Unit testing with JMUnit and NetBeans IDE

This article needs to be updated: If you found this article useful, please fix the problems below then delete the {{ArticleNeedsUpdate}} template from the article to remove this warning.

Reasons: hamishwillee (talk) (24 Jul 2013)
Instructions are not clear and it is not obvious what versions of Netbeans and JMUnit and what devices have been tested on. It is likely that this is very out of date.

Unit testing is common practice in professional software development. Especially in the Java™ Platform, Enterprise Edition (Java EE) space it's widely used to ensure stable high quality software.

While unit testing is a methodology and doesn't depend on a special framework it's a good idea to use a mature framework.

One of the frameworks available in the Java Platfrom, Micro Edition (Java ME) space is JMUnit.

Here I will give you a quick overview of the steps involved to use JMUnit and NetBeans 6.1 to easily write and run Java Platfrom, Micro Edition (Java ME) unit tests and how to create a release build without any test artifacts contained.

When you have created a Mobile Java Appication right click on your desired package node and choose "New / Empty JUnit Test".

Now create your Test-Class. It's a good idea to have a naming convention like "*Test" for test classes.

NetBeans 6.1 automatically adds a dependency on JMUnit4CLDC10. If you intend to use CLDC 1.1 you should remove that dependency and use JMUnit4CLDC11. Don't forget to change the import statement of the generated class.

Now you have a new test class consisting of a constructor and a method named "test(int testNumber)".

In the constructor there's a call to the super constructor. Here you have to give the total number of tests to run.

In the test method you should dispatch the call to the actual test method.

Example:

```
public class NewEmptyJMUnitTest extends TestCase {

    public NewEmptyJMUnitTest() {
        //The first parameter of inherited constructor is the number of test cases
        super(1,"NewEmptyJMUnitTest");
    }

    public void test(int testNumber) throws Throwable {
        switch(testNumber){
            case 0:
                testOne();
                break;
        }
    }

    private void testOne() throws Exception{
        // here we run our test, use assertXXX methods to check results
        String tmp = "Hello World";
        assertEquals(tmp, "Hello World!");
    }
}
```

Now it's time to run the test. Right click on the project node, choose "Properties". Choose "Application Descriptor".

Activate "MIDlets" and click "Add...". Choose the newly created test class and give it a good name indicating that it's the testrunner.

When you launch the application there are two MIDlets to choose from: Your original MIDlet and the Test-Runner-MIDlet.

If there was any error it's details are dumped to the console. For the above test the output should look like this:

```
Assert Equals failed.
Expected Hello World, but was Hello World!
jmunit.framework.cldc11.AssertionFailedException
        at jmunit.framework.cldc11.Assertion.fail(Assertion.java:1066)
        at jmunit.framework.cldc11.Assertion.fail(Assertion.java:1054)
        at jmunit.framework.cldc11.Assertion.assertEquals(Assertion.java:150)
        at jmunit.framework.cldc11.Assertion.assertEquals(Assertion.java:594)
        at hello.NewEmptyJMUnitTest.testOne(+9)
        at hello.NewEmptyJMUnitTest.test(NewEmptyJMUnitTest.java:26)
        at jmunit.framework.cldc11.TestCase.test(TestCase.java:65)
        at jmunit.framework.cldc11.Screen.run(Screen.java:157)
```

Now it's time to correct the error and add tests to the test class. Don't forget to add the method call to the "test(int)" method and raise the number of tests in the call to the super constructor.

When everything works fine and you are about to release your work you most likely don't want to ship the testing stuff with it.

To do this you can create an additional project configuration. Right click on the project node and choose "Properties".

Choose "Add configuration..." from the project configuration combo box. Choose a good name for the configuration. (e.g. "Release")

In the release configuration choose MIDlets. Uncheck the "Use values from "DefaultConfiguration"" and remove the test runner MIDlet.

Choose "Build/Libraries & Resources" and remove the JMUnit dependency.

Choose "Build/Sources Filtering" and check the "Exclude Test Souces". If there are still test resources left, uncheck them too.

You might wish to change other settings (e.g. obfuscation level) for the release, too.

Now you can build the release e.g. via NetBeans' batch build. You will find the release binaries in ".../dist/Release".