FlexMonkey 1.0 Code Generation

FlexMonkey can record and run tests that interact directly with your application views, allowing quick, easy creation of GUI tests. FlexMonkey can also save your tests as Actionscript test classes, allowing them to be run in Continuous Integration or customized by editing the generated Actionscript.

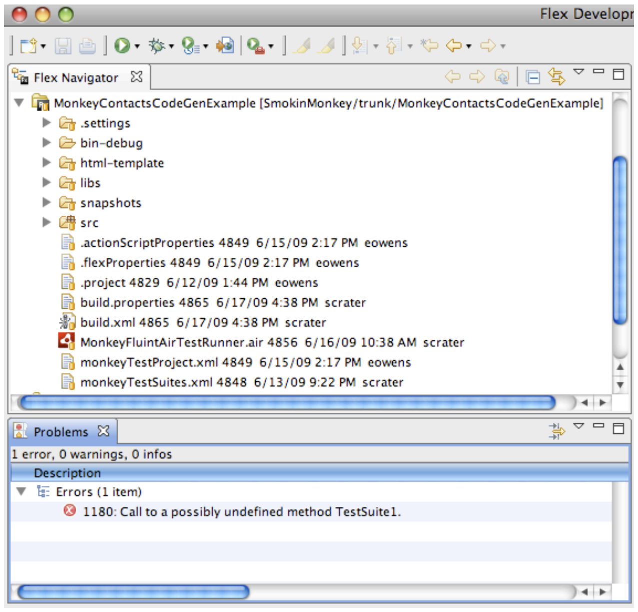
This article shows how to generate a simple FlexMonkey test to Actionscript and run it using either an Air test runner application or a Flex test launcher in a browser.   In either case the test is invoked from an Ant build script; if you are new to Ant, see http://www.adobe.com/devnet/flex/articles/flex_ant_pt1_03.html for an intro on using Ant with FlexBuilder.

GENERATING THE CODE

MonkeyContactsCodeGenExample is the same MonkeyContacts sample application from *Getting Started With FlexMonkey 1.0,* complete with a FlexMonkey test project already set up for you, a Flex module MXML file to package up the compiled test code, and a simple Ant build script which runs the test and produces a test report. A complete test has already been created in the FlexMonkey project, so all you need to do is load the project, generate code, compile, and run; however, a few settings will need to be fixed to match your machine's configuration. These are called out in the steps below, but for reference here are the settings and where they are set:

Setting	**Set in**
Target SWF Location	FlexMonkey Project Properties
Source Directory for Generated Code	FlexMonkey Project Properties
MonkeyFluintAirTestRunner Location	build.properties

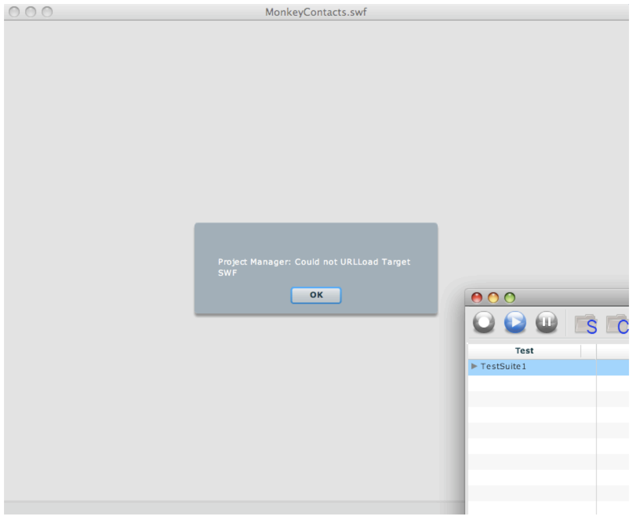
Extract *MonkeyContactsCodeGenExample.zip* to your workspace and import it as a FlexBuilder project (you can also check out MonkeyContactsCodeGenExample via SVN here <URL>.) Next, copy the contents of *MonkeyAccessories.zip/libs* to the example project's *libs* directory. Don't be concerned by the compile error in *MonkeyContactsCodeGenExampleTestModule.mxml* - that will disappear in a moment. Note that FlexBuilder successfully builds the target application, *MonkeyContacts.swf*, in spite of this error.



```
MonkeyContactsCodeGenExample [SmokinMonkey/trunk/MonkeyContactsCodeGenExample]
    .settings
    bin-debug
    html-template
    libs
    snapshots
    src
    .actionScriptProperties 4849  6/15/09 2:17 PM  eowens
    .flexProperties 4849  6/15/09 2:17 PM  eowens
    .project 4829  6/12/09 1:44 PM  eowens
    build.properties 4865  6/17/09 4:38 PM  scrater
    build.xml 4865  6/17/09 4:38 PM  scrater
    MonkeyFluintAirTestRunner.air 4856  6/16/09 10:38 AM  scrater
    monkeyTestProject.xml 4849  6/15/09 2:17 PM  eowens
    monkeyTestSuites.xml 4848  6/13/09 9:22 PM  scrater
```

