InterSystems
Health | Business | Government

Unit Test Coverage in ObjectScript

Timothy Leavitt
Developer, HealthShare

Agenda

Background

- Unit Tests in ObjectScript
- Continuous Integration, Jenkins

Test Coverage

- Concepts
- Other languages
- Our solution

Demo

Applications

Key Idea

Unit tests are an essential tool for building high-quality software. Measuring how well they cover the code under test can help developers write more valuable tests and can encourage adoption of unit testing within an organization.

Background

Unit Tests and Continuous Integration

Background: Unit Testing in ObjectScript

Subclass %UnitTest.TestCase

Method names starting with “Test”

$$$Assert* macros

Run with %UnitTest.Manager

See: Caché Unit Test Tutorial in our Online Documentation

Background: Unit Testing in ObjectScript

```
Class MyPackage.Tests Extends %UnitTest.TestCase {

Method TestAdd() {
  Do $$$AssertEquals(##class(MyPackage.TestMe).Add(2,2),4,"Test Add(2,2)=4")
  Do $$$AssertNotEquals(##class(MyPackage.TestMe).Add(2,2),5,"Test Add(2,2)'=5")
}
}
```


Background: Unit Testing in ObjectScript

Run unit tests as follows:

```
Set ^UnitTestRoot = "C:\path\to\tests\"

Do ##class(%UnitTest.Manager).RunTest(,"/nodelete")
```


Background: Continuous Integration

Automatically builds, tests when changes are made

Reports on build failures, test results

May produce artifacts for use later

Many tools; we use Jenkins

Demo

Unit tests in Jenkins for WidgetsDirect

Test Coverage

Why? What? How?

User Stories for Test Coverage

As a software developer, I want to know if my unit tests are sufficient and how to improve them, so that I can avoid bugs and an increasing burden of technical debt.

As a manager, I want to know at a glance how well my team is adopting and using unit tests, so that I can provide appropriate feedback and direction.

As a customer or prospective customer, I want to know that the product I am considering or using for my critical business operations is well-tested and high-quality, so that I can buy and operate with confidence.

Test Coverage Metrics

Coverage can be measured at the following levels:

- Package/class (at least one line was run by tests)
- Method (at least one line was run by tests)
- Line (at least one statement on the line was run by tests)
- Branch (a particular path was run by tests)
- Condition (a particular Boolean condition evaluated to true/false during tests)

Also consider cyclomatic complexity

Test Coverage for Other Languages

Java: Cobertura, Clover, and more

JavaScript: Istanbul, others

Two ways these work, generally speaking:

- Instrument + recompile code
- Instrument bytecode

Our Solution

On GitHub: https://github.com/intersystems/TestCoverage

- Works like %UnitTest.Manager, with additional options:
  - Do ##class(TestCoverage.Manager).RunTest(,,.extraOptions)
- Works with your existing unit tests
- Works on InterSystems IRIS, but also reasonably-modern Caché, Ensemble
- Requires specification of classes/routines for which coverage should be measured
- Measures coverage at the package, class/routine, method, line level
- Includes simple UI for viewing aggregated/detailed results
- Allows export in Cobertura format for use in CI tools
- Supported via Developer Community, GitHub issues, community contributions

Demo

Test coverage for the WidgetsDirect application

How it Works

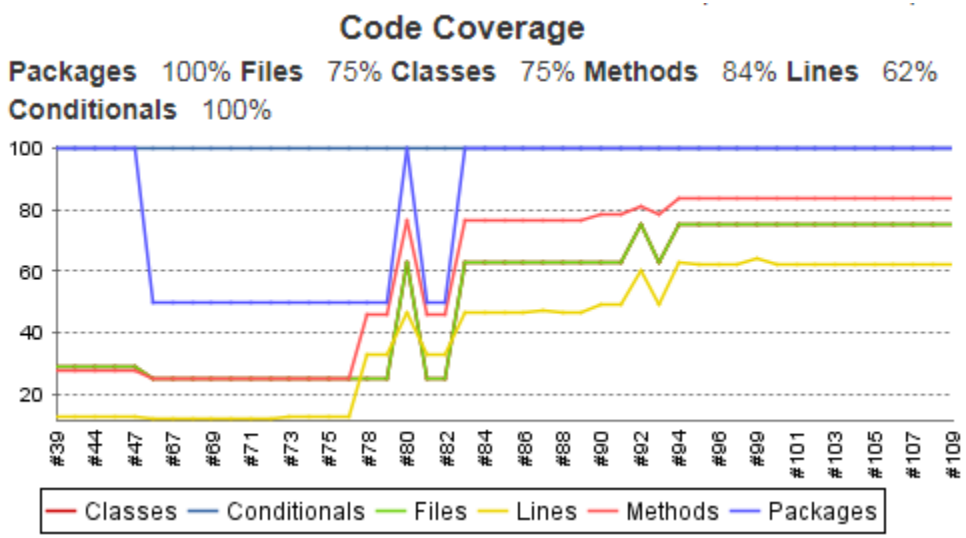
Line-by-line monitor (%Monitor.System.LineByLine) to track coverage at .INT level

