SciTE for Simple Java Projects on Linux

Adrian P. Robson
adrian.robson@nepsweb.co.uk

10th June, 2011

This article gives a SciTE local properties file for Java programming on Ubuntu platforms. It improves upon the default commands by putting class files in a separate folder, adding a clean command, and changing some of the default settings.

Contents

1 Introduction

The SciTE programming editor works well for most Java projects, and it is multi-platform. It can be obtained from

`http://www.scintilla.org/SciTE.html`

SciTE is a little unusual, in that most of its configuration is done using manually edited *properties files.* Here we provide a local properties file that changes SciTE's defaults to make Java programming a little easier. It changes tab and indent settings; changes the output panel orientation; selects a non-proportional font; and modifies the tools menu. It is suitable for Java projects where *all the source code is stored in a single folder.*

2 Setting Up a Project

To set up the folders and files for a project do the following:

1. Create a root folder for the project
2. Create a `SciTE.properties` file from §7, or obtain the file from `nepsweb.co.uk`, and put it in the project's root folder.

   You also might want to modify your user properties as described in §5.
3. Create a sub-folder called `classes` in the root folder, which is where all the programs class files will be put by the compiler.
4. Put any existing source files in the root folder.

3 Using SciTE with Java

Once the project folders and local properties file are created, the SciTe editor can used to edit Java source code files that are in the project's root folder. The editor has syntax highlighting, brace matching and code folding.

To compile and execute the program, the following tools menu commands can be used

Compile: Compiles the current Java file.

Build: Compiles all the Java files in the project,

Go: Executed the program provided the current file has a main method.

Clean: Delete all the class files.

Compilation errors are displayed the output pane, and a double mouse click will take you to the offending line of code.

3.1 Problems with Command Line Input

Java command line input does not work with SciTE on Linux ...

Java programs launched with the tools menu go command cannot use their `System.in` stream. None of the various read methods work as expected. The input stream always returns End-of-File, implying that the stream is always empty. However, the output streams of `System.out` successfully print to ScTE's output pane. There is no SciTE documentation relating to this problem.

A 'work around' is to open a terminal window, go to the project folder and run the program with

```
java -classpath classes MyProgram
```

where `MyProgram` is the name of the class with the main method.

Being unable to use command line input is not important for most Java applications, but it unfortunately limits SciTE's usefulness as a teaching tool.

4 Local Properties

The properties file is given in §7. There are some general settings that modify the editors display and behaviour; and some tools menu commands.

4.1 General Settings

These are some useful general settings that can be used for all programming languages.

The editors automatic indentation and tab width can be set to three spaces, and the tabs converted to spaces with:

```
tabsize=3
indent.size=3
use.tabs=0
```

This is a personal preference, but converting tabs to spaces gives consistent display and printing.

Finally the editor's font can be made permanently non-proportional with the rather verbose:

```
font.base=$(font.monospace)
font.small=$(font.monospace)
font.comment=$(font.monospace)
font.text=$(font.monospace)
font.text.comment=$(font.monospace)
font.embedded.base=$(font.monospace)
font.embedded.comment=$(font.monospace)
font.vbs=$(font.monospace)
```

This affects *all* files, and the option menu's 'use monospaced font' toggle stops working.

4.2 Tools Menu

These tools are similar to the SciTE Java defaults, but class management is improved by putting all class files in a separate folder, and adding a clean command.

4.2.1 Standard Commands

The following replace SciTE's standard Java compile, build and go commands:

```
1 command.compile.*.java=javac -d classes\
2  -classpath classes $(FileNameExt)

3 command.build.*.java=javac -d classes\
4  -classpath classes *.java

5 command.go.*.java=java -classpath classes $(FileName)
```

The command on lines 1 and 2 compiles just the current file. The build command on lines 3 and 4 compiles *all* Java files in the root folder. Finally, the go command on line 5 executes the program if the current file is a class with a `main` method.

4.2.2 Clean

This adds an item to the tools menu that deletes all files in the `classes` folder.

```
1 delcommand=rm
2 file.patterns.source=*.java
3 command.name.0.$(file.patterns.source)=Clean
4 command.0.$(file.patterns.source)=$(delcommand) classes/*
```

The delete command used here is specific to Ubuntu, but it can be changes by modifying line 1.

5 User Properties

Setting the output pane orientation cannot be done in a local properties file, and has to be done in user or global properties. So the following should be put in the user properties to permanently display the pane horizontally, and clear it each time a tools menu command is used.

```
split.vertical=0
clear.before.execute=1
```

A horizontal pane displays compiler errors better than the default vertical pane, but it can be temporarily changed to vertical from the options menu.

These settings are included in the sample properties file, and can by cut and pasted to user properties from there. However, if you mostly use SciTE for programming, and you want all of your Java project handled the way this article describes, you can simply *put all of the sample file in your user properties.* Then you will not have to bother with a `SciTE.properties` file when you create a Java project.

6 Limitations

This SciTE configuration works well for simple Java projects, but it has a number of limitations:

- Some of its commands are specific to Ubuntu
- All the source files must be in the project's root folder. Source code subfolders are not allowed.
- The name of the `classes` folder cannot be easily changed.
- Programs that use `Syatem.in` will not work correctly if they are launched with the go command.
- Custom packages cannot be used.
- Launching programs with command line arguments is not supported.
- Debugging is not supported.

7 SciTE Properties File

```
############################################
# Custom commands
############################################

# Java clean - delete all class files
#delcommand=cmd /c del # windows
delcommand=rm -r
file.patterns.source=*.java
command.name.0.$(file.patterns.source)=Clean
command.0.$(file.patterns.source)=$(delcommand) classes/*.class

############################################
# Standard commands
############################################

command.compile.*.java=javac -d classes $(FileNameExt)
command.build.*.java=javac -d classes *.java
command.go.*.java=java -classpath classes $(FileName)

############################################
# Common programming configuration
############################################

# set select shade
#selection.alpha=50

# Tab is 3 spaces
tabsize=3
indent.size=3
use.tabs=0

# use monospace font
font.base=$(font.monospace)
font.small=$(font.monospace)
font.comment=$(font.monospace)
font.text=$(font.monospace)
font.text.comment=$(font.monospace)
font.embedded.base=$(font.monospace)
font.embedded.comment=$(font.monospace)
font.vbs=$(font.monospace)

############################################
# For user properties
############################################

# horizontal split
split.vertical=0
clear.before.execute=1
```