

Improve Your ColdFusion Code Through Unit Testing

- Adobe ColdFusion Developer Week
 - Tuesday, September 13, 2011, 10:00 AM Pacific
- Jamie Krug
 - UGM, Rochester NY ColdFusion User Group
 - <http://jamiekrug.com/blog>
 - <http://twitter.com/jamiekrug>



Overview

- What Is Unit Testing?
- Why Unit Testing?
- MXUnit Testing Framework
- MXUnit Eclipse Plugin
- Test Cases, Assertions, Test Driven Development (TDD)
- Debugging and Other Real World Examples
- Advanced Topics, What's Next...



What Is Unit Testing?

- Testing individual units of code
 - In CF, a unit is generally a CFC or Interface
- Test cases ensure code behaves as intended
- Test cases are run in isolation
- Unit testing is NOT integration testing



Why Unit Testing?

- Facilitate refactoring
 - Repeatable tests == freedom!
- Simplify integration testing
 - Easier when the "parts" are already tested
- Application Design & Documentation
 - Built-in, maintained and always current!



MXUnit Unit Testing Framework

- Modeled after other xUnit frameworks (e.g., JUnit)
- Conveniences for writing and running unit tests
 - Assertions, test cases, test suites
 - Test runners: HTML, Eclipse, directory runner
 - Results output: XML, HTML, JUnit reports, etc.
 - Test private methods
 - Data providers
 - Mocking and stubbing



MXUnit Framework Installation

1. Download MXUnit: <http://mxunit.org/download.cfm>
2. Unzip into Web root: {webroot}/mxunit
3. Test the install: [http://host\[:port\]/mxunit/index.cfm](http://host[:port]/mxunit/index.cfm)

DEMO...



MXUnit Eclipse Plugin

- Eclipse Update URL: <http://mxunit.org/update>
- Global settings: Window > Preferences > MXUnit
- Project settings: Project > Properties > MXUnit Properties

DEMO...



Anatomy of a Test Case

- The file is a component (a .cfc file).
- The file name starts or ends with "Test."
- The component extends `mxunit.framework.TestCase`.
- All public methods will be run as tests (regardless of their names).
- `setUp()` will run prior to each and every test.
- `tearDown()` will run after to each and every test.
- `beforeTests()` will run once before any tests are run.
- `afterTests()` will run once after all tests have run.
- Private methods are not considered tests (not run by MXUnit).

EXAMPLE...



Assertions

- MXUnit base assertions
 - `assertTrue(boolean condition)`
 - `assertEquals(any expected, any actual)`
 - `assertSame(any obj1, any obj2)`
- MXUnit assertion extensions
 - e.g., `assertIsTypeOf(component obj, string type)`
- Custom assertions
- Assertion patterns



What About Test Driven Development (TDD)?

- Write a failing test.
 - This defines desired improvement or new function.
- Write code to pass the test.
- Refactor the new code to reasonable standards.

DEMO...



Viewing Debug Output

- Use `debug()` wherever you would normally use `cfdump` or `writeDump()`.
- `cfdump/writeDump()` or `cfoutput/writeOutput()` will show up in test output, but if your test fails or errors, the output will now show up.
- You can also use `request.debug()` to see variables in your component under test.

DEMO...



Testing Private Methods

```
component extends="mxunit.framework.TestCase"
{
    function testSomePrivateDefaultBehavior()
    {
        var myObj = new Something();

        makePublic( myObj, "somePrivate" );

        var result = myObj.somePrivate( "blah" );

        assertEquals( "blah", result );
    }
}
```



Testing Expected Exceptions

```
function foo_shouldFailBecauseExpectedExceptionListNotThrown()  
    mxunit:expectedException="Database,MyCustomException"  
{  
    new Foo().doSomethingToThrowADatabaseOrMyCustomException();  
}
```



MXUnit Data Providers

- Data driven testing
- Execute tests with a wide variety of input data
- Efficient test coverage
- Provide data collection reference and MXUnit will iterate over collection and repeat test execution for each item
- Accepted data: array, query, list, and a CSV or Excel file

EXAMPLE...



Mocking and Stubbing

- Easily create mock objects to simulate real objects
- Easily stub out data returned from a mock
- Important tools to isolate units to test from dependencies

EXAMPLE...



Creating Test Suites

```
testSuite = new mxunit.framework.TestSuite().TestSuite();
testSuite.addAll( 'mxunit.samples.MyComponentTest' );
testSuite.addAll( 'mxunit.samples.MyOtherComponentTest' );
testSuite.add( 'mxunit.samples.YetAnotherComponentTest',
    'aTestFunction,anotherTestFunction'
);
results = testSuite.run();
writeOutput( results.getResultsOutput( 'html' ) );
```



What Else?

- Test driven development (TDD)
- Automation (ANT, CI, etc.)
- Snippets
- Keyboard shortcuts for MXUnit Eclipse plugin



Yikes! Where Do I Start?!?

1. Write one test
2. Write tests for bug fixes and new code
3. Automate testing
4. Expand test coverage



Resources

- <http://wiki.mxunit.org/>
- <http://groups.google.com/group/mxunit>
- <http://blog.mxunit.org/>
- Open source projects' tests



Questions?

- Thank you!
- Jamie Krug
 - <http://jamiekrug.com/blog>
 - <http://twitter.com/jamiekrug>

