統合開発環境 Eclipse のスクリプト言語用ライブラリの開発

—Eclipse 上の作業を自動化するために—

1. 背景

近年、Java 開発者を中心に、統合開発環境 Eclipse が急速に普及してきている。Eclipse は、プラグイン機構によって自分自身を拡張できることを特徴とする柔軟な開発環境である。Eclipse のユーザは、Eclipse に足りない機能があれば、それをプラグインとして開発することで、Eclipse の機能を拡張することができる。プラグイン機構による拡張性は非常に高く、Eclipse に標準で付属する Java 開発環境である JDT（図 1）すら Eclipse のプラグインとして提供されている程である。また、サードパーティによって多数のプラグインが開発されており、その数は 1000 を超える。

しかし、Eclipse のプラグインを開発するには、プラグインの構成定義ファイルの記述、プラグインのデバッグ、プラグインのインストールといった多くの手順を踏む必要があり、普通の Java プログラムの開発と比べて面倒である。また、プラグインのプログラムは Java 言語で記述する必要があるため、プログラムの記述量は多くなりがちである。

このため、Unix のシェルスクリプトのように Eclipse 上の作業をプログラムによって自動化したい場合には、プラグインは向いていない。というのは、プラグインは記述にかかるコストが高いため、そのような作業を行うプラグインを作るコストが手作業で行うコストよりも高くなってしまう場合が多いのである。

2. 目的

以上の問題点を解決するために、本プロジェクトでは、（1）Eclipse 上の作業をプログラミングして、その場ですぐに実行することができるプログラム環境を提供すること、（2）Eclipse が提供しているプラグイン開発用ライブラリを、スクリプト言語 Groovy から呼び出すための高レベルなライブラリを提供することで、Eclipse のユーザが、Eclipse 上で Unix のシェルスクリプトのような感覚で作業をプログラミングできるようにすることを目的とする。

3. 開発の内容

3.1. リファクタリング機能の開発

Eclipse が提供している Java プログラムのリファクタリング機能を、ユーザプログラムから簡単に呼び出せるようにするためのライブラリを開発した。Eclipse が提供している Java プログラムのリファクタリング機能は、主にプラグインの一部として組み込むことを想定しているため、リファクタリング作業を自動化するには、冗長で扱いにくい。そこで、この問題を解決するために、Eclipse が提供している Java プログラムのリファクタリング機能を、より簡単に使えるようなラッパーライブラリを開発した。このライブラリを使うことで、プロジェクト名、クラス名、メソッド名などの単位での名前変更を、簡単な Groovy スクリプトを書くだけで行えるようになる。例えば、指定したパッケージにある全ての public な Java のクラスに、指定したプレフィクスを付加するという作業は、図 1 のような短い Groovy スクリプトを書くだけで行うことができた。このスクリプトを実行すると、いくつかのダイアログが表示され、それにプロ

ジェクト名やパッケージ名などを入力するだけで、指定したパッケージにある全ての public な Java のクラスに、プレフィクスを付加するという作業を行うことができる。

