SWTBot improvements

Test recorder enhancements

outline

1 - Overview....................3
2 - Enhancements....................4
2.1 - The recorded code should compile....................4
2.1.1 - Do not use deprecated classes....................4
2.1.2 - Delegate to the proper bots....................4
2.1.3 - Context menus on tree items are not functional....................4
2.2 - Record double-clicks as validating....................5
2.3 - Differentiate between the visible elements....................5
2.4 - Tree item expansions should be recorded....................5
2.5 - The recorder should be run in the same environment as the tests....................5
2.6 - The recorder window should never be unfocusable....................6
2.7 - Allow users to delete recorded actions....................6
2.8 - Allow users to save the recorded test....................6
2.9 - Tie the recorder with the new SWTBot project wizard....................6
2.10 - The recorder should be stopped when its dialog is closed....................6
2.11 - Starting the recorder should remember user choices....................7
2.12 - Provide a « spy » tool....................7
3 - Delta between recorded and functional test....................9
3.1 - Recorded code....................10
3.2 - Functional test case....................10
3.3 - Diff....................11

1 - Overview

The SWTBot test recorder allows users to record the actions they do in the UI of a given Eclipse application into a code that can reproduce them. The purpose of this document is to provide means to enhance the recorder to reduce its limitations or make it a better user experience.

2 - Enhancements

By default, the recorder will assume we're extending the deprecated *SWTBotEclipseTestCase* which uses the deprecated *SWTEclipseBot*. This can only be understood when the users manually selects the « JDTDialog » instead of the « BasicDialog » though. With the basic dialog, the same assumption is made but the user gets no hint about it.

We'll use the following recorded example as a sample for the following points :

```
1 import org.eclipse.swtbot.eclipse.finder.SWTBotEclipseTestCase;
2
3 public class RecordedTestCase extends SWTBotEclipseTestCase {
4         public void testCreateProject(){
5                 bot.viewByTitle("Package Explorer").show();
6                 bot.contextMenu("New").menu("Other...").click();
7                 bot.tree().getTreeItem("Java Project").select();
8                 bot.textWithLabel("&Project name:").setText("test");
9                 bot.textWithLabel("&Project name:").pressShortcut(SWT.SHIFT, '.');
10                bot.textWithLabel("&Project name:").setText("test.project");
11                bot.button("Finish").click();
12                bot.contextMenu("Properties").click();
13                bot.tree().getTreeItem("Java Build Path").select();
14                bot.tabItem("&Source").activate();
15                bot.tree().getTreeItem("test.project/src").getNode("Excluded: (None)").select();
16                bot.button("Cancel").click();
17        }
18 }
```

2.1 - The recorded code should compile

2.1.1 - Do not use deprecated classes

SWTBotEclipseTestCase is deprecated. Create an *SWTBotWorkbenchTestCase* overriding the basic *SWTBot* instance with an *SWTWorkbenchBot* instead.

2.1.2 - Delegate to the proper bots

« *contextMenu(String)* » is a method that exists neither within the old (*SWTEclipseBot*) nor within the new (*SWTWorkbenchBot*) APIs. The recorder should make sure to check where we're clicking and delegate to the correct bots. In this case, line 6 should read « *bot.tree().contextMenu(...)* » whilst line 12 cannot work.

2.1.3 - Context menus on tree items are not functional

Our recorded line 12 is our user trying to access the properties of his newly created project.

```
bot.contextMenu("Properties").click();
```

Will not compile.

```
bot.tree().contextMenu("Properties").click();
```

Will not work : no visible UI action taken by the bot with this line and no error message, so the user is left in the dark as to what's failing.

```
bot.tree().getTreeItem("test.project").contextMenu("Properties").click();
```

Will not work either, so it doesn't seem like it's a context issue.

In this particular instance, the issue can be worked around by using the global menu bar's items… But the user has to figure that out by himself with no help from either the recorder or the test itself since there are no error messages on this.

