

Java is Dead. Long Live Groovy!

Bob Brown

Transentia Pty. Ltd.

<http://www.transentia.com.au>

bob@transentia.com.au



Thursday, 9 October 2008



What is Groovy?

- <http://groovy.codehaus.org>

Groovy...

- *is an agile and dynamic language for the Java Virtual Machine*
- *builds upon the strengths of Java but has **additional power features** inspired by languages like Python, Ruby and Smalltalk*
- *makes **modern programming features** available to Java developers with **almost-zero learning curve***
- *supports **Domain-Specific Languages** and other compact syntax so your code becomes **easy to read and maintain***
- *makes writing shell and build scripts easy with its **powerful processing primitives**, OO abilities and an Ant DSL*
- *increases developer productivity by **reducing scaffolding code** when developing web, GUI, database or console applications*
- ***simplifies testing** by supporting unit testing and mocking out-of-the-box*
- *seamlessly integrates with all existing Java objects and **libraries***
- *compiles straight to Java bytecode so you can use it anywhere you can use Java*



What is Groovy?...

- new open source scripting language
- formalised via JSR 241
 - led by Guillaume Laforge

*"... a new programming language for the Java Platform—one that is **on equal footing** with the Java programming language. Groovy is an agile, dynamic programming language like Python, Perl and Ruby, but it's designed specifically for the Java Platform and is **completely interoperable with conventional Java programs**.*

...Groovy represents the beginning of a new era in the Java platform, one in which the Java community embraces language diversification and harnesses the full potential of the Java platform....

*So why Groovy? Why not Jython or JRuby?...Groovy is the best choice because it was built from the ground up for the Java Platform and uses syntax that is familiar to Java developers,....Jython and JRuby are excellent examples of how existing languages can be ported to the Java platform, but they are, after all, ports. They use syntax that is not designed with Java developers in mind and they are founded on a **completely different set of code libraries**."*

—http://weblogs.java.net/blog/monsonhaefel/archive/2004/03/jsr241_groovy_a.html

"We've always had lisp people.... While most of them are dead or retired now, the...autistic genes that caused that sick disease is still rampant in society, and we're seeing a new generation of infections in the form of...ruby, groovy, and scala."

—<http://www.bibleblog.org/2008/05/java-haters-gtfo/>

Example

- had to happen...

old and
busted...

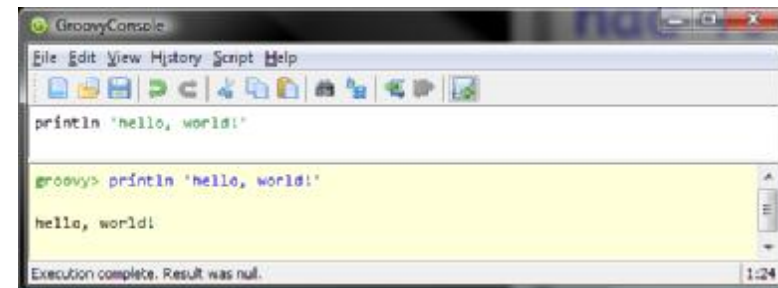
```
public class Hello {  
    public static void main(String [] args) {  
        System.out.println("hello, world!");  
    }  
}
```

...new and
shiny...

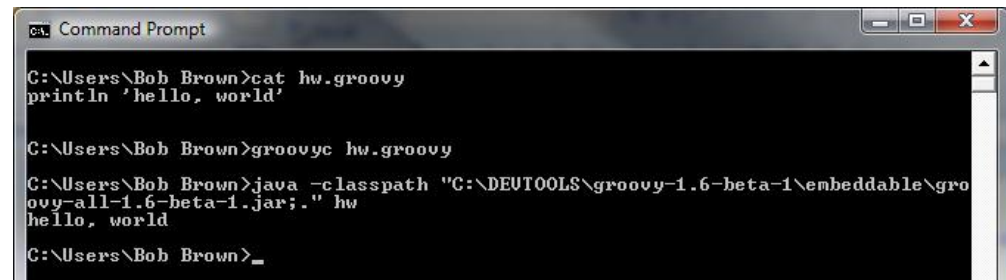
```
class Hello {  
    static main(args) {  
        println 'hello, world!'  
    }  
}
```

```
println 'hello, world!'
```

...time to put on
the shades!



A screenshot of the GroovyConsole application. The window title is "GroovyConsole". The menu bar includes "File", "Edit", "View", "History", "Script", and "Help". The main area shows a Groovy script with the line `println 'hello, world!'`. Below the script, the output shows `hello, world!`. At the bottom, it says "Execution complete. Result was null." and the time is 1:24.



A screenshot of a Windows Command Prompt window. The window title is "Command Prompt". The prompt shows the following commands and output:
`C:\Users\Bob Brown>cat hw.groovy`
`println 'hello, world'`
`C:\Users\Bob Brown>groovyc hw.groovy`
`C:\Users\Bob Brown>java -classpath "C:\DEVTOOLS\groovy-1.6-beta-1\embeddable\groovy-all-1.6-beta-1.jar;" hw`
`hello, world`
`C:\Users\Bob Brown>`

Example...

```
// file:XMLReader.java
import java.io.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class XMLReader {
    public static void main(String argv[]) throws Exception {
        File file = new File("items.xml");
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        DocumentBuilder db = dbf.newDocumentBuilder();
        Document doc = db.parse(file);
        doc.getDocumentElement().normalize();
        NodeList nodeList = doc.getElementsByTagName("an-item");
        for (int s = 0; s < nodeList.getLength(); s++) {
            Element anItem = (Element) nodeList.item(s);
            System.out.println(anItem.getAttribute("the-id") + ": " +
                anItem.getChildNodes().item(0).getNodeValue());
        }
    }
}
```

old and
busted...

```
C:\Windows\system32\cmd.exe
C:\Users\Bob Brown\Desktop\XML>groovy XMLReader.groovy
0: This is item 0
1: This is item 1
2: This is item 2
3: This is item 3
4: This is item 4
5: This is item 5

C:\Users\Bob Brown\Desktop\XML>javac XMLReader.java

C:\Users\Bob Brown\Desktop\XML>java XMLReader
0: This is item 0
1: This is item 1
2: This is item 2
3: This is item 3
4: This is item 4
5: This is item 5

C:\Users\Bob Brown\Desktop\XML>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<items>
  <an-item the-id="0">This is item 0</an-item>
  <an-item the-id="1">This is item 1</an-item>
  <an-item the-id="2">This is item 2</an-item>
  <an-item the-id="3">This is item 3</an-item>
  <an-item the-id="4">This is item 4</an-item>
  <an-item the-id="5">This is item 5</an-item>
</items>
```

"Nothing Makes You Want Groovy More Than XML..."

—<http://www.groovyongrails.com/article/63>

...new and
shiny!

```
// file:XMLReader.groovy
items = new XmlSlurper().parse(new File('items.xml'))

items?. 'an-item'.each {
    println it.@the-id.text() + ': ' + it.text()
}
```

Couldn't Say It Better... Dr. Paul King's Slides

A Better Java...

```
import java.util.List;
import java.util.ArrayList;

class Erase {
    private List filterLongerThan(List strings, int length) {
        List result = new ArrayList();
        for (int i = 0; i < strings.size(); i++) {
            String s = (String) strings.get(i);
            if (s.length() <= length) {
                result.add(s);
            }
        }
        return result;
    }
    public static void main(String[] args) {
        List names = new ArrayList();
        names.add("Ted"); names.add("Fred");
        names.add("Jed"); names.add("Ned");
        System.out.println(names);
        Erase e = new Erase();
        List shortNames = e.filterLongerThan(names, 3);
        System.out.println(shortNames.size());
        for (int i = 0; i < shortNames.size(); i++) {
            String s = (String) shortNames.get(i);
            System.out.println(s);
        }
    }
}
```

Submission 631 @ ASERT 2007

This code
is valid
Java and
valid Groovy

*Based on an
example by
Jim Weirich
& Ted Leung*

Agile 2007 - 8

Couldn't Say It Better... Dr. Paul King's Slides...

...A Better Java...

```
import java.util.List;
import java.util.ArrayList;

class Erase {
    private List filterLongerThan(List strings, int length) {
        List result = new ArrayList();
        for (int i = 0; i < strings.size(); i++) {
            String s = (String) strings.get(i);
            if (s.length() <= length) {
                result.add(s);
            }
        }
        return result;
    }
    public static void main(String[] args) {
        List names = new ArrayList();
        names.add("Ted"); names.add("Fred");
        names.add("Jed"); names.add("Ned");
        System.out.println(names);
        Erase e = new Erase();
        List shortNames = e.filterLongerThan(names, 3);
        System.out.println(shortNames.size());
        for (int i = 0; i < shortNames.size(); i++) {
            String s = (String) shortNames.get(i);
            System.out.println(s);
        }
    }
}
```

Submission 631 @ ASERT 2007

Do the
semicolons
add anything?
And shouldn't
we use more
modern list
notation?
Why not
import common
libraries?

Agile 2007 - 9

Couldn't Say It Better... Dr. Paul King's Slides...

Submission 631 @ ASERT 2007

...A Better Java...

```
class Erase {
    private List filterLongerThan(List strings, int length) {
        List result = new ArrayList()
        for (String s in strings) {
            if (s.length() <= length) {
                result.add(s)
            }
        }
        return result
    }

    public static void main(String[] args) {
        List names = new ArrayList()
        names.add("Ted"); names.add("Fred")
        names.add("Jed"); names.add("Ned")
        System.out.println(names)
        Erase e = new Erase()
        List shortNames = e.filterLongerThan(names, 3)
        System.out.println(shortNames.size())
        for (String s in shortNames) {
            System.out.println(s)
        }
    }
}
```

Agile 2007 - 10

Couldn't Say It Better... Dr. Paul King's Slides...

...A Better Java...

```
class Erase {
    private List filterLongerThan(List strings, int length) {
        List result = new ArrayList()
        for (String s in strings) {
            if (s.length() <= length) {
                result.add(s)
            }
        }
        return result
    }

    public static void main(String[] args) {
        List names = new ArrayList()
        names.add("Ted"); names.add("Fred")
        names.add("Jed"); names.add("Ned")
        System.out.println(names)
        Erase e = new Erase()
        List shortNames = e.filterLongerThan(names, 3)
        System.out.println(shortNames.size())
        for (String s in shortNames) {
            System.out.println(s)
        }
    }
}
```

Submission 631 @ ASERT 2007

Do we need
the static types?
Must we always
have a main
method and
class definition?
How about
improved
consistency?

Agile 2007 - 11

Couldn't Say It Better... Dr. Paul King's Slides...

Submission 631 @ ASERT 2007

...A Better Java...

```
def filterLongerThan(strings, length) {  
  def result = new ArrayList()  
  for (s in strings) {  
    if (s.size() <= length) {  
      result.add(s)  
    }  
  }  
  return result  
}  
  
names = new ArrayList()  
names.add("Ted")  
names.add("Fred")  
names.add("Jed")  
names.add("Ned")  
System.out.println(names)  
shortNames = filterLongerThan(names, 3)  
System.out.println(shortNames.size())  
for (s in shortNames) {  
  System.out.println(s)  
}
```

Agile 2007 - 12

9 October, 2008

Couldn't Say It Better... Dr. Paul King's Slides...

Submission 631 @ ASERT 2007

...A Better Java...

```
def filterLongerThan(strings, length) {  
  def result = new ArrayList()  
  for (s in strings) {  
    if (s.size() <= length) {  
      result.add(s)  
    }  
  }  
  return result  
}  
  
names = new ArrayList()  
names.add("Ted")  
names.add("Fred")  
names.add("Jed")  
names.add("Ned")  
System.out.println(names)  
shortNames = filterLongerThan(names, 3)  
System.out.println(shortNames.size())  
for (s in shortNames) {  
  System.out.println(s)  
}
```

Shouldn't we
have special
notation for lists?
And special
facilities for
list processing?

