Java Programming with SciTE for Simple Projects on Windows Platforms

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

16th May, 2018

Contents

1 Introduction **1**

2 Property Files **2**

3 Setting Up Simple Projects **2**
3.1 Stand-Alone Local Property File 3
3.2 User and Local Property Files 3
3.3 User and Directory Property Files 3
3.4 Adding a JAR Tool 4

4 Property Files and Code **5**
4.1 Stand Alone Local Property File 5
4.2 User Property File 5
4.3 Local or Directory Property File 6
4.4 Adding Build JAR Tool 6

1 Introduction

The SciTE is a simple text editor with facilities for building and executing programs. It can be used on Windows, Mac and Unix platforms. Here we show how SciTE (4.0.3) can be configured on Windows to

- save all buffers before build,
- have tab as three hard spaces,
- use the default mono-space font,
- have a horizontal output pane and
- use a sub-folder for all byte code files (`.class`).

SciTE is a little unusual, in that most of its configuration is done using manually edited *property files*. These can be stored with a project's source

code files (`.java`), which allows us to tailor the editor's behaviour precisely to a project's needs.

In this article, we describe how to set up a simple Java project that uses SciTE as it development tool on a Windows computer. A *simple project* has all its source code in a single folder. Complex projects that have code sub-folders are discussed in a separate report []. We also describe how a build JAR tool can be added to SciTE's tools menu.

The SciTE program and some documentation can be obtained from

`www.scintilla.org/SciTE.html`

and there is an an interest group at

`groups.google.com/forum/#!forum/scite-interest`

2 Property Files

SciTE has four types of property files:

Local property file called `SciTE.properties` that can be in the same directory as the file being edited. If present, it applies to only to the files in that folder.

Directory property file called `SciTEDirectory.properties` that can in the same folder as as the file being edited or in a parent folder. If present, it applies to all the files in the folder and in its sub-folder hierarchy.

User property file called `SciTEUser.properties` on Windows. It is stored in the user's account/root folder, and it applies to any of the user's files being edited with SciTE.

Global property file called `SciTEGlobal.properties`. This is provided as part of SciTE's installation, and it applies to any file in the computer's file system being edited with SciTE. *It is normally best if the global properties are not changed.*

There are also various language specific property files[1], but like global properties these should not be altered.

The four types of property files given above, override each other as follows, left to right:

$$\text{local} \xrightarrow{overrides} \text{directory} \xrightarrow{overrides} \text{user} \xrightarrow{overrides} \text{global}$$

3 Setting Up Simple Projects

Simple projects are those that store *all* source code files (`.java`) in their root folder. They are mostly easy to setup. There are three methods described. The author's preference is for a user and local property files combination, which is described in §3.2 below. A method for adding a JAR tool is also described.

[1] The default Java settings are in `cpp.properties`.

3.1 Stand-Alone Local Property File

This is the simplest method. All properties are in a single file. *The properties apply to only the files in the project's root folder.* The setup procedure is:

1. Make a suitably named root folder for the project.
2. Make a sub-folder called `classes` in the project's root folder.
3. Create a local property file called `SciTE.properties` in the project's root folder.

   Copy its contents from §4.1, or download the complete file from the `nepsweb.co.uk` site.
4. Start using SciTE to edit Java code files in the project's root folder. Then compiling and running programs using the editor's tools menu.

3.2 User and Local Property Files

With this method we have two property files. A common user file provides general settings; and a local property file gives settings specific to a particular Java project. *It has the advantage that the general user properties apply to any file being edited not just those in the project's root folder.* The setup procedure is:

1. If a user property file already exists and you like its settings, there is no need to replace it. Otherwise, create a file called `SciTEUser.properties` in your account/root folder.

   Get its contents from §4.2, or download a complete file from the `nepsweb.co.uk` site.

   This step is only needed once.
2. Make a suitably named folder for the project.
3. Make a sub-folder called `classes` in this project folder.
4. Create a local property file called `SciTE.properties` in the project folder.

   Copy its contents from §4.3, or download a complete file from the `nepsweb.co.uk` site.

   Compiler options can be changed by editing the file and uncommenting the `opt1`, `opt2`, `opt3` or opt4 variable assignments.
5. Start using SciTE to edit Java code files in the project's root folder. Then compiling and running programs using the editor's tools menu.