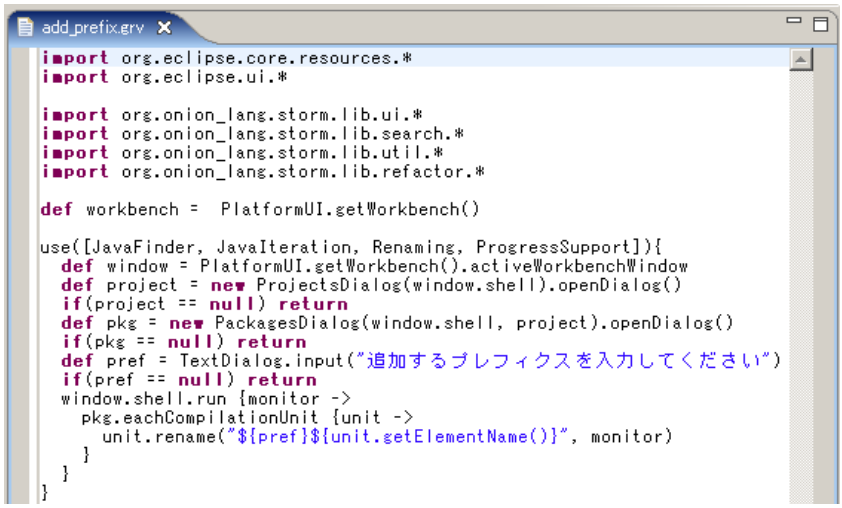

図 1

3.2. ファイル検索機能の開発

Eclipse のプロジェクト中にあるファイルを、ユーザプログラムから簡単に検索できるようにするためのライブラリを開発した。このライブラリを使うことで、プログラムを使ったファイルの検索作業を、簡単な Groovy スクリプトを書くだけで行えるようになる。例えば、図2のような Groovy スクリプトを書くことで、プロジェクト onion の build ディレクトリ中（子ディレクトリも含む）にある全てのクラスファイル（拡張子が.class のファイル）を削除することができる。

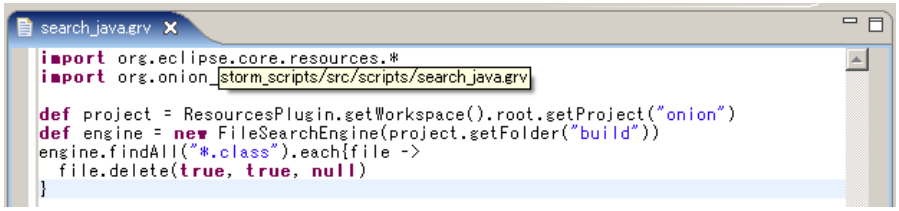

図 2

3.3. 構文解析機能の開発

Eclipse が提供している Java プログラムの抽象構文木を組み立てるための機能を、ユーザプログラムから簡単に呼び出すためのライブラリを作成した。Eclipse が提供している Java プログラムの抽象構文木を組み立てるためのライブラリは、冗長であり使いづらい。そこで、ユーザプログラムから簡単に Java プログラムの抽象構文木を組み立てることができるライブラリを開発した。Groovy には、任意の木構造を簡単に組み立てることができるようにするための、Groovy Markup と呼ばれる枠組みがあるため、これを利用してユーザが簡単に Java プログラムの抽象構文木を組み立てられるようなライブラリを開発した。このライブラリを使うと、簡単なプログラムで、Java の抽象構文木を組み立てられるようになる。例えば、図3のような Groovy スクリプトを書くことで、Hello World!を出力するクラスを抽象構文

木から組み立てることができる。

```
make_hello_world.grv
import org.onion_lang.storm.lib.ui.*
import org.onion_lang.storm.lib.parser.*
import org.eclipse.jdt.core.dom.*
import org.eclipse.core.resources.*
import java.io.*

def className = TextDialog.input("生成するクラス名を入力してください。")

if(className == null) return

def builder = new JavaASTBuilder(AST.newAST(AST.JLS3))
def unit = builder.unit {
  import_("java.util.*")
  type(className, access:["public"], extends_:"java.lang.Object"){
    method(
      access:["public", "static"],
      type:"void", name:"main", args:[["args", "String[]"]]){
      block("""
        System.out.println("Hello, World!");
      """)
    }
  }
}

def root = ResourcesPlugin.getWorkspace().root
def file = root.getProject("onion").getFile("src/${className}.java")
def input = new ByteArrayInputStream(unit.toString().getBytes())
file.create(input, true, null)
```

図 3

3.4. プラグイン機能の開発

Groovy スクリプトを実行して、Eclipse の機能を呼び出すためのプログラム環境をプラグインとして開発した。プラグインの場合、開発したプラグインを動作させるためには、プラグインをインストールして Eclipse を再起動する必要があるが、このプログラム環境を使うことで、Eclipse を再起動することなく、Eclipse の機能を呼び出すための Groovy スクリプトを実行することができる。Groovy スクリプトの実行は、図4のような画面で行うことができる。この設定画面で、スクリプトがあるプロジェクトと、実行するスクリプトのファイル名を指定して、「実行」ボタンを押すと、スクリプトが実行される。