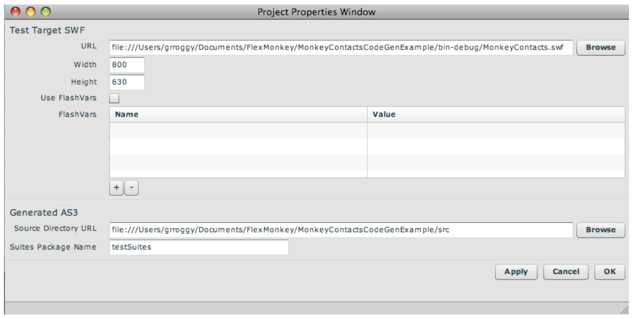
```
1 error, 0 warnings, 0 infos
Description
Errors (1 item)
    1180: Call to a possibly undefined method TestSuite1.
```

Run FlexMonkey and open the MonkeyContactsCodeGenExample project. The target application window will show an error that the target SWF can't be loaded. Click OK on the error.



Project Manager: Could not URLLoad Target SWF

Now open Project Properties from the menu. You will see that the Test Target SWF URL needs to be changed. (FlexMonkey 1.0 does not support relative paths in project files.) Browse to and select MonkeyContacts.swf in your MonkeyContactsCodeGenExample FlexBuilder project's bin-debug folder. Also set the Source Directory URL to your FlexBuilder project's src folder. Your Project Properties Window should now have valid paths for your machine. Click OK to save these changes; the MonkeyContacts application will appear



Test Target SWF

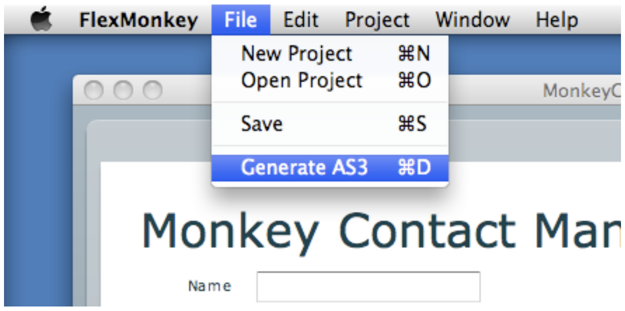
URL: file:///Users/grroggy/Documents/FlexMonkey/MonkeyContactsCodeGenExample/bin-debug/MonkeyContacts.swf
Width: 800
Height: 630
Use FlashVars: ☐
FlashVars:

