Санкт-Петербургский Государственный Университет
Математико-механический факультет

Курсовая работа

Визуализация кода встроенных языков в Microsoft Visual Studio IDE

Выполнил:
студент 243 группы
Ершов Александр

Научный руководитель:
старший преподаватель кафедры системного программирования
Григорьев С.В.

Санкт-Петербург
2014

Содержание

1 Введение

1.1 Встроенные языки

Встроенные языки — это языки, команды которых выполняются из базового языка. В основном это предметно-ориентированные языки (специализированные под конкретную область), но вообще в качестве встроенного может выступать любой язык программирования. В качестве примера можно привести JavaScript в Java или SQL в C#.

SQL в C#

```
public void SelectByName(int cond)
        {
            var baseQuery = "drop procedure";
            string tableName;
            switch (cond)
            {
                case 1:
                    tableName = "some_else_table";
                    break;
                case 2:
                    tableName = "some_else_table_2";
                    break;
                default:
                    tableName = "default_table d";
                    break;
            }

            Program.ExecuteImmediate(baseQuery + tableName);
        }
```

1.2 Проблема

Для компилятора код встроенного языка - просто строка, и статически он не ищет в ней ошибок. В проекте YaccConstructor[1] разрабатывается плагин[2] для статического анализа встроенных языков. Он подчеркивает синтаксически неверные участки кода. Но если строка формируется динамически, тогда возможно большое количество вариантов и сложно ориентироваться в получившейся структуре. Представление всех вариантов строк в виде графа должно упростить пользователю понимание кода встроенного языка.

1.3 Постановка задачи

- Разобраться с библиотекой GraphX. GraphX [3] это .NET библиотека для визуализации графов, основанная на алгоритмах из Graph#, которая использует WPF для визуализации.
- Разобраться с ReSharper и с ReSharper SDK[4] (С ToolWindow[5] в SDK). ReSharper - это плагин для Visual Studio, который проводит статический анализ кода, помогает с рефакторингом, автозаполнением и т.д..
- Собрать и запустить YC.ReSharper.AbstractAnalysis plug-in[2]
- Добавить в него возможность просмотра кода встроенного языка в графическом виде. Он должен будет выводить граф, на ребрах которого должны быть части исходного кода встроенного языка. Он не зависит от используемого языка, так как анализ абстрактный.

```
string t1 = "table1"
string t2 = "table2"
execute("select * from " + if check then t1 else t2)
```

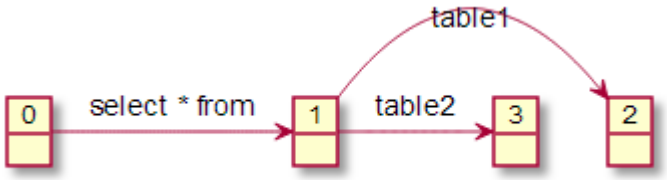

Рис. 1: Что требуется

1.4 Существующие решения

Graphviz Graphviz[6] - программное обеспечение с открытым исходным кодом для визуализации графов. Описание графа осуществляется на языке DOT, а на выходе получаем граф, который можно сохранить в виде изображения, PDF, SVG и PostScript. Имеется возможность настройки цвета, шрифтов, создание гиперссылок и т.д.

Alvor Alvor[7] - плагин для Eclipse, который статически проверяет встроенный SQL в Java. Проверяет синтаксическую и семантическую корректность, а также наличие объекта через JDBC. Находит синтаксически неверные строки в запросе, написанные с ошибками имена строк/столбцов, несоответствия типов и т.д.