2.2 - Record double-clicks as validating

The recorder hasn't realized that the « Java Project » selection on line 7 was actually a double-click, thus leaving us on the « new » wizard instead of following through to the « new Java Project » wizard, failing our test on line 8.

2.3 - Differentiate between the visible elements

The recorder doesn't differentiate between visible elements when they cannot be selected by title/caption. The recorded line 15 will fail our test because of that since there are two « trees » visible in the properties page, and as it turns out it's the second one we've clicked on.

2.4 - Tree item expansions should be recorded

The recorder discards expansions of tree item as irrelevant, whereas some trees will require them. It is more harmful not to record it (and create a test that might fail while leaving the responsibility of fixing the failure to the user) than to record it even if it's useless to the test.

In our example, line 15 will fail since the node we're trying to select is not currently visible, the recorder having discarded the expansion of its parent node.

2.5 - The recorder should be run in the same environment as the tests

When starting the recorder (File > New > Other… > Test Recorder), the user can either « record on current eclipse instance » or « record on another application ». With this second choice, the user expects to be started in the same « pristine » environment as the tests themselves would be running (if we go to a test and « run as > SWTBot test »). This is not the case since the « run as » menu will start a new product (*org.eclipse.sdk.ide*) whilst the recorder will run in a trimmed down application (*org.eclipse.ui.ide.workbench*).

The most visible differences are that the recorder will never show the welcome screen, whereas running the tests always will, and the recorder starts in the « Resource » perspective by default whereas running tests will start in the Java perspective.

As a result, recorded tests will almost never run out of the box since the welcome screen will take the whole available space and all other views are minimized by default when we run the tests themselves. Likewise, the recorder will most likely show a switch perspective dialog when creating projects that the tests themselves won't. The recorder should add a « setup » method that will close the welcome page by default (or use a new SWTBot « fail-safe » API that will closeIfPresent()?).

2.6 - The recorder window should never be unfocusable

The recording window should always be focusable in one way or another. In our current use case, as soon as the user has selected « File > New > Other... », the new file wizard is opened, and it is modal. This prevents us from pausing the recorder altogether since we can no longer focus it until we've somehow finished with the new file wizard.

2.7 - Allow users to delete recorded actions

There might be unwanted, useless actions recorded in the skeleton, that the user might want to remove before copying the recorded stub in an actual test class for the sake of readability, without having to re-record everything without them.

This can happen because of the previous point (the user couldn't pause the recording and is thus recording more than he wants to), because of the actual action being recorded (see our lines 8 to 10 in the example above, wherein only line 10 is relevant and the two others can be removed) or because of miss-clicks during the recording.

2.8 - Allow users to save the recorded test

The recorder currently forces the user to copy its content in order to paste it in an actual test class. An action to actually save the test would be nice instead.

2.9 - Tie the recorder with the new SWTBot project wizard

Related to 2.8 above, users should be allowed to create a new SWTBot test project from the recorder or have the option of directly start recording when creating a new SWTBot project.

As it stands, in order for the recorded code to even have a chance to compile, the user has to :

- Create a new SWTBot project
- Create a new class
  - Making sure the class name matches the name the user has selected when he started the recorder if he chose the « JDT Dialog »
- Copy/paste the recorded code in that new class, overriding everything

If the user had started from a regular project, he will also have to manually add the dependencies towards the proper plugins (e.g. *org.eclipse.swtbot.go*).

2.10 - The recorder should be stopped when its dialog is closed

- File > New > Other… > Run Test Recorder
- Select the current eclipse instance as AUT and « Finish »
- « start » the recording
- Close the recorder dialog.

Every action after this will be logged as an error in the error log, with additional « error » modal dialogs showing if the error log is currently visible.

Even without counting the errors, since the recorder is still running, starting another recorder window will log the actions twice and will not allow the user to « stop » the previously closed recorder. Restarting eclipse is the only option to recover from this state.

