Easier integration testing in OSGi: The open source osgi-test project

BJ Hargrave (IBM)
Raymond Augé (Liferay Inc.)
Jeremy Krieg (Greek Orthodox Archdiocese of Australia - Greek Welfare Centre of SA)
Who are we?

- BJ Hargrave
- Raymond Augé
- Fr Jeremy Krieg
What is osgi-test?

- Part of the OSGi Alliance & initiative of some of the core Bnd/Bndtools developers
- Goal: make it easier to write automated tests that run *inside* an OSGi framework.
- Is itself a series of OSGi bundles containing useful test utilities:
  - Common utility library (also used internally by other osgi-test bundles)
  - OSGi-specific AssertJ custom assertions
  - JUnit 4 TestRules and JUnit Jupiter test Extensions
- Uses Maven as the build environment using Bnd’s maven plugins (somewhat of a reference implementation for integration testing in a bnd-maven project).
- Current early-adopter release 0.10.0 available in Maven Central.
AssertJ support
What is AssertJ?

- [https://assertj.github.io/doc/](https://assertj.github.io/doc/), lead developer Joel Costigliola
- A series of open-source, domain-specific assertion libraries:
  - JDK (assertj-core)
  - Joda time
  - Guava
  - Neo4J
  - DB
  - Swing
- Core module ([https://github.com/assertj/assertj-core](https://github.com/assertj/assertj-core) - 1.8k stars, 432 forks) is a fork of the original *Fixtures for Easy Software Testing* (FEST) project by Alex Ruiz
  - [https://github.com/alexruiz/fest-assert-2.x/](https://github.com/alexruiz/fest-assert-2.x/)
Why AssertJ?

- Fluent API - easy to understand (even for non-developers).
- API design leverages the IDE’s autocompletion to make writing tests easier.
- Object-specific assertions provide more informative feedback about test failures.
- “Soft” assertions allow multiple assertions per test method (useful for slow code).
- Assertions on a single object can be chained, making code more concise.
- Extensible - lends itself to creating your own domain-specific assertions.
- Test framework agnostic - works in JUnit 4, JUnit 5, TestNG, etc.
- OSGi-friendly - the osgi-test team has made some contributions to assertj-core to make sure it works well in an OSGi framework.
Example - Code

- **Regular Jupiter assertions:**

  ```java
  assertEquals(bundle.getState() & RESOLVED, RESOLVED, "mybundle.state");
  assertEquals(bundle.getSymbolicName(), "my.bundle.name", "mybundle.bsn");
  ```

- **AssertJ Core:**

  ```java
  assertThat(bundle.getState() & RESOLVED).as("mybundle.state").isEqualTo(RESOLVED);
  assertThat(bundle.getSymbolicName()).as("mybundle.bsn").isEqualTo("my.bundle.name");
  ```

- **AssertJ custom assertion from osgi-test:**

  ```java
  assertThat(bundle).as("mybundle")
    .isInState(RESOLVED)
    .hasSymbolicName("my.bundle.name");
  ```
Example - IDE completion

```java
@Test
void assertJOSGi() {
    Bundle bundle = FrameworkUtil.getBundle(PlayerTest.class);
    assertThat(bundle).as("mybundle.state").isIn
}
```
assertJ example - regular Jupiter output

```java
assertEquals(bundle.getState() & RESOLVED, RESOLVED, "mybundle.state");
```
yields:

```
org.opentest4j.AssertionFailedError: mybundle.state ==> 
expected: <0> but was: <4>
```

(oops, accidentally swapped expected and actual...)
AssertJ example - regular AssertJ

```java
assertThat(bundle.getState() &
    RESOLVED).as("mybundle.state").isEqualTo(RESOLVED);
```
yields:

```java
org.opentest4j.AssertionFailedError: [mybundle.state]
Expecting:
<0>
to be equal to:
<4>
but was not.
(But what is state “4” again? And what state was it actually in?)
```
assertThat(bundle).as("mybundle").isInState(RESOLVED);

yields:
java.lang.AssertionError: [mybundle]
Expecting
   <my.test.project.test [7]>
to be in state:
   <4:RESOLVED>
but was in state:
   <32:ACTIVE>
JUnit support (4 & 5)
What is JUnit?

- JUnit 4 has been the mainstay of Java unit testing for some time.
- JUnit 5 is its successor, comprised of:
  - JUnit Platform - a framework for discovering and running tests for custom `TestEngine` implementations (1.7 just released).
  - JUnit Jupiter (Jupiter = 5th planet from the Sun) - a new `TestEngine` implementation (5.7 just released).
  - JUnit Vintage - a `TestEngine` wrapper around JUnit 4.
JUnit in OSGi

- JUnit 4 has had OSGi bundled versions for some time (e.g., Apache servicemix)
- Eclipse (since Oxygen) also ships with OSGi bundles of JUnit 5, available in Orbit.
- More recently, Ray Augé added OSGi metadata to the JUnit 5 jars to make them bundles (shipped since JUnit Platform 1.6/Jupiter 5.6/Vintage 5.6) so they can be used directly out of Maven Central.
- PDE supports JUnit 4 & 5. osgi-test should work with PDE (*note: this has not been tested by us*).
- The osgi-test team prefers to work with Bnd/Bndtools, which since 5.0 supports JUnit 5 through the `biz.aQute.tester.junit-platform tester bundle`. 
JUnit support in osgi-test

- BundleContextExtension/BundleContextRule
- ServiceExtension/ServiceRule
- ConfigurationExtension
- JUnit 4 rules have similar functionality to their counterpart Jupiter extensions, however...
- Most recent (and future) development has focused (and will focus) on Jupiter.
- This presentation focuses on the Jupiter extensions
- The osgi-test team recommends migrating to Jupiter where possible to fully leverage osgi-test.
JUnit support - BundleContextExtension

- Injects a BundleContext into the test as a field or test method parameter.
- Automatically rolls back changes to the BundleContext made within the test scope.
- Provides InstallBundle interface for installing *embedded* bundles into the running framework during the test.
- Supports multiple levels of scope (ie, @BeforeAll, @BeforeEach, @Nested, etc).
JUnit support - ServiceExtension

- Injects services into the test as a field or parameter.
- Configuration via the @InjectService annotation (cardinality, filter, timeout, etc)
- Takes care of:
  - setting up the ServiceTracker;
  - waiting for the required number of services to arrive (often a source of bugs if you try to roll your own);
  - releasing service references at the end of the test scope.
- Provides a ServiceAware interface for introspection of the underlying ServiceTracker state.
JUnit support - ConfigurationExtension

- Injects Configuration instances into the running test as a field or parameter.
- Allows you to declaratively create configurations with the OSGi ConfigurationAdmin service.
- Will automatically delete any configurations it has created once the test scope ends.
- No JUnit 4 TestRule counterpart implemented
-.osgi-test team wishes to acknowledge Stefan Bischof (another Bnd contributor) for this contribution.
Live TDD demonstration
Demo code repository

https://github.com/rotty3000/osgi-test-example-mvn
Conclusion
Try it out and then feedback and contribute!

- Available from Maven Central
  - OSGi is currently using osgi-test in some of its compliance tests for Release 8
    - Some projects like Apache Aries and idempiere-test are starting to use it
- GitHub repo
  - https://github.com/osgi/osgi-test/
- We would love any feedback
  - Open issues and PRs
Thank you!

Join the conversation:

@EclipseCon | #EclipseCon
Evaluate the Sessions

Sign in and vote at Eclipsecon.org:

-1  0  +1