Name	Value

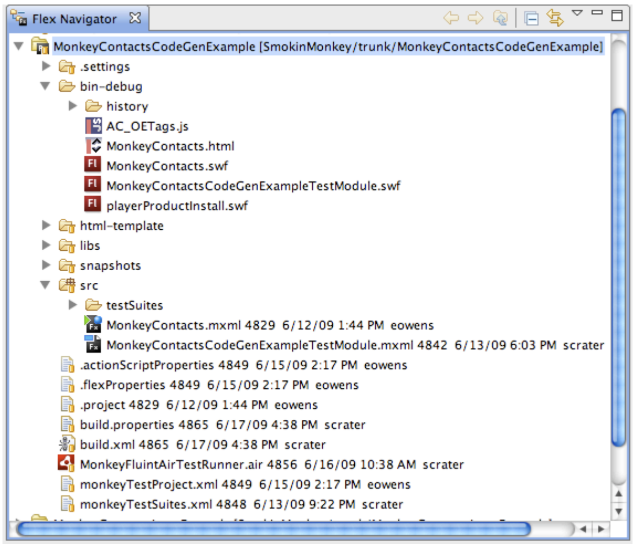
Generated AS3

Source Directory URL: file:///Users/grroggy/Documents/FlexMonkey/MonkeyContactsCodeGenExample/src
Suites Package Name: testSuites

Now, select Generate AS3 from the mail File menu.



Refresh your FlexBuilder project and you should see the testSuite package appear under src. When you rebuild the project, you should see the error in MonkeyContactsCodeGenExampleTestModule.mxml go away as the missing class references have now been resolved. Take a quick look at testSuites.TestSuite1.tests.TestCase1; you may recognize the MonkeyCommand objects there from the FlexMonkey test.



```
MonkeyContactsCodeGenExample [SmokinMonkey/trunk/MonkeyContactsCodeGenExample]
  .settings
  bin-debug
    history
    AC_OETags.js
    MonkeyContacts.html
    MonkeyContacts.swf
    MonkeyContactsCodeGenExampleTestModule.swf
    playerProductInstall.swf
  html-template
  libs
  snapshots
  src
    testSuites
    MonkeyContacts.mxml 4829  6/12/09 1:44 PM  eowens
    MonkeyContactsCodeGenExampleTestModule.mxml 4842  6/13/09 6:03 PM  scrater
  .actionScriptProperties 4849  6/15/09 2:17 PM  eowens
  .flexProperties 4849  6/15/09 2:17 PM  eowens
  .project 4829  6/12/09 1:44 PM  eowens
  build.properties 4865  6/17/09 4:38 PM  scrater
  build.xml 4865  6/17/09 4:38 PM  scrater
  MonkeyFluintAirTestRunner.air 4856  6/16/09 10:38 AM  scrater
  monkeyTestProject.xml 4849  6/15/09 2:17 PM  eowens
  monkeyTestSuites.xml 4848  6/13/09 9:22 PM  scrater
```

RUNNING THE ANT BUILD – LAUNCHER OPTION

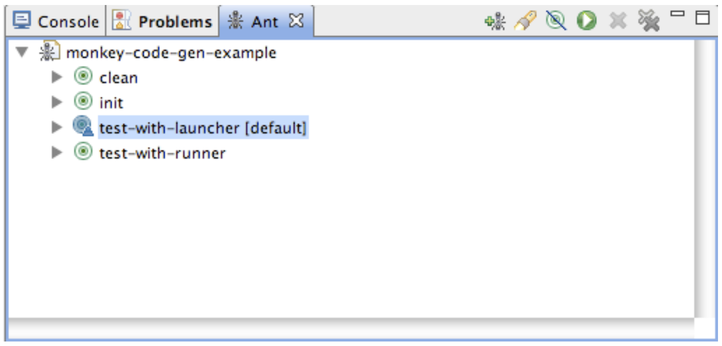
Now that the test has compiled we can run it from Ant. In the same way that there are two ways to connect FlexMonkey to your application, there are two ways to run your FlexMonkey generated test code from Ant. The MonkeyTestLauncher is a Flex application that runs in a browser, while the the FluintMonkeyAIRTestRunner is an AIR application. We will run the launcher first. The launcher consists of three files, *MonkeyTestLauncher.swf, MonkeyTestLauncher.html,* and *AC_OETags.js*. Copy these from *MonkeyAccessories.zip* into the example's *bin-debug* directory. You should also copy monkey-ant-task.jar from *MonkeyAccessories.zip* to the example's *libs* directory.

