Advanced Groovy

Rod Cope

CTO and Founder
OpenLogic, Inc.
http://www.openlogic.com

TS-9720

Groovy Goal

What you'll get out of this session

Learn the most powerful features of Groovy and put them to use today!

Agenda

Code Sample

Basic Features

Markup

Advanced Features

Demos

Extras

Conclusion

Agenda

Code Sample

Basic Features

Markup

Advanced Features

Demos

Extras

Conclusion

Sample: Java™ Code and Groovy!

```
public class Filter {

  public static void main( String[] args ) {
     List list = new ArrayList();
     list.add("Rod");  list.add("Neeta");
     list.add("Eric"); list.add("Missy");


     Filter filter = new Filter();
     List shorts = filter.filterLongerThan(list, 4);
     System.out.println(shorts.size());


     Iterator iter = shorts.iterator();
     while (iter.hasNext()) {
       System.out.println(iter.next());
     }
   }

  public List filterLongerThan(List list, int length) {
     List result = new ArrayList();
     Iterator iter = list.iterator();
     while (iter.hasNext()) {
       String item = (String) iter.next();
       if (item.length() <= length) { result.add(item); }
     }
     return result;
   }

}
```

Sample: Groovy!

```
def list = ["Rod", "Neeta", "Eric", "Missy"]
def shorts = list.findAll { it.size() <= 4 }
println shorts.size()
shorts.each { println it }
```

```
-> 2
-> Rod
   Eric
```

Sample in Java Code (27 Lines)

```
public class Filter {

  public static void main( String[] args ) {
     List list = new ArrayList();
     list.add("Rod");  list.add("Neeta");
     list.add("Eric"); list.add("Missy");


     Filter filter = new Filter();
     List shorts = filter.filterLongerThan(list, 4);
     System.out.println(shorts.size());


     Iterator iter = shorts.iterator();
     while (iter.hasNext()) {
       System.out.println(iter.next());
     }
   }

  public List filterLongerThan(List list, int length) {
     List result = new ArrayList();
     Iterator iter = list.iterator();
     while (iter.hasNext()) {
       String item = (String) iter.next();
       if (item.length() <= length) { result.add(item); }
     }
     return result;
   }

}
```

Sample in Groovy (4 lines)

```
def list = ["Rod", "Neeta", "Eric", "Missy"]
def shorts = list.findAll { it.size() <= 4 }
println shorts.size()
shorts.each { println it }
```

Agenda

Code Sample

Basic Features

Markup

Advanced Features

Demos

Extras

Conclusion

Basic Features

- Dynamic and (optional) static typing

```
int a = 2
def str = "Hello"
```

- Native syntax for lists, maps, arrays, beans, etc.

```
def list = ["Rod", 3, new Date()]
def myMap = ["Neeta":33, "Eric":35]
```

- Closures

```
myMap.each { name, age ->
                println "$name is $age years old" }
  -> Eric is 35 years old
  -> Neeta is 33 years old
```

Basic Features (Cont.)

- Regex built-in

```
if ("name" ==~ "na.*") { println "match!" }
    -> match!
```

- Operator overloading

```
def list = [1, 2, 3] + [4, 5, 6]
list.each { print it }
    -> 123456
```

- Autoboxing and polymorphism across collection, array, map, bean, String, iterators, etc.

```
String[] array = ['cat', 'dog', 'mouse']
def str = 'hello'
println "${array.size()},${str.size()},${list.size()}"
-> 3,5,6
```

Basic Features: Groovy Java™ Development Kit (JDK™)

- Groovy-er JDK software: adds convenient methods to JDK software
- String
  - `contains(), count(), execute(), padLeft(), center(), padRight(), reverse(), tokenize(), each(), etc.`
- Collection
  - `count(), collect(), join(), each(), reverseEach(), find/All(), min(), max(), inject(), sort(), etc.`
- File
  - `eachFile(), eachLine(), withPrintWriter(), write(), getText(), etc.`
- Lots there and growing all the time
- You can add methods programmatically

Agenda

Code Sample

Basic Features

Markup

Advanced Features

Demos

Extras

Conclusion

Groovy Markup

- Native support for hierarchical structures in code
  - XML
  - XHTML
  - Ant
  - Swing
  - SWT
- Relatively easy to add your own

Groovy Markup Example: Ant

```
ant = new groovy.util.AntBuilder()

ant.echo('starting...')
ant.sequential {
    def mydir = 'c:/backups'
    mkdir(dir: mydir)
    copy(todir: mydir) {
        fileset(dir: 'src/test') {
            includes(name:'**/*.groovy')
        }
    }
    echo("done!")
}
```

