Using JVM Testing

Jim Wilson
MOBILE SOLUTIONS DEVELOPER & ARCHITECT

@hedgehogjim   blog.jwhh.com

What to Expect from This Module

Testing Basics

Creating Unit Tests

Executing Unit Tests

Assuring Test Consistency

Test Driven Development

Testing

Testing needs to be a core task

- Essential to delivering quality software

Functional testing

- Verify behaves as expected
- Detect breaking changes

Testing

Unit testing

- Testing of units of code
- Each unit test is relatively simple
  - Tests a specific feature/behavior
- Generally will have many unit tests

Integration testing

- Testing of pieces being put together
- Application behaviors
- Often involves testing of UI

Unit Testing

Unit tests should be run often

- After code changes
- Before check-in to main source branch

Generally want to run all unit tests

- No change is complete until all tests pass

Ideally can be run reasonably quickly

Android Application Testing

Challenges of testing Android apps

- Full testing needs Android environment
- Requires emulator or a physical device

Need way to efficiently run unit tests

- Limit how often full environment needed

Efficiently Running Unit Tests

Android applications

- Java-based behavior
- Android-based behavior

Separate tests

- Tests for Java-based behavior
- Tests for Android-based behavior

Efficiently running unit tests

- Tests Java parts of app locally
- Leverage JVM on desktop

Local JVM Tests

Android Studio JVM testing

- Separate source set for JVM tests
- Uses JUnit 4

Managing tests with Android Studio

- Can run or debug tests
  - Single test, group of tests, or all tests
- Displays tests results
  - Success/failure indicated by color

Testing with JUnit

Each unit test is a separate method

- Marked with @Test annotation
- JUnit handles details of running method

Tests grouped within classes

- Primarily for organization convenience
- Allows execution grouping
- Allows setup/teardown grouping

Testing with JUnit

Assert class

- Use to indicate expected results
- Fails test when expectation not met

Testing with JUnit

Example Assert class methods

- assertSame
  - Two references to same object
- assertEqual
  - Two objects equal (equals method)
- assertNull
  - Reference is null

Negative versions of most methods

- i.e. assertNotSame, etc.

Assuring Test Consistency

Test reliability

- Each test must always run consistently
- Can’t depend on action of another test
- Can’t be impacted by side effects of other tests

Test must always start from same state

- Test order is not guaranteed
- Need way to set/reset test state

Assuring Test Consistency

Test pre-processing

- Method with @Before annotation
- Will run before each test in class

Test post-processing

- Method with @After annotation
- Will run after each test in class

Assuring Test Consistency

Once per class pre-processing

- Method with @BeforeClass annotation
- Will run once before all tests in class
- Method must be static

Once per class post-processing

- Method with @AfterClass annotation
- Will run once after all tests in class
- Method must be static

Assuring Test Consistency

Multiple pre/post-processing methods

- Multiples in a class valid
- All will run
- Sequence is not guaranteed

Summary

Testing needs to be a core task

- Essential to delivering quality software

Unit Testing

- Units of relatively simple tests
- Focus is on specific feature/behavior
- Tests should be run frequently

Summary

Local JVM testing

- Focused on Java aspects of our app
- Run directly on desktop

Managing tests in Android Studio

- Can run or debug tests
- Displays test success or failure

Summary

Test methods

- Marked with @Test annotation
- Grouped in classes

Assert class

- Use to indicate expected results
- Fails test when expectation not met

Summary

Tests need to be reliable

- Need to assure consistent testing

Can run test pre/post-processing

- Use @Before and @After
- Methods run for each @Test method

Can run test class pre/post-processing

- Use @BeforeClass and @AfterClass
- Methods run once for test class
- Methods must be static