You may want to take a look at *build.xml* to see what we are about to run. The `test-with-launcher` target will run the test launcher in your default browser and send it the target SWF and test module SWF parameters specified. After the test is run, the `junitreport` task will format the results.

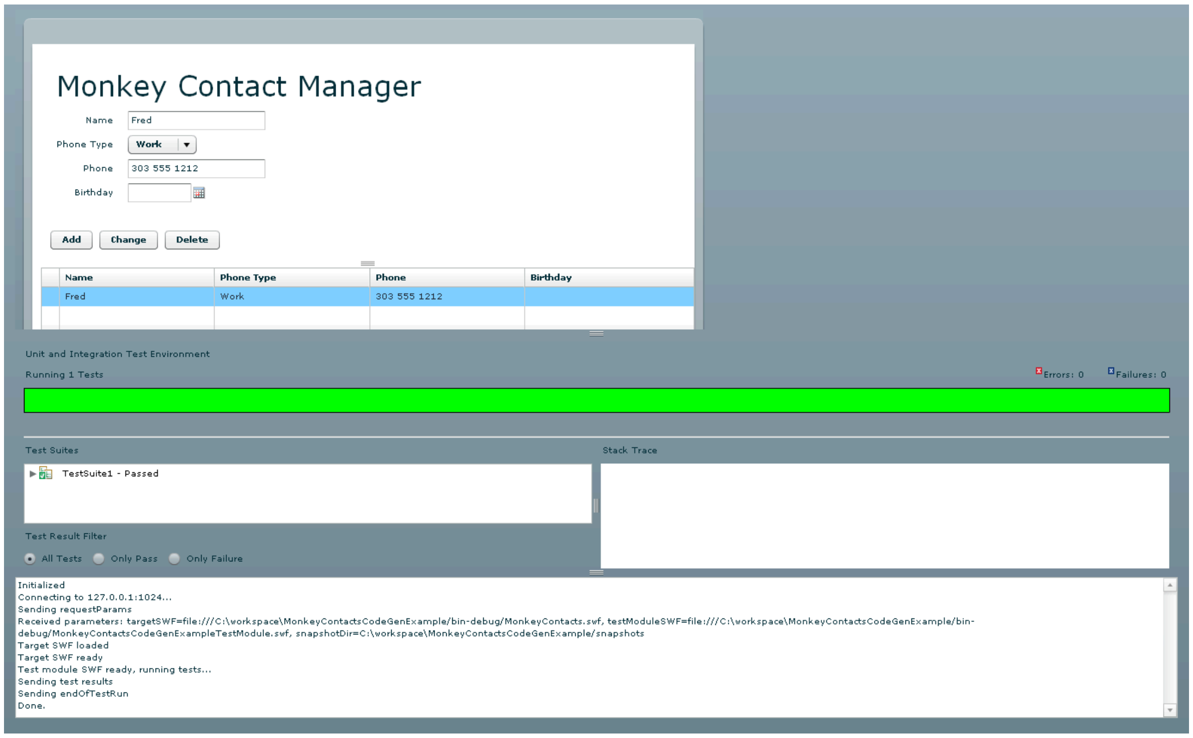
```
<target name="test-with-launcher" depends="init">
   <monkey-test-launcher
      timeout="0"
      launcher="${basedir}/bin-debug/MonkeyTestLauncher.html"
      targetSwf="file:///${basedir}/bin-debug/MonkeyContacts.swf"
      testModuleSwf="file:///${basedir}/bin-debug/MonkeyContactsCodeGenExampleTestModule.swf"
      snapshotDir="${basedir}/snapshots"
      toDir="${report.dir}"
      haltonfailure="false" />

   <junitreport todir="${report.dir}">
      <fileset dir="${report.dir}">
         <include name="TEST-*.xml"/>
      </fileset>
      <report format="frames" todir="${report.dir}/html"/>
   </junitreport>
</target>
```

Open the Ant view in FlexBuilder (Window/Other Views/Ant). Drag the example *build.xml* to the Ant view, open it to display its tasks and and double-click on the `test-with-launcher` task.



