Groovy, but without a cheesy presentation title...

Ken Rimple, Chariot Solutions

Emerging Technologies for the Enterprise 2009

All About Me...

- Ken Rimple
  - Mentoring/Education Services lead for Chariot Solutions
  - Host of Chariot Tech Cast podcast
  - Groovy/Grails enthusiast

Agenda

- Why Groovy?
- The Groovy Language
- POGOs, collections, closures
- GDK / Groovyizing the JDK
- Manipulating Resources
- How to use Groovy at work...

Do you hate...

- Generating Code (Getters/setters/constructors)
- Iterators, casting, exception handling?
- Parsing XML
- Writing Swing UIs
- JDBC
- Java Date Math

Try Groovy!

- Dynamically typed language
- Uses a superset of Java syntax and dynamic language ‘syntactic sugar’
- Groovy classes can extend Java classes (and vice-versa)
- Makes Java productive and actually fun!

The Groovy Language can be...

- Compiled to bytecode in .class files
- Interpreted at runtime as a Script
- Accessed as a Scripting engine via JSR-223
- Embedded into Spring and other frameworks

What does it bring to Java?

- Closures
- Dynamic Typing
- Groovy-ized Java Classes
- Dirt-simple XML parsing support
- Meta-programming
- Easy DSL
- Much more...

Groovy sweetens development

- Auto imports: java. packages
  - .lang, .util, .net, .io, .math.Big*, groovy.lang and groovy.util
- Semicolons, return statements and parentheses are optional
- The def keyword allows dynamic (or duck) typing

Groovy and Strings

- A GString allows for ${} syntax within a string
  - Automatically substituted at runtime
  - Must be surrounded by double quotes
- Examples
  - `"Hello there, ${name}"`
  - `"Your current balance is ${acct.bal}"`
  -
```
"Test passed?
   ${kid.grade >65 ? 'yes' : 'no'} "
```

How to access Groovy

- Groovy Scripts (groovy interpreter)
- Groovy Console (testing scripts)
- Groovyc compiler
  - ANT Task for Joint Compiler
  - Maven plugin (gmaven)

Getters, Setters and Properties

- Groovy automatically creates getters/setters for properties
  - Java code can call get and set methods
  - Groovy code can just reference properties
  - This is a shortcut – properties are private but the Groovy compiler generates the get/set methods
- Switch to POGOs and reduce code in your application!

Groovy and Collections

- A groovy list:
  - `def myList = ['A', 234, '234']`
- A groovy map:
  -
```
def myMap = [ firstName: "Joe",
              lastName : "Jones",
            "Key with Spaces" : "Value"]
```
  - Provides overloaded operators:
  - `myList << '234234'`
  - `myMap['abcde'] = 'hijkl'`

Demos

- A Java Bean –vs- a Groovy Bean
- Collections example

- Groovy eschews ceremony at every possible point

What is a Closure?

- A named block of code
  - Can be injected as a parameter into another method
- Huh?
  - Best to look at a few examples

Dirt Simple Closure

- Dreaded "Hello World"
- 'it' is the value passed to the closure

```
def announceVisitor = {
  println "Welcome to class ${it}"
  println "That will be \$5.00"
}

announceVisitor "Ken Rimple"
```

```
Welcome to class Ken Rimple
That will be $5.00
```

Closures and Parameters

- Looks much like a method...
- Just dynamically typed parameters

```
def announceVisitor = {
  firstName, lastName ->
  println "Welcome to class ${firstName} ${lastName}"
  println "That will be \$5.00"
}

announceVisitor "Ken", "Rimple"
```

```
Welcome to class Ken Rimple
That will be $5.00
```

Example Closure use - Iteration

- Groovy adds .each to collections
- Much less code than iteration

```
def myNumbers = [2, 4, 5, 6, 7, 8, 88, 2, 43, 5]
def max
myNumbers.each {
    if (it > max) max = it
}
println "Max : ${max}"
```

```
Max : 88
```

The .each method for Maps

- One method provides key and value
- Another provides the entry

```
def mymap = [One: 1, Two: 2]

mymap.each { k,v ->
  println "${k} = ${v}"
}

mymap.each { entry ->
  println "${entry.key} = ${entry.value}"
}
```

Writing methods with Closures

```
def myeach(List mylist, Closure c) {
  Iterator it = mylist.iterator()
  while (it.hasNext()) {
     c(it.next())
  }
}

// use it!
def list = [1, 234, 324234, 23, 1]

myeach(list) {
  println "Iteration value ${it}"
}
```

Java – Dates – What were they thinking??

