Unit Testing With at-should Annotation

Please note that the @should annotation and the corresponding Eclipse plugin are no longer supported. This page is kept out of historical interest.

Simple JUnit testing usually means writing one large test for each method. However, when a test breaks, its often very difficult to debug, fix, or know if it should be removed.

Enter Behavior Driven Development (BDD). *However*, our current situation prevents us from jumping all the way to full BDD.

Enter Dan North's introduction to BDD. The first several points about sentence styled test names and its ability to keep tests simple really are what we want to adopt:

- Each unit test name should be a sentence
- Each unit test should only test one thing
- That one thing the unit test focuses on should be a behavior of the test, not the whole test

The Problem

1. Test driven development is hard.
    - The developer routinely does not have the time/desire/gumption to create tests before/during development. However, the developer is usually the best one to know what a method is meant
2. We have a large number of existing methods that do not have tests
    - Only the more experienced OpenMRS developers know what each method does and the tests needed
3. We potentially have a large number of developers that have a small amount of time to be able to donate
    - Tests are in one class, methods in the other, which ones are done? which methods need more?

Our Solution

Introducing the "@should" javadoc annotation. Each api method will get one or more @should annotations that simply state a behavior of that method that needs testing:

- @should not fail given null parameter
- @should return empty list if no results
- @should execute in less than thirty seconds
- @should find patient given partial name
- @should allow, not allow, fail, return something to, on, with, when, for doing something else

The four examples above would become unit tests named (given the method was findPatient):

- public void findPatient_shouldNotFailGivenNullParameter()
- public void findPatient_shouldReturnEmptyListIfNotResults()
- public void findPatient_shouldExecuteInLessThanThirtySeconds()
- public void findPatient_shouldFindPatientGivenPartialName()

See our Conventions for using @should annotations page for more detail on how @should annotations should be phrased.

Eclipse Plugin To Make It Easy

Typing out the format for each @should annotation and each junit test is tedious. We created an eclipse plugin that will create the unit test and annotation to match a given @should annotation.

Generate Test Case Plugin

Example

In /src/api/org/openmrs/somepackage/SomeObject.java:

```
package org.openmrs.somepackage;

public class SomeObject {

  **
  * (Descriptive text)
  *
  * @param x (descriptive text)
  * @return (descriptive text)
  *  
  * *@should get name as written by user with spaces*
  */
  public y methodName(x) {
    // do stuff
  }
}
```

In /test/api/org/openmrs/somepackage/SomeObjectTest.java:

```
package org.openmrs.somepackage;

public class SomeObjectTest {

  **
  * @see \{@link SomeObject#methodName()\}
  * @verifies get name as written by user with spaces
  */
  @Test
  public void methodName_shouldGetNameAsWrittenByUserWithSpaces() throws Exception \{
    //TODO auto-generated
    Assert.fail("Not yet implemented");
  }

}
```