

CHICAGO

INTERNATIONAL  
SOFTWARE DEVELOPMENT  
CONFERENCE 2015

goto;  
conference

# Level Up Your Automated Tests

*Trisha Gee @trisha\_gee*



Using <Technology X>  
Will Fix Your Problems

Why Don't We Write  
Tests?



# What Are Tests For?

Check it

works

What Are Tests Actually  
Good For?

What testing  
is good for



# **How Can We Change ~~Attitudes~~?**

# **How Can We Change Behaviour?**

# Having a Champion

Quality becomes a habit

This can only get you so  
far

# Remaining Problems

Complicated matrix  
of capabilities

Only Happy Path



No unit tests

lots of setup

Testing too  
many things

Horrible test  
names

Difficult to

see what's under  
test

Hard to  
read

Many similar  
tests

It needs to be easy



# Possible Solutions

- EasyMock / Mockito / JMock
- Home-grown mocking/stubbing
- Standards / Examples
- DSL – Domain Specific Language
- Hamcrest matchers
- Spock

**Along came Spock**

# DBCollectionFunctionalSpecification

```
def 'should update multiple documents'() {  
  given:  
  collection.insert([  
    [x: 1] as BasicDBObject,  
    [x: 1] as BasicDBObject])  
  
  when:  
  collection.update([  
    [x: 1] as BasicDBObject,  
    [$set: [x: 2]] as BasicDBObject,  
    false, true);  
  
  then:  
  collection.count([  
    [x: 2] as BasicDBObject]) == 2  
}
```

How it fixes the  
problems

# Hard to read

```
def 'should update multiple documents'() {  
  given:  
  collection.insert([  
    [x: 1] as BasicDBObject,  
    [x: 1] as BasicDBObject])  
  
  when:  
  collection.update([  
    [x: 1] as BasicDBObject,  
    [$set: [x: 2]] as BasicDBObject,  
    false, true);  
  
  then:  
  collection.count([x: 2] as BasicDBObject) == 2  
}
```

# Horrible Test Names

```
def 'should update multiple documents'() {
```

# Difficult to tell what's under test

```
def 'should return the name of the collection the results are contained in if it
given:
  def expectedCollectionName = 'collectionForResults';
  def outputCollection = database.getCollection(expectedCollectionName)
  def results = outputCollection.find()

  @Subject
  def mapReduceOutput = new MapReduceOutput(new BasicDBObject(), results, null,

  when:
  def collectionName = mapReduceOutput.getCollectionName();

  then:
  collectionName != null
  collectionName == expectedCollectionName
}
```

# Lots of setup

```
def 'should return the name of the collection the results are contained in if it
given:
  def expectedCollectionName = 'collectionForResults';
  def outputCollection = database.getCollection(expectedCollectionName)
  def results = outputCollection.find()

  @Subject
  def mapReduceOutput = new MapReduceOutput(new BasicDBObject(), results, null,

  when:
  def collectionName = mapReduceOutput.getCollectionName();

  then:
  collectionName != null
  collectionName == expectedCollectionName
}
```



# No Unit Tests

```
class IterableCodecSpecification extends Specification {  
    def bsonWriter = Mock(BsonWriter)  
  
    @Subject  
    private final IterableCodec iterableCodec = new IterableCodec();  
  
    public void 'should encode list of strings'() {  
        given:  
            List<String> stringList = ['Uno', 'Dos', 'Tres']  
  
        when:  
            iterableCodec.encode(bsonWriter, stringList, EncoderContext.builder().build())  
  
        then:  
            1 * bsonWriter.writeStartArray()  
        then:  
            1 * bsonWriter.writeString('Uno')  
        then:  
            1 * bsonWriter.writeString('Dos')  
        then:  
            1 * bsonWriter.writeString('Tres')  
        then:  
            1 * bsonWriter.writeEndArray();  
    }  
}
```

# Too Few Unhappy Paths

```
def 'should throw Exception if URI does not have a trailing slash'() {  
  when:  
    new MongoClientURI('mongodb://localhost?wtimeoutMS=5');  
  
  then:  
    thrown(IllegalArgumentException)  
}
```

```
def 'should not throw an Exception if URI contains an unknown option'() {  
  when:  
    new MongoClientURI('mongodb://localhost/?unknownOption=5');  
  
  then:  
    notThrown(IllegalArgumentException)  
}
```