Agile 2007 - 13

Couldn't Say It Better... Dr. Paul King's Slides...

Submission 631 @ ASERT 2007

...A Better Java...

```
def filterLongerThan(strings, length) {  
    return strings.findAll{ it.size() <= length }  
}  
  
names = ["Ted", "Fred", "Jed", "Ned"]  
System.out.println(names)  
shortNames = filterLongerThan(names, 3)  
System.out.println(shortNames.size())  
shortNames.each{ System.out.println(s) }
```

Agile 2007 - 14

9 October, 2008

Couldn't Say It Better... Dr. Paul King's Slides...

Submission 631 @ ASERT 2007

...A Better Java...

```
def filterLongerThan(strings, length) {  
    return strings.findAll{ it.size() <= length }  
}  
  
names = ["Ted", "Fred", "Jed", "Ned"]  
System.out.println(names)  
shortNames = filterLongerThan(names, 3)  
System.out.println(shortNames.size())  
shortNames.each{ System.out.println(s) }
```

Is the method
now needed?
Easier ways to
use common
methods?
Are brackets
required here?

Couldn't Say It Better... Dr. Paul King's Slides...

Submission 631 © ASERT 2007

...A Better Java...

```
names = ["Ted", "Fred", "Jed", "Ned"]
println names
shortNames = names.findAll{ it.size() <= 3 }
println shortNames.size()
shortNames.each{ println it }
```

```
["Ted", "Fred", "Jed", "Ned"]
3
Ted
Jed
Ned
```

Agile 2007 - 16

...A Better Java

```
names = ["Ted", "Fred", "Jed", "Ned"]
println names
shortNames = names.findAll{ it.size() <= 3 }
println shortNames.size()
shortNames.each{ println it }
```

```
import java.util.List;
import java.util.ArrayList;

class Erase {
    private List filterLongerThan(List strings, int length) {
        List result = new ArrayList();
        for (int i = 0; i < strings.size(); i++) {
            String s = (String) strings.get(i);
            if (s.length() <= length) {
                result.add(s);
            }
        }
        return result;
    }
    public static void main(String[] args) {
        List names = new ArrayList();
        names.add("Ted"); names.add("Fred");
        names.add("Jed"); names.add("Ned");
        System.out.println(names);
        Erase e = new Erase();
        List shortNames = e.filterLongerThan(names, 3);
        System.out.println(shortNames.size());
        for (int i = 0; i < shortNames.size(); i++) {
            String s = (String) shortNames.get(i);
            System.out.println(s);
        }
    }
}
```

"Hear me out. In 2 to 3 years from now there we will see strong indications that Groovy is replacing the Java language as the dominant language on the JVM. Actually outnumbering the Java language will be very hard, given its 10+ years of history. But the trend will be clearly with Groovy."

—<http://java.dzone.com/news/groovy-will-replace-java-langu>

"So, if you're using Java, do yourself a favor and check out Groovy. If for no other reason than being able to write scripts in a language that takes advantage of the Java platform, instead of having to keep remembering Perl, sed, awk, Bourne shell, etc."

—<http://www.vitarara.org/cms/node/93>

"...I just spent 2 weeks trying to embed Groovy into an ongoing java app. Funny, the lines of Groovy code kept getting shorter and shorter. Now I want to go back and try it all Groovy..."

—<http://www.manning-sandbox.com/thread.jspa?threadID=18241&tstart=0>

"Groovy is just Java with flava."
—<http://codeforfun.wordpress.com/category/programming/groovy/>

*"Whenever I'm in Java, and I need to write this utility, I feel like I'm about to buy a house. There is a lot of paperwork involved and the process is **not particularly fun**. Plus, there is a conceptual mismatch: if I wanted to buy a house, that would be one thing, but I just want to stay overnight.*

*By contrast, when working with Groovy, I feel like I'm staying at the Ritz.... I feel as though there are people attending to my needs, and they are **friendly**. Start the bath running, pour some wine, and relax...."*

—<http://codetojoy.blogspot.com/2008/08/groovy-file-io-staying-at-ritz.html>

"...I found the time to dive into the Groovy language.

My God, did I like what I read!!

*I will probably **never script again in bash**, whenever I am allowed to choose!!*

*And the fact that java and groovy can coexist!! And pointers on methods!!
And... and... and..."*

—<http://jgrasstechtips.blogspot.com/2008/04/groovy-groovy-groovy-groovy-like-kaiser.html>

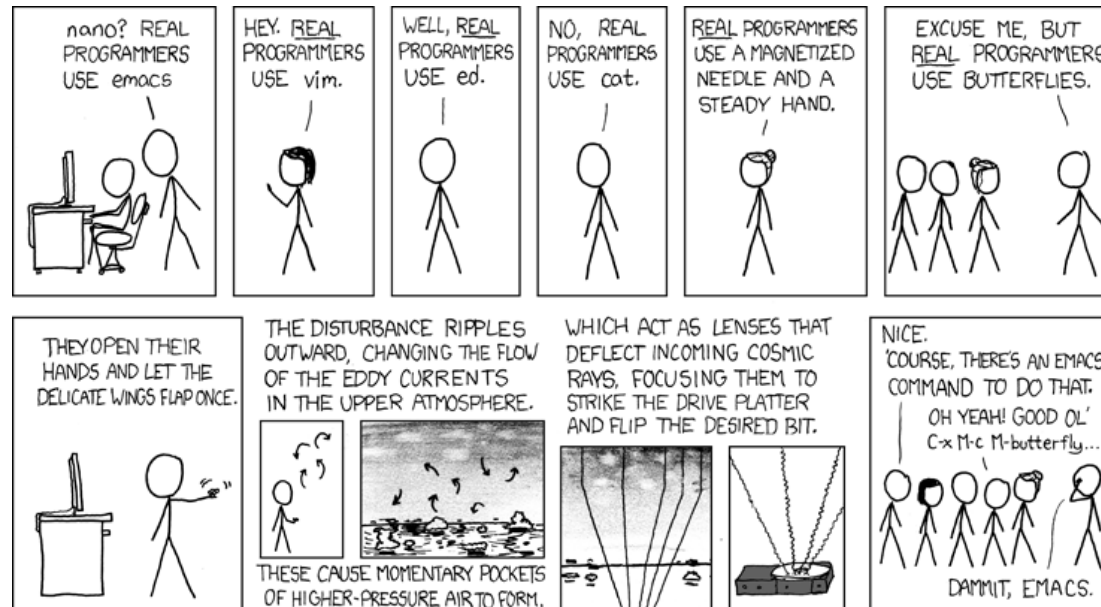
"...the Next Big Language..."

I was thinking Ruby or Groovy until I read this post. Okay, maybe not Groovy unless they come up with a "management friendly" version of the name (Neal Ford's idea is ebXI or 'Enterprise Buisness Execution Language' I think)."

—<http://steve-yegge.blogspot.com/2007/02/next-big-language.html>

- **IDE support**
 - Eclipse, Netbeans
 - IntelliJ
 - best (so far...)
 - not JDeveloper (yet!)
 - currently often need to drop into the command line
- **IDEs are over-rated...**

“Although Oracle JDeveloper does not yet come with special features for Groovy and Grails, Oracle JDeveloper’s external tools capability enables us to quite easily set up Oracle JDeveloper for Grails development.”
—<http://www.oracle.com/technology/pub/articles/oak-grails.html>



<http://xkcd.com/378/>

- **cleaning up the cruft**
 - no more ';', bye-bye "()"
 - length, length(), size()... => size()
 - unchecked exceptions throughout
 - "switch on steroids" / isCase()
 - *much enhanced over Java's anaemic version*
- **everything is an object**
 - int => java.math.BigInteger, etc.
- **convenient collections**
 - maps, lists, sets, ranges
 - plus a few supporting operators
- **GStrings**
- **native regexp**

"Write Once, Groove Anywhere"
—<http://www.darwinsys.com/groovy/jugslides-20041102.pdf>

A Short Tour

```
switch (10)
{
  case 0 :
  case 0..9 :
  case [8,9,11] :
  case Float :
  case {it % 3 == 0} : assert false; break;
  case ~/.. / : assert true; break;
  case "ten": assert false; break;
  default : assert false; break;
}
```

```
new File('copy.txt').withWriter { file ->
  new File('orig.txt').eachLine { line ->
    file.writeLine(line)
  }
}
```

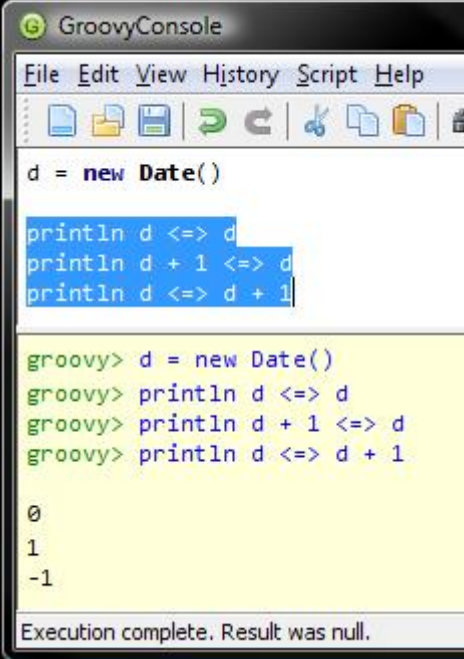
```
def reader = new FileReader(/can't.find.me!!!/)
```

```
composer = 'August Wilhelmj'
println "\"Air on a G string\" was written by $composer"
```

```
def brisbane = [ state:'QLD',
                  'current population': 1800000,
                  location: [ latitude: /27° 30' South/, longitude: /153° 00' East/ ],
                  timezone: 'GMT+10' ]
```

```
for (i in brisbane)
  println i
```

```
println ""Welcome to Sunny Brisbane!
State: ${brisbane.state}
Current population: ${brisbane['current population']}""
```



The screenshot shows a GroovyConsole window with a menu bar (File, Edit, View, History, Script, Help) and a toolbar. The code in the editor is:

```
d = new Date()
println d <=> d
println d + 1 <=> d
println d <=> d + 1
```

The output of the execution is:

```
groovy> d = new Date()
groovy> println d <=> d
groovy> println d + 1 <=> d
groovy> println d <=> d + 1

0
1
-1
```

At the bottom, it says "Execution complete. Result was null."

