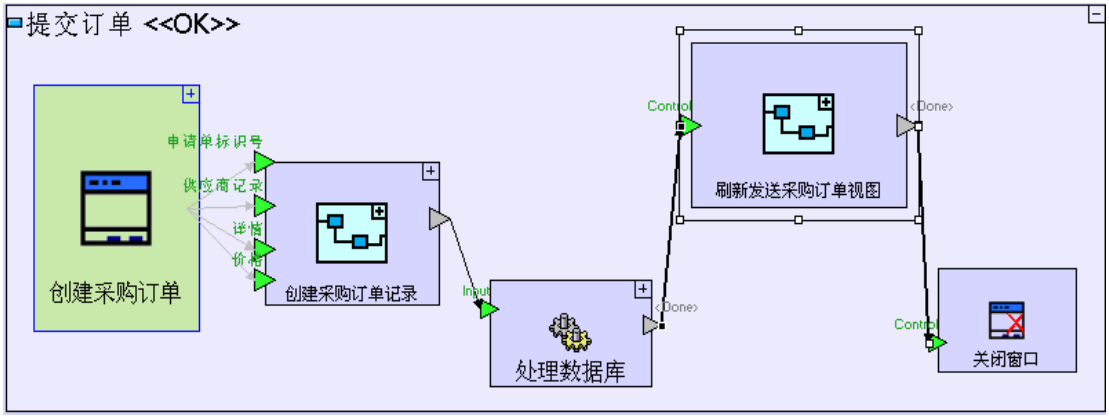
进入“**提交订单**”（**Submit Order**）按钮。

添加一个“**基本/动作**”（**Basic/Action**）。把它命名为 “***刷新发送采购订单视图***”（**Refresh Issue Purchase Orders View**）。并给它添加一个“**控制**”（**Control**）触发器和一个“**<完成>**”（**<Done>**）出口（使用“**添加单元**”（**Add Element**）情景子菜单(context sub-menu)）。

选定连接“**处理数据库/<完成>**”（**Handle Database/<Done>**）出口和“**关闭窗口/控制**”（**Close Window/Control**）触发器的流程，并把目标拖入到“**刷新发送采购订单视图/控制**”（**Refresh Issue Purchase OrdersView/Control**）触发器。

创建连接“**刷新发送采购订单视图/<完成>**”（**Refresh Issue Purchase Orders View/<Done>**）出口和“**关闭窗口/控制**”（**Close Window/Control**）触发器的流程。

“**提交订单**”（**Submit Order**）按钮模型外观应该与下图类似：



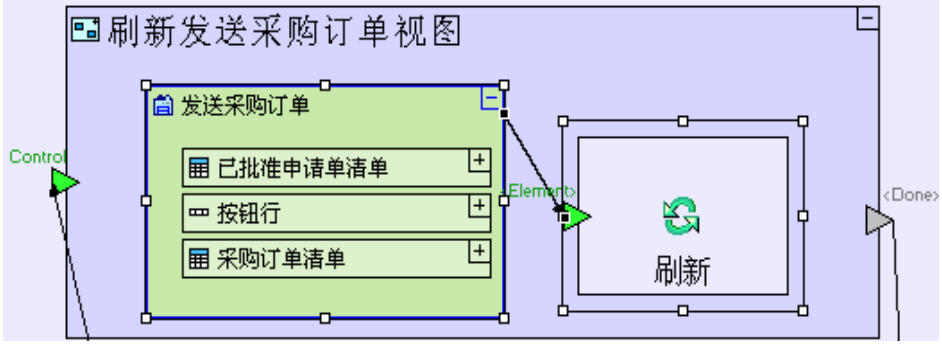
放大到“**刷新发送采购订单视图**”（**Refresh Issue Purchase Orders View**）。

给“**发送采购订单**”（**Issue Purchase Orders**）视图添加一个上级索引。

添加一个“**显示动作/刷新**”（**Display Actions/Refresh**）（ ）。

创建连接“**发送采购订单**”（**Issue Purchase Orders**）和“**刷新/<单元>**”（**Refresh/<Element>**）的流程。