The test launcher should appear in a browser, and the MonkeyContacts application should come up in its upper panel. The bottom panel is a console window that logs the launcher's progress; after a few seconds it should look something like this:

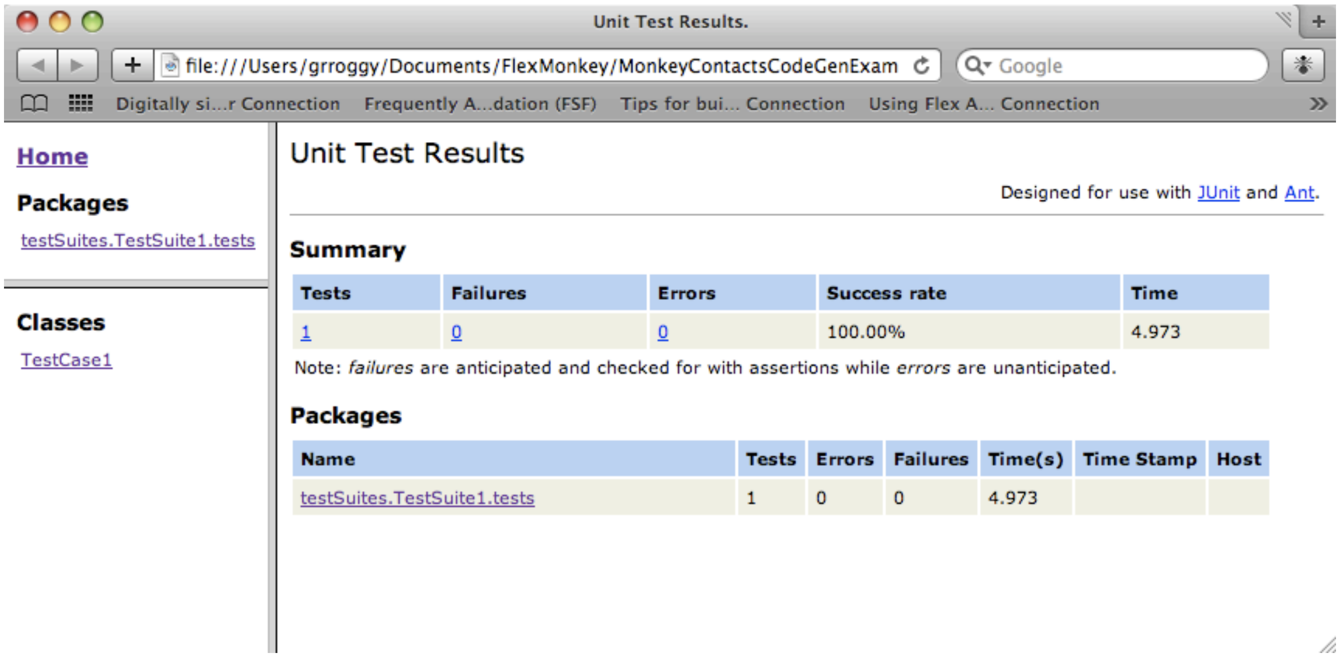


Name	Phone Type	Phone	Birthday
Fred	Work	303 555 1212	

```
Initialized
Connecting to 127.0.0.1:1024...
Sending requestParams
Received parameters: targetSWF=file:///C:\workspace\MonkeyContactsCodeGenExample/bin-debug/MonkeyContacts.swf, testModuleSWF=file:///C:\workspace\MonkeyContactsCodeGenExample/bin-
debug/MonkeyContactsCodeGenExampleTestModule.swf, snapshotDir=C:\workspace\MonkeyContactsCodeGenExample/snapshots
Target SWF loaded
Target SWF ready
Test module SWF ready, running tests...
Sending test results
Sending endOfTestRun
Done.
```

If MonkeyContacts has not come up, or if the test has not run, this console may provide a clue about what's wrong.

