Creating JUnit test case

Let's use MagicDraw as an example describing the creation of JUnit test cases, as other modeling tools developed by No Magic Inc. use the same naming.

On this page

JUnit 3 test case

The JUnit test case should be created by extending the abstract *com.nomagic.magicdraw.tests.MagicDrawTestCase* class from Open API (see the following *MyTest* sample). The *MagicDrawTestCase* class starts MagicDraw automatically for each test case method defined in *MagicDrawTestCase* and performs the memory leaks test after each completed test case.

MagicDrawTestCase provides default and specific constructors for creating a test case instance with a specific custom name for the specific test case method. Test cases with specific names might be helpful when several instances of the same test case are created and analyzed.

The *MagicDrawTestCase* initialization and tear down should be implemented in overridden *setUpTest()* and *tearDownTest()* methods. The standard JUnit method *suite* might be used for preparing the tests suite with several instances of test cases which may use different test data.

```
    import com.nomagic.magicdraw.tests.MagicDrawTestCase;
    public class MyTest extends MagicDrawTestCase
    {
        public MyTest(String testMethodToRun, String testName)
        {
            super(testMethodToRun, testName);
        }

        @Override
        protected void setUpTest() throws Exception
        {
            super.setUpTest();
            //do setup here
        }

        @Override
        protected void tearDownTest() throws Exception
        {
            super.tearDownTest();
            //do tear down here
        }

        public void testSomething()
        {
              //implement the unit test here
        }

        public static Test suite() throws Exception
        {
            //you may create a test suite with several instances of the
test.
            TestSuite suite = new TestSuite();
            suite.addTest(new MyTest("testSomething", "MagicDraw Test
Sample"));
            suite.addTest(new MyTest("testSomething", "Another MagicDraw
Test Sample"));
            return suite;
        }
    }
```

MagicDrawTestCase also provides methods for opening, saving, and closing MagicDraw projects. It also performs the memory leak test after the test case has been completed and after the project has been closed. The memory leak test for the whole test case can be disabled using the *setMemoryTestReady (boolean)* method, while the *setSkipMemoryTest(boolean)* method can be used to disable the memory leaks test on closing the MagicDraw project.

The list of required MagicDraw plugins for the test case can be configured by overriding the *getRequiredPlugins()* method of *MagicDrawTestCase*. The overridden method implementation should return the list of required plugins IDs. Textual information about required plugins and their loading status can be obtained using *getRequiredPluginsDescription()* and *isRequiredPluginsLoaded()* methods.

Textual information about the test process can be logged using the Log4j logger. The test case logger for the TEST category can be accessed using the *getLogger()* method of *MagicDrawTestCase*. More information about the Log4j logger configuration and usage can be found at http://logging.apache.org/log4j/1.2/manual.html.

JUnit 4 test case

The JUnit test class should be annotaited to use *com.nomagic.magicdraw.tests.MagicDrawTestRunner* from Open API (see code sample bellow). The *MagicDrawTestRunner* class starts MagicDraw automatically and performs the memory leaks test after each completed test case.

```
@RunWith(MagicDrawTestRunner.class)
public class MyTest
{
    @Before
    public void beforeTestBegins()
    {
        //do setup here
    }

    @After
    public void afterTestDone()
    {
        //do tear down here
    }

    @Test
    public void testSomething()
    {
        //implement the unit test here
    }
}
```