










Java Testing on the Fast Lane

About the speaker

-  Java developer since the early days
-  Open source believer since 1997
-  Member of the Groovy Dev team since 2007
-  Co-founder of the Griffon project
-  <http://jroller.com/aalmiray>
-  @aalmiray

Agenda

-  What is Groovy
-  Groovy + Testing Frameworks
-  How Groovy Helps
-  Mocking with Groovy
-  XML Processing
-  Functional Testing
-  Resources

What is Groovy

- Ⓜ Dynamic and agile language for the JVM
- Ⓜ Built upon the strenghts of the Java Language
- Ⓜ Brings modern features from Python, Ruby & Smalltalk
- Ⓜ Works perfectly with Java
- Ⓜ Supports DSLs and compact syntax
- Ⓜ Makes programming fun again

HelloWorld.java

```
public class HelloWorld {  
    String name;  
  
    public void setName(String name)  
    { this.name = name; }  
    public String getName(){ return name; }  
  
    public String greet()  
    { return "Hello " + name; }  
  
    public static void main(String[] args){  
        HelloWorld helloWorld = new HelloWorld();  
        helloWorld.setName("Groovy");  
        System.out.println(helloWorld.greet());  
    }  
}
```

HelloWorld.groovy

```
public class HelloWorld {  
    String name;  
  
    public void setName(String name)  
    { this.name = name; }  
    public String getName(){ return name; }  
  
    public String greet()  
    { return "Hello "+ name; }  
  
    public static void main(String[] args){  
        HelloWorld helloWorld = new HelloWorld();  
        helloWorld.setName("Groovy");  
        System.out.println(helloWorld.greet());  
    }  
}
```

GroovierHelloWorld.groovy

```
class HelloWorld {  
    String name  
    def greet() { "Hello $name" }  
}  
  
def helloWorld = new HelloWorld(name: "Groovy")  
println helloWorld.greet()
```

Groovy loves Java

- A Java class is a Groovy and viceversa
- No artificial bridge between languages
- Full JDK5 support: annotations, generics, varargs, enums
- 98% of Java code is valid Groovy code
- Rename *.java to *.groovy and compile!

Scott Davis' 1st Mantra

Groovy is Java & Java is Groovy



Groovy provides...

Closures!, operator overloading, turbocharged POJOs, native syntax for Maps & Lists & Ranges, iterator methods, regexps as first-class citizens, metaprogramming capabilities, optional typing, dynamic types, everything is an object, and more!

Scott Davis' 2nd Mantra

Groovy is Java & Groovy is NOT Java



Groovy is what the Java language would look like had it be written in the 21st century



Groovy + Testing Frameworks

- Any Groovy script may become a testcase

assert keyword enabled by default

- Groovy provides a `GroovyTestCase` as base class

Easier to test exception throwing code

- JUnit 4.x and TestNG ready. Groovy supports JDK5

Annotations, generics, static imports


How Groovy Helps

 Write less with optional keywords

`public, return, arg types & return types`

 Terser syntax for property access

 Native syntax for Lists and Maps

 Closures

 AST Transformations – compile time metaprogramming

Accessing Properties

// Java

```
public class Bean {  
    private String name;  
    private void setName(String n) { name = n; }  
    private String getName() { return name; }  
}
```

// Groovy

```
Bean bean = new Bean(name: "Duke")  
assert bean.name == "Duke"  
bean.name = "Tux"  
assert bean.name == "Tux"  
assert bean.name == bean.getName()
```


Native Syntax for Maps/Lists

```
Map map = [:]
assert map instanceof java.util.Map
map["key1"] = "value1"
map.key2 = "value2"
println map
assert map.size() == 2
assert map.key1 == "value1"
assert map["key2"] == "value2"

List list = []
assert list instanceof java.util.List
list.add("One")
list << "Two"
println list
assert list.size() == 2
assert ["One", "Two"] == list
```


Closures (1)

```
int count = 0
closure = { k ->
    0.upto(k) { count += it }
}

closure(10)
assert count == (10*11)/2

runnable = closure.curry(20) as Runnable
assert runnable instanceof java.lang.Runnable
count = 0
runnable.run()
assert count == (20*21)/2
```

Closures (2)

```
getSlope = { x, y, b = 0 ->  
  println "x: $x, y: $y, b: $b"  
  (y - b) / x  
}
```

```
assert 1 == getSlope(2, 2)  
getSlopeX = getSlope.curry(5)
```

```
assert 1 == getSlopeX(5)  
assert 0 == getSlopeX(2.5, 2.5)
```


AST Transformations

```
import java.text.SimpleDateFormat
class Event {
    @Delegate Date when
    String title, url


    String toString() {
        "title: $title, url: $url
when: $when"
    }
}

df = new SimpleDateFormat("MM/dd/yyyy")
so2gx = new Event(title: "SpringOne2GX",
    url: "http://springone2gx.com",
    when: df.parse("10/19/2009"))
oredev = new Event(title: "Oredev",
    url: "http://oredev.org",
    when: df.parse("11/02/2009"))
println so2gx
println oredev
assert oredev.after(so2gx.when)
```

AST Transformations

 @Singleton

 @Lazy

 @Immutable

 @Bindable

 @Newify

 @Delegate

 And a few more...