Agenda

Code Sample

Basic Features

Markup

Advanced Features

Demos

Extras

Conclusion

Advanced Features

- Safe Navigation
- Expando
- Template Engines
- Default Parameters
- Single Object Iteration/Identity Support
- Currying
- Dynamic Language Extensions (Enhancing JDK software)
- Aliases

Safe Navigation

- Use the safe navigation operator

```
people = ['rod': ['age':36, 'height':"5'9"]]
println people.rod.age
    -> 36
println people.joe.age
    -> throws NullPointerException
println people?.joe?.age
    -> null
```

Expando: The Dynamic Object

```
import groovy.util.Expando
rod = new Expando(name: 'Rod', age: 36)
rod.drinkWater = { num ->
     num.times { println "yummy!" }
}

println rod.age
    -> 36
rod.drinkWater(2)
    -> yummy!
       yummy!
```

Closure as Template Engine

```
t = { p -> "${p.name} is ${p.age()}" }
rod = new Person(name: 'Rod', birth: '3/31/71')
println t(rod)
    -> Rod is 36
joe = new Person(name: 'Joe', birth: '1/17/92')
println t(joe)
    -> Joe is 15
```

GStringTemplateEngine

```
import groovy.text.GStringTemplateEngine
t = new GStringTemplateEngine()
t.createTemplate(
        '${person.name} is ${person.age()}')
binding = ['person':
     new Person(name: 'Rod', birth: '3/31/71')]
println t.make(binding).toString()
   -> Rod is 36
binding.person =
    new Person(name: 'Joe', birth: '1/17/92')
println t.make(binding).toString()
   -> Joe is 15
```

Default Parameters

```
class Person

{
String name
int age
def yearsToRetirement(retAge = 65) { return retAge - age }

}

p = new Person(name: 'Rod', age: 36)

println p.yearsToRetirement(40)

     -> 4

println p.yearsToRetirement()

     -> 29

println p.yearsToRetirement('dog')

     -> dog

To prevent this last problem use:

      yearsToRetirement(int retAge = 65)
```

Single Object Iteration/Identity

- What?
  - Groovy lets you iterate over any object
- Why?
  - To fake a “with” construct; don’t need to know object vs. collection
- Examples

```
currentCustomer.employees['joe'].manager.secretary.each {
    it.salary *= 1.25
   it.bonus = 1000
   println it.location.state
}
currentCustomer.employees['joe'].manager.secretary.identity {
    println "salary=$salary, bonus=$bonus"
}  // attribute changes in here don't stick!
```

Currying

- Trivial example

```
c1 = { a, b -> a + b }
c2 = c1.curry("Hi ")    Result: c2 = { b -> "Hi " + b }
println c2("there")
   -> "Hi there"
```

- More realistic example

```
c1 = { date, account, action, amount ->
 println "${date}: ${action} of $${amount} to #${account}" }
[later in the code…]
c2 = c1.curry(new Date(), 12469)
[still later in the code…]
c3 = c2.curry('Deposit')
[and finally…]
c3(21.82)
    -> "Tue May 11: Deposit of $21.82 to #12469"
```

Dynamic Language Extensions

```
class PropertiesHelper
 {
   public static List getPropertyNames(Object bean)
   {
      def methodNames = bean.class.methods.name.findAll {
                            it.startsWith('get') }
      def goodNames = methodNames -
                  ['getMetaClass','getClass','getProperty']
      def propertyNames = goodNames.sort().collect {
         // "getName"  -> "n" + "ame" -> "name"
         it[3].toLowerCase() + it[4..-1]
      }
      return propertyNames
   }
 }
```

Dynamic Language Extensions (Cont.)

```
class Person
{
   String firstName; String lastName; int age
}
rod = new Person(firstName:'Rod',lastName:'Cope',age:36)
use(PropertiesHelper) {
    for (prop in rod.propertyNames) {
       println "${prop} = ${rod[prop]}"
    }
}
-> age = 36
-> firstName = Rod
-> lastName = Cope
```

Aliases

```
def p = System.out.&println
p('hi')    -> hi
p "hi"     -> hi

def doSomething(method) { method("dog") }
doSomething(p)
    -> dog

def doIt = { println it.size() }
def list = ['cat']
[p, doIt, list.&add].each { doSomething(it) }
    -> dog, 3, list == ['cat', 'dog']
```

