Advanced Java Testing

What’s next?
Agenda

• Context & Current status quo
• Coverage testing
• Testing for backward compatibility
• Mutation testing
• Environment testing
Open source wiki

14 years

10-15 active committers

Very extensible, scripting in wiki pages

Platform for developing ad-hoc web applications

Strong build practices using Maven and lots of “Quality” plugins

Using Jenkins & custom pipeline library for the CI

http://xwiki.org
Context: STAMP

- Automatic Test Amplification
- XWiki SAS participating
- Experiment on XWiki project

![STAMP diagram with cycle of software development phases: code, plan, test, build, release, deploy, monitor, operate, and configuration testing. Highlighted areas include Mutation testing and Environment Testing.]
Current Testing Status

- 10414 automated tests (in 2.5 hours):
  - Unit tests (using Mockito)
  - Integration tests (using Mockito)
  - Functional (UI) tests (using Selenium/Webdriver)
New questions

• Are my tests testing enough? **Coverage** ✅

• How can I prevent breaking my users (when I expose some APIs)? **Backward compatibility** ✅

• How good are my tests? **Mutation testing** ⚠

• Do my software work in various setups? **Environment testing** ⚠

= in place w/ strategy   ⚠ = in progress
Test Coverage

- Using Jacoco and Clover

**Strategy** - “Ratchet effect”:

- Each Maven module has a threshold
- Jacoco Maven plugin fails if new code has less coverage than before in %
- Dev is allowed to increase threshold

- Global Clover TPC **computed automatically** once per month on Jenkins for all repos combined

Source: http://massol.myxwiki.org/xwiki/bin/view/Blog/ComparingCloverReports
Backward Compatibility

• For APIs (and SPIs)

• Using the Revapi Maven plugin

• Supports source and binary compatibility

• Strategy:

  • Break the build on backward compatibility violations

  • Add ignores in pom.xml if really needed and ok

  • Add ignore list in release notes to warn users of your APIs
Backward Compatibility

• Strategy continued:

• Use `@Deprecated` annotation

• Once no more code uses deprecated API, move it to Legacy module. We don’t break backward compatibility!

• Use AspectJ in Legacy module to generate and aspectified API (JAR)

• Makes code clean and not able to use the Legacy modules (build-enforced)

• Distribute the legacy modules
Backward Compatibility

• Strategy continued:

  • For young APIs, use @Unstable + @Since
  • Enforced by build (to make sure @Since is there). Custom checkstyle rule (or Spoon rule)
  • Max duration is one cycle (i.e. 1 year).
  • Enforced by build (fails the build if beyond).

```java
/**
 * ...
 * @since 9.7RC1
 */
@Unstable
public List<MacroDescriptor> getMacroDescriptors(Syntax syntax) throws MacroLookupException {
  ...
}
```

Mention that as a result XWiki extensions and scripts can still run in XWiki even several years after they were released.
Mutation Testing

- Using PIT and Descartes

Concepts

- Modify code under test (mutants) and run tests
- Good tests kill mutants
- Generates a mutation score similar to the coverage %
- Descartes = extreme mutations that execute fast and have high values
Mutation - Descartes
## Pit Test Coverage Report

### Project Summary

<table>
<thead>
<tr>
<th>Number of Classes</th>
<th>Line Coverage</th>
<th>Mutation Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>80%</td>
<td>71%</td>
</tr>
<tr>
<td></td>
<td>294/369</td>
<td>110/155</td>
</tr>
</tbody>
</table>

### Breakdown by Package

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Classes</th>
<th>Line Coverage</th>
<th>Mutation Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.xwiki.rendering.internal.macro</td>
<td>4</td>
<td>81%</td>
<td>76%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>96/118</td>
<td>42/55</td>
</tr>
<tr>
<td>org.xwiki.rendering.internal.transformation.macro</td>
<td>3</td>
<td>83%</td>
<td>65%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>85/102</td>
<td>26/40</td>
</tr>
<tr>
<td>org.xwiki.rendering.macro</td>
<td>2</td>
<td>87%</td>
<td>69%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>48/55</td>
<td>20/29</td>
</tr>
<tr>
<td>org.xwiki.rendering.macro.descriptor</td>
<td>4</td>
<td>57%</td>
<td>55%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>36/63</td>
<td>11/20</td>
</tr>
<tr>
<td>org.xwiki.rendering.transformation</td>
<td>1</td>
<td>94%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>29/31</td>
<td>11/11</td>
</tr>
</tbody>
</table>

Report generated by PIT 1.2.3
Mutation - Example

```java
@Override
public boolean equals(Object object)
{
    boolean result;


    if ((object == null) || (object.getClass() != this.getClass()))
    {
        result = false;
    }
    else {
        MacroId macroId = (MacroId) object;
        result =
            (getId() == macroId.getId()) || (getId() != null && getId().equals(macroId.getId()))
            && (getSyntax() == macroId.getSyntax()) || (getSyntax() != null && getSyntax().equals(macroId.getSyntax()));
    }

    return result;
}
```
@Test
public void testEquality()
{
    MacroId id1 = new MacroId("id", Syntax.XWIKI_2_0);
    MacroId id2 = new MacroId("id", Syntax.XWIKI_2_0);
    MacroId id3 = new MacroId("otherid", Syntax.XWIKI_2_0);
    MacroId id4 = new MacroId("id", Syntax.XHTML_1_0);
    MacroId id5 = new MacroId("otherid", Syntax.XHTML_1_0);
    MacroId id6 = new MacroId("id");
    MacroId id7 = new MacroId("id");