“**刷新发送采购订单视图**”（**Refresh Issue Purchase Orders View**）模型外观应该与下图类似：



> **“刷新”**（**Refresh**）动作重新载入传送给它的显示单元，清除这个显示单元并运行它所包含的所有过程 – 这与刚开始单元被载入时的动作相同。

保存您的工作成果并在浏览器中查看应用程序。

创建一个新的订单，您可以看到当前申请单的总金额已经更新。

完成阶段 14 Completing Stage 14

输入项目样本“**申请单管理系统**”（**Requisition Management System**）。

> 如果您使用安装程序安装了“**Tersus 工作室**”（**Tersus Studio**），就不需要输入这个项目，这个项目应该会在工作室里面。
> 如果您需要输入此项目，请参阅**阶段 2** 最后的“**输入项目样本**”的部分。

这个项目样本具备到目前为止我们建立的模型的所有功能和完成此应用程序所需的附加功能。

我们看一下所添加的“**管理采购订单**”（**Manage Purchase Orders**）视图模型，这个视图模型执行以下功能：

- 一个已发送采购订单的清单。
- 一个“**订单已收到**”（**Order Received**）按钮。

“**订单已收到**”（**Order Received**）按钮会执行以下动作：

1. 将选定采购订单的状态更新为“**已收到**”（**Received**）。
2. 如果采购订单所属的申请单里面还有其他的采购订单尚未收到，那么它就会把申请单的状态修改为“**部分完成**”（**Partly Fulfilled**）。否则申请单的状态就会被修改为“**完成**”（**Fulfilled**）。

3. 刷新采购订单显示。

实例 See It Live

点击此处在另一个窗口里面打开实际项目。

附录 A –Tersus 工作室功能和工具　Appendix A – Tersus Studio Features and Tools

附录目标 Appendix Goals

本附录概述“**Tersus 工作室**”（**Tersus Studio**）和支持建模所提供的不同工具（如“**模型编辑器**”(Model Editor)、“**大纲**”(Outline)、“**储存库管理器**”(Repository Explorer)、“**模板库**”(Template Library)）和嵌入式应用程序和数据库服务器。

Eclipse 平台 The Eclipse Platform

“Tersus 工作室”（Tersus Studio）使用“**Eclipse** 平台”（**Eclipse** platform），这是一个行业标准的 IDE 框架，通过菜单和工具栏提供各种功能和重要的灵活性，包括可以按您的喜好重新排列显示。

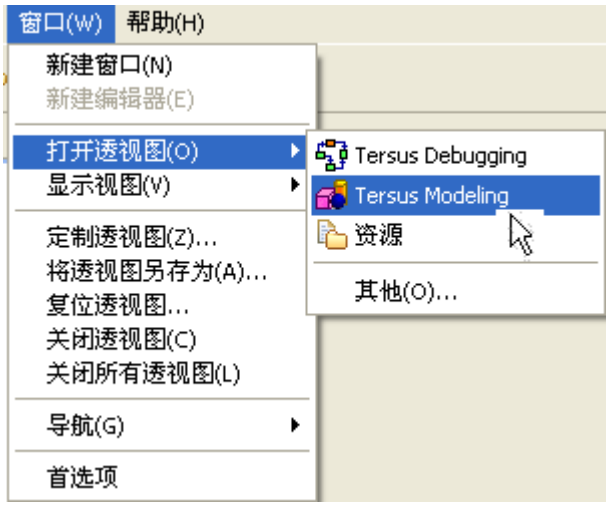
“**Eclipse** 平台”(**Eclipse** platform)使用“**透视图**”(**Perspective**)的理念显示一个或多个“**视图**”(**Views**)并把视图按特定方式进行排列。在透视图间转换会改变视图的组成（make-up）和排列。上面的屏幕快照显示了“**Tersus 建模**”(**Tersus Modeling**)透视图，该透视图在缺省情况下包含了以下视图：“**模型编辑器**”(Model Editor)、“**选项板**”（**Palette**）、“**大纲**”(Outline)、“**储存库管理器**”(Repository Explorer)、“**属性**”（**Properties**）和“**验证**”（**validation**）。

- 要在一个透视图中转换，使用“**窗口 -> 打开透视图**”（**Window -> Open Perspective**）。
- 要改变一个透视图中视图的排列，点击某个特定视图的标题栏（title bar）把它拖放到新位置上。

想了解更多关于 Eclipse 提供的功能的信息，请参阅 Eclipse 平台帮助系统(通过“**帮助 -> 帮助内容**”(**Help ->Help Contents**)进入）。

在继续之前，请确保显示“**Tersus 建模**”（**Tersus Modeling**）透视图。想知道当前正在使用哪个透视图，请看一下 Eclipse 窗口标题栏。如果标题栏不是以“**Tersus 建模…**”（**Tersus Modeling…**）开头，那么当前显示的就是另一个透视图。如果是这种情况：