Agenda

Code Sample

Basic Features

Markup

Advanced Features

Demos

Extras

Conclusion

DEMO 1

XML-RPC

XML-RPC

- import groovy.net.xmlrpc.*
- Server

```
server = new XMLRPCServer()
server.testme = { name -> name + " is cool!" }
server.multiply = { number -> number * 10 }
serverSocket = new ServerSocket(9047)
server.startServer(serverSocket)
```

- Client

```
serverProxy = new XMLRPCServerProxy("http://127.0.0.1:9047")
println serverProxy.testme("Groovy")
 -> "Groovy is cool!"
println serverProxy.multiply(7)
 -> 70
server.stopServer()
```

DEMO 2

Active Proxies

Active Proxies: Excel

- Easy native Windows access through Groovy
- Uses Jacob Library (danadler.com/jacob)
- Import org.codehaus.groovy.scriptom.ActiveXProxy

```
excel = new ActiveXProxy("Excel.Application")
excel.visible = true
workbook = excel.workbooks.add()
sheet = workbook.ActiveSheet
a1 = sheet.range('A1')
a2 = sheet.range('A2')
a1.value = 125.3
a2.formula = '=A1 * 2'
println "a2: ${a2.Value.value}"
   -> 250.6
workbook.close(false, null, false)
excel.Quit()
```

Active Proxies: Excel (Cont.)

```
a1 = sheet.range('A1'); a2 = sheet.range('A2')
b1 = sheet.range('B1'); b2 = sheet.range('B2')
a1.value = 125.3; a2.value = 97.1
b1.formula = '=A1 * 2'; b2.formula = '=A2 * 3'
range = sheet.range('A1:B2')
range.font.size = 16; range.font.bold = true
range.copy()
chartObject = sheet.chartObjects.add(50,50,400,200)
chart = chartObject.chart
chart.axes(1).hasTitle = true
chart.axes(1).axisTitle.text = "Groovy!"
chart.seriesCollection.item(1).name = "Cool"
```

Active Proxies: Excel (Cont.)

```
count = 1

swing = new groovy.swing.SwingBuilder()

mybutton = swing.button(text: "Click me!")

mybutton.actionPerformed = {
   a1.value = new Random().nextFloat() * 200
   a2.value = new Random().nextFloat() * 500
   chart.export("c:\\temp\\mychart${count}.gif")
   mybutton.icon = new
   javax.swing.ImageIcon("c:\\temp\\mychart${count}.gif")
   count += 1 }

frame = swing.frame(title:"The chart", size:[500,400]) {
   panel() { widget(mybutton) } }

frame.show()

chart.seriesCollection.item(1).name = "That rocks!"
```

Agenda

Code Sample

Basic Features

Markup

Advanced Features

Demos

Extras

Conclusion

Groovy Extras

- Eclipse, IntelliJ, JEdit: Groovy plug-ins available
- Grails: Like Ruby on Rails, but Groovy, Hibernate, Spring
- Processes: `"cmd /c dir".execute().text`
- Threading: `Thread.start { any code }`
- Testing: `GroovyTestCase, GroovyMock`
- SWT: Full support for SWT building, like SwingBuilder
- Groovy Pages/Template Engine: GSP, Groovlets, etc.
- UNIX® Scripting: Groovy API for pipe, cat, grep, etc.
- JMX API: Small sample available

Agenda

Code Sample

Basic Features

Markup

Advanced Features

Demos

Extras

Conclusion

Some Trouble in Paradise

- Weak and Missing Features
  - No support for inner classes
  - Tool support (Eclipse, IntelliJ, etc.) not great, getting better
- Debugging/Scripting Hell
  - Immature parser: hard to find “real” bugs
    - Not uncommon to find compile-time issues at run-time
  - Lots of rope: easy to hang yourself with dynamic code

Conclusion

- Status
  - Java™ Specification Request (JSR) 241
  - 1.0, targeting 1.1 release by end of year
- Development time
  - Half that of Java technology (except for debugging hell factor)
- Performance
  - 20–90% of Java technology depending on usage
  - Very little tuning so far
- Recommendations
  - Ready for small, non-mission-critical projects and scripting
  - Try it! Very easy to learn and lots of fun!

References and Links

- Groovy Home Page
  - http://groovy.codehaus.org
- Download
  - http://dist.codehaus.org/groovy/distributions/groovy-1.0.zip
- GDK Javadoc™ Tool
  - http://groovy.codehaus.org/groovy-jdk.html
- JSR 241: Groovy Language Specification
  - http://www.jcp.org/en/jsr/detail?id=241

Q&A

Rod Cope, CTO and Founder
OpenLogic, Inc.

Advanced Groovy

Rod Cope

CTO and Founder
OpenLogic, Inc.
http://www.openlogic.com

TS-9720