A Short Tour...

```
def now = new Date()

println now
(now - 1..now + 7).each { println it }
```

```
use(org.codehaus.groovy.runtime.TimeCategory) {

    def now = new Date()

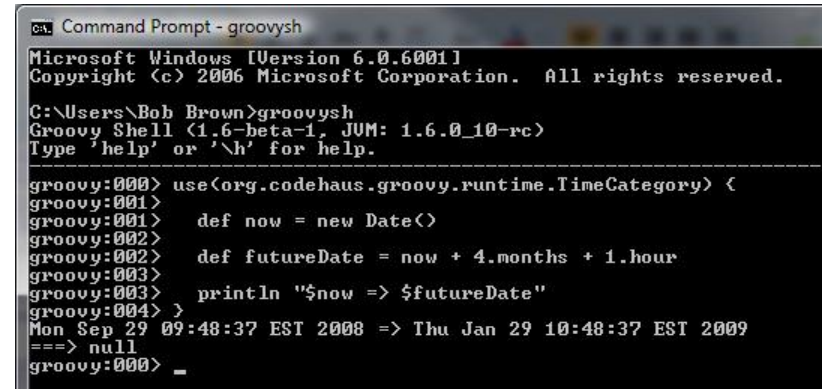
    def futureDate = now + 4.months + 1.hour

    println "$now => $futureDate"
}
```

```
int [] x = new int [3]
for (i in 1..<4)
    x[i - 1] = i
assert [1,2,3].size() == x.size()
```

```
4.times { println it }
(-2..2).each { println it }
```

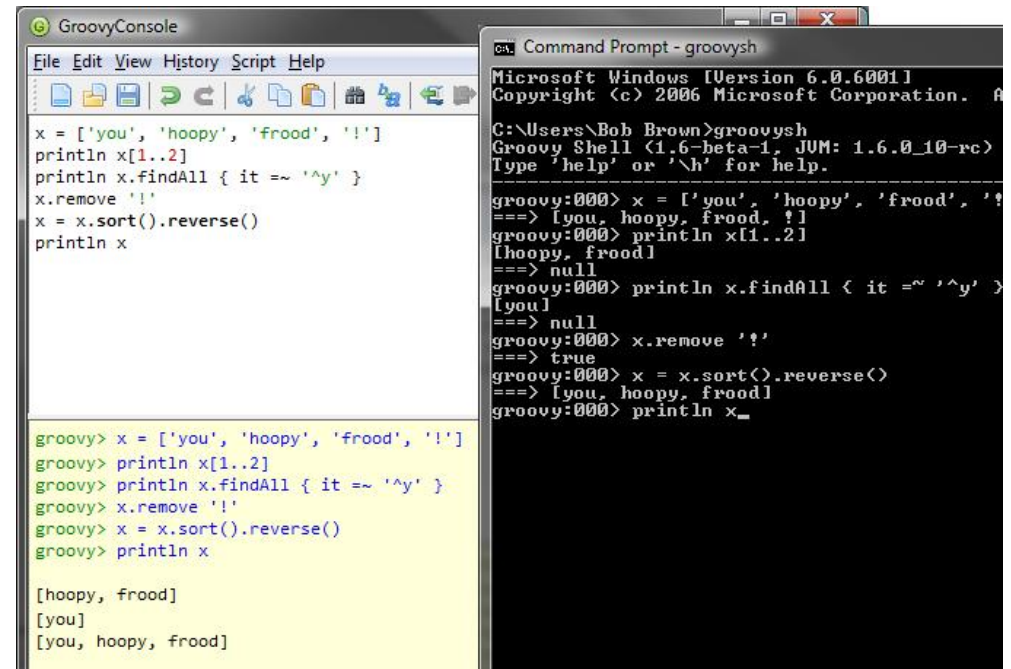
```
x = ['you', 'hoopy', 'frood', '!']
println x[1..2]
println x.findAll { it =~ '^y' }
x.remove '!'
x = x.sort().reverse()
println x
```



```
ca: Command Prompt - groovysh
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

C:\Users\Bob Brown>groovysh
Groovy Shell (1.6-beta-1, JVM: 1.6.0_10-rc)
Type 'help' or '\h' for help.

groovy:000> use(org.codehaus.groovy.runtime.TimeCategory) {
groovy:001>
groovy:001>     def now = new Date()
groovy:002>
groovy:002>     def futureDate = now + 4.months + 1.hour
groovy:003>
groovy:003>     println "$now => $futureDate"
groovy:004> }
Mon Sep 29 09:48:37 EST 2008 => Thu Jan 29 10:48:37 EST 2009
==> null
groovy:000> _
```



```
GroovyConsole
File Edit View History Script Help

x = ['you', 'hoopy', 'frood', '!']
println x[1..2]
println x.findAll { it =~ '^y' }
x.remove '!'
x = x.sort().reverse()
println x

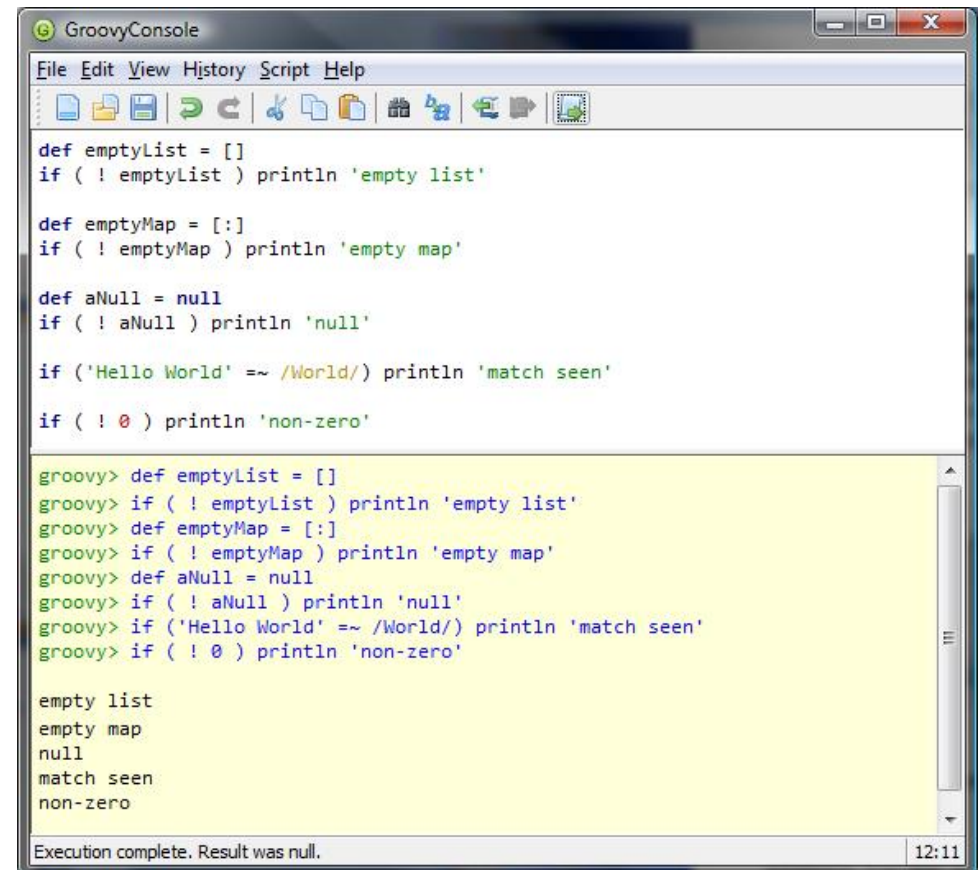
[hoopy, frood]
[you]
[you, hoopy, frood]

ca: Command Prompt - groovysh
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

C:\Users\Bob Brown>groovysh
Groovy Shell (1.6-beta-1, JVM: 1.6.0_10-rc)
Type 'help' or '\h' for help.

groovy:000> x = ['you', 'hoopy', 'frood', '!']
==> [you, hoopy, frood, ?]
groovy:000> println x[1..2]
[hoopy, frood]
==> null
groovy:000> println x.findAll { it =~ '^y' }
[you]
==> null
groovy:000> x.remove '!'
==> true
groovy:000> x = x.sort().reverse()
==> [you, hoopy, frood]
groovy:000> println x_
```

- sensible defaults to make life easier
 - special rules for coercing non-boolean objects to a boolean value
 - collections
 - pattern matchers
 - strings
 - numbers
 - objects
 - much cleaner



```
GroovyConsole
File Edit View History Script Help
def emptyList = []
if ( ! emptyList ) println 'empty list'

def emptyMap = [:]
if ( ! emptyMap ) println 'empty map'

def aNull = null
if ( ! aNull ) println 'null'

if ( 'Hello World' =~ /World/) println 'match seen'

if ( ! 0 ) println 'non-zero'

groovy> def emptyList = []
groovy> if ( ! emptyList ) println 'empty list'
groovy> def emptyMap = [:]
groovy> if ( ! emptyMap ) println 'empty map'
groovy> def aNull = null
groovy> if ( ! aNull ) println 'null'
groovy> if ( 'Hello World' =~ /World/) println 'match seen'
groovy> if ( ! 0 ) println 'non-zero'

empty list
empty map
null
match seen
non-zero

Execution complete. Result was null. 12:11
```

- **duck (optional) typing**
 - objects are treated polymorphically **without** being related by a common base class or interface
 - *"if it walks like a duck and quacks like a duck, its a duck"*
- **Gpaths**
 - object graph navigation
- **operator overloading**
 - and a few new operators
- **dynamic classpath**

```
import Groovy.sql.Sql
```

```
this.class.classLoader.rootLoader.addURL(new URL("file:/lib/ojdbc14.jar"))  
def driver="oracle.jdbc.driver.OracleDriver"  
def sql = Sql.newInstance("jdbc:oracle:thin:@localhost:1521:MyDB",  
    "scott", "tiger", driver);
```



```
final String s = "hello, Groovy world!"
int no = 0;
for (int i in 0..<s.size())
    if (s[i] == 'o')
        no ++
println "Found $no 'o' chars"
```

types
optional

```
final s = "hello, Groovy world!"
no = 0;
for (i in 0..<s.size())
    if (s[i] == 'o')
        no ++
println "Found $no 'o' chars"
```

```
def add(int x) { x + x }
```

```
println add(2)
println add('hello') ❌
```

```
def add(x) { x + x }
```

```
println add(2)
println add('hello')
```

```
println user?.config.loginDetail.passwordDetail.expiryDate
println user?.config.editorDetail.tabs.position
```

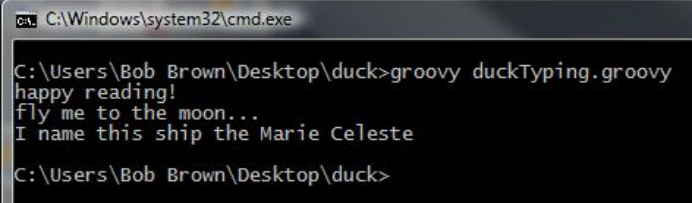
iteration across 'tabs' collection ↗

```
// 'duck' typing in action...
[new Book(), new Rocket(), new Ship()].each { it.launch() }
```

```
class Book {
    def launch() { println 'happy reading!' }
}
```

```
class Rocket {
    def launch() { println 'fly me to the moon...' }
}
```

```
class Ship {
    def launch() { println 'I name this ship the Marie Celeste' }
}
```



```
cmd. C:\Windows\system32\cmd.exe
C:\Users\Bob Brown\Desktop\duck>groovy duckTyping.groovy
happy reading!
fly me to the moon...
I name this ship the Marie Celeste
C:\Users\Bob Brown\Desktop\duck>
```

- operator overloading
 - all handle nulls gracefully, no NPE

```
class JukeBox {
    def songs = []

    def plus(song) { songs << song }

    def minus(song) {
        songs.remove(songs.lastIndexOf(song))
    }

    def printPlayList() {
        songs.each { song -> println "$song" }
    }
}

jbox = new JukeBox()
sng = "RaceWithDevilSpanishHighway.mp3"
jbox + "SpanishEyes.mp3"
jbox + sng
jbox + "Nena.mp3"
jbox.printPlayList()
jbox - sng
jbox.printPlayList()
```

a + b	a.plus(b)
a ^ b	a.xor(b)
a-- or --a	a.previous()
a[b]	a.getAt(b)
a[b] = c	a.putAt(b, c)
a << b	a.leftShift(b)
switch(a) { case(b) : }	b.isCase(a)
a <=> b	a.compareTo(b)
a > b	a.compareTo(b) > 0