选择“**窗口 -> 打开透视图->Tersus 建模**”（**Window -> Open Perspective-> Tersus Modeling**）。



如果没有出现“**Tersus 建模**”（**Tersus Modeling**）选项：

选择“**窗口-> 打开透视图->其他…**”（**Window ->Open Perspective->Other…**）。

从清单中选择“**Tersus 建模**”（**Tersus Modeling**），然后点击“**确定**”（**OK**）。

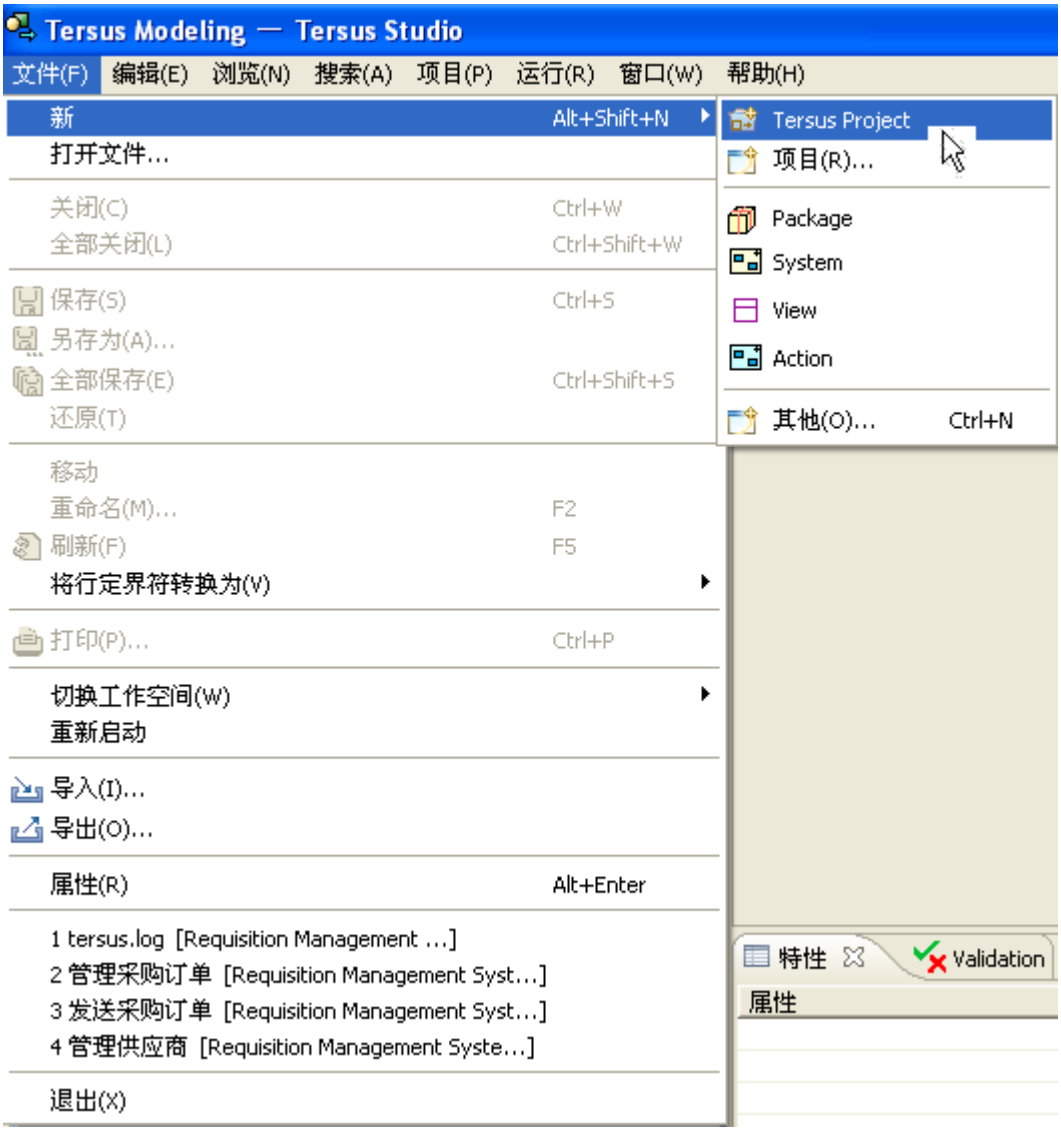
现在 Eclipse 应该转换到“**Tersus 建模**”（**Tersus Modeling**）透视图并在 Eclipse 窗口标题栏显示透视图的名称。