    Assert.assertEquals(id2, id1);
    // Equal objects must have equal hashcode
    Assert.assertTrue(id1.hashCode() == id2.hashCode());

    Assert.assertFalse(id3 == id1);
    Assert.assertFalse(id4 == id1);
    Assert.assertFalse(id5 == id3);
    Assert.assertFalse(id6 == id1);

    Assert.assertEquals(id7, id6);
    // Equal objects must have equal hashcode
    Assert.assertTrue(id6.hashCode() == id7.hashCode());
}

Not testing for inequality!

Improved thanks to Descartes!
Mutation - Limitations

• Takes time to find interesting things to look at and decide if that’s an issue to handle or not. Need better categorisation in report:
  
  • *Strong pseudo-tested methods*: The worst! No matter what the return values are the tests always fail
  
  • *Pseudo-tested methods*: Grey area. The tests pass with at least one modified value.

• Multi module support - PITmp

• Slow on large projects (e.g. 7+ hours just for xwiki-rendering)
Mutation - Strategy

- Work in progress, no feedback yet! ☢️

- Fail the build when the mutation score of a given module is below a defined threshold in the pom.xml

- The idea is that new tests should, in average, be of quality equal or better than past tests.

- Other idea: hook on CI to run it only on modified code/tests.

**Ideally: replace coverage check by mutation check(∗)**

(∗) But too slow for now to replace coverage, can be done in addition (in a Maven profile for example, or executed on CI). Timeouts are a problem for example.
Mutation: Going further

- Using DSpot
- Uses PIT/Descartes but injects results to generate new tests
  - Adds assertions to existing tests
  - Generate new test methods
Generated test

```java
public void escapeAttributeValue2() {
    String escapedText = XMLUtils.escapeAttributeValue("a < a'
            && a' < a" => a < a" {";
    // AssertGenerator add assertion
    Assert.assertEquals("a &60; a&39; &#38;&#38; a&39; &#60; a&#34;; =&#62; a &60; a&34; &123;", escapedText);
    // AssertGenerator create local variable with return value of invocation
    boolean o_escapeAttributeValue__3 = escapedText.contains("<");
    // AssertGenerator add assertion
    Assert.assertFalse(o_escapeAttributeValue__3);
    // AssertGenerator create local variable with return value of invocation
    boolean o_escapeAttributeValue__4 = escapedText.contains(">");
    // AssertGenerator add assertion
    Assert.assertFalse(o_escapeAttributeValue__4);
    // AssertGenerator create local variable with return value of invocation
    boolean o_escapeAttributeValue__5 = escapedText.contains(""");
    // AssertGenerator add assertion
    Assert.assertFalse(o_escapeAttributeValue__5);
    // AssertGenerator create local variable with return value of invocation
    boolean o_escapeAttributeValue__6 = escapedText.contains("\"";
    // AssertGenerator add assertion
    Assert.assertFalse(o_escapeAttributeValue__6);
}
```

Original test

```java
public void escapeAttributeValue() {
    String escapedText = XMLUtils.escapeAttributeValue("a < a'
            && a' < a" => a < a" {";
    // AssertGenerator add assertion
    Assert.assertFalse("Failed to escape <", escapedText.contains("<");
    // AssertGenerator add assertion
    Assert.assertFalse("Failed to escape >", escapedText.contains(">");
    // AssertGenerator add assertion
    Assert.assertFalse("Failed to escape ", escapedText.contains("\"";
    // AssertGenerator add assertion
    Assert.assertFalse("Failed to escape &", escapedText.contains("&");
    // AssertGenerator add assertion
    Assert.assertFalse("Failed to escape {", escapedText.contains("{";
}
```
Mutation: Dspot Strategy

- DSpot is very slow to execute.

- One strategy is to run it on CI from time to time and in the pipeline commit generated tests in a different source root.

- Configure Maven to add a new test directory source using the Maven Build Helper plugin.

- Another idea: run it as GitHub commit hook so that it only executed on the modified code.
Environment Testing

- Environment = combination of Servlet container & version, DB & version, OS, Browser & version, etc

- Using Docker

- Need: Be able to run functional tests on local dev machines as well as on CI

  - Lead to using Fabric8 Docker Maven Plugin (DMP)
Environment

- One maven module to generate the XWiki Docker image for the official distribution
- Another maven module to generate the XWiki Maven Docker image (Maven + Browsers installed)
- In each functional test module, use the DMP to start the DB Docker image + the XWiki image + the XWiki Maven image.
- Execute Maven's verify goal inside the XWiki Maven image
Parting words

- Experiment, push the limit!

- Some other types of tests not covered and that also need automation
  - Performance/Stress testing
  - Usability testing
  - others?
Q&A

Me
Vincent Massol

vincent@xwiki.com
skype: vmassol
http://about.me/vmassol

http://xwiki.org
http://xwiki.com