- new operators
 - is; changed ==
 - safe dereference
 - 'elvis'

```
GroovyConsole
File Edit View History Script Help
[Icons]
def x = [ moniker: 'fred' ]
println "x has the value ${x.name?.size()}"
println "${x.name ?: 'missing'}"

groovy> def x = [ moniker: 'fred' ]
groovy> println "x has the value ${x.name?.size()}"
groovy> println "${x.name ?: 'missing'}"

x has the value null
missing

Execution complete. Result was null.
```

- default set of imports
 - java.{lang, util, io, net}
 - groovy.{lang, util}
 - java.math.Big{Integer, Decimal}

- enhanced import

```
import third.party.vendors.library.UsefulStuff as BugRiddenLib

// on-the-fly, ad-hoc, just-in-time, bespoke, bug fix...
class UsefulStuff extends BugRiddenLib {
    String buggyMethod(Integer arg) { "good value" }
}

// fix or no fix, nothing changes below here
def u = new UsefulStuff()
println u.buggyMethod(1)
```

- scripts don't have to define a main class
 - scripts get one created for them
 - along with `static main(args)`
- relaxed source/class relationship

- GroovyBeans
 - properties
 - constructors
 - implicit
 - named parameters
 - default parameter values



the real furball

```
class CatGroovyBean {
    static final UNKNOWN_AGE=-99

    String name
    short age=UNKNOWN_AGE
    String disposition

    def doActivity(a='sleep a lot') { println "I $a" }
}

furball = [name:'Furball', age:(short)3, disposition:'roly-poly'] as CatGroovyBean
println furball.dump()
furball.doActivity('hunt geckos')

snowball = new CatGroovyBean(name:'Snowball')
snowball.age = 4
snowball.disposition = 'voted for Sideshow Bob'
println snowball.dump()
snowball.doActivity('lick my fur')

garfield = new CatGroovyBean(name:'Garfield',
    disposition:'am too fat and lazy to do anything much')
println garfield.dump()
garfield.doActivity()
```

- **programs as data**

"Essentially a closure is a block of code that can be passed as an argument to a function call."

—<http://martinfowler.com/bliki/Closure.html>

- **uses**

- **anonymous methods**

- **callbacks, listeners**

```
(1..10).each { int i ->
  println i
}
```

- **sandwich code**

- **transactional, timing, logging, etc.**

```
def timeSandwich(filling) {
  start = System.currentTimeMillis()
  result = filling()
  [System.currentTimeMillis() - start, result]
}
```

```
[millis,res] = timeSandwich { int i = 0; (1..1000).each { i += it }; i }
println "$millis: $res"
```

- simple examples

```
def x(f) { f * 2 }
def y = this.&x
```

```
assert y(4) == 8
```

```
def myMP3s=new File(".")
  .listFiles( { dir, file -> file.toLowerCase().endsWith(".mp3") } as FilenameFilter)*.name
myMP3s.each { println it }
```

- currying

- "semi-precompiling" a function with several params into one with fewer params

```
def tellFortune = { fortune, date ->
  println "Fortune for ${date} is '${fortune}'"
}

fOne = tellFortune.curry("Fortune smiles on you")
fTwo = tellFortune.curry("Beware raven-haired cats")

today = new Date()
fOne today
fTwo today + 3
```

```
def eachMember = { fn, list ->
  list.collect { fn(it) } }

def doubler = eachMember.curry( {
  it * it } )
def tripler = eachMember.curry( {
  it * it * it } )

assert doubler([1,2,3]) == [1,4,9]
assert tripler([1,2,3]) == [1,8,27]
```

```
new File('/etc/passwd').splitEachLine(':') { tokens ->
    user = tokens[0]
    home = tokens[5]
    if ( ! home)
        println "${user}: HOME NOT DEFINED"
    else {
        exists = new File(home).exists()
        println "${user}: HOME(${home}) ${exists ? 'exists' : 'DOES NOT EXIST'}"
    }
}
```

```
#!/usr/bin/groovy

// uvi - vi a file without changing it's last modified time
if (args.size() != 1)
{
    println "usage: uvi filename"
    System.exit(1)
}

file = args[0]
origTime = new File(file).lastModified()
proc = "vi $file".execute()
proc.waitFor()
new File(file).setLastModified(origTime)
```

http://pleac.sourceforge.net/pleac_groovy/directories.html

```
// file:cleanSvn.groovy; delete all .svn directories
new AntBuilder().delete(includeemptydirs: true) {
    dirset(dir: '.') {
        include(name: "**/.svn")
    }
}
```

Digressions...

```
import org.apache.commons.lang.time.DurationFormatUtils as DFU
```

```
userMap = [:]  
[ "sh" , "-c" , "last" ].execute().text.splitEachLine(/\s/) {  
    user = it[0]  
    ms = millis(it[-1])  
    if (ms >= 0)  
        userMap[user] = userMap.get(user, 0) + ms  
}
```

```
println "ID\tCumulative Logged-in Time"  
userMap.each { u, ms -> printf "%-10s\t%s\n",  
    u, DFU.formatDuration(ms, 'HH:mm') }
```

```
def millis(str) {  
    def matcher = str =~ /[ \(\) \{ \} : \d { 2 } ] [ \] ] /  
    secs = -1  
    if (matcher.find()) {  
        h = Integer.valueOf(matcher.group(1))  
        m = Integer.valueOf(matcher.group(2))  
        secs = ((h * 60) + m) * 60  
    }  
    secs * 1000  
}
```

ID	Cumulative Logged-in Time
user0	04:43
user1	09:29
user222	04:50
bigusernm	03:20
me	01:18
bundy	04:43
rum	05:49
hare	00:00
reabbit	00:23
war	00:04
peace	33:21

The screenshot shows a Groovy console window with the following code:

```
server = new ServerSocket(5000)  
while (true)  
{  
    server.accept() { socket ->  
        socket.withStreams { input, output ->  
            // ignore input and just serve dummy content  
            output.withWriter { writer ->  
                writer << "HTTP/1.1 200 OK\n"  
                writer << "Content-Type: text/html\n"  
                writer << "<html><body>Hello World! It's ${new  
Date()}/</body></html>\n"  
            }  
        }  
    }  
}
```

A dialog box titled "Groovy executing" is overlaid on the console, with the message "Groovy is now executing. Please wait." and an "Interrupt" button.

Below the console, a Windows Internet Explorer window is open to <http://localhost:5000/>, displaying the output: "Hello World! It's Tue Sep 16 09:55:09 EST 2008".

Digressions...

```
// file: generatePlaylist.groovy
dirs = []
new File('.').eachDir { d ->
    dirs << d
}

count = 0
str = ""
dirs.sort { it.path }.each { d ->
    d.eachFileMatch( ~/.*wma/ ) { f ->
        str += "<media src=\"${f.path[2..-1]}\" />\n"
        count++
    }
}

println ""
<?wpl version="1.0"?>
<smil>
    <head>
        <meta name="ItemCount" content="${count}"/>
        <title>First Lensman</title>
    </head>
    <body>
${str}
    </body>
</smil>
""
```

```
<?wpl version="1.0"?>
<smil>
    <head>
        <meta name="ItemCount" content="197"/>
        <title>First Lensman</title>
    </head>
    <body>
<media src="Disk 1\01 Track 1.wma" />
<media src="Disk 1\02 Track 2.wma" />
<media src="Disk 1\03 Track 3.wma" />
<media src="Disk 1\04 Track 4.wma" />
<media src="Disk 1\05 Track 5.wma" />
...
<media src="Disk 9\19 Track 19.wma" />
<media src="Disk 9\20 Track 20.wma" />
<media src="Disk 9\21 Track 21.wma" />

    </body>
</smil>
```

```
import com.ibm.as400.access.*

def usage = {
    println "Usage: groovy jobs.groovy <user>"
    println "Display active job info for <user> (can be *ALL)"
}

if (this.args.length != 1)
    usage()
else {
    user = this.args[0]
    AS400 system = new AS400()
    JobList jobList = new JobList(system);
    jobList.addJobSelectionCriteria(JobList.SELECTION_USER_NAME, user);
    Enumeration list = jobList.getJobs();
    while (list.hasMoreElements()) {
        Job j = (Job) list.nextElement();
        println "Name: ${j.name}, Number: ${j.number}, User: ${j.user}, <br>CPU Used: ${j.cPUUsed}"
        println "Date entered: ${j.date}, Status: ${j.status}, Type: ${j.type}"
        println "Func Name: ${j.functionName}, Func Type: ${j.functionType}"
        println ""
    }
    println "Done."
}
```

<http://www.itjungle.com/fhg/fhg051607-story01.html>

```
Name: QPADEV0001, Number: 018735, User: **, CPU Used: 112
Date entered: Fri May 11 07:50:17 EDT 2007, Status: *ACTIVE, Type: I
Func Name: STRQSH, Func Type: C
```

```
Name: QZSHSH, Number: 018736, User: **, CPU Used: 31
Date entered: Fri May 11 07:52:39 EDT 2007, Status: *ACTIVE, Type: B
Func Name: QZSHSH, Func Type: P
```

```
Name: QP0ZSPWP, Number: 018765, User: **, CPU Used: 1975
Date entered: Fri May 11 08:39:52 EDT 2007, Status: *ACTIVE, Type: B
Func Name: GroovyStar, Func Type: J
```

```
import alice.tuprolog.*

//extra method so engine can be accessed just like regex patterns...
class Extras{
    static eachMatch(Prolog engine, String query, Closure action){
        def info= engine.solve(query)
        while( info.success ){
            action(info)
            if( engine.hasOpenAlternatives() ) info= engine.solveNext()
            else break
        }
    }
}
def engine= new Prolog()

//example rules describing how to append to a list...
engine.theory= new Theory(''
    myAppend([],X,X) .
    myAppend([X|L1],L2,[X|L3]) :-myAppend(L1,L2,L3) .
'')

//find all ways to produce list [1,2,3] using myAppend...
use(Extras){
    engine.eachMatch('myAppend(X,Y,[1,2,3]).'){
        println
        "$it.solution; bindings: ${it.toString().replaceAll('\n',' ')}"
    }
}

myAppend([], [1,2,3], [1,2,3]); bindings: yes. X / [] Y / [1,2,3]
myAppend([1], [2,3], [1,2,3]); bindings: yes. X / [1] Y / [2,3]
myAppend([1,2], [3], [1,2,3]); bindings: yes. X / [1,2] Y / [3]
myAppend([1,2,3], [], [1,2,3]); bindings: yes. X / [1,2,3] Y / []
```

Interoperability

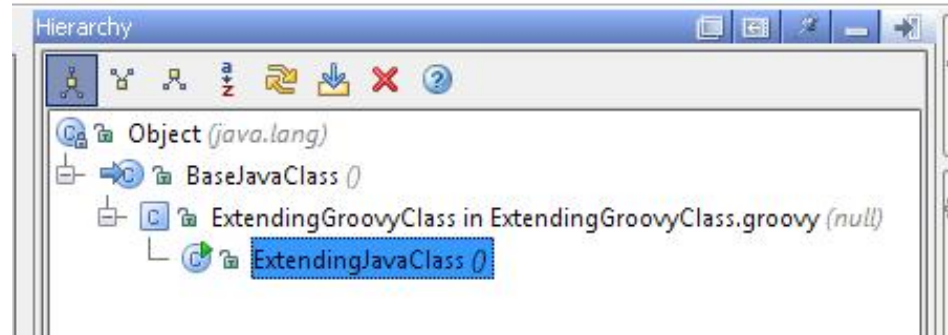
```
// file: BaseJavaClass.java
public class BaseJavaClass {
    protected Integer baseInteger;
}
```

```
// file: ExtendingGroovyClass.groovy
class ExtendingGroovyClass
    extends BaseJavaClass {
    def extendingGroovyLong
}
```

```
// file: ExtendingJavaClass.java
public class ExtendingJavaClass
    extends ExtendingGroovyClass {
    private String myString;
```

```
    private ExtendingJavaClass ()
    {
        baseInteger = new Integer(99);
        setExtendingGroovyLong(1234L);
        myString = "hello, World";
    }
```

```
    public static void main(String [] args)
    {
        ExtendingJavaClass ejc = new ExtendingJavaClass();
        System.out.printf(
            "It's a crazy, mixed-up world!\n\nbaseInteger: %d\nextendingGroovyLong: %d\nmyString: %s\n",
            ejc.baseInteger, ejc.getExtendingGroovyLong(), ejc.myString);
    }
}
```



"...you can now use the groovyc compiler to jointly compile both your Groovy and Java classes. And here's the kicker: it manages all the dependencies for you! Groovy class A extends Java class B which references Groovy class C? No problem."

