Язык Рефал Плюс на платформе Java

Ю.А. Климов, А.Ю. Орлов, С.А. Романенко

Введение

При разработке больших программных проектов становится всё более популярным использовать богатый арсенал языков программирования для решения различных подзадач. Это связано с тем, что части проекта могут сильно отличаться по требованиям к срокам реализации, надёжности и скорости работы, по алгоритмической сложности. Выбор подходящего языка может дать существенное преимущество по наиболее важным из этих критериев.

В настоящее время активно развиваются платформы Java и Microsoft .NET, обеспечивающие возможность прозрачного взаимодействия частей проекта, описанных на различных языках программирования.

Для платформы Java уже существует довольно богатый набор интегрированных языков программирования, и этот набор постоянно расширяется. В рамках проекта Refal+ [1,6,7] выполнена реализация языка Рефал Плюс для платформы Java, которая позволяет удобным и естественным образом вызывать из Рефал-программ методы классов Java, и, наоборот, вызывать из языков платформы Java функции, описанные на языке Рефал.

Что такое Рефал

Язык программирования Рефал разработан В.Ф. Турчиным в шестидесятые годы прошлого столетия как функциональный язык, ориентированный на задачи, связанные с преобразованием символьной информации (symbol manipulation) [3].

Основная структура данных в Рефале, так называемое *Рефал-выражение*, представляет собой дерево произвольной арности, узлы которого можно строить и читать как слева направо, так и справа налево. Использование такой структуры выделяет Рефал среди остальных языков программирования высокого уровня.

Для разбора Рефал-выражений используется *сопоставление с образцом* -- мощная и выразительная конструкция, которая позволяет записывать алгоритмы на Рефале естественным и понятным образом. Цитируя автора языка: "*Рефал дает свободу и удобство в создании структур данных наряду с использованием лишь математически простых механизмов управления -- сопоставления с образцом и подстановки. Это именно то, что нужно для символьной обработки и для искусственного интеллекта*" [4].

Использование языка Рефал облегчает создание и сопровождение алгоритмов обработки символьной информации. Такие подзадачи возникают во всех достаточно крупных проектах. На платформе Java они часто связаны с преобразованием XML-документов [5].

Платформа Java даёт возможность исполнять программу на различных конфигурациях аппаратного и программного обеспечения. Для неё существует богатый набор библиотек, в частности, для создания распределённых и параллельных приложений. Эти возможности позволяют использовать Рефал при создании серьёзных и востребованных на сегодняшний день продуктов, например, опирающихся на распределённую работу через интернет, параллельные вычисления и т.д.

Рефал Плюс -- это диалект языка Рефал, разработанный Р.Ф. Гуриным и С.А. Романенко в 1991-ом году. В 2006-ом году синтаксис языка был адаптирован для более гладкого взаимодействия с современными программными средами [7].

В основе системы Refal+ лежит компилятор с языка Рефал Плюс в байткод для виртуальной машины Java. Он поддерживает автоматические преобразования между данными Рефала и Java при вызове методов Java из кода на Рефале и наоборот.

Преобразование данных

В качестве элементов выражения (*термов*) могут выступать как встроенные рефальские типы данных (такие, как *символы-слова* или *символы-числа*), так и объекты произвольных классов, определённых на Java.

Объекты Java можно сопоставлять как "по ссылке" (с помощью оператора ==), так и "по содержанию" (с помощью метода equals). При использовании объектов Java в Рефал-выражении, необходимо указать способ их отождествления в Рефале.

Произвольные объекты Java могут быть использованы в Рефал-выражении при условии, что они будут отождествляться "по ссылке". Отождествление "по содержанию" может быть использовано только для объектов таких классов, которые реализуют интерфейс java.lang.Comparable. Встроенные рефальские типы реализованы с помощью таких классов. Например, *символы-литеры* -- с помощью класса java.lang.Character, а *символы-числа* -- с помощью класса java.math.BigInteger.

Примитивные данные типа char и byte, short, int, long естественным образом преобразуются в *символы-литеры* и *символы-числа* соответственно. Значения true и false типа boolean преобразуются в *символы-слова* "True" и "False" соответственно.

Некоторые объектные типы также удобно преобразовывать в рефальские типы данных. Например, объекты класса java.lang.Integer можно преобразовывать в *символы-числа*, а массивы -- в Рефал-выражения.

Описание форматов

Для того чтобы использовать метод Java в языке Рефал Плюс, у него должен быть описан Рефал-формат. Такое описание задаётся в Рефал-модуле с помощью обычного объявления Рефал-функции, а также сопоставления этой Рефал-функции прототипа метода Java при помощи конструкции "$native Java":

```
$func Thread s.runnable = s.thread;
$func IsInterrupted s.thread = s.is_interrupted;

$native Java {
   Thread = "Thread (Runnable)"; // Конструктор класса java.lang.Thread
   IsInterrupted = "boolean Thread.IsInterrupted ()"; // Метод класса
java.lang.Thread
};
```

В примере для конструктора и метода класса java.lang.Thread определены имена и форматы соответствующих им Рефал-функций. Аргументы метода Java сопоставляются с аргументами рефальской функции в порядке следования (имена не существенны, для нестатических методов считается, что this передаётся в

качестве первого аргумента).