创建新项目 Creating a new Project

要开始一个新的 Tersus 项目，请按以下步骤操作：

选择“**文件->新**”（**File->New**）。

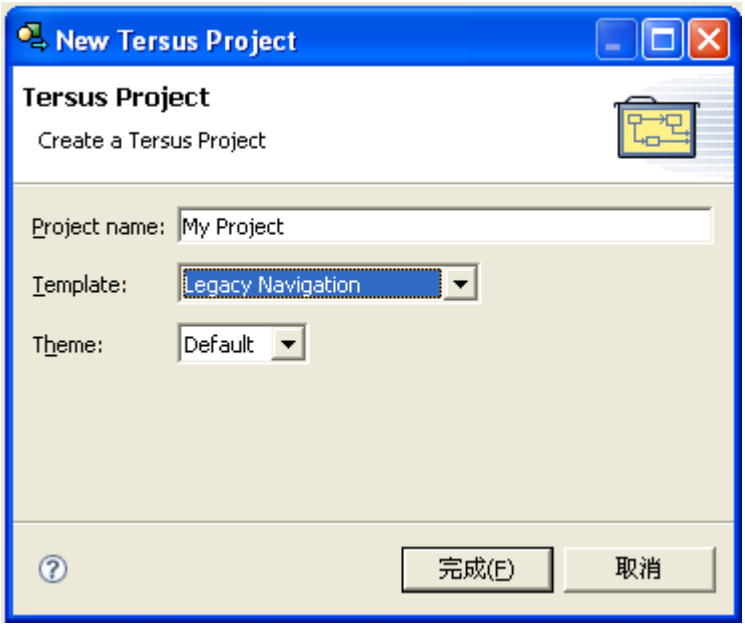
就会出现下面的子菜单：



选择“Tersus 项目”（Tersus Project）。

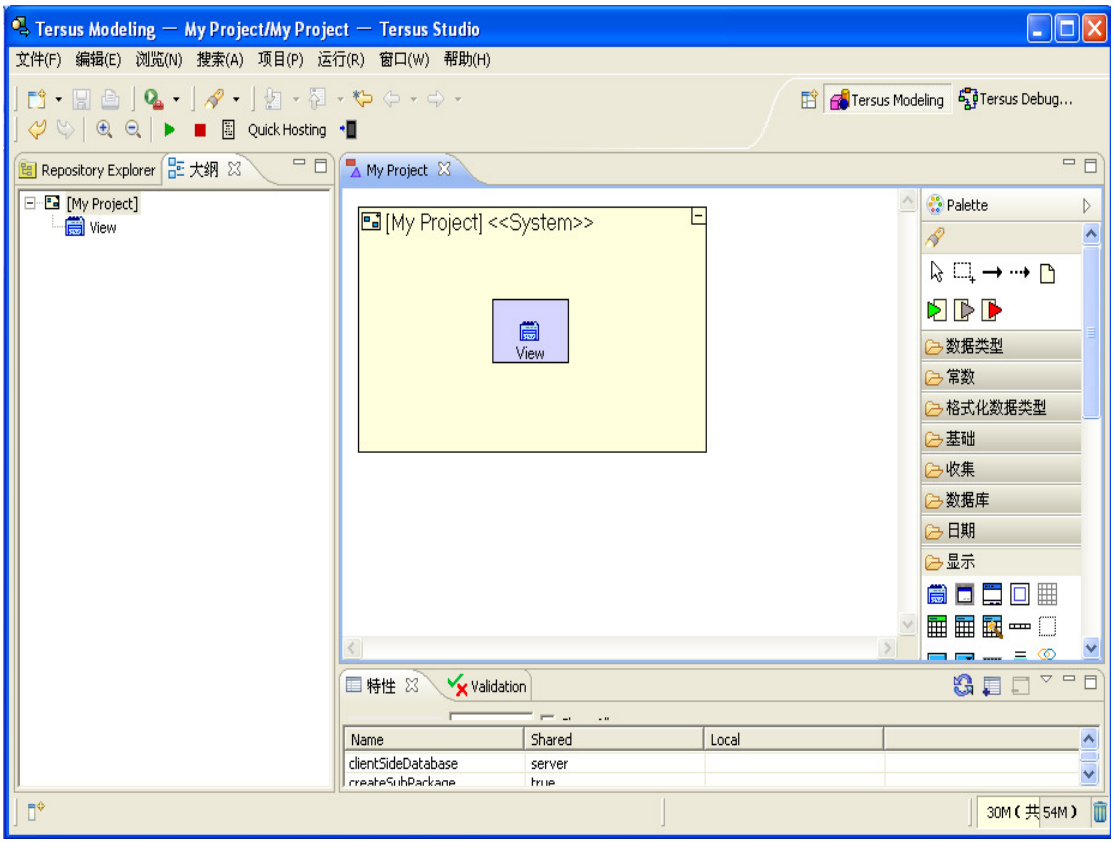
> 注意：菜单中有两个项目选项—确保您选择的是第一个项目（“Tersus 项目”（Tersus Project））而不是第二个项目（“项目 ...”（Project ...））。