—<http://jasonrudolph.com/blog/2007/07/05/groovy-11-beta-2-released-introduces-joint-compiler-for-java-groovy/>

- it's war out there, people...

*"the JRuby team are planning on ditching java.util.regex and implementing their own regex parser to match the Ruby regex semantics. ...this is the essence of the debate over Java integration. So whenever you drop out of Java and into Ruby-land you need to remember that the semantics differ between regex implementations. To be clear: **Java integration means more than just being able to call a method on a Java class.**"*

—<http://graemerocher.blogspot.com/2007/03/jruby-groovy-java-integration.html>



"It's alright to let yourself go, as long as you can get yourself back..."

—Mick Jagger

- *"The Groovy Way to Blow a Buttoned-Down Java Developer's Mind"**
- changes the 'feel' of the world...
 - expandos
 - metaClass
 - adding new methods to classes at runtime - even if they were originally implemented in Java, and even if they were declared final!
 - JDK → GDK
 - dynamic invocation
 - calling methods that don't exist and avoiding the dreaded `MethodNotFoundException`

- **groovy humour**

groovy source file: Uh, hi compiler, I need to invoke a method on an object — the thing is, I'm not totally sure that the object has this method.

groovyc: Relax, man. No worries, I'll just go ahead and generate some byte code for you.

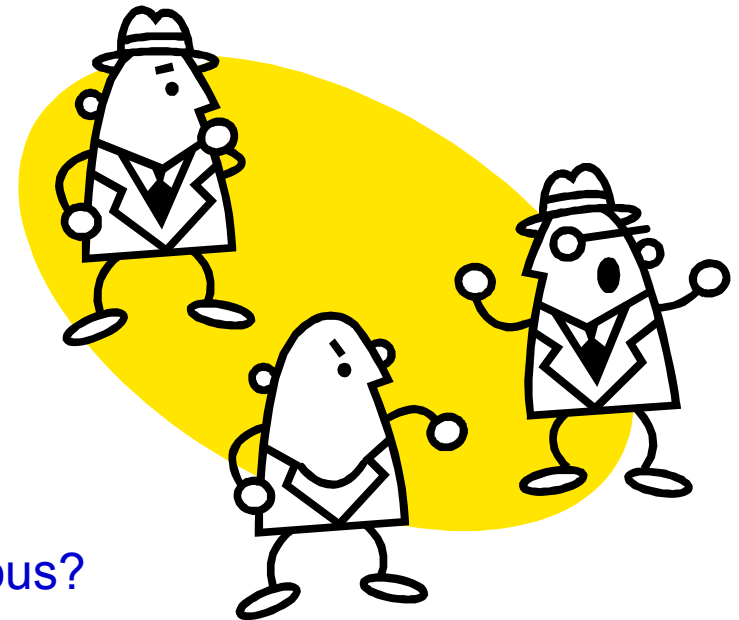
groovy source file: Really? You're not going to check to see if that method exists first?

groovyc: Static checking is for squares, man! We'll just let the Runtime handle it.

groovy source file: Isn't that dangerous?

groovyc: You're starting to sound like my old man, Java, and you're totally killing my buzz.

javac: I heard that! You kids with your dynamic method invocation! You're going to get RuntimeExceptions all over yourselves!



- run-time tests

```
def foo = "Hello World!"
if(foo.respondsTo(foo, "toUpperCase"))
    println foo.toUpperCase()
```

```
if(foo.metaClass.hasProperty("bytes"))
    println foo.bytes.encodeAsBase64()
```

- augment existing functionality...

- the "Pimp My Library" pattern

- this is used a lot

- to create groovy's *GDK*

```
// file:meta.groovy
import java.util.zip.*

File.metaClass.zip = { String destination ->
    def result = new ZipOutputStream(new FileOutputStream(destination))
    result.withStream { zipOutputStream ->
        delegate.eachFileRecurse { f ->
            if (!f.isDirectory()) {
                zipOutputStream.putNextEntry(new ZipEntry(f.getPath()))
                new FileInputStream(f).withStream { inStream ->
                    def buffer = new byte[1024]
                    def count
                    while ((count = inStream.read(buffer, 0, 1024)) != -1) {
                        zipOutputStream.write(buffer)
                    }
                }
                zipOutputStream.closeEntry()
            }
        }
    }
}

new File("stuff").zip("./stuff.zip")
```

<http://blog.xebia.com/2008/05/25/powerful-groovy/>

- dynamic dispatch
 - based on runtime type
 - not declared type as in Java
 - at the last moment

```
inl = System.in.newReader().&readLine

input1 = inl()
input2 = inl()
input3 = inl()

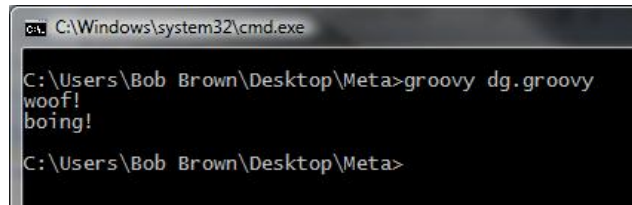
def f() { println "this is f()" }

def g() { println "this is g()" }

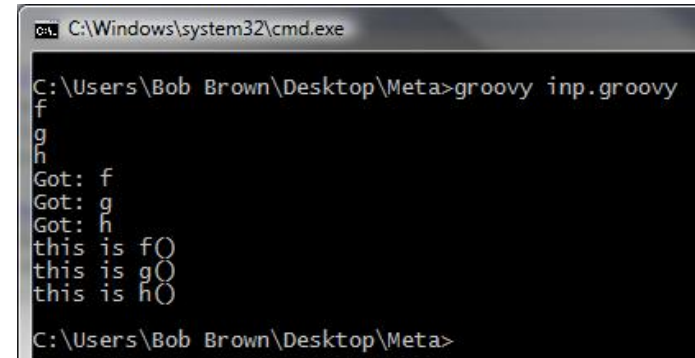
def h() { println "this is h()" }

[input1, input2, input3].each { println "Got: $it" }

"$input1"()
"$input2"()
"$input3"()
```



```
C:\Windows\system32\cmd.exe
C:\Users\Bob Brown\Desktop\Meta>groovy dg.groovy
woof!
boing!
C:\Users\Bob Brown\Desktop\Meta>
```



```
C:\Windows\system32\cmd.exe
C:\Users\Bob Brown\Desktop\Meta>groovy inp.groovy
f
g
h
Got: f
Got: g
Got: h
this is f()
this is g()
this is h()
C:\Users\Bob Brown\Desktop\Meta>
```

```
class Dog {
    def bark() { println "woof!" }
    def sit() { println "(sitting)" }
    def jump() { println "boing!" }
}

def doAction(animal, action) {
    animal."$action"()
}

def rex = new Dog()

doAction(rex, "bark")
doAction(rex, "jump")
```

<http://groovy.codehaus.org/Dynamic+Groovy>

Multimethods

```
public class BabyColors {  
  
    String color(Boy b) { return "Blue"; }  
    String color(Girl g) { return "Pink"; }  
    String color(Baby a) { return "Green"; }  
  
    String whatColor(Baby a) { return color(a); }  
  
    public static void main(String[] s) {  
        BabyColors bc = new BabyColors();  
        Baby a = new Baby();  
        Baby bb = new Boy();  
        Baby bg = new Girl();  
        Boy b = new Boy();  
        Girl g = new Girl();  
        System.out.println("Indirect -----");  
        System.out.println("unknown   - " + bc.whatColor(a));  
        System.out.println("baby boy   - " + bc.whatColor(bb));  
        System.out.println("baby girl  - " + bc.whatColor(bg));  
        System.out.println("      boy   - " + bc.whatColor(b));  
        System.out.println("      girl  - " + bc.whatColor(g));  
    }  
}  
  
class Baby {}  
class Boy extends Baby {}  
class Girl extends Baby {}
```

"does the type safe thing
and always buys green"

java

groovy

"peeks into the babies diaper
and buys the appropriate color"

```
C:\Windows\system32\cmd.exe  
C:\Users\Bob Brown\Desktop\b>javac BabyColors.java  
C:\Users\Bob Brown\Desktop\b>java BabyColors  
Indirect -----  
unknown   - Green  
baby boy   - Green  
baby girl  - Green  
      boy   - Green  
      girl  - Green  
C:\Users\Bob Brown\Desktop\b>
```

```
C:\Windows\system32\cmd.exe  
C:\Users\Bob Brown\Desktop\b>copy BabyColors.java BC.groovy  
1 file(s) copied.  
C:\Users\Bob Brown\Desktop\b>groovy BC.groovy  
Indirect -----  
unknown   - Green  
baby boy   - Blue  
baby girl  - Pink  
      boy   - Blue  
      girl  - Pink  
C:\Users\Bob Brown\Desktop\b>
```

9 October, 2008

- adding missing methods
 - per class or per instance
 - at runtime...

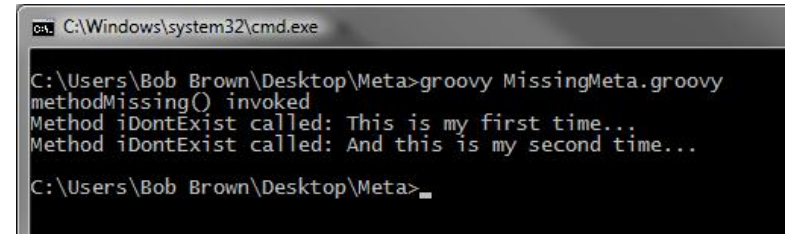
```
// file:MissingMeta.groovy
class Missing {

    def methodMissing(String name, args) {
        println "methodMissing() invoked"
        def method = {
            println "Method $name called: $it"
        }

        Missing.metaClass."$name" = method
        return method(args)
    }

}

new Missing().iDontExist('This is my first time...')
new Missing().iDontExist('And this is my second time...')
```



```
C:\Windows\system32\cmd.exe
C:\Users\Bob Brown\Desktop\Meta>groovy MissingMeta.groovy
methodMissing() invoked
Method iDontExist called: This is my first time...
Method iDontExist called: And this is my second time...
C:\Users\Bob Brown\Desktop\Meta>
```

"Which is, actually, a bit freaky when you think about it: you're changing a metaClass which subsequent instances would use, but you can't. Which means you can create bugs for people down the pipe which you won't experience.

Fun, ain't it?"

—http://groups.google.com/group/groovymn/browse_thread/thread/0bf1498ff2801683

- dynamically adding fields, etc.

