Testing

Running tests in Eclipse
JUnit best practice
Mocking objects
SwtBot

Running unit tests in Eclipse

see wiki to setup eclipse test fragments : http://talendforge.org/wiki/doku.php?id=dev:rcp_unit_test

JUnit 4 best practice

JUnit annotations

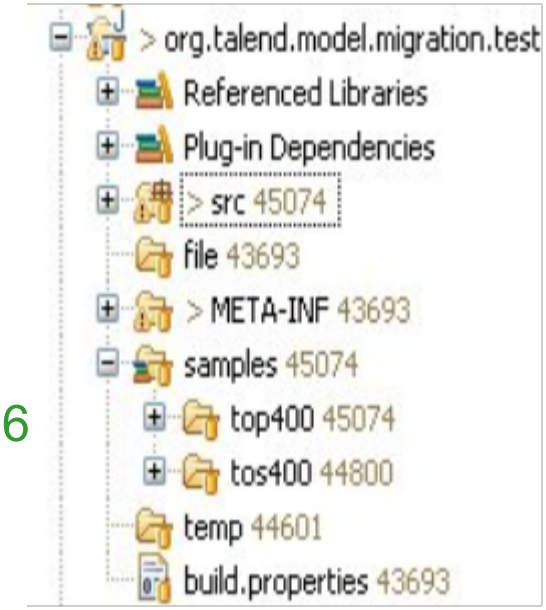
```
public class CalculatorTest {
private Calculator calculator;

  @Test
public void canAdd() {
System.out.println("canAdd");
    assertEquals(5, calculator.add(2, 3));
}


  @Test(expected = DivByZeroException.class)
public void shouldNotDivByZero() {
System.out.println("shouldNotDivByZero");
    calculator.divide(100, 0);
}

  @Ignore("still under development")
  @Test
public void canMultiply() {
System.out.println("canMultiply -- this will not print!");
    assertEquals(15, calculator.multiply(3, 5));
}
```

```
  @BeforeClass
public static void beforeClass(){
System.out.println("beforeClass");
}

  @Before
public void setupCalculator() {
System.out.println("setupCalculator");
    calculator = new Calculator();
}

  @After
public void shutdownCalculator() {
System.out.println("shutdownCalculator");
    calculator.shutdown();
}

  @AfterClass
public static void afterClass(){
System.out.println("afterClass");
}
}
```

JUnit 4 best practice

Don't assume the order in which the test cases can run
Test case should not depend on other test cases
So always cleanup everything after the test cases
Do not load data from hard coded location on file system :

```
public void setUp () {
FileInputStream fis= new FileInputStream("C:\\TestData\\det1.dat");
}
```



Place and load data in the test bundle :

```
URL url = getClass().getResource("/top400/");
// bundleresource://875.fwk22949069:4/top400/
URL unescapedURL = FileLocator.toFileURL(url);//bug 145096
// file:/E:sv n/org.talend.model.migration.test/samples/top400/
URI escapedURI = new URI(unescapedURL .getProtocol(),
unescapedURL .getPath(), unescapedURL .getQuery());
//file:/E:sv%20n/org.talend.model.migration.test/samples/top400/
File sampleFolder = URIUtil.toFile(escapedURI );
// E:\sv n\org.talend.model.migration.test\samples\top400
```

Note that *Sample* folder was added to the plug-in Class-path and the initial forward slash to look for resource in the classpath and not relative to the current class

JUnit 4 best practice

Use Throws declaration to handle unexpected exceptions

```
public void exampleTest () {
try {
// do some test
} catch (SomeException e) {
    fail ("Caught SomeException");
}
}
```

```
public void exampleTest ()
throws SomeApplicationException {
// do some test
}
```

Other pratice are summarised here : http://geosoft.no/development/unittesting.html

Mocking objects

mock objects are simulated objects that mimic the behavior of real objects in controlled ways.
Mock Objects for Verifications :

```
import static org.mockito.Mockito.*;

//mock creation
List mockedList = mock(List.class);

//using mock object
  mockedList.add("one");
  mockedList.clear();

//verification
 verify(mockedList).add("one");
 verify(mockedList).clear();
```