When the test has completed, open MonkeyContactsCodeGenExample/output/flex/index.html; this is the test report generated by the `junitreport` ant task. Links in the report allow you to navigate to the results for specific tests.


Unit Test Results.
file:///Users/grroggy/Documents/FlexMonkey/MonkeyContactsCodeGenExam
Google
Digitally si...r Connection
Frequently A...dation (FSF)
Tips for bui... Connection
Using Flex A... Connection
Home
Packages
testSuites.TestSuite1.tests
Classes
TestCase1

Unit Test Results

Designed for use with JUnit and Ant.

Summary

Tests	Failures	Errors	Success rate	Time
1	0	0	100.00%	4.973

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

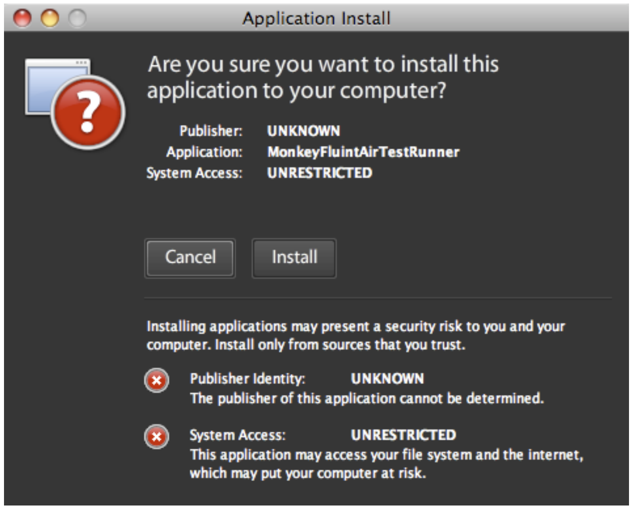
Packages

Name	Tests	Errors	Failures	Time(s)	Time Stamp	Host
testSuites.TestSuite1.tests	1	0	0	4.973		

RUNNING THE ANT BUILD – AIR TEST RUNNER OPTION

Another option for running FlexMonkey tests is MonkeyFluintAirTestRunner, also included in the FlexMonkey suite. Users needing to test AIR specific functionality can do so using MonkeyFluintAirTestRunner, and some users may find this option easier to integrate into their builds.

The installer MonkeyFluintAirTestRunner.air is included for your convenience in MonkeyContactsCodeGenExample. Double click that now to install the application. Note the location of the installed application, as we will need that for the next step. The installer will bring the application up, but it won't be very interesting to look at yet, so close it.


Application Install

Are you sure you want to install this application to your computer?

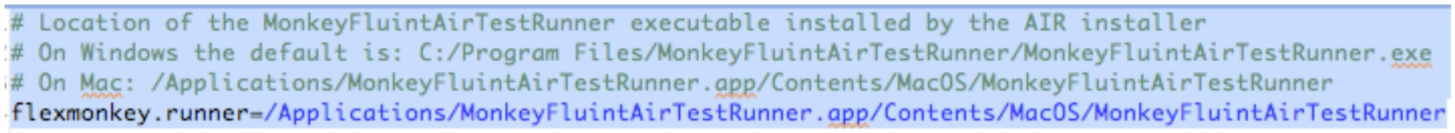
Publisher: UNKNOWN
Application: MonkeyFluintAirTestRunner
System Access: UNRESTRICTED

Installing applications may present a security risk to you and your computer. Install only from sources that you trust.

