Open APIs for Open Minds

Unit Testing DITA-OT Plugin Extensions

Jason Fox (FIWARE Foundation)

FIWARE

Disclaimer

The views presented in this workshop are my own and do not necessarily reflect the views of the FIWARE Foundation

FIWARE

New Employer - FIWARE Foundation

WHAT IS FIWARE?

FIWARE IS A CURATED FRAMEWORK OF OPEN SOURCE PLATFORM COMPONENTS TO ACCELERATE THE DEVELOPMENT OF **SMART SOLUTIONS**

OPEN SOURCE

A market-ready open source software, combining components that enable the connection to IoT with Context Information Management and Big Data services in the Cloud.

SMART USAGE OF DATA

Standard APIs for data management and exchange, as well as harmonised data models.

SMART SOLUTIONS & SERVICES

Automation of processes across the entire value chain. Easy plug&play integration with other solutions and services. Part of a marketplace of portable and interoperable solutions.

FIWARE FOUNDATION

Actively promotes the FIWARE Adoption, supports the community providing shared resources and validates the FIWARE technologies.

FIWARE SUMMITS

A meeting place for developers, entrepreneurs, political decision makers, thought leaders, business executives and investors.

FIWARE ECOSYSTEM

More than 100 cities, 11 iHubs, various accelerator programmes, and strategic partnerships with GSMA, TM Forum, CEF, and ETSI, amongst others.

