目录

[1. 上下文菜单项Action注册 2](#_Toc419989000)

[2. 工作流上下文菜单提供器 2](#_Toc419989001)

[3. 根据不同类型节点的右击事件构建上下文菜单 3](#_Toc419989002)

[4. 附录 GraphicalEditor方法介绍 4](#_Toc419989003)

**Data Studio上下文菜单管理**

**By 王方舟**

 Data Studio的工作流编辑器中，针对不同类型的节点，配置了不同的上下文菜单，并且实现了上下文菜单的动态构建。目前工作流当中的节点类型主要分为数据节点、算法节点、模型节点和可视化节点。其中，数据节点主要负责数据的读入、处理和处理结果表的预览；算法节点主要负责执行相应算法，读入数据并输出模型节点；模型节点以数据作为输入，输出包含预测结果的数据；可视化节点负责将输入数据转化成各式图表，对数据做可视化展现。

 不同类型节点的上下文菜单如下所示：



图表 1 数据节点菜单



图表 2 算法节点菜单



图表 3 可视化节点菜单

 本文将介绍GEF框架中的EditorPart上下文菜单创建流程，以及在Data Studio上的实践过程。

# 上下文菜单项Action注册

RCP程序的EditorPart中最常用主要是两类：TextEditor和GraphicalEditor，Eclipse的Editor就是一种TextEditor，而Data Studio中的Editor则是后者。

GraphicalEditor隶属GEF框架，与大多数图形编辑框架类似，GEF框架也是一种MVC框架，此外GEF提供了不同的编辑部分和图案类，供用户扩展。在GraphicalEditor中添加上下文菜单，首先要注册菜单项的各种Action，GraphicalEditor类提供了createAction方法，注册编辑器中右键后需要展现的事件列表。父类中只提供了简单的撤销、恢复、全选、删除、保存这5个基本的IAction，如果要添加自定义IAction，则需要在子类里面覆写该方法，先调用父类的createAction方法，将父类的5个基本事件注册，再注册子类的事件到ActionRegistery对象中。

注册好的事件通过createGraphicalViewer方法，传给WorkflowGraphicalViewerCreator的实例，在其中的createViewer方法中，传递给WorkflowContextMenuProvider的上下文菜单提供器，提供注册事件列表，在该类中动态的构建上下文菜单。

# 工作流上下文菜单提供器

工作流上下文菜单提供器中包含的成员有上述所说的事件注册列表、WorkflowGraphicalViewerCreator所创建的GraphicalViewer对象，以及鼠标右击事件的发生位置Point，在其继承自父类ContextMenuProvider的buildContextMenu的方法里面，通过ID在传递的注册事件列表拿到对应的上下文菜单IAction，最终形成一个完整的菜单列表。

上下文菜单提供分组的功能，通过一个GroupID来创建一个Separator对象，该对象将添加到该分组ID的IAction与其他IAction分隔开来，在UI上面体现为一条分隔线。

特别地，buildContextMenu方法的参数为一个IMenuManager对象，调用该对象的add方法可以将Separator对象添加到菜单中，调用remove方法可以删除某个ID的IAction，调用appendToGroup方法可以将IAction对象添加到指定分组中。

# 根据不同类型节点的右击事件构建上下文菜单

那么ContextMenuProvider是如何根据不同的节点类型来构建不同的菜单实例的呢？在上节所说的GraphicalViewer对象，可以通过getSelectedEditParts方法拿到当前编辑器中被用户选中的EditParts对象列表，当用户使用WorkflowSelectionTool选中了多个EditPart对象时，理想状态下当然不能弹出多个上下文菜单，因此在这里会取该列表的第1个EditPart作为应显示其上下文菜单的载体，当然这在用户只选择了一个EditPart对象时是自然的。

拿到当前所选择的EditPart之后，只是拿到了在图形编辑器上UI的对象，如果要做模型的类型区分，那必须要通过EditPart的getModel方法，拿到其中封装的模型，在Data Studio中，该模型对象是一个NodeContainer对象。

NodeContainer在设计器中充当一层节点的容器封装，其中它的继承关系如下所示：



图表 4 NodeContainer继承关系

其中WorkflowManager为整个工作流中的节点集合类，SingleNodeContainer用来表示一个节点的容器类，NativeNodeContainer表示其本身只有一个节点，而SubNodeContainer中可能有多个节点，但在外部表现成一个节点（类似于一个元节点）。

特殊地，在NativeNodeContainer中，会封装一个Node对象，这个对象该容器类中封装的节点对象，getNode之后再getFactory则会取到该节点对象的工厂类。

工厂类的子类从功能上区分了三种不同类型的节点，数据节点、算法节点和可视化节点的工厂类分别为DataNodeFactory 、AlgorithmNodeFactory、VizNodeFactory，根据节点工厂类的类型，可以构建分支创建不同的上下文菜单。

# 附录 GraphicalEditor方法介绍

* createActions()——为该编辑器创建操作，子类应该覆盖该方法以使用ActionRegistery创建并注册操作。
* getCommandStack()——返回命令栈。
* getEditDomain()——返回编辑域。
* initializeGraphicalViewer()——覆盖以在GraphicalViewer创建后设置它的内容。
* setEditDomain(DefaultEditDomain)——为该EditorPart设置EditDomain。