2.11 - Starting the recorder should remember user choices

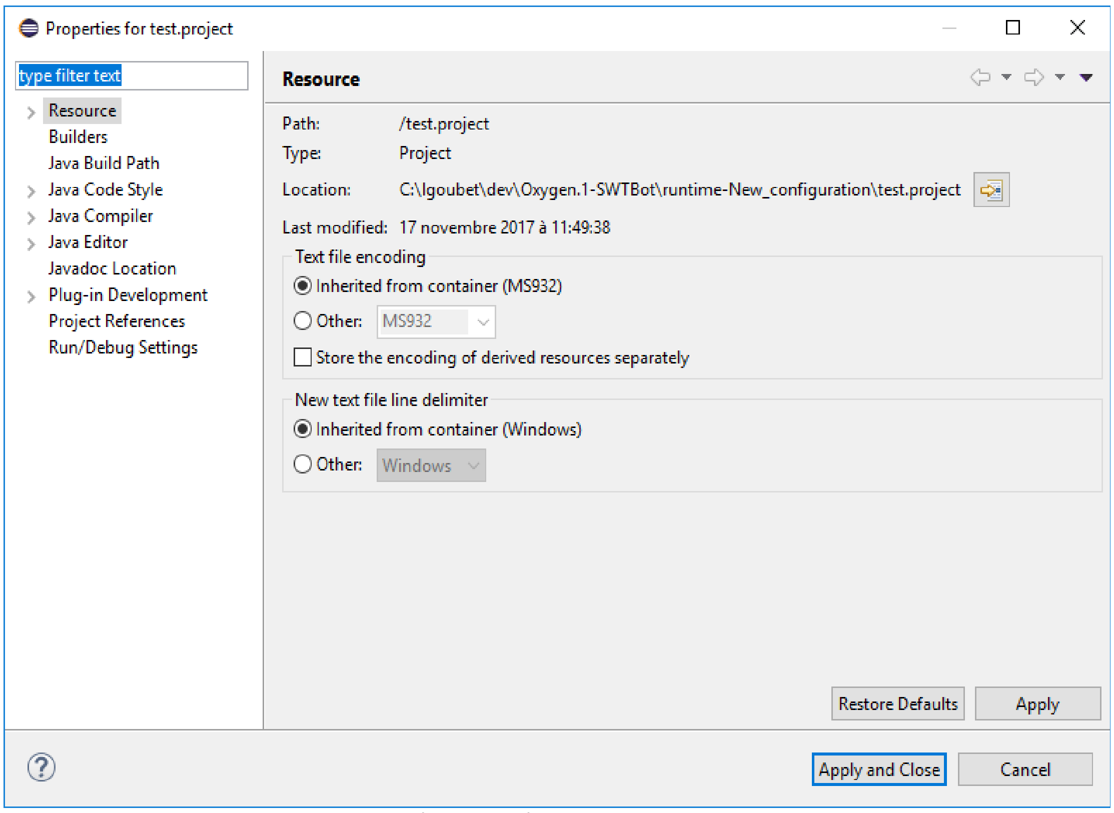
When the user has selected the « JDT dialog » and to « record in a new instance », it is more likely that these choices will also apply on next launch than he'd like the default « Basic Dialog » recording « on the current instance » instead.

2.12 - Provide a « spy » tool

The recorder is mostly there to provide a stub that the user can augment with assertions to check that the UI shows what he expects it to. However, it is very difficult to properly navigate to the widgets through the SWTBot APIs.

The recorder should provide some kind of a « spy » that would allow users to visualize the widgets (e.g. when the mouse hovers on them) that would allow selection, so that a skeleton of the code needed to make assertions on that given widget is generated for the user.

This is especially true on widgets such as labels or read-only text fields. For example, if the user has the following UI and wishes to assert that the « Type » of the current resource is indeed « Project », it would be more than interesting to be able to just « click » on the desired label and have the recorder generate the required stub to access that widget so that he can add the assertions on it.