- Someone ought to write a law..
  - Date Formats – default is bizarre
  - Not easy to move to/from Strings
- Groovy "Groovyizes" Dates
  - Easy String formatting
  - Simple defaults

Groovy: Have a nice Date!

```
def d= new Date()
println "date : ${d.getDateString()}"
println "time : ${d.getTimeString()}"
println "One day later ${d + 1}"
println "23 days later ${d + 23}"
def d2 = Date.parse("MM/dd/yyyy", "01/15/2003")
print "${d2}"
```

```
date : 3/22/09
time : 5:58:37 PM
One day later Mon Mar 23 17:58:37 EDT 2009
23 days later Tue Apr 14 17:58:37 EDT 2009
Wed Jan 15 00:00:00 EST 2003
```

Nulls and Ceremony

- Groovy's ?. operator can reduce coding headaches
  - Use it before querying attributes
  - Short-circuits and stops evaluation
- Elvis Operator (?: another handy shortcut)
  - Shortcut for (expr) ? true : false

```
customer?.flights?.each {
    print "${it.arrivalTime} : ${it.gate} "
    println "${it.notes ?: 'No information'}"
}
```

Let's get things done

- Process Files
- Create XML
- Parse XML
- Integrate with Java Applications
- Build some SWING UIs
- Write killer web apps

Processing Files

- Groovy enhances java.io.File
  - eachDir and eachFile
  - eachLine – reads file line at a time
  - getText – read the entire as a String
  - << and >> to write to / read from a file

```
def f = new File("myfile")
f << "Hi there\n"
f << "It's nice today\n"

def str = new File("myfile").getText()
print str
```

```
Hi there
It's nice today
```

Creating XML

- Create via ' ' ' strings...
- Groovy MarkupBuilder
  - Uses closures to generate XML Documents
  - method names -> tag names
  - attribute names -> tag attribute names
  - non-named string value in parameters -> body of tag
- StreamingMarkupBuilder
  - SAX events version of same process

Groovy " " " String XML Creation

- Great for quick test data

```
def channelName = 'News at 10'
def title = 'Chariot Tech Cast'
def episodeTitle = "Interview with Dan Allen"
def description = 'Dan allen discusses Seam'

def myXML = """
<?xml version='1.0'?>
<channel name="${title}">
    <items>
        <item title="${episodeTitle}">
            <description>${description}</description>
        </item>
    </items>
</channel>
"""
```

Groovy Markup Builder Example

```
import groovy.xml.MarkupBuilder

def writer = new StringWriter()
def xml = new MarkupBuilder(writer)

xml.channel {
    title 'Chariot Tech Cast'
    link 'http://techcast.chariotsolutions.com'
    item(id:'123') {
        title 'Episode 23 - Dan Allen on Seam, Part 1'
        description 'Last Thursday I had an opportunity ...'
    }
}

// print out the XML:
println(writer)
```

Streaming Markup Example

```
import groovy.xml.StreamingMarkupBuilder

def channel = {
   title 'Chariot Tech Cast'
   link 'http://techcast.chariotsolutions.com'
   item (title : 'Episode 23 - Dan Allen on Seam, Part 1') {
       description 'Last Thursday I had an opportunity ...'
   }
}

def xml = new StreamingMarkupBuilder()
fw = new FileWriter('xmldemo')
doc = xml.bind (channel)
fw << doc

f = new File('xmldemo')
print (f.getText())
```

Parsing XML

- Two Groovy options
  - XML Parser
    - DOM-oriented
    - Read/Write – can update document
  - XML Slurper
    - SAX-like slurping of XML in streams
    - Read-only

The XMLSlurper

```
def myXML = '''
<courses>
  <course id='1'>
     <name>Intro to Groovy</name>
     <duration>3</duration>
  </course>
  <course id='2'>
     <name>Intro to COBOL and JCL</name>
     <duration>10</duration>
  </course>
</courses>
'''
println myXML

def courseXML = new XmlSlurper().parseText(myXML)

courseXML.course.each {
  println "Course - ${it.@id} : ${it.name.text()}"
}
```

Groovy and Java Platforms

- Compile Groovy into .class files, add groovy-all.jar to the classpath
- Use JSR-223 Scripting Engine
- Spring
  - Can also install scripting language support

Example Spring Bean Config

```
<lang:groovy id="groovyServer"
     script-source="classpath:...ProcessOrders.groovy"
     refresh-check-delay="10000">
     <property name="orderDao" ref="orderDao"/>
</lang:groovy>
```

- refresh-check-delay – how long to wait before scanning for changes
- You can inject properties from other beans
- Constructor injection not allowed...

Swing Builder

- Groovy's DSL for Swing
  - Supports all swing widgets
  - Greatly reduces coding
- Griffon is an MVC-style app framework for Groovy + Swing

Grails

- You did attend Jeff Brown's Talk?
- Grails is an excellent web framework
  - GORM – Domain Centric ORM
  - MVC – easy to use and flexible
  - Vast plugin library
- www.grails.org

APIs we don't have time to talk about..

- Grails
- Domain Specific Languages
- Groovy SQL
- GORM
- MetaBuilder (build your own DSL builders)
- AST Annotations (@Singleton, @Immutable, @Bindable, @Lazy, @Delegate, etc.)
- Grape (dependency injection from scripts)
- MetaProgramming w/ Expando Meta-Class (intercepting 'missing' methods to build your own languages)
- Groovy and OSGi
- Griffon

Resources

- Groovy
  - groovy.codehaus.org
  - groovyblogs.org
  - GroovyMag.com
- Grails
  - grails.org
- My presentation, other random thoughts
  - www.rimple.com/tech
- Chariot TechCast Podcast
  - techcast.chariotsolutions.com