

Introduction to Groovy

Scott Hickey
Bass & Associates



Agenda

- Overview of the Language
- Groovy Code Examples
- Making Java Better with Groovy
- Practical Things to Know

Overview

- Simple Definition
- How Is Groovy Being Used

What is Groovy?

Groovy is an agile dynamic language for the Java Platform with many features that inspired languages like Python, Ruby and Smalltalk, making them available to Java developers using a Java-like syntax.

What is Groovy?

- Dynamic – interpreted, dynamically typed
- Java Platform – compiles to Java byte code
- Ruby, Smalltalk – everything is an object
- Java Syntax – looks kind of like Java

What is Groovy?

- Dynamic
 - Optional typing
 - Optional Compilation
- Relaxed Script-like Syntax
 - Optional semicolons
 - Optional parentheses

Scripting Example

```
/* linecount.groovy */
count = 0
file = new File(args[0]).eachLine {
    count++
}
println "line count for ${args[0]} is ${count}"
```

```
>cat fileReading.txt
Scott,Hickey,15.2
Tom,Hickey,32
Jerry,Hickey,13.4
```

```
>groovy lineCount fileReading.txt
line count for fileReading.txt is 3
```

Scripting Example

```
>groovyc lineCount.groovy  
>java lineCount fileReading.txt  
line count for fileReading.txt is 3
```

What is Groovy? Java

- Run anywhere there's a JVM
- Use Java Libraries
- Embed in a Desktop Java Application
- Embed in a J2EE Application
- Test with JUnit
- Debug with Eclipse, JSwat

Like Ruby and Smalltalk

- Except when it's not :)
- Everything's an object
- Closures / Code Blocks
- Compact code
- Meta Object Protocol

Sample Code

```
// everything's an object - numbers
x = 1          // Integer, not int
y = 1.1       // BigDecimal, not float or double
println x + y

// everything's an object - code
f = {k,v -> println "key=${k}, value=${v}" }
me = [first:"scott",last:"hickey", golfHandicap:14.3]
me.each( f )
```

```
>groovy likeRuby
2.1
key=golfHandicap, value=14.3
key=first, value=scott
key=last, value=hickey
```

What is Groovy? Like

- Except when it's not :)
- Semicolons are optional
- Triple quoted strings
- GStrings
- Multiple public classes in the same file
- No anonymous inner classes
- == means .equals()

Overview

- Simple Definition
- How Is Groovy Being Used

What is it good for?

- Scripting
- Prototype Development
- Readability
- Sweet Spot: decimal math, collections, XML, Java libraries

What's it good for?

- Custom MQ implementation
- Rails like web application framework
- Financial applications
- Testing

Agenda

- Overview of the Language
 - ✓ *Simple Definition*
 - ✓ *How Is Groovy Being Used*
- Groovy Code Examples
- Making Java Better with Groovy

Groovy Code

- Decimal Math
- GJDK: Collection, File
- SQL – DataSet
- XML - GPath
- Closures - Benchmark Example
- Swing Example
- GSAP dynamic CSV indexing

Decimal Math Example

```
BigDecimal interpolate(final BigDecimal lowerBound,
    final BigDecimal upperBound, final BigDecimal value,
    final BigDecimal lowerValue, final BigDecimal upperValue) {

    final BigDecimal uMinusv = upperBound.subtract(value);
    final BigDecimal vMinusl = value.subtract(lowerBound);
    final BigDecimal uMinusl = upperBound.subtract(lowerBound);
    return lowerValue.multiply(uMinusv)
        .add(upperValue.multiply(vMinusl))
        .divide(uMinusl, 10, BigDecimal.ROUND_HALF_UP);
}
```

Decimal Math Example

```
def interpolate(lowerBound, upperBound,  
               value, lowerValue, upperValue) {  
    return (lowerValue * (upperBound-value) +  
           upperValue * (value-lowerBound) ) /  
           (upperBound-lowerBound)  
}
```

A Groovy Closure is...

- Abstraction of execution
- Like an anonymous function
- Carries lexical scope and can modify variables in that scope

Groovy Code -

```
// cool things to do with lists
golfer = ["scott","hickey",14.3]
golfer.each { println it}
csv = golfer.join(",")
println csv

golfer.eachWithIndex {obj,i->
    println "item ${i} is ${obj}"
}
println golfer[1]

golfers = []
golfers << golfer
golfers << ["tom","hickey",30.0]
golfers << ["jerry","hickey",12.2]
me = golfers.find { it.contains("scott") }
println me
```

Collections

- Methods work on Arrays, Lists, Maps
- any, collect, count, each, every, find, findAll, getAt, inject, join, leftShift, max, min, sort, sum

File Example

```
// Read a CSV file and
// parse each line into a list a values

new File("fileReading.txt").splitEachLine(",") { line ->
    println line
}
```

Output:

```
[ "Scott", "Hickey", "15.2"]
[ "Tom", "Hickey", "32"]
[ "Jerry", "Hickey", "13.4"]
```

