NiagaraST Query Maker GUI
CS-410 Winter 2012

Dustin Schmidt, Weston Wedding, Brian Liu

21 March 2012

Contents

1 Overview

The project goal is to design an application for graphically building NiagaraST query plans using Java and Swing. The program will allow a user to assemble NiagaraST query plans by constructing a directed graph of operators. The user will select from a list of operators to generate an operator node. The graphic representation of the operator node will contain fields for entering the operator parameters. The list of operators available to the user and the parameters each operator needs will be generated from the NiagaraST source code for logical operator representation. In this way the application will be scalable as new operators are added. The data flow through the query plan will be constructed by graphically connecting the inputs and outputs of various operators. When the query plan construction is complete an XML query plan file will be generated that will successfully run in NiagaraST.

2 Class Descriptions

2.1 DTDInterpreter

This class is used by QueryPlan to interpret a DTD file which provides the logical definition for NiagaraST operators.

2.2 GraphNode

This class extends Operator to include a draw function to draw the operator on the graph canvas. GraphNode is constructed with an operator template which defines it's corresponding operator name and attributes. The draw method accepts a reference to an mxGraphComponent and it's parent and draws the graphical

representation of the operator onto the graph component.
The graphical representation of an operator contains three main components. An input port, a main body, and an output port. The main body of the node is mainBox which is an mxCell whose value is set to this GraphNode. Setting a cells value to an object causes the string representation of the that object to be displayed in the cell, hence the mainBox contains a string representation of this Operator GraphNode as defined by the toStringMethod. The input port and output port are both instances of the IOPort subclass.

2.2.1 GraphNode.IOPort

This is a nested class used to represent the input and output of an operator. IOPort extends mxCell but includes a boolean value input and boolean methods isInput and isOutput. These methods are used to validate edge creation in MainForm's EdgeListener.

2.3 MainFrame

This class is the main frame of the application. It contains an mxGraph component which is a swing component for editing graphs.

2.4 Operator

This class represents an instance of an operator. The Operator is constructed with and OperatorTemplate which defines the name and attributes of this operator. Once instaniated the Operator attributes are manipulated through the get and set Attribute methods. These methods modify the private HashMaps which map attribute names to their values. There is also a method for generating the XML representation of this operator. The current implementation of the Operator class treats sub elements (such as predicates for select) as a single string *elements* and not as meaningful content. There is also a member String *comments* which specifies a comment string to precede the XML representation of the operator.

2.5 OperatorTemplate

The operator template class is used by most classes in this project. The OperatorTemplate provides a representation of an unstansiated operator. The operator template contains a listing of attributes and elements and the name of an operator. This template is used to construct an Operator instance with the given attributes, elements and name.

2.6 OperatorSelectorDialog

This dialog is used by MainFrame to display a list of candidate operators. The user clicks an operator (or ctrl+click many or shift+click many) and then clicks the "Add" button and the operators are drawn on the graph.

2.7 PropertyDialog

This dialog allows the user to set the attributes of an operator. This dialog is displayed by right clicking an operator in the graph and clicking "Properties" After changes are made the user must click the "Update" button in order to apply the changes. If the operator has an "input" attribute it is displayed as read-only. This is because inputs to an operator are set by connection the input and output ports of operators on the graph.

2.8 QueryPlan

This class contains a logical representation of the query plan currently being edited by the user. This class maintains a listing of all operator templates (*opTemplates*) drawn from a DTD file, and a listing of all instantiated operators currently in the plan (*opList*). When the user makes changes to an operator either on the graph or through the property dialog the state of the corresponding operator in opList is updated.

When the user has completed a Query the method QueryPlan.generateXML is called and given a fileName, the query is then rendered to the given file name.

3 Third Party Libraries Used

- JGraphX - obtained from jgraph.com, provided in the file jgraphx.jar
- DTDParser - obtained from , provided in the file dtdparser.jar
- XML???? - obtained from ???