Mocking objects

Mock Objects for Stubbing :

```
//You can mock concrete classes, not only interfaces
LinkedList mockedList = mock(LinkedList.class);

//stubbing
when(mockedList.get(0)).thenReturn("first");
when(mockedList.get(1)).thenThrow(new RuntimeException());

//following prints "first"
System.out.println(mockedList.get(0));

//following throws runtime exception
System.out.println(mockedList.get(1));

//following prints "null" because get(999) was not stubbed
System.out.println(mockedList.get(999));
```

Mocking objects

One plugin in svn repository :
http://talendforge.org/svn/commons/development/org.mokito

SwtBot

Framework to develop Graphical User Interface test cases.

Naming the test plugins

org.talend.whatever_you_want.swtbot.test

```
/**
* Initialization
*/
bot = new SWTWorkbenchBot();
bot.waitUntil(Conditions.shellIsActive("Generation Engine Initialization in progress..."));
shell = bot.shell("Generation Engine Initialization in progress...");
shell.activate();
bot.waitUntil(Conditions.shellCloses(shell), 60000 * 5);
```

SwtBot

Every widget can be reached from the bot instance
Please do not use the sleep() method whenever possible.

A plugin is availalble in svn *http://talendforge.org/svn/commons/development/org.talend.swtbot.commons*
This plugin provides all the dependencies for SwtBot and a class that should be extended by all swt bot test case for TOS.
It provides an initialisation method to wait for the end of the generation engine and some helper methods.

SwtBot

About finding Widget.
Most of the widget should be found using their textual name.
Do not use message.properties directly just use the displayed name.

1. It is going to be easier to create test case : no need to look for the key, no need to depend on other plugins and therefore more independence
2. So it is faster to create.
3. Test cases are more readable.
4. They are less robust to change in the naming that is why we should not be using internationalised properties but the original “message.properties” file; therefore tests should be launched using some unsupported language so that default properties files are used. Let’s us pick the “ta” language for talend and “Tamil” which I doubt we will support.

SwtBot

Job creation test case

```
@RunWith(SWTBotJunit4ClassRunner.class)
public class SwtBot2Test extends TalendSwtBotForTos {
   @Test
public void testJobCreation() {//Create a new job
      SWTBotView view = bot.viewByTitle("Repository");
      view.setFocus();
      SWTBotTree tree = new SWTBotTree((Tree) bot.widget(WidgetOfType.widgetOfType
(Tree.class), view.getWidget()));
      tree.setFocus();
      tree.select("Job Designs").contextMenu("Create job").click();
      bot.waitUntil(Conditions.shellIsActive("New job"));
      SWTBotShell shell = bot.shell("New job");
      shell.activate();
      bot.textWithLabel("Name").setText("test02");
      bot.button("Finish").click();
      bot.waitUntil(Conditions.shellCloses(shell));
      SWTBotTree jobTreeNode = tree.select(JOB_NAME + " 0.1");
      Assert.assertNotNull(jobTreeNode);
}
```

SwtBot

Job creation test case with component design

```
SWTBotGefEditor gefEditor = bot.gefEditor("Job " + JOB_NAME + " 0.1");
gefEditor.activateTool("tRowGenerator").click(200, 200);
gefEditor.activateTool("tLogRow").click(400, 200);
//retreive the tRowGenerator component
SWTBotGefEditPart rowGen = getTalendComponentPart(gefEditor, "tRowGenerator_1");
Assert.assertNotNull(rowGen);
gefEditor.select(rowGen);
//create a link between rowgen and logRow
gefEditor.clickContextMenu("Main");
SWTBotGefEditPart logRow = getTalendComponentPart(gefEditor, "tLogRow_1");
Assert.assertNotNull(logRow);
gefEditor.click(logRow);
//check the link is created
SWTBotGefEditPart connPart = gefEditor.getEditPart("row1 (Main)");
Assert.assertNotNull(connPart);
```