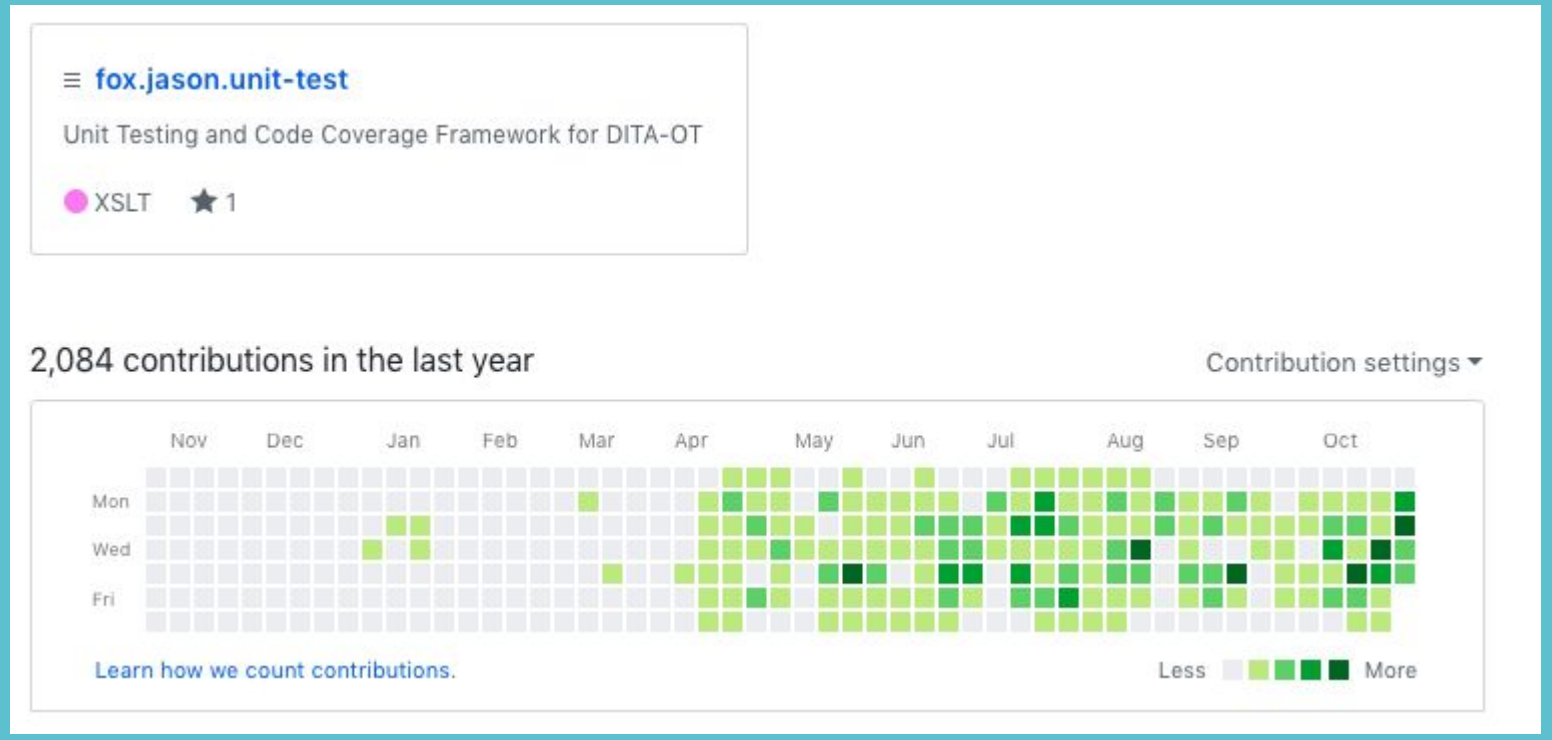
FIWARE

New Employer - FIWARE Foundation

Open Source Friendly - Everything is available on GitHub:

Website: **fiware.org**

@FIWARE on Twitter



https://github.com/jason-fox/fox.jason.unit-test

Why Unit Testing?

- Validate that it does what you want
- Check that you haven't broken something else
- Essential safety net for refactoring

Why not unit testing?

- Writing tests is difficult
- No support tools
- Spend more time debugging the tests than the code

FIWARE

Why Code Coverage?

- Validate your tests are comprehensive
- Check that no dead code is left in the code base
- Maintain standards

Why Profiling?

- Check where bottlenecks occur

FIWARE

Demonstration

- UNIT TESTING
- CODE COVERAGE
- ANT PROFILING

FIWARE

Unit Test Report and Coverage Report

Unit Test Results

Designed for use with AntUnit and Ant.

Summary

Tests	Failures	Errors	Success rate	Time
9	0	0	100.00%	361.872

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

Test Suites

Note: test suite statistics are not computed recursively, they only sum up all of its testsuites numbers.

Name	Tests	Errors	Failures	Time(s)
org_dita_html5	4	0	0	140.795
org_dita_pdf2	5	0	0	221.077

Tests for org_dita_html5

Name	Status	Type	Time(s)
test: Expect that a codeblock displays in a monospace font	Success		37.922
test: Expect that HTML body text should be displayed in the standard font	Success		30.702
test: Expect that a codeblock usually has a background except when processing within tables	Success		36.182
test: Expect that an HTML codeblock can contain links	Success		35.890

Back to top

Tests for org_dita_pdf2

Name	Status	Type	Time(s)
test: Expect that a PDF codeblock should be displayed in a monospace font.	Success		47.596
test: Expect that additional PDF processing for codeblocks and allows codeblocks to contain sub-nodes.	Success		56.892
test: Expect that a PDF codeblock usually has a background except when processing within tables	Success		37.599

