FarCry Plugins : testMXUnit

This page last changed on Jan 14, 2009 by rob.

Prerequisite

Farcry unit tests uses http://mxunit.org/ with a wrapper that adds in Farcry hooks and webskin compatibility. You can get the latest copy of the plugin by checking out the code from the following SVN repository URL:

```
http://svn9.cvsdude.com/modius/farcry/plugins/testMXUnit
```

Once you have the code checked out into your plugins folder, you will need to add an Alias (in Apache) or a Virtual Directory in IIS. The path should be similar to the following:

```
Alias /mxunit /Users/robrohan/Sites/blarg/Yadda/plugins/testMXUnit/www/mxunit
```

Note the path goes in one level further than most Farcry alias'.

You will then need to add the plugin to your plugin list in your *www/farcryConstructor.cfm* file. For example:

```
<cfset THIS.plugins =
"farcrypoll,farcryradiostar,farcrycms,farcryverity,farcrycfximage,testMXUnit" />
```

And finally, you will need to run *updateapp* on your application.

Running Tests

To run unit tests, create a folder in your project's webskin folder called *testMXUnit*. Within that folder create a file called *displayPageStandard.cfm*. The contents of *displayPageStandard.cfm* should initially match the contents of the file *displayPageStandard.cfm* in the *testMXUnit* plugin itself. In fact, you can just copy that file into your project.

Once you have the *displayPageStandard.cfm* file in your project, you can add test suites by calling *addAll* on the *testSuite* object. For example, around line 22 you should see the example test suite addition:

```
<cfscript>
////////////////////////////////////////////////////////////////////////
// Example of adding single tests for your application
testSuite.addAll("farcry.plugins.testMXUnit.tests.ExampleTest");
results = testSuite.run(testMethod=url.method);
</cfscript>
```

To actually run this test, point your browser to the url:

```
http://[your_project]/index.cfm?type=testMXUnit
```

You can also run a single specific method in the unit test cfc by adding the URL parameter *method*. For example:

```
http://[your_project]/index.cfm?type=testMXUnit&method=testMyThing
```

If you get an error, be sure you have run *updateapp*, you have the web mapping setup, and the plugin added to your *www/farcryConstructor.cfm* file.

Writing Tests

The standard for tests is to create a directory call *tests* in the root level of your plugin or project. In other words, the directory should be at the same level as *packages*, *tags*, *webskin*, etc. Within the test directory you will create CFCs that hold all your tests.

The CFCs must extend *mxunit.framework.TestCase*. There is a file called *ExampleTest.cfc* in the *testMXUnit* plugin that you can use as a template. The contents of that file at the time of this writing are the following:

```
<cfcomponent extends="mxunit.framework.TestCase">
<!--- setup and teardown --->
<cffunction name="setUp" returntype="void" access="public">
<!--- Any code needed to return your environment to normal goes here --->
</cffunction>

<cffunction name="tearDown" returntype="void" access="public">
<!--- Any code needed to return your environment to normal goes here --->
</cffunction>

<!--- ////////////////////////////////////////////////////////////// --->

<cffunction name="passTest" returntype="void" access="public">
<!--- Any code needed to return your environment to normal goes here --->
<cfset assertEquals(true,true) />
</cffunction>

<cffunction name="failTest" returntype="void" access="public">
<!--- Any code needed to return your environment to normal goes here --->
<cfset assertEquals("yes", "no") />
</cffunction>

</cfcomponent>
```

Once your tests are written, you can add them your *testSuite.addAll()* in your projects *testMXUnit displayPageStandard.cfm* file.

Since the tests are running from within the context of Farcry, all of the Farcry tags and functions are available from within your test. While it is not pedantically correct to write tests that use items outside of the unit test and method you are testing, it is often quite helpful (and sometimes required).

For documentation and tutorials on MXUnit see http://mxunit.org/doc/index.cfm

Known Issues

The extjs display version of the unit test does not work.