就会出现下面的对话框：



为您的新项目输入一个“**项目名称**”（**Project Name**）：“***My Project***”。
按“**完成**”（**Finish**）按键以创建您的第一个 **Tersus** 项目。

您应该可以看到以下显示：



您可以使用您刚刚创建的项目“**My Project**”测试本阶段所谈到的不同的平台的功能。

熟悉模型编辑器 Familiarizing Yourself with the Model Editor

“选项板”（The Palette）

在“**模型编辑器”(Model Editor)**的右侧，您可以看到“**选项板**”（**Palette**）（ 

）。

选项板包含四种类型的单元：

1. “**编辑工具**”（**Editing Tools**）

用于定义和控制单元在编辑器中的布局（layout）。

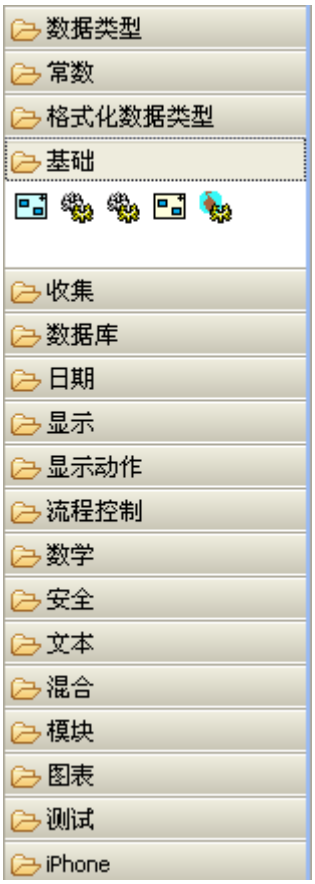
- （“**选择**”（**Select**））在编辑器中选择并移动单元。
- （“**选取框**”（**Marquee**））选择编辑器特定区域所有单元。
- （“**流程**”（**Flow**））在编辑器的单元之间创建一个（常规）流程。
- （“**删除**”（**Remove**））在编辑器的单元之间创建一个删除流程。
- （“**注释**”（**Note**））给编辑器中现有单元添加一个注释（文档/评论）。

2. “**槽**”（**Slots**）

用来定义过程的输入/输出“端口”（ports）。

- （“**触发器**”（**Trigger**））用来激活输入并将输入传送到过程里面的端口。
- （“**出口**”（**Exit**））用来把输出传送给过程的端口。
- （“**错误出口**”(**Error Exit**)）用来从过程中传送错误信息（例外信息）的端口。

3. “**模板库**”（**Template Library**）



模板是预先定义好的单元，它们是建模的结构件(building blocks)。模板有几种类别(如“数据类型”(Data Types)、“数据库”(Database)和“显示”(Display)　等等)，每个类别都含有几个模板。

4. “**搜索**”(**Search**)按键 

可按名称(或部分名称)搜索选项板中某个特定的模板。

在模型中插入新单元 Inserting a New Element to the Model

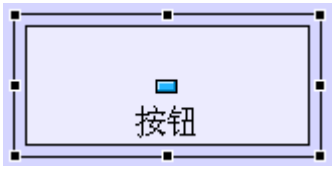
当您从“选项板”(Palette)中选择一个模板并回到编辑器中时，光标形状就会改变，以表示即将插入一个新单元到模板中。从选项板中插入新单元有 2 个方法：