Summary

Title	Templates	Covered	Percent
com.oxygenxml.editlink	84	54	64.29%

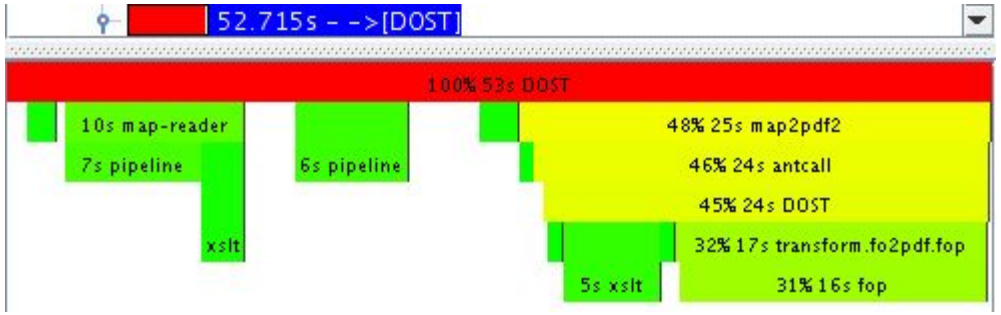
Test suites

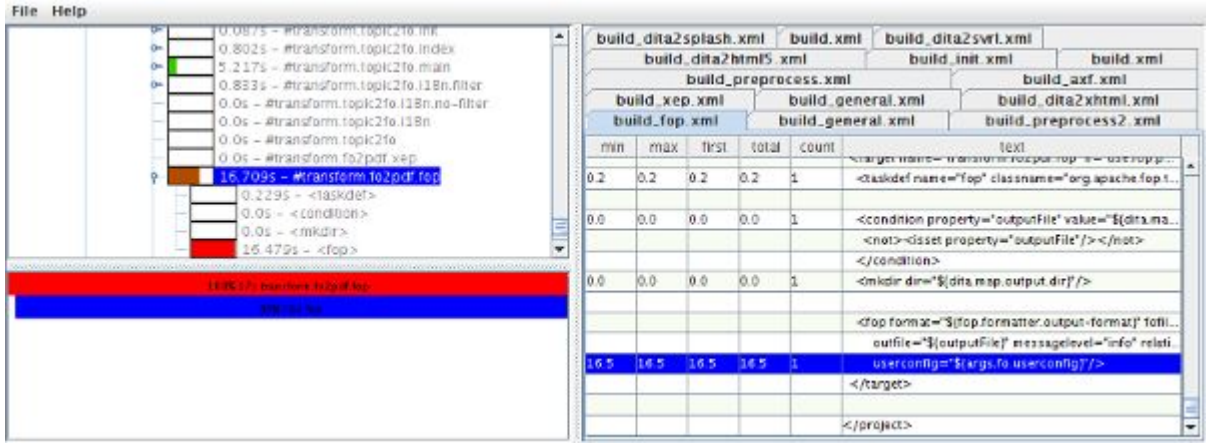
com.oxygenxml.editlink

Id	Templates	Count
xhtml.xsl	`<xsl:template match="*[contains(@class, ' topic/topic ')]/*[contains(@class, ' topic/title ')]">`	1
	`<xsl:choose>`	1
	`<xsl:when test="string-length($editlink.remote.ditamap.url) > 0 or $editlink.present.only.path.to.topic = 'true'"/>`	1
	`<xsl:otherwise/>`	0
	`</xsl:choose>`	
	`</xsl:template>`	
	`<xsl:template match="*[starts-with(local-name(), 'h')]" mode="add-edit-link">`	1
	`<xsl:choose>`	1
	`<xsl:when test="$editlink.present.only.path.to.topic = 'true'"/>`	0
	`<xsl:otherwise/>`	1
	`</xsl:choose>`	
	`</xsl:template>`	
	`<xsl:template match="node() | @*" mode="add-edit-link"/>`	0
pdf5.xsl	`<xsl:template match="*[contains(@class, ' topic/topic ')]/*[contains(@class, ' topic/title ')]"/>`	0
xslfo.xsl	`<xsl:template match="*" mode="processTopicTitle">`	1
	`<xsl:choose>`	1
	`<xsl:when test="string-length($editlink.remote.ditamap.url) > 0 or $editlink.present.only.path.to.topic = 'true'">`	1
	`<xsl:if test="$level = 1"/>`	0
	`<xsl:if test="$level = 2"/>`	1
	`<xsl:if test="@xtrf">`	1
	`<xsl:choose>`	1
	`<xsl:when test="$editlink.present.only.path.to.topic = 'true'"/>`	0
	`<xsl:otherwise/>`	1
	`</xsl:choose>`	
	`</xsl:if>`	
	`</xsl:when>`	
	`<xsl:otherwise/>`	0
	`</xsl:choose>`	
	`</xsl:template>`	

FIWARE

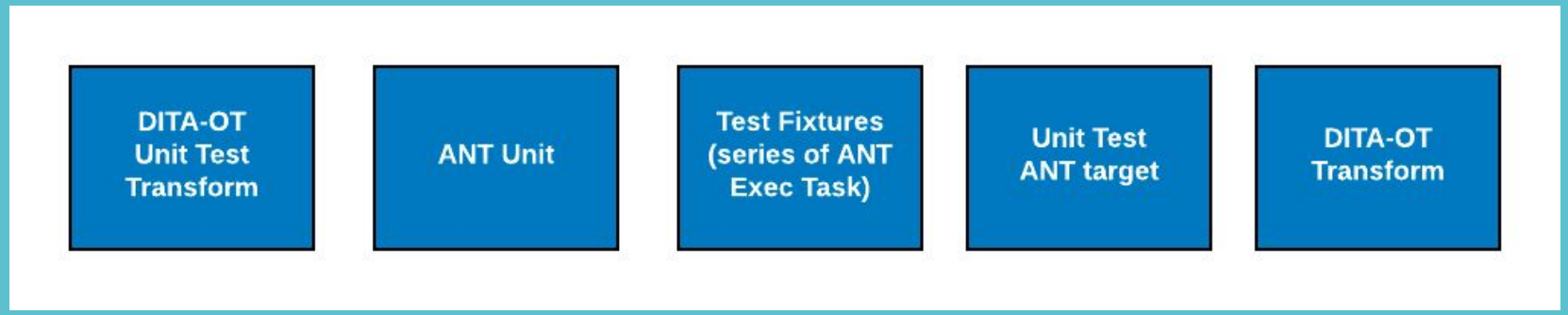
Profiling Report





Deep Dive - Unit Testing

```
./bin/dita -f unit-test -o ../out -i plugins/fox.jason.splash/
```