3.3 User and Directory Property Files

This method requires all Java project folders to be in or below a common root folder, which we shall call the *Java root folder.*

We use a common user property file to provide general settings for all edited files; and a common directory property file that gives Java specific settings for

files in the Java root folder and its sub-folder tree. Furthermore, some individual projects can have local property files if they need different properties.

It is the most versatile of the simple project methods. But it is more complicated, and it might not be worth the effort compared with just using user and local property files (§3.2). The setup procedure is:

1. If a user property file already exists and you like its settings, there is no need to replace it. But it must have directory property files enabled with

   `properties.directory.enable=1`

   Otherwise, create a file called `SciTEUser.properties` in your account's root folder.

   Get its contents from §4.2, or download a complete file from the `nepsweb.co.uk` site.

   This step is only needed once.

2. Make a suitably named Java root folder that will hold your Java project folders.

3. Create a directory property file called `SciTEDirectory.properties` in the Java root folder.

   Get its contents from §4.3, or download a complete file from the `nepsweb.co.uk` site.

4. For each Java project:

   (a) Make a suitably named project folder in the Java root folder. Java project folders can be grouped in parent folders an long as these are in the Java root folder's sub-folder tree..

   (b) Make a sub-folder called `classes` in the project folder.

   (c) Start using SciTE to edit Java code files in the project's root folder. Then compiling and running programs using the editor's tools menu.

3.4 Adding a JAR Tool

In all of the methods described above, a build JAR tool can be added. This will create a JAR file containing the contents of the project's `classes` folder package in such a way the the whole program can be shipped as a single file and executed with

`java -jar` *jarfile*

For a stand-alone local property file (§3.1) or user with local property files (§3.2), add the code in §4.4 to the project's `SciTE.properties` file; for user with directory files (§3.3), add it to the shared `SciTEDirectory.properties` file. The code can be alternatively downloaded from the `nepsweb.co.uk` site.

Once in place the new properties *must be edited*. The variables `jarname` and `jarmain` must also be given correct values. The variable `jarmain` is the name of the JAR file to be built; and `jarmain` is the name of a class with a main method that is the entry point to the program. For example:

```
jarname=myprog
jarmain=MainClass
```

4 Property Files and Code

Here is some property file code that can be used to create the files discussed in in §3 above.

4.1 Stand Alone Local Property File

These are the SciTE properties for a stand alone local property file that is discussed in §3.1 above. It can be downloaded from the `nepsweb.co.uk` site.

```
# SciTE.properties
############################################
# Java command overrides (Tools menu)
############################################

command.compile.*.java=javac -d classes $(FileNameExt)
command.build.*.java=javac -d classes *.java
command.go.*.java=java -classpath classes $(FileName)
command.clean.*.java=\
@echo deleting class folder &\
rd /s/q classes &\
@echo recreating class folder &\
md classes &\
@echo done

############################################
# Editor configuration
############################################

# Save all buffers for build
save.all.for.build=1

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

# use default 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)
```

4.2 User Property File

These are the SciTE properties for a user property file for use with the local and directory files given in §4.3 and discussed in §3.2 and §3.3.

```
# SciTEUser.properties

# Directory properties enabled
#
properties.directory.enable=1

# Output Pane is horizontal and cleared
#
```

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

# Save all buffers for build
#
save.all.for.build=1

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

# Always use default 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)
```

4.3 Local or Directory Property File

These are SciTE properties for a local or directory property files. This version is used with the user property file given in §4.2, as discussed in §3.2 and §3.3 above.

```
# SciTE.properties
#  or
# SciTEDirectory.properties
##########################################
# Java command overrides (Tools menu)
##########################################

command.compile.*.java=javac -d classes $(FileNameExt)
command.build.*.java=javac -d classes *.java
command.go.*.java=java -classpath classes $(FileName)
command.clean.*.java=\
@echo deleting class folder &\
rd /s/q classes &\
@echo recreating class folder &\
md classes &\
@echo done
```

4.4 Adding Build JAR Tool

This properties code will add a build JAR command to SciTE's tools menu. Its use is explained in §3.4 above.

```
##########################################################
# JAR build command
##########################################################
# Change as required to setup JAR tool, where:
#   jarname is the name of the JAR file
#   jarmain is the name of the class with the main method
#
jarname=helloworld
```

```
jarmain=HelloWorld
command.name.2.*.java=Build JAR
command.2.*.java=\
jar cfe $(jarname).jar $(jarmain) -C classes .
```