В данном случае мы имеем дело с классами Java, не реализующими интерфейс Comparable, поэтому соответствующие рефальские термы будут сравниваться "по ссылке". Функция Thread ожидает на вход *символ-ссылку* на объект, реализующий интерфейс java.lang.Runnable, и возвращает *символ-ссылку* на объект класса java.lang.Thread. Функция IsInterrupted ожидает на вход символ-ссылку на объект класса java.lang.Thread и возвращает символ-слово "True" или символ-слово "False". Эти функции можно использовать как обычные Рефал-функции.

Для того чтобы использовать в Java функцию, реализованную на языке Рефал Плюс, необходимо определить для неё Java-прототип. Он определяется аналогичным образом в конструкции "$native Java".

```
// Reverse.rf

$module Reverse;

$func Reverse e.source = e.result;

Reverse {
   /* empty */ = /* empty */;
   t.head e.tail = <Reverse e.tail> t.head;
};

$native Java {
   "public !String reverse (!String)" = Reverse;
   "public Object[] reverse (Object[])" = Reverse;
};
```

В программе Reverse.rf объявлена и определена Рефал-функция Reverse, обращающая выражение. Далее этой функции сопоставлены два Java-прототипа статических методов reverse: для обращения строки и массива.

Восклицательный знак перед типом указывает компилятору на необходимость применения одного из встроенных преобразований данных. Конкретное преобразование определяется по типу Java и по Рефал-формату аргументов и результата. В данном случае объект класса java.lang.String, наследника интерфейса java.lang.CharSequence, будет переведён в выражение, состоящее из *символов-литер*, и наоборот. Компилятор предоставляет встроенные преобразования между числовыми объектами (такими как java.lang.Integer или java.lang.Long) и рефальскими *символами-числами*, между java.lang.Boolean и *символами-словами* (аналогично примитивным данным типа boolean) и многие другие.

Определённые выше методы Reverse.reverse доступны для вызова из Java-программы:

```
...
public static bool isPalindrome (String phrase) {
   return phrase.equals(Reverse.reverse(phrase));
}
...
```

Таким образом, если для функции в Рефал-модуле заданы и Рефал-формат, и Java-прототип, то возможен вызов такой функции как из Рефала, так и из Java, вне

зависимости от того, на каком языке реализована эта функция. Заметим, что определённый таким образом Java-интерфейс для рефальской функции позволяет использовать её из любых языков программирования, интегрированных с платформой Java, также как обычный статический метод Java.

Заключение

Язык Рефал Плюс является удобным инструментом для решения задач, связанных с обработкой символьной информации. Его интеграция с платформой Java, выполненная в рамках проекта Refal+, предоставляет программисту возможность использовать как весь арсенал средств, реализованных для платформы Java, так и уникальные возможности языка Рефал. Одним из достоинств реализации является полная прозрачность при взаимодействии кода на Рефале с платформой Java.

Также реализовано расширение популярной для создания Java-проектов среды разработки Eclipse [2], которое предоставляет рабочее окружение для эффективного создания и поддержки Рефал-программ. Это даёт возможность легко интегрировать проекты, написанные на разных языках программирования.

Авторы выражают благодарность всем участникам Рефал-семинара за плодотворные идеи и обсуждения, Сергею Абрамову за активную поддержку и Аркадию Климову (автору Refal-Java [8]) за ценные комментарии по предварительному тексту данных тезисов.

Работа выполнена при поддержке гранта РФФИ 06-01-00574.

Литература

1. Refal+. http://wiki.botik.ru/Refaldevel/
2. Refal Plus development tools for Eclipse. http://rfp.botik.ru/eclipse/
3. В.Ф. Турчин "Метаязык для формального описания алгоритмических языков" // В сб.: Цифровая вычислительная техника и программирование, М.:Сов. Радио, 1966, с.116-124
4. V.F. Turchin "Refal-5, Programming Guide and Reference Manual" // New England Publishing Co., Holyoke, Massachusetts, 1989 (http://refal.ru/rf5_frm.htm)
5. В.Ф. Турчин "Рефал как язык для обработки xml-документов" // Журнал "Компьютерра" 25 от 02 июля 2001 года (http://refal.ru/xmlref_1.htm)
6. С.М. Абрамов, А.Ю. Орлов, Л.В. Парменова, С.М. Пономарева, А.Ф. Слепухин "Новый подход к реализации системы программирования Рефал Плюс" // Труды международной конференции "Программные системы: теория и приложения", г. Переславль-Залесский, май 2004, М.: Физматлит, Т. I, с.373-401
7. Р.Ф. Гурин, С.А. Романенко "Язык программирования Рефал Плюс" // Переславль-Залесский: "Университет города Переславля" им. А.К. Айламазяна, 2006
8. Refal-6 - Java. http://www.refal.ru/~arklimov/refal6/refal-j.htm