But how do I run it?

Pick your favorite IDE!

IDEA

Eclipse

NetBeans

Command line tools

Ant / Gant

Maven / GMaven

Gradle

Good ol' Groovy shell/console

Testing Exceptions (Java)

```
public class MyService {  
    public void doSomething() {  
        throw new UnsupportedOperationException();  
    }  
}  
  
public class JavaExceptionTestCase extends TestCase {  
    public void testExceptionThrowingCode() {  
        try {  
            new MyService().doSomething();  
            fail("MyService.doSomething has been implemented");  
        } catch ( UnsupportedOperationException expected ) {  
            // everything is ok if we reach this block  
        }  
    }  
}
```

Testing Exceptions (Groovy)

```
public class MyService {  
    public void doSomething() {  
        throw new UnsupportedOperationException();  
    }  
}  
  
class GroovyExceptionTest extends GroovyTestCase {  
    void testExceptionThrowingCode() {  
        shouldFail( UnsupportedOperationException ){  
            new MyService().doSomething()  
        }  
    }  
}
```

Mocking with Groovy

- Use known (Java) mocking libraries

Easymock – record/replay

JMock – write expectations as you go

Mockito – new kid on the block

- Use dynamic proxies as stubs

- Use StubFor / MockFor

Inspired by Easymock

No external libraries required

Dynamic Proxies

```
class StringProvider {
    String getString() { "" }
}

class StringDecorator {
    // This is a property declaration, meaning that
    // the Groovy compiler will generate a pair of
    // get/set methods
    StringProvider provider

    def getValue() { provider.string + "Decorated" }
}

// Here comes the proxy
def provider = [
    getString: { -> "Groovy" }
] as StringProvider
// it looks like JSON, doesn't it?

def decorator = new StringDecorator( provider: provider )
// the following would have worked too
// def decorator = new StringDecorator()
// decorator.setProvider( provider )
assert "GroovyDecorated" == decorator.value
assert decorator.provider instanceof StringProvider
```

StubFor/MockFor

- Caller – Collaborator
- Mocks/Stubs define expectations on collaborators

Mocks are strict

Stubs are relaxed

- CAVEAT – caller must be Groovy

StubFor Example

```
import groovy.mock.interceptor.StubFor

class StringProvider {
    String getString() { "" }
}

class StringDecorator {
    StringProvider provider = new StringProvider()
    String getValue(){
        provider.string + "Decorated"
    }
}

def providerStub = new StubFor(StringProvider)
providerStub.demand.getString() { "Groovy" }
providerStub.use {
    def decorator = new StringDecorator()
    assert "GroovyDecorated" == decorator.value
}
```

XML Processing

- DbUnit: a JUnit extension for testing databases
- Several options at your disposal

Old school – extend DatabaseTestCase

Flexible – use an IDatabaseTester impl

Roll your own Database testcase

Inline XML Dataset

```
import org.dbunit.*
import org.junit.*

class MyDBTestCase {
    IDatabaseTester db

    @BeforeClass void init(){
        db = new JdbcDatabaseTester("org.hsqldb.jdbcDriver",
            "jdbc:hsqldb:sample", "sa", "" )
        def dataset = """
        <dataset>
            <company name="Acme"/>
            <employee name="Duke", company_id="1"/>
        </dataset>
        """
        db.dataset = new FlatXmlDataSet( new StringReader(dataset) )
        db.onSetUp()
    }

    @AfterClass void exit() { db.onTearDown() }
}
```

Compile Checked Dataset

```
import org.dbunit.*
import org.junit.*

class MyDBTestCase {
    IDatabaseTester db

    @BeforeClass void init(){
        db = new JdbcDatabaseTester("org.hsqldb.jdbcDriver",
            "jdbc:hsqldb:sample", "sa", "" )
        def dataset = new MarkupBuilder().dataset {
            company( name: Acme )
            employee( name: "Duke", company_id: 1 )
        }
        db.dataset = new FlatXmlDataSet( new StringReader(dataset) )
        db.onSetUp()
    }

    @AfterClass void exit() { db.onTearDown() }
}
```

Functional Testing

- Ⓜ These tests usually require more setup
- Ⓜ Non-developers usually like to drive these tests
- Ⓜ Developers usually don't like to code these tests
- Ⓜ What can we do about this?

Groovy to the rescue!

Web:

Canoo WebTest – Leverages AntBuilder

Tellurium – a Groovier Selenium

Desktop:

FEST – next generation Swing Testing

BDD:

Easyb, Spock, JBehave, Cuke4duke

Resources

 <http://groovy.codehaus.org>

 <http://groovy.dzone.org>

 <http://easytesting.org>

 <http://easyb.org>

This presentation made with



Q & A

Thank you!