TestExpert v0.1 Release Notes

# General Remarks

This is the first release of TestExpert. It is a newly, beta-release of the product.

# New Features

In fact everything is new here. ☺

# Improvements/Enhancements

# Performance Improvements/Enhancements

None at this time

## Other Improvements/Enhancements

None at this time

# Additional Support

jMockit and EasyMock are now fully supported. See roadmap for more in the future support for mocking frameworks.

# Bug Fixes

Although it is not entirely interesting a short bug list which **SHOULD BE FIXED** is mentioned below:

As of 2013, November 18th, there were 34 verified issues that have been resolved:

Sometimes an import of an innerclass contains an invalid $

A variable which is not in the Fixture file should be treated as a literal and NOT as a variable in the testclass

Although TestExpert supports jMockit some issues may rise

There might be some security issues when a class can not be opened by TestExpert

Array types in the in- and out variable are not fully compliant. You have to change the generated class yourselve, which is trivial (2 seconds)

This list can be found at: <<https://issues.apache.org/ooo/buglist.cgi?cmdtype=dorem&remaction=run&namedcmd=4.0.1RelaseBlockersResolved>>

You will need an OpenOffice Bugzilla login to view it.

(An [OpenOffice Bugzilla](https://issues.apache.org/ooo/) login will enable you to search for any bugs that may interest you.)

## Important bug fixes have been made in these areas:

* None at this time.

**The QE team has been tracking additional fixes as well. See their complete reports at:**

<http://wiki.services.openoffice.org/wiki/QA/Report/WeeklyReport>

# Known Issues

TestExpert is not fully compliant with the Google Web Toolkit (GWT).

Some classes are not automatically imported. A click on Ctrl-O (Eclipse)

Sometimes the type of String and int are confused. This can be fixed be changing the code in the generated test class. It will be fixed in the next release.

Jad should be installed on the developer’s pc.

RoadMap

Jad should not be a requirement.

More successcase per method (max. one at this moment)

More unsuccessful cases per method (zero at this moment)

The jMockit.jar should appear above or below the Junit 4.8.2 jar. // rloman: dit nog even uitzoeken straks.

# Installation guide

Download and unzip the latest greatest source or binary from Github.

On your Java project add the TestExpert.jar from the dist directory to your classpath.

Also add the lib folder to the classpath.

Optional: install the content from the demo directory in your Eclipse environment.

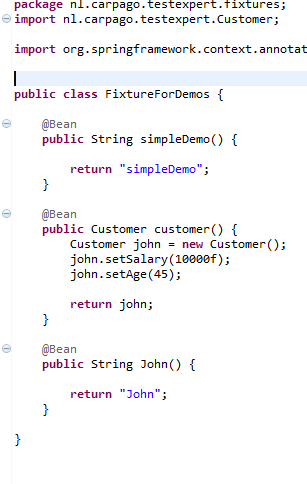
# Getting Started Guide

After completing the installation guide above TestExpert is ready for your first experience.

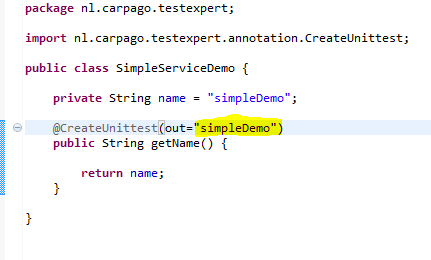
In a Java class with a method wherefore you would like to create some unittests open the desbetreffende method.

There a two ways to make a hint for TestExpert to create a unittest:

1. Use a literal: this is only possible for an int and a String
2. Use a variable: first create a so called Fixture file / class e.g.



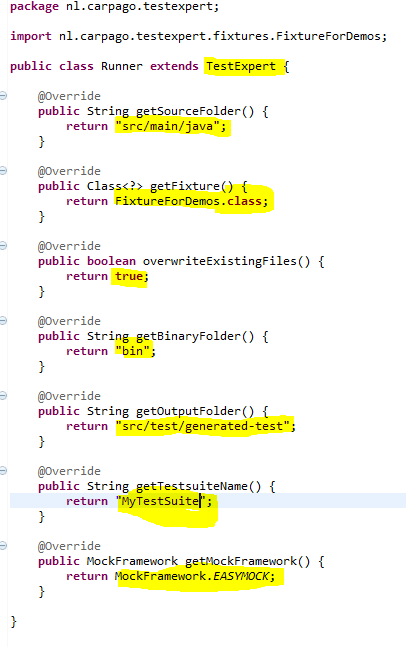
After creating a file like this you can use the method names as a variable in your class under test WITHOUT the parenthesis! So ‘simpleDemo()’ will be ‘simpleDemo’ and ‘customer’ will be ‘customer’



After added the @CreateUnittest annotations – the heart of TestExpert – it is time for the last step to generate your tests:

Create a class which extends from TestExpert. Let Eclipse or your favourite IDE implement the stubs for the needed template methods.

Implement the the template methods to your situation:



The methods of the overridden class contains several overriden methods which are explained below:

String getSourceFolder(): should return a String where the source files of your project are found. (TestExpert will use this a startpoint and will recursively traverse all subdirectory)

Class getFixture: should return a class which represents your Fixtures. See example above.

boolean overwriteExistingFiles: should return a boolean which indicates whether TestExpert should overwrite previously created testcases. In fact: when TestExpert has created them you may tweak and update them manually. Then you might return a false.

String getBinaryFolder(): should return the folder in which the compiled (.class) files exist. This is necessary for inspecting the JVM instructions for generating collaborating methods calls and such stuff.

String getOutputFolder(): the folder in which TestExpert will create the generated Testclasses. You should beforehand create that folder and add that folder to the sourcepath of your Java project.

String getTestsuiteName(): a suiteable name for the class which will also be generated after the creation of all your testclasses. It will contains calls to the by TestExpert generated Testclasses so you don’t have to call them all by hand in your Eclipse / IDE environment.

MockFramework getMockFramework(): should return an enum instance from the MockFramework enum. At this moment only EasyMock (thoroughly tested) and jMockit (good tested) are supported.

After implementing this class above, we are ready to launce this as a Junit unittest.

The testclasses will now be generated in the src/test/generated-test folder (the returned String in getOutputFolder).

Have fun!

Continue here: rloman: 2013, 11-18 d.d.

**Stay informed about Carpago’s TestExpert**

You are encouraged to subscribe to the Carpago’s TestExpert announcement mailing list, to be receive important notifications such as product updates and security patches.  To subscribe you can send an email to:  [ooo-announce-subscribe@incubator.apache.org](mailto:ooo-announce-subscribe@incubator.apache.org).

You can also follow the project on [Twitter](https://twitter.com/#!/apacheoo), [Facebook](http://www.facebook.com/ApacheOO) and [Google+](https://plus.google.com/u/0/114598373874764163668/posts).