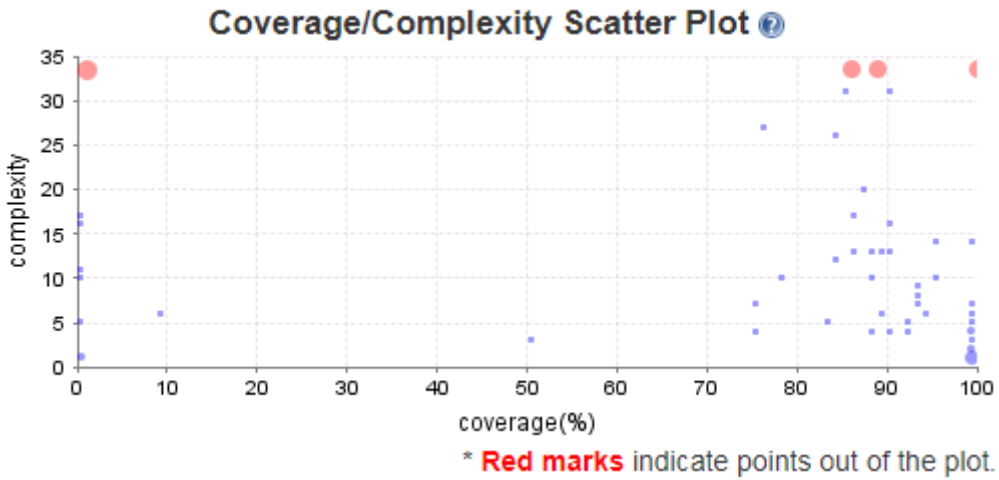
Debug map to convert to coverage at .MAC/.CLS level

%Library.SyntaxColor to figure out which code is “executable”

Use Case: HealthShare

Jenkins reports on Test Coverage, Complexity/Coverage





Use Case: HealthShare

Results at a glance in build notification e-mails

BUILD SUCCESS

Build URL https://hsjenkins.iscinternal.com/job/HS.JSON/109/
Project: HS.JSON
Date of build: Thu, 27 Sep 2018 08:47:56 -0400
Build duration: 1 min 32 sec

CHANGES

No Changes

JUnit Tests

Name: UnitTest.HS.JSON Failed: 0 test(s), Passed: 71 test(s), Skipped: 0 test(s), Total: 71 test(s)
Name: UnitTest.HS.JSON.Reader Failed: 0 test(s), Passed: 39 test(s), Skipped: 0 test(s), Total: 39 test(s)

Cobertura Report

Project Coverage Summary

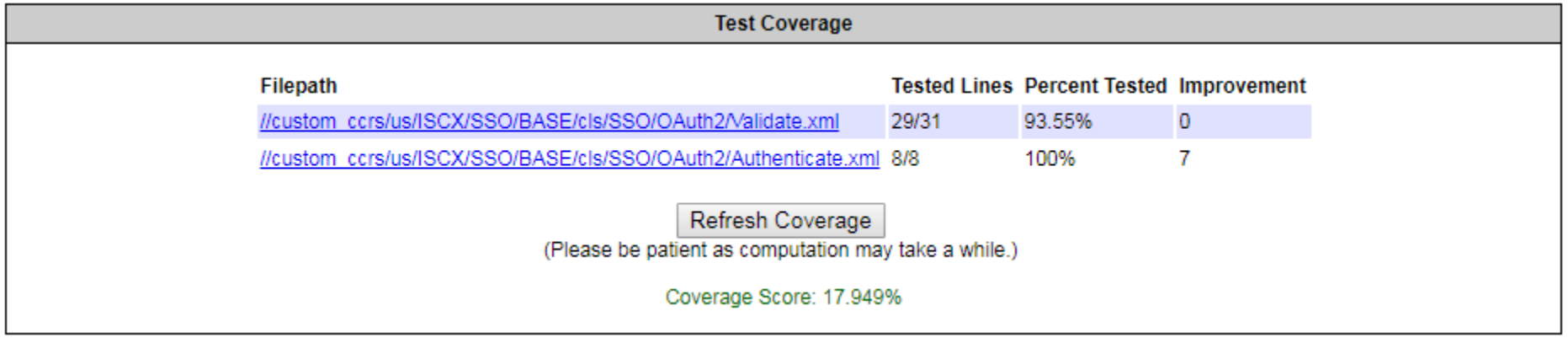
Name	Packages	Files	Classes	Methods	Lines	Conditionals
Cobertura Coverage Report	100% (2/2)	75% (6/8)	75% (6/8)	84% (67/80)	62% (1139/1836)	100% (0/0)

Coverage Breakdown by Package

Name	Files	Classes	Methods	Lines	Conditionals
HS.JSON	71% (5/7)	71% (5/7)	82% (59/72)	60% (1035/1731)	100% (0/0)
HS.Util	100% (1/1)	100% (1/1)	100% (8/8)	99% (104/105)	100% (0/0)

Use Case: InterSystems Internal Applications

CCR (internal application for managing documentation/promotion of changes) pulls diffs in test coverage data from Jenkins:



Test Coverage

Filepath	Tested Lines	Percent Tested	Improvement
//custom_ccrs/us/ISCX/SSO/BASE/cls/SSO/OAuth2/Validate.xml	29/31	93.55%	0
//custom_ccrs/us/ISCX/SSO/BASE/cls/SSO/OAuth2/Authenticate.xml	8/8	100%	7

Refresh Coverage

(Please be patient as computation may take a while.)

Coverage Score: 17.949%

Caveats

Of the technique:

- Tests with great coverage and no assertions are bad tests, but will look good!
- 100% line coverage is likely an unreasonable goal, and may be unattainable *

Of our tool:

- Currently does not support branch/condition coverage
- Currently has limitations on measuring across multiple processes

* without design for dependency injection, or advanced techniques like mocking.

Questions?

Please complete the survey in the event app or text “summit” to 878787

Session recording and slides will be available at:
Learning.InterSystems.com
Search for “Global Summit 2018”

Visit the Tech Exchange to meet with a subject expert

The power behind what matters.

InterSystems®
Health | Business | Government

Thank you.