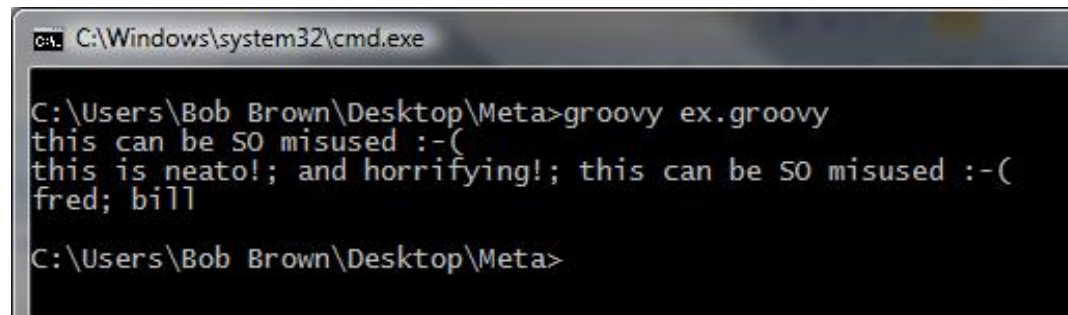
```
def x = new Expando()

x.field = "this is neat!"
x.otherField = "and horrifying!"
x.inputField = System.in.newReader().readLine()

println "${x.field}; ${x.otherField}; ${x.inputField}"

def y = new Expando(f0:'fred', f1:'bill')

println "${y.f0}; ${y.f1}"
```



```
C:\Windows\system32\cmd.exe
C:\Users\Bob Brown\Desktop\Meta>groovy ex.groovy
this can be SO misused :-(\
this is neat!; and horrifying!; this can be SO misused :-(\
fred; bill
C:\Users\Bob Brown\Desktop\Meta>
```

- add methods to classes for a limited scope
 - help create "self-documenting" code
 - and DSLs
 - adapted from Objective-C

```
class RandomHelper {
    static Random r = new Random()
    static Integer rand(Integer self) {
        r.nextInt(self)
    }
}

use (RandomHelper) {
    15.times { println 10.rand() }
}
```

```
class Multiplicity {
    static boolean isMultipleOf(Integer dividend,
                                Integer divisor) {
        dividend % divisor == 0
    }
}

use (Multiplicity.class) {
    (1..100).each { number ->
        if (number.isMultipleOf(5) &&
            number.isMultipleOf(7)) {
            print "fizzbuzz"
        } else if (number.isMultipleOf(5)) {
            print "fizz"
        } else if (number.isMultipleOf(7)) {
            print "buzz"
        } else {
            print "${number}"
        }
        print " "
    }
}
```

- support the creation of Domain Specific Languages
- out-of-the-box support
 - NodeBuilder, DOMBuilder, SAXBuilder
 - MarkupBuilder
 - XML/HTML
 - GraphicsBuilder
 - SwingBuilder
 - AntBuilder
- 3rd party
 - GoogleChartBuilder
 - <http://grails.org/Google+Chart+Plugin>
 - eclipse EMF Builder
 - <http://www.dinkla.net/groovy/emf.html>
 - PlanetarySystemBuilder
 - <http://thediscoblog.com/2007/07/06/builders-are-groovys-bag/>

```
import groovy.swing.SwingBuilder
import javax.swing.*
import java.awt.*
import javax.swing.event.*

tab = 0

pagesScanner = new AntBuilder().fileScanner {
    fileset(dir:'html', includes:'page*.xhtml')
}

private void tabStateChanged(ChangeEvent ce) {
    tab = ce.getSource().getSelectedIndex()
}

SwingBuilder.build() {

    tp = tabbedPane(stateChanged: this.&tabStateChanged) {
        for (f in pagesScanner) {
            editorPane(title: "${f.name.split('/')[0]}",
                contentType: 'text/html',
                text: f.text)
        }
    }

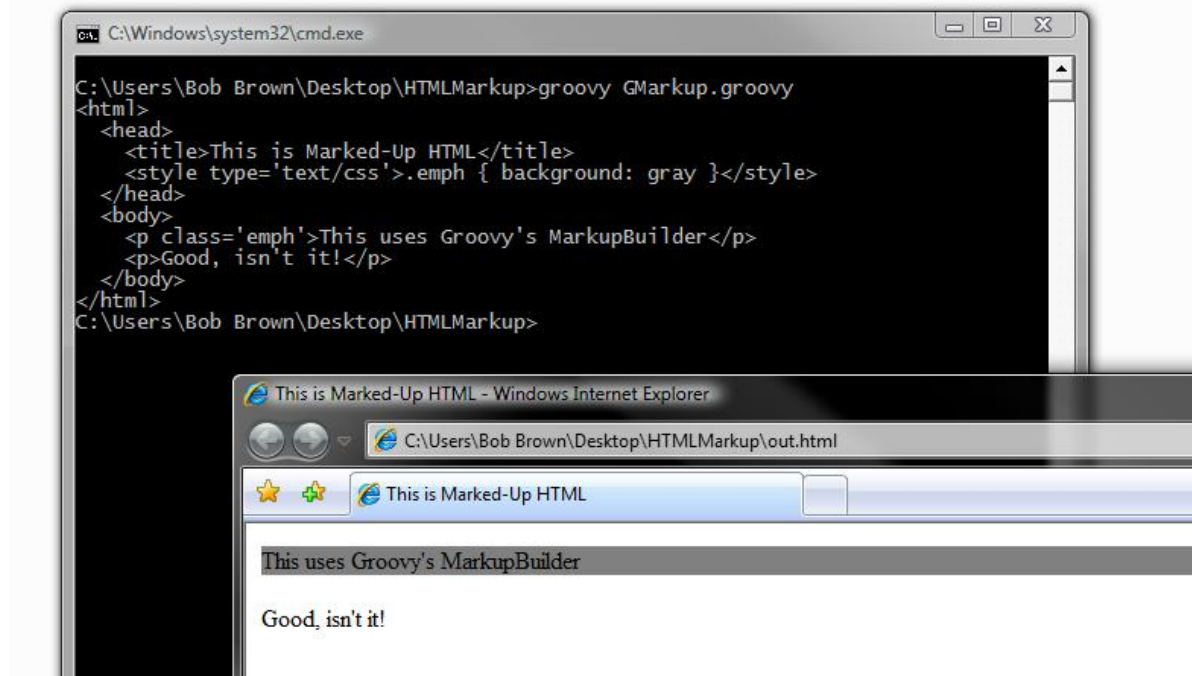
    frame(title:'Frame',
        size:[1024, 768],
        extendedState:Frame.MAXIMIZED_BOTH,
        defaultCloseOperation:JFrame.EXIT_ON_CLOSE,
        show:true) {
        widget(tp, constraints:BorderLayout.CENTER)
        panel(constraints:BorderLayout.SOUTH) {
            button(text:'Next',
                actionPerformed: { tp.setSelectedIndex(++tab) })
            button(text:'Previous',
                actionPerformed: { tp.setSelectedIndex(--tab) })
        }
    }
}
```



Builders...

```
import groovy.xml.MarkupBuilder

def builder = new MarkupBuilder ()
builder.html {
  head {
    title "This is Marked-Up HTML"
    style type:'text/css', ".emph { background: gray }"
  }
  body {
    p 'class':'emph', "This uses Groovy's MarkupBuilder"
    p (/Good, isn't it!/ )
  }
}
```



The screenshot shows a Windows command prompt window with the following text:

```
C:\Windows\system32\cmd.exe
C:\Users\Bob Brown\Desktop\HTMLMarkup>groovy GMarkup.groovy
<html>
<head>
  <title>This is Marked-Up HTML</title>
  <style type='text/css'>.emph { background: gray }</style>
</head>
<body>
  <p class='emph'>This uses Groovy's MarkupBuilder</p>
  <p>Good, isn't it!</p>
</body>
</html>
C:\Users\Bob Brown\Desktop\HTMLMarkup>
```

Below the command prompt is a Windows Internet Explorer browser window titled "This is Marked-Up HTML". The address bar shows "C:\Users\Bob Brown\Desktop\HTMLMarkup\out.html". The browser displays the rendered HTML output:

This uses Groovy's MarkupBuilder

Good, isn't it!

9 October, 2008

builders come into their own within Grails...

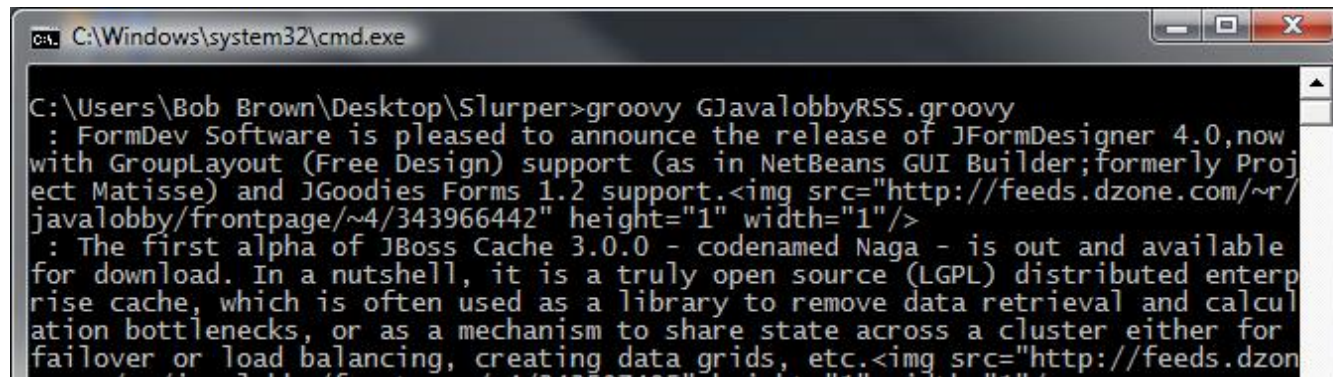
- consuming DSL formats
 - out-of-the-box
 - XMLSlurper
 - also HTML, of course
 - ConfigSlurper
 - properties files and groovy config files

"Nothing Makes You Want Groovy More Than XML..."

—<http://www.groovyongrails.com/article/63>

```
import groovy.util.XmlSlurper as S

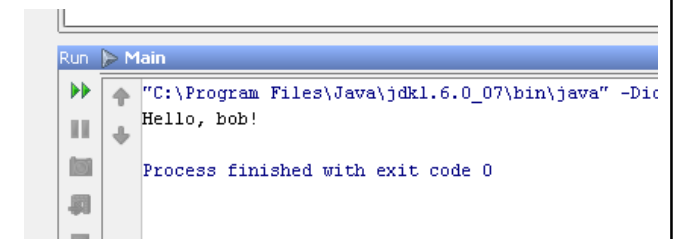
new S().parse('http://feeds.dzone.com/javalloby/frontpage')
    .channel[0].item.each {
println "$it.author : $it.description"
}
```



```
C:\Windows\system32\cmd.exe

C:\Users\Bob Brown\Desktop\Slurper>groovy GJavalobbyRSS.groovy
: FormDev Software is pleased to announce the release of JFormDesigner 4.0, now
with GroupLayout (Free Design) support (as in NetBeans GUI Builder; formerly Proj
ect Matisse) and JGoodies Forms 1.2 support.
: The first alpha of JBoss Cache 3.0.0 - codenamed Naga - is out and available
for download. In a nutshell, it is a truly open source (LGPL) distributed enterp
rise cache, which is often used as a library to remove data retrieval and calcul
ation bottlenecks, or as a mechanism to share state across a cluster either for
failover or load balancing, creating data grids, etc.= 1) ? args[0] : "world");
        gse.run("hello.groovy", binding);
        System.out.println(binding.getVariable("output"));
    }
}
```

```
// file:scripts/hello.groovy
output = "Hello, ${input}!"
```



- even more vital for dynamic languages
- Groovy contains support for basic stubbing and mocking
 - mockFor() and stubFor()
- encourages testing
 - dependency injection makes life easier
 - generates test harnesses alongside 'proper' code
- some 'groovy' tools out there...
 - Canoo WebTest
 - SoapUI
 - Hudson
 - TestNG
 - Cobertura