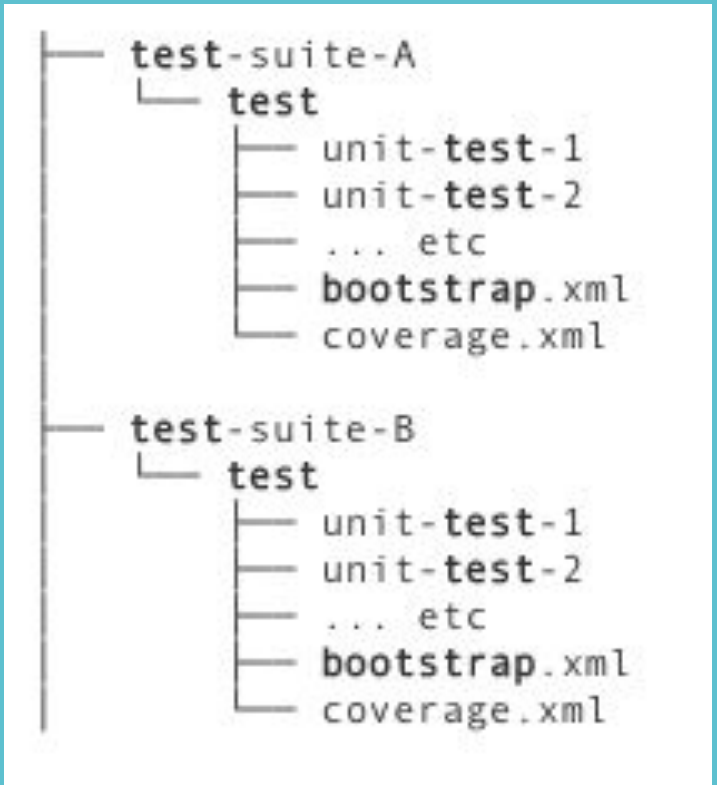
- build-init
- unit-test.init
- generate-fixtures
- unit-test
- coverage.report
- unit-test.report

```
Trying to override old definition of task fail

[SUCCESS] All tests have passed
```


FIWARE

Anatomy of a Test Suite Directory



- Series of named sub directories:
  - One Test = One Dir
- Essential file in root
  - `bootstrap.xml`
- Other Special Files
  - `coverage.xml`
  - `test.properties`
  - `disabled.txt`

FIWARE

Anatomy of a Unit Test

```
├── build.xml
├── document.ditamap
├── expected.html
├── test.properties (optional)
└── topics
    └── *.dita files
```

```
<project basedir="." default="unit-test">
  <import file="../../bootstrap.xml"/>
  <description>
    Body text should be displayed in the standard font
  </description>
  <target name="unit-test">
      <exec-html5/>
      <get-html-article from="topics/body-text.html"/>
      <compare-output suffix=".html"/>
  </target>
</project>
```

- Standard DITA-OT inputs
  - **`document.ditamap`**
  - **`*.dita`** files
- Output becomes Test expectation
  - **`expected.html`**
  - **`expected.fo`**
  - etc
- **`build.xml`** defines the flow of the test by calling ANT tasks defined in the plugin

FIWARE

Simplifying the Unit Tests

- `cfg` - simplifies the Output
  - `attributes.xml` - Attributes to ignore in the output
  - `colors.xml` - Switches from HEX to readable names
  - `fonts.xml` - Switches from your fonts to placeholders (e.g. Monospace)

- `resource` - simplifies the Input - see API Reference
  - Compare-Output
  - Contains-Text
  - Exec-HTML5
  - Exec-PDF
  - Exec-Transtype
  - Get-HMTL-Article
  - Get-PDF-Article