# Too many similar tests

```
@Unroll
def 'should sort with #sortBy and filter with #criteria before selecting first result'() {
  given:
  collection.drop()
  collection.insert([_id: 1, x: 100, y: 'abc'] as BasicDBObject)
  collection.insert([_id: 2, x: 200, y: 'abc'] as BasicDBObject)
  collection.insert([_id: 3, x: 1, y: 'abc'] as BasicDBObject)
  collection.insert([_id: 4, x: -100, y: 'xyz'] as BasicDBObject)
  collection.insert([_id: 5, x: -50, y: 'zzz'] as BasicDBObject)
  collection.insert([_id: 6, x: 9, y: 'aaa'] as BasicDBObject)

  expect:
  collection.findOne(criteria, null, sortBy)['_id'] == expectedId;

  where:
  criteria                                | sortBy                                | expectedId
  new BasicDBObject()                    | [x: 1] as BasicDBObject              | 4
  new BasicDBObject()                    | [x: -1] as BasicDBObject             | 2
  [x: 1] as BasicDBObject                 | [x: 1, y: 1] as BasicDBObject       | 3
  QueryBuilder.start('x').lessThan(2).get() | [y: -1] as BasicDBObject            | 5
}
```

# Testing too many things

```
@Unroll
def 'should sort with #sortBy and filter with #criteria before selecting first result'() {
  given:
  collection.drop()
  collection.insert([_id: 1, x: 100, y: 'abc'] as BasicDBObject)
  collection.insert([_id: 2, x: 200, y: 'abc'] as BasicDBObject)
  collection.insert([_id: 3, x: 1, y: 'abc'] as BasicDBObject)
  collection.insert([_id: 4, x: -100, y: 'xyz'] as BasicDBObject)
  collection.insert([_id: 5, x: -50, y: 'zzz'] as BasicDBObject)
  collection.insert([_id: 6, x: 9, y: 'aaa'] as BasicDBObject)

  expect:
  collection.findOne(criteria, null, sortBy)['_id'] == expectedId;

  where:
  criteria                                | sortBy                                | expectedId
  new BasicDBObject()                    | [x: 1] as BasicDBObject              | 4
  new BasicDBObject()                    | [x: -1] as BasicDBObject             | 2
  [x: 1] as BasicDBObject                 | [x: 1, y: 1] as BasicDBObject       | 3
  QueryBuilder.start('x').lessThan(2).get() | [y: -1] as BasicDBObject            | 5
}
```

# Testing too many things

The screenshot shows a test runner interface with a sidebar on the left and a main panel on the right. The sidebar lists four test cases under the heading "DBCollectionFunctionalSpecification". The second test case, "should sort with [x:-1] and filter with [:] before selecting first result (DBColl", is marked with a red exclamation mark icon, indicating it failed. The other three test cases are marked with green "OK" icons. The main panel displays the error message "Condition not satisfied:" followed by a comparison of the actual result with the expected result. The actual result is a document with fields "\_id" (value 2) and "x" (value -1). The expected result is a document with fields "\_id" (value 3) and "x" (value -1). The comparison shows that the "\_id" field does not match, resulting in a "false" value. Below the comparison, there is a link to "Click to see difference".

Done: 4 of 4 Failed: 1 (in 0.879 s)

DBCollectionFunctionalSpecification

- OK should sort with [x:1] and filter with [:] before selecting first result (DBColl
- ! should sort with [x:-1] and filter with [:] before selecting first result (DBColl
- OK should sort with [x:1, y:1] and filter with [x:1] before selecting first result (DBColl
- OK should sort with [y:-1] and filter with [x:\${t:2}] before selecting first result (DBColl

Condition not satisfied:

```
collection.findOne(criteria, null, sortBy)['_id'] == expectedI
```

		[:]	[x:-1]	2	3
		[_id:2, x:200, y:abc]			false

DBCollection{database=DB{name='JavaDriverTest'}, name='DBColle

[Click to see difference](#)

# Complicated Matrix

```
@IgnoreIf({ serverVersionAtLeast(asList(3, 0, 0)) })
def 'should support legacy dropDups when creating a unique index'() {
    when:
        collection.drop()
```

```
@IgnoreIf({ System.getProperty('java.version').startsWith('1.6.') })
def 'should be equal to another MongoClientURI with the same string values'() {
    expect:
        uri1 == uri2
```

```
@IgnoreIf({ !ClusterFixture.isDiscoverableReplicaSet() })
def 'should throw bulk write exception with a write concern error w'() {
    given:
        def op = new MixedBulkWriteOperation(getNamespace())
```

Proving itself

# Issues



How can we:

**a) write tests?**

How can we:

**b) write readable tests?**

How can we:

**c) write meaningful  
tests?**

# Conclusions

**Make it easy**

**Automate everything**

Zero tolerance for  
failures

Have a champion



Let it go

# Pairing or code review

Focus on the purposes  
of testing

# Resources

**<http://bit.ly/GroovyVsJava>**

@trisha\_gee

CHICAGO

INTERNATIONAL  
SOFTWARE DEVELOPMENT  
CONFERENCE 2015

goto;  
conference

# Questions?

*Please remember to evaluate via the GOTO  
Guide App*

# ENTER to WIN FREE VIDEO TRAINING



**Come to the Meet and Greet and  
Enter to Win**

**When: Today! During the 11:50 break**

**Where: Power Lounge**

**Located in the Executive Room off the exhibit hall**

Save 50% off at  
[informit.com/gotochgo](http://informit.com/gotochgo)