1. 点击—创建一个大小为默认尺寸(由一个虚线矩形(dashed rectangle)表示)的新单元。
2. 点击并拖拽—创建一个由用户定义其大小的新单元。

我们也可以使用拖放(drag-and-drop)的方法从“储存库”(Repository)和“大纲”(Outline)中插入新单元。创建的新单元大小为默认尺寸。

选择单元 Selecting an Element

当你点击编辑器中显示的一个单元时，单元就会被选定并用一个边框标识出来(如以下屏幕快照所示)：



我们还可以通过使用如下方法选择多个单元：

1. 选择第一个单元之后，按住[CTRL]键，并继续选择其他单元。

2. 使用选项板中的“**选取框**”（**Marquee**）  工具在编辑器中规定一个选定区域，标记区域里面的所有单元都会被选定。

移动单元 Moving an element

选定一个单元之后，我们可以拖拽单元以改变单元在编辑器中的位置。

调整单元大小 Resizing an element

选定一个单元之后，您可以通过拖拽选择边框周围出现的 8 个选择锚点（selection anchors）中的任何一个调整单元的大小。

使用转角锚点（corner anchor）时，某些类型的单元（特别是“显示单元”、“过程单元”和“系统单元”（Display, Process and System elements））调整尺寸的方式会有所不同。这些单元可能含有其他单元，因此使用转角锚点（corner anchor）调整大小时，单元和它的子单元的纵横比（aspect ratio）（宽-高比例）保持不变。

下钻（Drill-down）

模型是分级的，也就是说一个单元可能含有其它单元（组成子模型），而每个子单元也可以含有其他子模型，以此类推。

模型编辑器提供下钻功能，让我们可以不同的详细程度查看模型的各个部分。

下钻（或上钻）有两种方法：

1. “**扩展/折叠**”（**Expand/Collapse**）- 如果模型中的单元含有其它单元，它会显示出一个小的（扩展按钮）或  （折叠按钮）。点击扩展按钮/折叠按钮会使模型编辑器显示单元的内容（扩展），或隐藏单元的内容（折叠）。

2. “**双击**”（**Double-clicking**）模型中的一个单元就会扩展单元并显示单元的内容，双击还会使编辑器把视图在模型上居中，并放大或缩小视图直到模型刚好在视图里面。

放大/缩小(Zoom-in/out)

放大和缩小有各种不同的方法：

1. 在“**模型编辑器**”（**model editor**）中“**双击**”（**Double-clicking**）一个单元，会使编辑器放大/缩小，并在单元上居中（并扩展，如果适用的话）。
2. 在“**大纲**”（**outline**）中“**双击**”（**Double-clicking**）一个单元与在编辑器中“双击”（Double-clicking)相似。如果我们想直接移到模型不在当前显示范围内的部分，这个办法会很有用。
3. “**工具栏**”（**toolbar**）提供放大（  ）和缩小（  ）按钮。

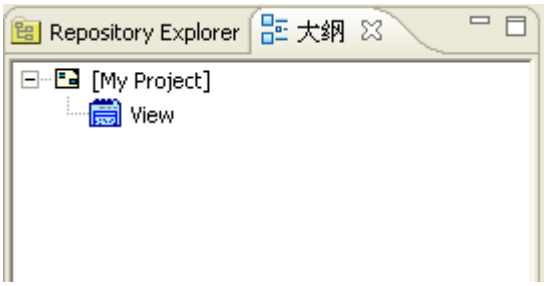
缩放对数据单元的作用有点不同。只有双击最上层的数据单元时，双击才会起作用，双击任何下级数据单元不会起作用。

撤消/重复（Undo/Redo）

模型编辑器提供完整的“**撤消/重复**”（**Undo/Redo**）功能。这使您能尝试不同的建模策略并快速地对模型进行修改，而不用担心丢失有效的功能。只要模型编辑器窗口处于打开状态，“撤销/重复”在整个编辑过程都可以用（保存并运行应用程序不会阻止后面的“**撤销**”（Undo）的运行）。

您可以从“**编辑**”（**Edit**）菜单中找到“撤销/重复”功能，您也可以通过**[CTRL-Z]/[CTRL-Y]**键盘快捷方式使用“撤销/重复”功能。

大纲 The Outline



出现在模型编辑器左边的大纲（Outline）面板提供了应用程序模板的另一个视图，这是一个组成完整模型的单元的树状分层视图。大纲反映模型本身的层次性质（包含其它单元的单元等）并总是与模型本身的层次性质同步。