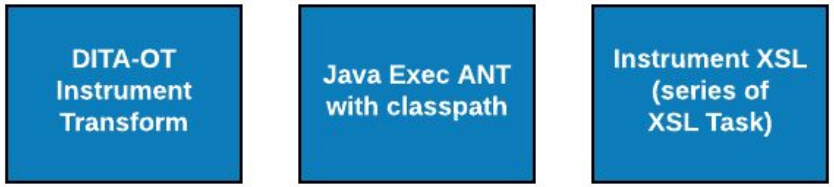
Writing your own Unit Tests

- **Copy** an existing test from the samples in the plugin to get a simple PDF or HTML output
- **Update** the `expectation.html/fo` Volkswagening `test.copy` parameter
- **Run** `build.xml` directly to get verbose output (useful when debugging)
- **Repeat…**

FIWARE

Deep Dive - Code Coverage

```
./bin/dita -v -f xsl-instrument  -i plugins/fox.jason.splash/
```




```
./bin/dita -f unit-test -o ../out -i plugins/fox.jason.splash/
```

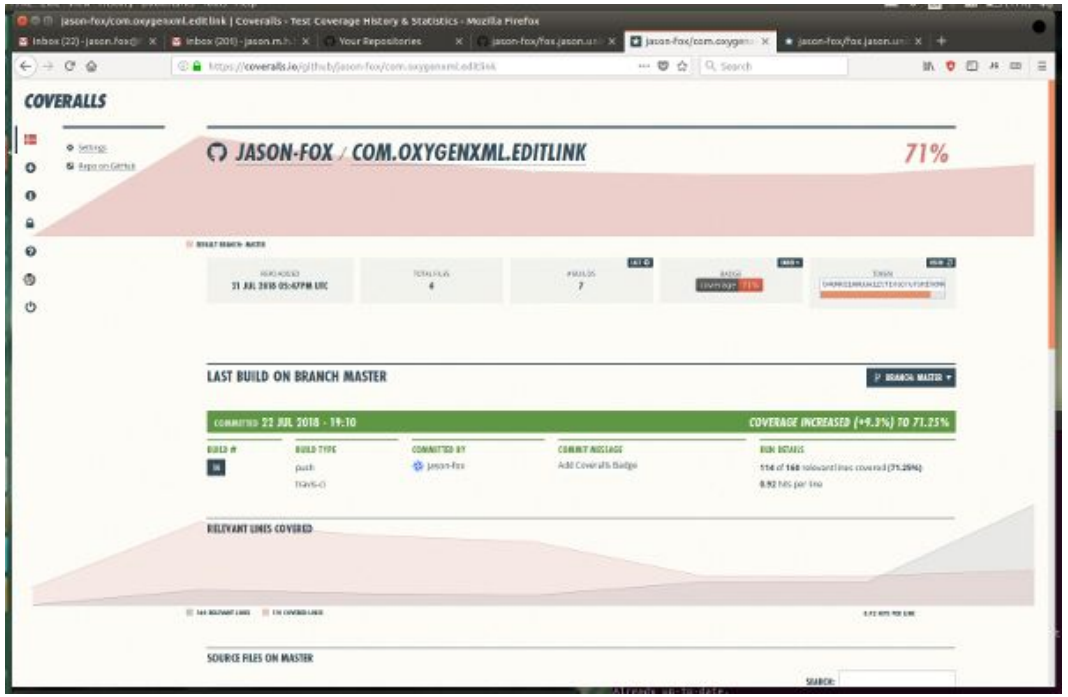

```
coverage.report:
[UNIT002I][INFO] Running an XSL coverage report

[SUCCESS] All tests have passed
```

- Instrument
  - Adds **`<xsl:message>`** to each decision point
  - Unique ID based on number of preceding nodes
  - Create an empty **`coverage.xml`** file with line numbers and IDs
- Coverage Report
  - When unit tests are run, IDs are piped to output.
  - coverage-report target updates number of hits
  - XSL transform to HTML and Cobertura

FIWARE

Integration - Travis CI + Coveralls



- Use any CI to run tests
- **`coverage.xml`** is Cobertura-like output
- Use standard **`pom.xml`** to read file and upload online
- Can track coverage over time
- Legitimately display 

and 

Badges

FIWARE

Deep Dive - Profiling

```
./bin/dita -f antro --test.transform=pdf2 -i document.ditamap
```

Antro

- Existing ANT listener task
- Add to classpath and as a listener

```
./bin/dita -f antro-ui  -i document.ditamap
```

Antro UI

- GUI can be invoked as a Java Jar
- Add to classpath and run Java

FIWARE

Thank you!

http://fiware.org
Follow @FIWARE on Twitter

FIWARE