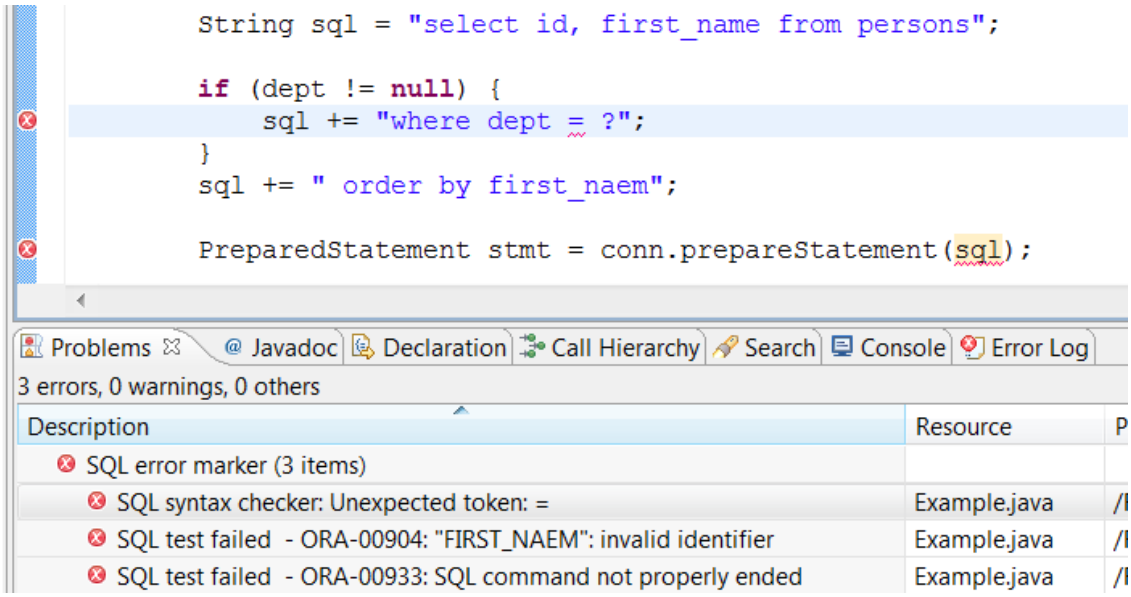

Рис. 2: Alvor пример работы

2 Реализация

2.1 Общее описание

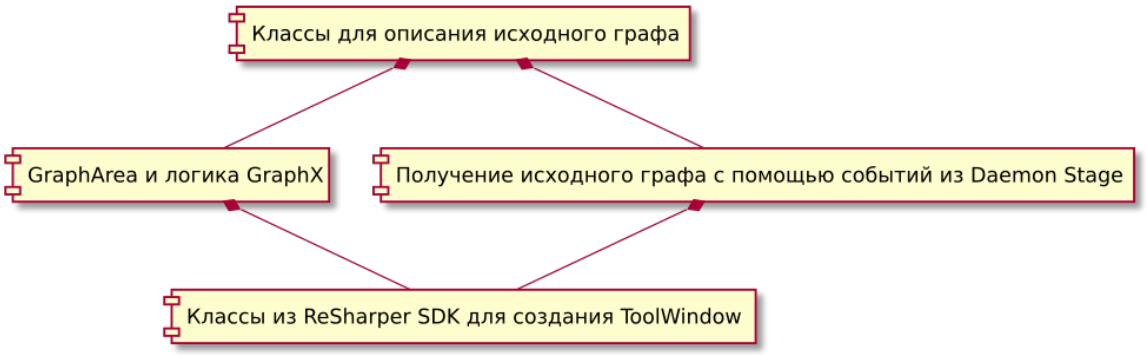

Рис. 3: Архитектура

Классы для описания графа представляют из себя три класса : DataEdge, DataVertex и DataGraph. Все они наследуются от классов из GraphX[3]. Вся нужная информация хранится в DataEdge : часть кода встроенного языка, который надо выводить вместе с этим ребром и BackRef - координаты данного сегмента в основном коде.

GraphArea представляет собой класс, визуализирующий исходный граф. В данном решении создано два класса EmptyGraphArea : стандартный GraphArea и ReadyGraphArea - с применением XAML шаблонов, потому что некоторые опции визуализации (например отображение лейблов у ребер) можно настроить только с помощью XAML шаблонов. Вся логика находится в классе LogicCore. С помощью него настраиваются алгоритмы раскладки, отображение текста на ребрах и т.д.

DaemonStageProcess[8] представляет из себя класс, где генерируется граф для исходного кода. Плагин работает в фоновом режиме, и прямого доступа к экземпляру этого класса нет. Поэтому граф получается с помощью двух событий : первое извещает

DaemonStageProcess о том, что граф надо отправить, а второе непосредственно отправляет.

ToolWindow представляет из себя два класса из ReSharper SDK для создания окна в стиле Visual Studio.