与模型编辑器同步化 Synchronization with the Model Editor

大纲（Outline）中选定的项目总是与模型编辑器中选定的单元保持同步。选定这两者之中的一个，就会选定另一个中匹配的项目。

双击动作（Double-Click Behavior）

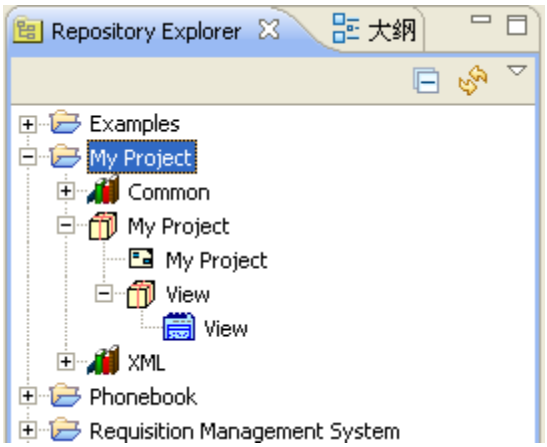
在大纲中双击一个模型会将模型放入当前模型编辑器中并进入模型。这个方法在我们想进入一个特定单元时会很有用（尤其在复杂的有不同层次的模型中更是如此）。

拖放动作（Drag-and-Drop Behavior）

从大纲中拖拽一个模型并把模型放在编辑器中会重新使用模型，这就意味着在一定的限制条件下（教程后面的阶段会谈到这一点），可以再次使用已经建模的功能，而不必从头开始重新建模。

重新使用时，同一个单元在大纲树状结构中出现多次（在不同的位置）。但有时单独的、不相关的模型有相同的名称；如果这些模型在不同的数据包中，这种情况是可能出现的（请参阅“**储存库管理器**”（**Repository Explorer**）部分以了解更多详情）。

储存库管理器 The Repositiory Explorer



出现在模型编辑器左边的储存库管理器（Repository Explorer）提供了一份组成应用程序的所有模型的清单。

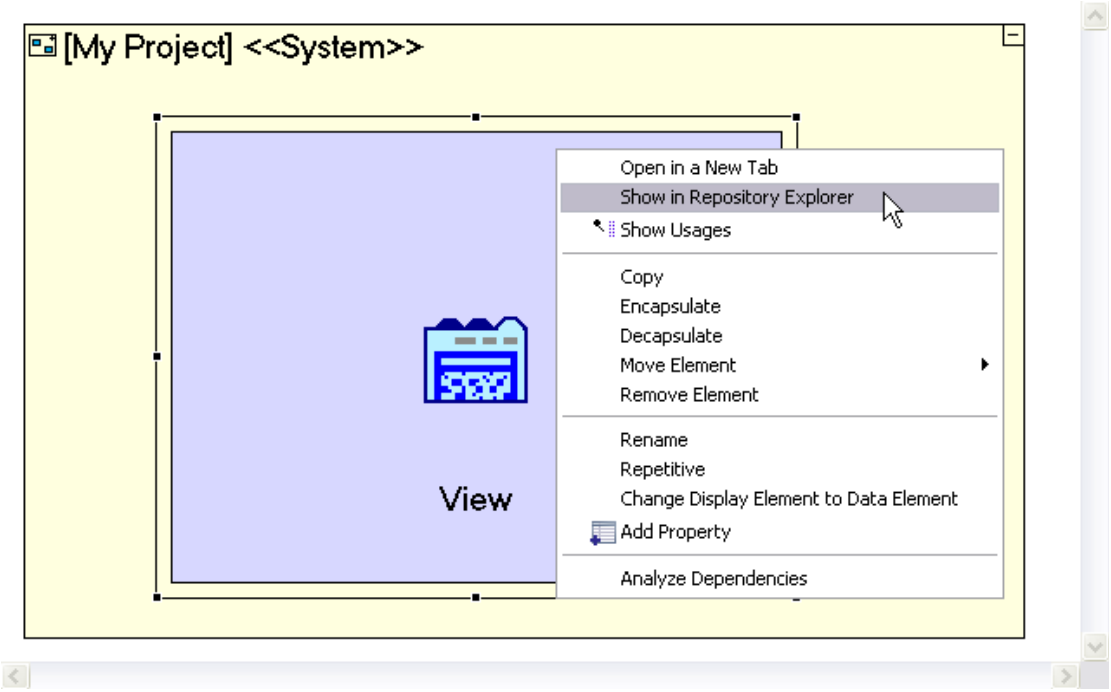
使用储存库管理器 Finding your way in the Repository