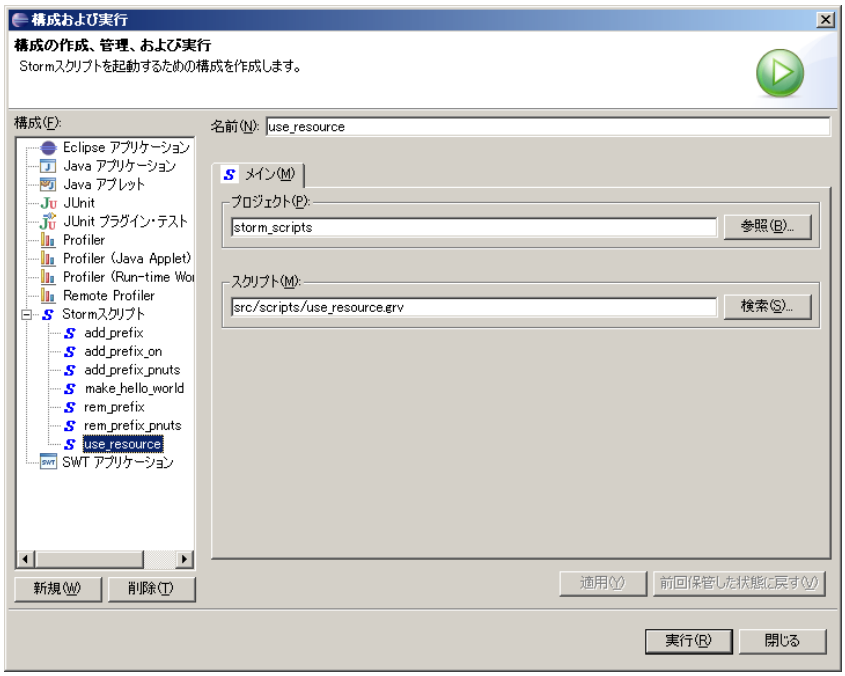

図 4

4. 従来の技術(または機能)との相違

本プロジェクトの開発成果を使用することで、プラグインを作成するのに比べて、プログラムを書いてEclipse上での様々な作業を自動化することがより簡単にできるようになる。例えば、プロジェクト上へのソースコードの自動生成や、プロジェクト上のファイル操作といった作業を、その場で簡単なプログラムを書いて実行するだけで行えるようになる。Eclipseでは、このような作業は従来ならプラグインとして記述するしかなかった。しかし、1章で述べたように、プラグインを作成するためには面倒な手順が必要なため、プログラムによる作業の自動化には向かない。

5. 期待される効果

本プロジェクトの開発成果によって、Eclipse上での作業を、プログラムを使って自動化することが容易になり、Eclipseを使ったソフトウェア開発の生産性が向上することが期待できる。また、それにとどまらず、EclipseプラグインをJava以外のプログラミング言語やスクリプト言語で開発することが盛んになることも期待できる。

6. 普及(または活用)の見通し

本プロジェクトの開発成果の利用者としては、元々EmacsやVimなどのエディタ上でプログラムを書いていたが、最近になってEclipseに移行してきたユーザを想定している。Emacsなどのスクリプト言語による拡張が可能なエディタを使っていたユーザからは、EclipseなどのIDEに対して、マクロ機能が無いことを批判する声が聞かれることがしばしばある。そのため、このような機能の需要が存在すると思われる。また、Eclipseをかなり使いこなしているユーザで、多くの作業がGUI上からしか行えないことを不満に思っているユーザなども利用者として想定している。

7. 開発者名(所属)

水島宏太(筑波大学大学院　システム情報工学研究科1年)
(参考)開発者URL:http://www.coins.tsukuba.ac.jp/~i021216/diary/