File Input / Output

- append, eachByte, eachDir, eachFile, eachFileRecurse, eachLine, filterLine, getText, readBytes, readLines,

Extend these examples

- Readers and Writers
- Input and Output Streams
- Sockets
- Strings
- Regular Expression operator
- Application Classes

Groovy Code

```
import groovy.sql.Sql

def sql = Sql.newInstance("jdbc:h2:golfscore",
    "scott","","org.h2.Driver")

def golferDDL = """
create table golfers(
    ID INT IDENTITY PRIMARY KEY,
    firstName VARCHAR,
    lastName VARCHAR,
    handicap DECIMAL(3,1)
);
"""

sql.execute("drop table golfers")
sql.execute(golferDDL)
```

Groovy Code

```
import groovy.sql.Sql

def sql = Sql.newInstance("jdbc:h2:golfscore",
                        "scott","","org.h2.Driver")

def golfers = sql.dataSet("golfers")

new File("golfers.txt").splitEachLine(",") { line ->
    golfers.add(
        firstName:line[0],
        lastName:line[1],
        handicap:line[2]
    )
}
```

Groovy Code

```
import groovy.sql.Sql

p = { println "${it.id} ${it.firstName} ${it.lastName} ${it.handicap}" }

def sql = Sql.newInstance("jdbc:h2:golfscore", "scott","","", "org.h2.Driver")

def golfers = sql.dataSet("golfers")

me = golfers.findAll {it.firstName == 'Scott' && it.lastName == 'Hickey' }

me.each( p )

sql.eachRow("select * from golfers where firstName = 'Scott'",p)
```

Groovy Code Examples:

```
import groovy.sql.Sql
sql = Sql.newInstance("jdbc:h2:golfscore","scott","", "org.h2.Driver")
writer = new StringWriter()
xmlBuilder = new groovy.xml.MarkupBuilder(writer)
xmlBuilder.golfers() {
    sql.eachRow("select * from golfers") { rs->
        golfer(id:rs.id, firstName:rs.firstName, lastName:rs.lastName) {
            handicap(rs.handicap)
        }
    }
}
new File("golfers.xml").withPrintWriter{ w->
    w.println writer.toString()
}
```

```
<golfers>
  <golfer firstName='Scott' id='1' lastName='Hickey'>
    <handicap>15.2</handicap>
  </golfer>
  <golfer firstName='Tom' id='2' lastName='Hickey'>
    <handicap>32.0</handicap>
  </golfer>
</golfers>
```

Groovy Code

```
node = new XmlSlurper().parse("golfers.xml")
golfers = node.golfer
me = golfers.find { it['@firstName'] == 'Scott' }
println "${me['@firstName']} has a handicap of ${me.handicap}"
```

```
>groovy findGolfersXml.groovy
Scott has a handicap of 15.2
```

Benchmark Example

```
public class GolferUtils {  
    // Returns the execution time in seconds  
  
    public static benchmark(closure) {  
        def start = System.currentTimeMillis()  
        closure.call()  
        def now = System.currentTimeMillis()  
        return (now - start) / 1000F  
    }  
}
```

Benchmark Example –

```
public class GolferUtilsTest extends GroovyTestCase {
    public void testBenchmark() {
        def x = 0
        def time = 0
        time = GolferUtils.benchmark {
            (1..10000).each { x += it }
        }
        println "benchmark time = ${time}"
        assert time > 0
    }
}
```

```
>groovy GolferUtilsTest
.benchmark time = 0.14
```

```
Time: 0.187
```

```
OK (1 test)
```

Groovy GJDK and

- Ant
- Iterators
- File I/O
- JDBC I/O
- GroovyBeans
- Regex
- Operator Overloading
- XML Builder
- Swing Builder
- Reflection / MOP
- Groovlets/GSP
- GroovySOAP
- XML-RPC

Agenda

- Overview of the Language
- Groovy Code Examples
- Making Java Better with Groovy
- Practical Things to Know

Java Idioms We'd Like

- JavaBeans
- Iterators
- Comparators
- Exceptions
- Interfaces
- Switch limitations

Improving Java

- Reducing Noise IBM Developworks Article

Agenda

- Overview of the Language
- Groovy Code Examples
- Making Java Better with Groovy
- Practical Things to Know

Things to Know

- Pre JSR vs. JSR
- FUD
- IDE Plugins
- It is in production
- Groovy is good enough already
- Groovy is more than scripting

My Keys to Groovy

- Start with layers that don't rely on frameworks
- Use ANT for builds
- Work with compiled code
- Write tests
- Monitor Groovy mail list
- Post questions
- Test Groovy libraries before incorporating into a project
- Use groovy-all.jar

Key's to Groovy's

- Documentation – Groovy in Action, Learning to Program, Grails Books
- 1.0 Release
- Success Stories
- IDE Support – *I'm working on it!*

Thanks!

For more information :

- groovy.codehaus.org
- **Practically Groovy** series @ ibm.com/java
- jshickey@yahoo.com