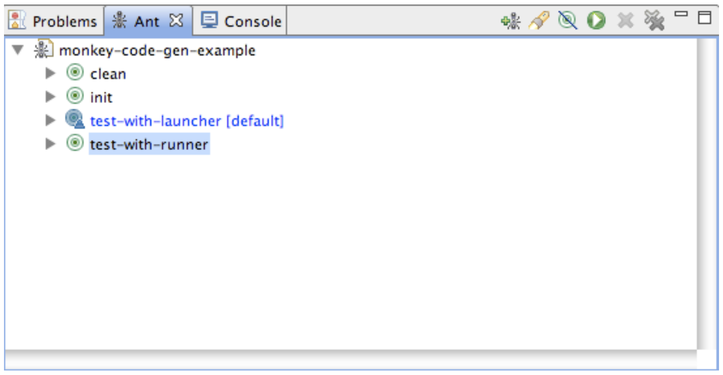
Publisher Identity: UNKNOWN
The publisher of this application cannot be determined.

System Access: UNRESTRICTED
This application may access your file system and the internet, which may put your computer at risk.

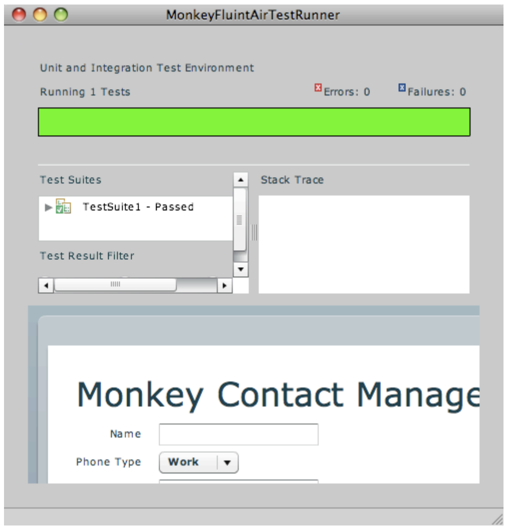
Open build.properties and set the flexmonkey.runner property to the location where the installer placed the MonkeyFluintAirTestRunner application:


```
# Location of the MonkeyFluintAirTestRunner executable installed by the AIR installer
# On Windows the default is: C:/Program Files/MonkeyFluintAirTestRunner/MonkeyFluintAirTestRunner.exe
# On Mac: /Applications/MonkeyFluintAirTestRunner.app/Contents/MacOS/MonkeyFluintAirTestRunner
flexmonkey.runner=/Applications/MonkeyFluintAirTestRunner.app/Contents/MacOS/MonkeyFluintAirTestRunner
```

Now run the `test-with-runner` task in build.xml.



You should see the MonkeyFluintAirTestRunner come up, the MonkeyContacts application come up in a panel of the MonkeyFluintAirTestRunner, and see some activity as the MonkeyCommands run. MonkeyFluintAirTestRunner will close when the test has completed.



As before, the `junitreport` task will format the test results when the run has completed.

NOTES

Here are a few things you will need to know as you begin to explore generating code from FlexMonkey:

- The test module class (called MonkeyContactsCodeGenExampleTestModule.mxml in the example) is not generated by FlexMonkey 1.0, so you will need to create it by hand and update it when you add or rename test suites.

- FlexMonkey generates class and method names based on the TestSuite, TestClass and Test names you assign in your test. FlexMonkey 1.0 does not check for uniqueness of these names, so if your names are not unique you will see compile errors in FlexBuilder when you generate Actionscript.

- You can write your own FlexMonkey Actionscript or modify the generated code. You do not need to use the VerifyMonkeyCommand to add verification code – you can call the Fluint assert() and fail() methods and their variants directly. See the Fluint documentation at http://code.google.com/p/fluint/.