Resource

Path: /test.project
Type: Project
Location: C:\lgoubet\dev\Oxygen.1-SWTBot\runtime-New_configuration\test.project
Last modified: 17 novembre 2017 à 11:49:38

Text file encoding
[✓ Inherited from container (MS932)]
Other: MS932
☐ Store the encoding of derived resources separately

New text file line delimiter
[✓ Inherited from container (Windows)]
Other: Windows

Here, clicking on the desired Label ("Project") would generate either :

```
bot.text(2)
```

or the more specific

```
bot.text("Project")
```

since a content for that text widget is available. It would also be possible with this to select the Composite containing our desired widgets and navigate through afterwards. The basic requirement is to be able to have the skeleton on how to access such or such widget on a given UI.

3 - Delta between recorded and functional test

We wish to run a simple test case that will create a new project in the package explorer, then open its properties and extend the tree showing its build path there. There will be no assertions in this test, just the bot part.

3.1 - Recorded code

```
public class RecordedTestCase extends SWTBotEclipseTestCase {
        public void testCreateProject(){
                bot.viewByTitle("Package Explorer").show();
                bot.contextMenu("New").menu("Other...").click();
                bot.tree().getTreeItem("Java Project").select();
                bot.textWithLabel("&Project name:").setText("test");
                bot.textWithLabel("&Project name:").pressShortcut(SWT.SHIFT, '.');
                bot.textWithLabel("&Project name:").setText("test.project");
                bot.button("Finish").click();
                bot.contextMenu("Properties").click();
                bot.tree().getTreeItem("Java Build Path").select();
                bot.tabItem("&Source").activate();
                bot.tree().getTreeItem("test.project/src").getNode("Excluded: (None)").select();
                bot.button("Cancel").click();
        }
}
```

3.2 - Functional test case

```
public class RecordedTestCase extends SWTBotEclipseTestCase {
        public void testCreateProject(){
                bot.viewByTitle("Welcome").close();
                bot.viewByTitle("Package Explorer").show();
                bot.tree().contextMenu("New").menu("Other...").click();
                bot.tree().getTreeItem("Java Project").select();
                bot.button("Next >").click();
                bot.textWithLabel("&Project name:").setText("test.project");
                bot.button("Finish").click();
                //bot.tree().getTreeItem("test.project").contextMenu("Properties").click();
                bot.tree().getTreeItem("test.project").select();
                bot.menu("File").menu("Properties").click();
                bot.tree().getTreeItem("Java Build Path").select();
                bot.tabItem("&Source").activate();
                bot.tree(1).getTreeItem("test.project/src").expand();
                bot.tree(1).getTreeItem("test.project/src").getNode("Excluded: (None)").select();
                bot.button("Cancel").click();
        }
}
```

3.3 - Diff

```
public class RecordedTestCase extends SWTBotEclipseTestCase {
    public void testCreateProject(){
+           bot.viewByTitle("Welcome").close();
            bot.viewByTitle("Package Explorer").show();
-           bot.contextMenu("New").menu("Other...").click();
+           bot.tree().contextMenu("New").menu("Other...").click();
            bot.tree().getTreeItem("Java Project").select();
-           bot.textWithLabel("&Project name:").setText("test");
-           bot.textWithLabel("&Project name:").pressShortcut(SWT.SHIFT, '.');
+           bot.button("Next >").click();
            bot.textWithLabel("&Project name:").setText("test.project");
            bot.button("Finish").click();
-           bot.contextMenu("Properties").click();
+           //bot.tree().getTreeItem("test.project").contextMenu("Properties").click();
+           bot.tree().getTreeItem("test.project").select();
+           bot.menu("File").menu("Properties").click();
            bot.tree().getTreeItem("Java Build Path").select();
            bot.tabItem("&Source").activate();
-           bot.tree().getTreeItem("test.project/src").getNode("Excluded: (None)").select();
+           bot.tree(1).getTreeItem("test.project/src").expand();
+           bot.tree(1).getTreeItem("test.project/src").getNode("Excluded: (None)").select();
            bot.button("Cancel").click();
    }
}
```