- **GroovyTestCase**
 - follows groovy's KISS path
 - enhanced JUnit facilities
 - also avoids the JUnit restriction of requiring all test* methods to be void

```
class TIntegrationTests extends GroovyTestCase {  
  
    void setUp() {  
        new T(i:99, s:'this is a new T').save()  
    }  
  
    void testThereIsNo42() {  
        shouldFail(NullPointerException) {  
            T.findByI(42).delete()  
        }  
    }  
  
    void testThereCanBeOnlyOne() {  
        assert T.count() == 1  
    }  
  
    void testTheOnlyOneIs99() {  
        assert T.findByI(99).i == 99  
    }  
}
```

Unit Test Results

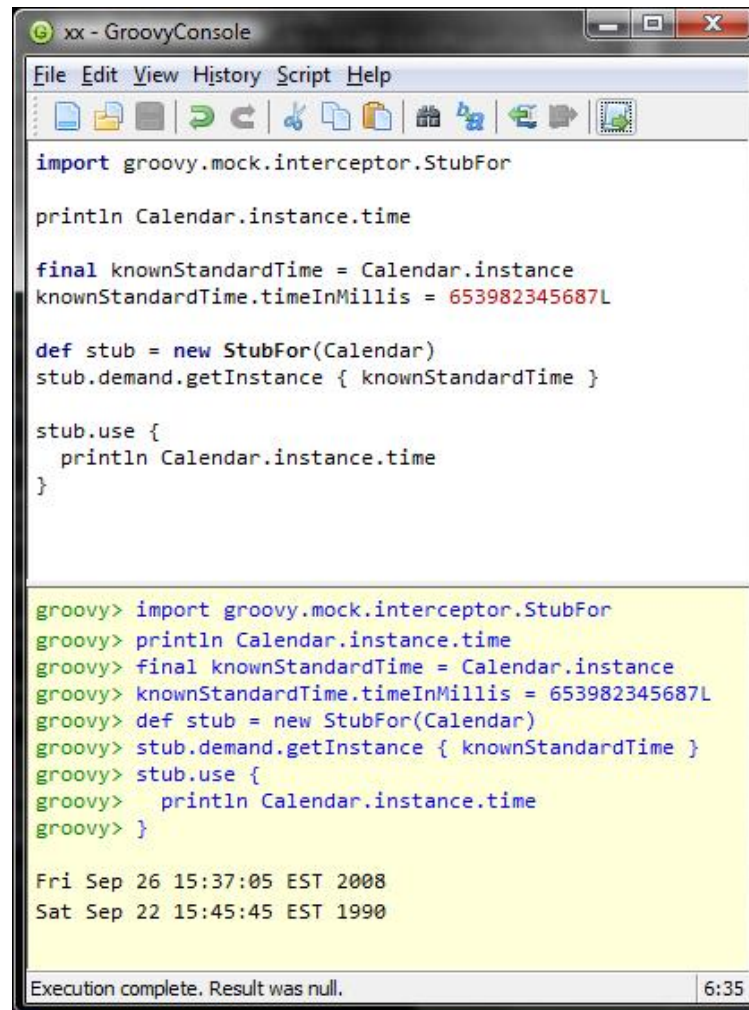
Class: TIntegrationTests

Name	Tests	Errors	Failures	Time(s)	Time Range	Start
TIntegrationTests	2	0	0	0.488	2008-08-19T03:24:31	19:19:10

Tests

Name	Status	Type	Time(s)
testThereCanBeOnlyOne	Success		0.304
testTheOnlyOneIs99	Success		0.028
testThereIsNo42	Success		0.008

- enable a Class Under Test (CUT) to run in isolation and lets you make assertions about state changes of the CUT



```
xx - GroovyConsole
File Edit View History Script Help
import groovy.mock.interceptor.StubFor
println Calendar.instance.time
final knownStandardTime = Calendar.instance
knownStandardTime.timeInMillis = 653982345687L
def stub = new StubFor(Calendar)
stub.demand.getInstance { knownStandardTime }
stub.use {
    println Calendar.instance.time
}

groovy> import groovy.mock.interceptor.StubFor
groovy> println Calendar.instance.time
groovy> final knownStandardTime = Calendar.instance
groovy> knownStandardTime.timeInMillis = 653982345687L
groovy> def stub = new StubFor(Calendar)
groovy> stub.demand.getInstance { knownStandardTime }
groovy> stub.use {
groovy>   println Calendar.instance.time
groovy> }

Fri Sep 26 15:37:05 EST 2008
Sat Sep 22 15:45:45 EST 1990

Execution complete. Result was null. 6:35
```

"One of the most amazing things I've ever seen done in software, something only a geek can appreciate..."

How often do you see software that acts that much like real magic? You've got to admit, that is beyond your expectations, really clean, and very cool."

—<http://leegrey.com/hmm/?m=200708>

- testing to ensure that the protocol for use of a class is correct

```
import groovy.mock.interceptor.MockFor

class GoodStuffClass {
    def open() { }
    def process() { }
    def close() { }
}

class GoodStuffTester {
    static main(args) {
        def mocked = new MockFor(GoodStuffClass)

        mocked.demand.open(1..1) { println 'opening'; true }
        mocked.demand.process(1..Integer.MAX_VALUE) { println 'processing'; true }
        mocked.demand.close(1..1) { println 'closing'; true }

        mocked.use {
            def systemUnderTest = new GoodStuffClass()
            systemUnderTest.open()
            (1..5).each { systemUnderTest.process() }
            // forgotten: systemUnderTest.close()
        }
    }
}
```

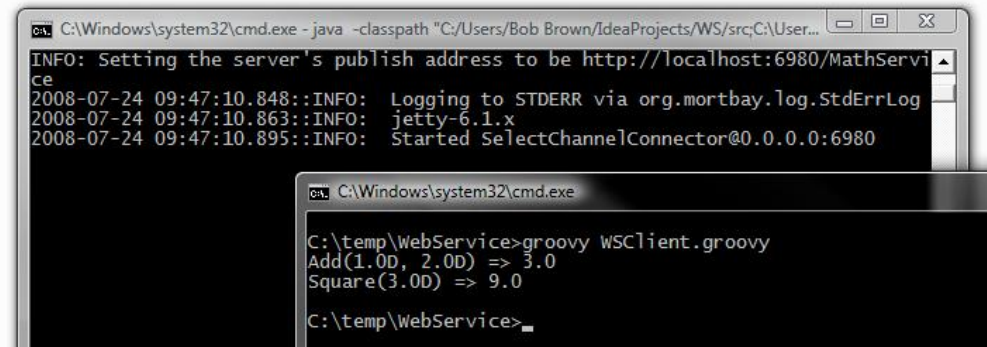


```
Run ▶ GoodStuffTester
"C:\Program Files\Java\jdk1.6.0_07\bin\java" -Didea.launcher.port=7537 "-Didea.launcher.bin.path=C:\Program Files\JetBrains\IntelliJ IDEA
opening
processing
processing
processing
processing
processing
Exception in thread "main" junit.framework.AssertionFailedError: verify[2]: expected 1..1 call(s) to 'close' but was called 0 time(s).
    at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
    at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:39)
```

"You won't believe how incredibly easy web services are in Groovy. Web services are suddenly easy to use, and not a major pain in the caboose... Traditionally, you need to deal with wsdl files, and stub classes, and all kind of icky, confusing nastiness. Not with Groovy."
— <http://joe.kueser.com/2007/10/13/more-groovy-goodies/>

```
//file:WSServer.groovy
class MathService {
    double add(double arg0, double arg1) {
        return (arg0 + arg1)
    }
    double square(double arg0) {
        return (arg0 * arg0)
    }
}

new groovyx.net.ws.WSServer().setNode('MathService', 'http://localhost:6980/MathService')
```



```
// file:WSClient.groovy
def proxy = new groovyx.net.ws.WSClient("http://localhost:6980/MathService?wsdl",
    this.class.classLoader)

println "Add(1.0D, 2.0D) => ${proxy.add(1.0D, 2.0D)}"
println "Square(3.0D) => ${proxy.square(3.0D)}"
```


- scripting Ant tasks using Groovy
 - overcome some of the limitations of plain ant
 - can have arbitrary Groovy methods within the build scripts

```
ant.property (file : 'build.properties')
antProperty = Ant.project.properties

target (welcome : 'Say hello to the world, groovy style') {
  echo welcome_description
  echo "Properties: $antProperty.prop1, $antProperty.prop2"
  (3..1).each { echo message:"\thello...$it!" }
  echo "FYI, GANT_HOME is ${antProperty.'environment.GANT_HOME'}"
}

target (cleanup : 'Start from a clean slate') {
  delete dir:'stuff\'
  echo "finished target \'$cleanup_description\'"
}

target (filesystemStuff : 'Do stuff with the filesystem') {
  depends cleanup,welcome
  mkdir dir:'stuff\'
  copy file:'build.gant', toDir:'stuff\'
  cleanup()
  echo 'Munged the filesystem'
}

setDefaultTarget filesystemStuff
```

A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The prompt is at "C:\Users\Bob Brown\Desktop\Gant>gant". The output shows the execution of the Gant build script, including messages for target completion, property echoing, file copying, and directory deletion.

```
C:\Windows\system32\cmd.exe
C:\Users\Bob Brown\Desktop\Gant>gant
[echo] finished target 'Start from a clean slate'
[echo] Say hello to the world, groovy style
[echo] Properties: hello, world
[echo]   hello...3!
[echo]   hello...2!
[echo]   hello...1!
[echo] FYI, GANT_HOME is C:\DEVTOOLS\gant-1.4.0
[mkdir] Created dir: C:\Users\Bob Brown\Desktop\Gant\stuff
[copy] Copying 1 file to C:\Users\Bob Brown\Desktop\Gant\stuff
[delete] Deleting directory C:\Users\Bob Brown\Desktop\Gant\stuff
[echo] finished target 'Start from a clean slate'
[echo] Munged the filesystem
C:\Users\Bob Brown\Desktop\Gant>
```

- simple database-oriented facilities
 - wrapping JDBC

```
import groovy.sql.Sql

def foo = 'cheese'
def sql = Sql.newInstance("jdbc:mysql://localhost:3306/mydb",
    "user", "pswd", "com.mysql.jdbc.Driver")

sql.eachRow("select * from FOOD where type=${foo}") {
    println "Gromit likes ${it.name}"
}
```

```
import groovy.sql.Sql

def sql = Sql.newInstance("jdbc:mysql://localhost:3306/mydb",
    "user", "pswd", "com.mysql.jdbc.Driver")
def food = sql.dataSet('FOOD')
def cheese = food.findAll { it.type == 'cheese' }
cheese.each {
    println "Eat ${it.name}"
}
```

One of Bob's rules for the groovy enterprise: *"thou shalt not SQL."*
(...Grails/GORM [*GORM is humane ORM*] shows the way...)