储存库被放到不同的层次的“**数据包**”（**Packages**）（和“**子数据包**”（**sub-packages**）中）里面，这些“**数据包**”（**Packages**）（和“**子数据包**”（**sub-packages**））把模型分成不同的功能类别。使用特定的模板（“**系统**”、“**视图**”、“**按键**”（**System**, **View**, **Button**））时，就会自动创建数据包；您也可以随时创建其它数据包，并按您的意愿将模型从一个数据包移到另一个数据包中。建模时，新单元会自动生成到与添加新单元的母模型相同的数据包中。

要在储存库中找到您当前编辑的模板，您可以使用以下快捷方式：

右击编辑器中的单元。

从情景菜单（context menu）中选择“**显示在储存库管理器中**”（**Show in Repository Explorer**）选项。



双击动作（Double-click Behavior）

双击储存库中的一个模型会在一个新模型编辑器窗口中打开模型。处理复杂的多层次的模型时，您可以使用这个方法打开并编辑某个特定的子模型。

拖放动作(Drag-and-drop Behavior)

从储存库中拖拽一个模型并把它放进编辑器中，这样操作会重新使用模型

（与从大纲中拖拽类似）。目标模型编辑器必须和源模型（source model）一样，都属于同个项目。

另外还可以在储存库内拖放模型，这样做会把模型从一个数据包移到另一个数据包中。

储存库管理器和大纲(The Repository vs. the Outline)

尽管储存库和大纲都以分层的方式显示项目内容，但是这两者之间也有一些差别，主要是：

储存库管理器	大纲
层次结构显示所有项目的所有模型。	层次结构映射到当前模型编辑器中并与当前模型编辑器同步。仅显示组成当前模型的单元。
重复使用的模型仅出现一次	重复使用的模型在当前模型中使用多少次，就出现多少次。
未使用模型出现在储存库中并可能被再次使用	未使用的模型不会在大纲中出现，因为他们不是当前模型的一部分。
每个模型显示的名称是“**模型名称**”（**Model Name**）。	每个模型显示的名称是“**单元名称**”（**Element Name**）。

嵌入式应用程序和数据库服务器 The Embedded Applicatin and Database Servers

Tersus包含一个捆绑式的小型应用程序和数据库服务器（Application & Database Server）；在建模过程的每个阶段，我们都可以用它来查看并测试已建模的应用程序。

我们可以通过工作室的工具栏控制嵌入式服务器：（  ）

 在应用程序服务器中“**启动应用程序**”（**Launch the application**）并打开浏览器。

 “**停止应用程序**”（**Stop the application**），停止应用程序在应用程序服务器中运行。

（  ）在文本编辑器窗口中“**显示应用程序日志文件**”（**Show the application log file**）。

您通过进行以下操作，尝试查看应用程序：

点击  按键，在服务器中启动应用程序并在浏览器中打开应用程序。

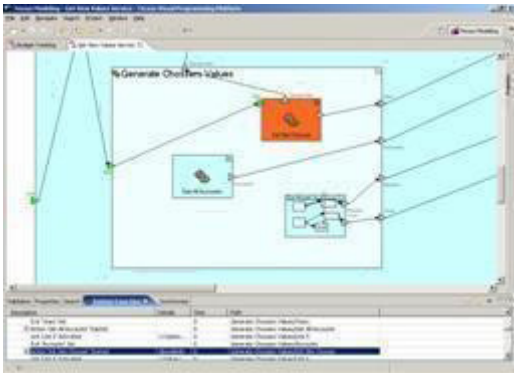
您可以转换回工作室（Studio）并修改您的建模过程。在保存您做的修改之后，如果您转回到浏览器，应用程序就会自动刷新并把您所做的最新的修改应用到应用程序模型和数据结构中。

附录 B-可视调试 Appendix B – Visual Debugging

“Tersus 可视编程平台”（Tersus Visual Programming Platform）使您能以定义应用程序所用的相同的图表，直观地追踪（trace）该应用程序的运行，这样就能够更容易地理解业务逻辑并发现缺陷（bugs）。

以追踪模式工作时，Tersus 服务器（Tersus Server）会记录应用程序运行的每一个步骤，我们可以随时回放这些步骤以查看应用程序的流程和每个数据单元的数值。

与只允许往前操作的常规调试器不同，Tersus 的追踪功能就像“时光机器”一样，您可以在运行过程中向后或向前操作。如果您看到某个点可能出错，您可以追踪回去以找到不当动作（improper behavior）的根本原因。



这个功能的在线文档可以从这里找到。

创建于 2010 年 2 月 24 日