2.2 Особенности реализации

В ходе работы стало ясно, что документация по ReSharper SDK (по ToolWindow[5]) устарела. Она не обновлялась, начиная с версии ReShaper 2. Многие старые классы, которые были указаны в документации, уже не существовали, поэтому пришлось разбираться, как создать ToolWindow[5].

С библиотекой GraphX[3] тоже есть некоторые трудности. В исходном графе большинство дуг параллельны,а поддержка лейблов для параллельных дуг появилась только в версии 2.0.2, которая вышла в мае. Параллельные дуги рисуются как линии и не входят в вершины.

Сам плагин работает в фоновом режиме, и прямого доступа к экземпляру класса, где анализируется код и создается граф, нет. Поэтому доступ к графу осуществляется с помощью события, которое передает класс Processor, в котором хранится исходный граф.

2.3 Результаты

В плагин добавлена возможность визуализации кода встроенного языка в виде графа. Проблема отображения параллельных дуг - они рисуются как прямые линии и не входят в вершину.

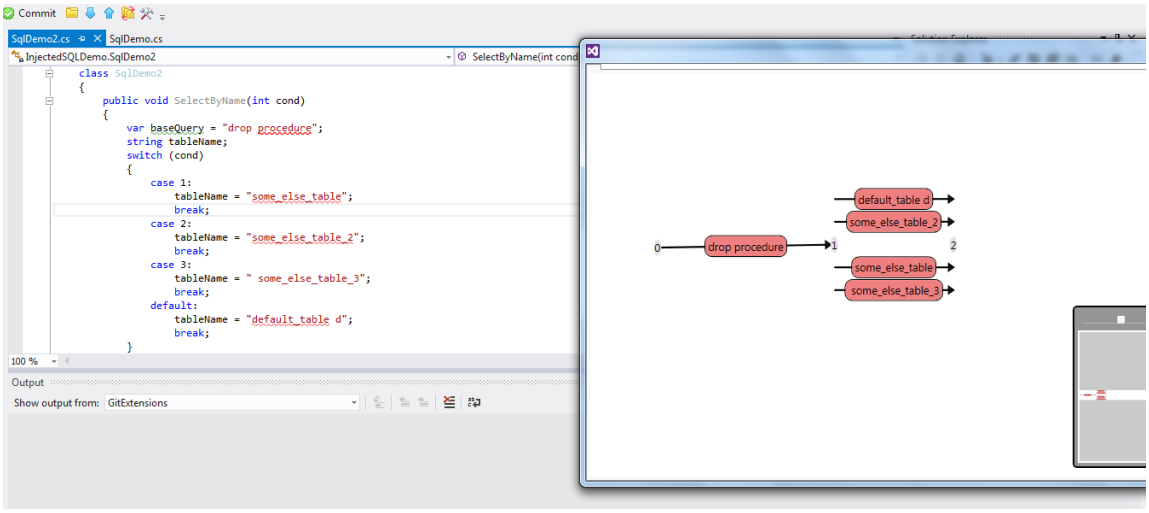

Рис. 4: Пример работы

2.4 Дальнейшее развитие

Добавить возможность по нажатию на ребро попадать в участок кода. В каждом ребре графа хранится координата сегмента строки, отображаемого им, во внешнем коде. Это позволит более удобно ориентироваться по коду встроенного языка.

Список литературы

[1] Yacc Constructor. https://code.google.com/p/recursive-ascent/.

[2] YC ReSharper plugin. https://code.google.com/p/recursive-ascent/wiki/ReSharper_AbstractAnalysis.

[3] GraphX. http://panthernet.ru/en/projects-en/graphx-en/14-graphx-en/33-graphx-doc-main.

[4] ReSharper SDK. http://confluence.jetbrains.com/display/NETCOM/ReSharper+8+Plugin+Development.

[5] ToolWindow. http://confluence.jetbrains.com/display/NETCOM/Creating+tool+windows.

[6] Graphviz. http://www.graphviz.org/.

[7] Alvor. https://code.google.com/p/alvor/.

[8] Daemon Stage. http://confluence.jetbrains.com/display/NETCOM/2.4+Daemons+and+Daemon+Stages.