- a COM automation library for Groovy
 - "...looks an awful lot like Windows Scripting Host (WSH) ...for Java."

```
import org.codehaus.groovy.scriptom.util.office.ExcelHelper as H
import java.text.SimpleDateFormat
```

```
final fmt = new SimpleDateFormat('MM/yyyy')
```

```
new H().process(new File("./report.xls")) { workbook ->
  def worksheet = workbook.Sheets.Item['DATA']
```

```
  SafeArray a = worksheet.UsedRange.Value
```

```
  a.bounds[0].each { row ->
    print "\t"
    a.bounds[1].each { col ->
      if (a[row, col] instanceof String)
        print "[${a[row, col]}".center(10)
      else if (a[row, col] instanceof Date)
        print fmt.format(a[row, col]).center(10)
      else
        print a[row, col].toString().center(10)
    }
  }
  println()
}
```

```
import org.codehaus.groovy.scriptom.ActiveXProxy

def outlook = new ActiveXProxy("Outlook.Application")
def message = outlook.CreateItem(0)
def emails = "user1@domain1.com;user2@domain2.com"
def rec = message.Recipients.add(emails)
rec.Type = 1 // To = 1, CC = 2, BCC = 3
message.Display(true)`
```

- enterprise-grade web framework
 - inspired by "Ruby-on-Rails"
- best-of-breed everything!
 - beat that! *seriously!*



*"...Grails is supported by **proven technologies**.*

***Hibernate**, a de facto standard in the software industry, provides the basis for the object-relational mapping (ORM) in Grails.*

*The **Spring Framework** supplies the core of the Grails Model-View-Controller (MVC) architecture and enables powerful dependency injection.*

***SiteMesh** brings flexible and effective layout management to Grails.*

*And, let's not forget **Java**. Because of Groovy's excellent Java integration, Grails applications not only have direct access to the multitude of Java libraries, but also to the enterprise services (distributed transactions, messaging, etc.) provided by JEE..."*

—<http://www.infoq.com/minibooks/grails>

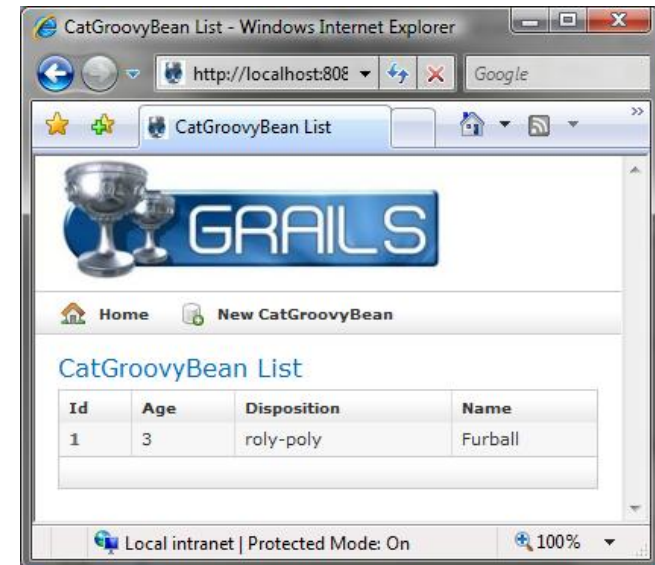
- convention over configuration
- scaffolding
 - generate full CRUD webapp with minimal effort

persistent domain class

```
class CatGroovyBean {  
    String name  
    short age  
    String disposition  
}
```

controller class

```
class CatGroovyBeanController {  
    static scaffold = CatGroovyBean  
}
```



- grails-like rich Swing client framework



- build system 'inspired' by Grails
 - '...by "inspired" I mean "taking large chunks of Grails code to bootstrap the codebase..."'
- a structure that supports/rewards MVC
- Groovy goodness: builders, @Bindable annotation, metaclass method injection, scripts, etc.
- declarative layout of GUI code in the view
- 3rd-party widget libraries
 - JIDE and SwingX are supported out of the box
- automatic packaging and signing for WebStart, Applet, and traditional application deployment
 - from the **SAME** source

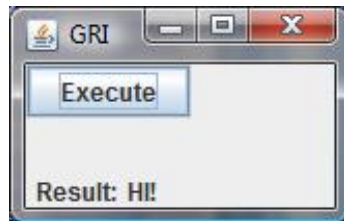
“Griffon is to the desktop what Grails is to the web.”

—<http://groovy.dzone.com/news/hello-griffon>

model

```
import groovy.beans.Bindable

class GRIModel {
    @Bindable def greeting = ""
}
```



```
application(title:'GRI',
            pack:true,
            locationByPlatform:true) {
    BorderLayout()
    hbox(constraints:NORTH) {
        button("Execute", actionPerformed:controller.&executeScript)
    }
    hbox(constraints:SOUTH) {
        hstrut(5)
        label("Result:")
        hstrut(5)
        label(text:bind {model.greeting})
    }
}
```

controller

```
import java.awt.event.ActionEvent

class GRIController {

    // these will be injected by Griffon
    def model
    def view

    def executeScript(ActionEvent evt =
null) {
        doOutside {
            model.greeting = 'HI!'
        }
    }
}
```

view

- ah...yes...well...

Groovy 1.0!

"...Java seems to be *three powers of ten faster* than Groovy, e.g. Groovy takes circa 2s at 103 test runs while Java takes 2s at 106 test runs..."

—<http://www.christianschenk.org/blog/performance-comparison-between-groovy-and-java/>

- programmer productivity vs. wall-clock time
- “good enough” is subjective
 - unless you need a weapon to use against groovy!
- things can only get better!
 - 1.5 up to 34% faster than 1.0
 - 1.6β-1



"...compared to the current Groovy 1.5.6 stable release, the **performance improvements range from 150% to 460%**....our main focus over the past 10 months has clearly been on performance. Between Groovy 1.0 and 1.5.1, on these same tests, we had already gained up to 80% speed improvements, and even between "dot releases" (1.5.1 and 1.5.6) we gained again up to 40% more."

—<http://glaforge.free.fr/weblog/index.php?itemid=241>

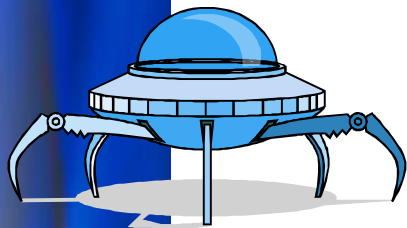
- what to do? what to do?
 - flail around looking for a silver(ruby?) bullet?
- be pragmatic
 - "good enough" is subjective



```
(0..untilYouBelieveIt).each { brain << "'Good Enough' is good enough" }
```

- **Groovy is Java**

- if Groovy isn't fast enough, write the 'hotspots' in Java



- if Java isn't fast enough, there's JNI/C
 - if C isn't fast enough, there's assembly
 - » if assembly can't cut it, buy more hardware?

- "premature optimization is the root of all evil"

• GJIT

"It's 200% faster than current Groovy, and ~20-30% slower than Java. There's still room to go. I hope to take some more steps closer to Java's speed."

—<http://chanwit.blogspot.com/2008/08/java-near-speed-groovy.html>

```
$ export JAVA_OPTS="-server"
$ time `java -cp bin alioth.PartialSumsJ 5000000`
real    0m10.246s
user    0m0.030s
sys     0m0.015s
$ time `groovy.bat test/partial_sums.groovy 5000000`
real    0m27.110s
user    0m0.030s
sys     0m0.016s
$ time `groovy.bat test/alioth/PartialSums.groovy 5000000`
real    0m28.409s
user    0m0.030s
sys     0m0.015s
$ export JAVA_OPTS="-server
-javaagent:C:\\groovy-ck1\\healthy\\target\\install\\lib\\gjit-0.1.jar"
$ time `groovy.bat test/alioth/PartialSums.groovy 5000000`
real    0m13.434s
user    0m0.030s
sys     0m0.030s
```

11 COMMENTS:

 Jacek Furmankiewicz said...

You guys should rush like crazy to make this a stable official build and get as close to Java as possible (if you get within 10% I think that will be enough for many to take notice).

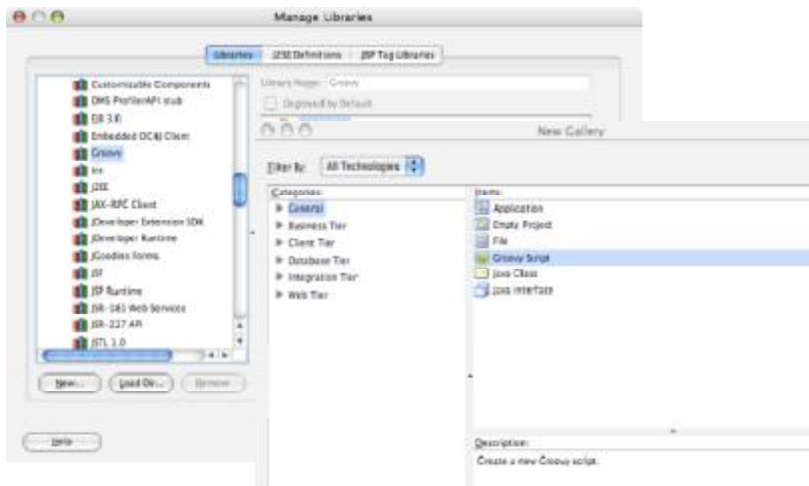
Compared to the lack of progress on Java lang features in Java 7 this may truly give us Java developers a reason to look at Groovy more seriously.

Great work, much respect.

2:56 AM

What about Oracle?

- **thought you'd never ask!**
 - JDeveloper 10g plugin
 - <http://docs.codehaus.org/display/GROOVY/Oracle+JDeveloper+Plugin>
 - "a work in progress"
 - ADF 11g will have Groovy support
 - <http://www.oracle.com/technology/oramag/oracle/07-nov/o67frame.html>
 - http://www.oracle.com/technology/pub/articles/oak-grails.html?rssid=rss_otn_articles
 - *expect to see much more*
 - directly, and via indirect influence
 - e.g.: <http://www.slideshare.net/tgrall/scripting-oracle-develop-2007>



groovy and ODI

You like Groovy ?

you like Oracle Data Integrator ?

How about using both ?



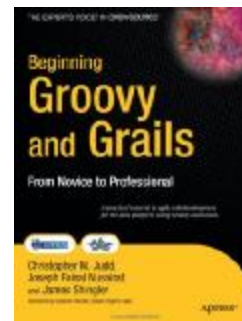
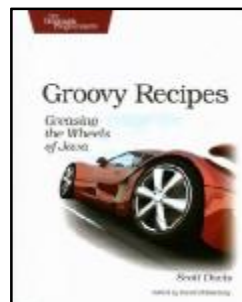
- **G4OO (Groovy for OpenOffice),** <http://extensions.services.openoffice.org/project/GroovyForOpenOffice>
- **IntelliJIDEA,** http://www.jetbrains.com/idea/features/groovy_grails.html
- **Netbeans, Eclipse IDEs**
- **Hudson,** <https://hudson.dev.java.net/>
- **Confluence,** <http://confluence.atlassian.com/display/CONFEXT/Groovy+Macro>
- **JBoss Seam,** <http://seamframework.org/>
- **SoapUI,** <http://www.soapui.org>
- **Freemind,** <http://freemind.sourceforge.net>
- **Gravl,** <http://code.google.com/p/gravl/>
- **CoW, a Controllable Wiki,** <http://gatewiki.sourceforge.net/>

"I know a few Fortune 500 companies...that trust tens of thousand lines of Groovy code....I'm still waiting for hearing about companies using Ruby for anything else than Rails-powered web sites..."

—<http://www.javalobby.org/java/forums/m92168317.html>

Resources

- <http://groovy.codehaus.org>
- <http://viva.sourceforge.net/talk/jug-mar-2004/slides.html>
- http://blogs.sun.com/sundararajan/entry/java_groovy_and_j_ruby
- the “Computer Language Benchmarks Game” tests,
<http://shootout.alioth.debian.org/>
- http://pleac.sourceforge.net/pleac_groovy/index.html
<http://groovy.codehaus.org/Oracle+JDeveloper+Plugin>
- http://www.nearinfinity.com/blogs/page/sfarley?entry=gpath_versus_x_path
- <http://tech.groups.yahoo.com/group/groovy-melbourne/>
- <http://groups.google.com/group/groovy-sydney>
- <http://www.infoq.com/minibooks/grails>
- aboutGroovy.com



Brisbane boy
makes good 😊

