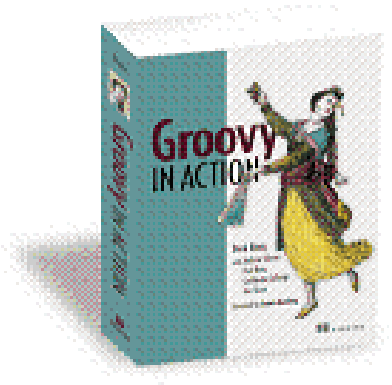


Groovy Tutorial

Dr Paul King

ASERT, Australia

Topics

- **Introduction**
- Language Basics
- Closures
- Builders
- Data Access
- Other Features
- Testing with Groovy
- Further Integration
- Grails
- More Information

What is Groovy?

- *"Groovy is like a **super version of Java**. It can leverage Java's enterprise capabilities but also has cool productivity features like closures, DSL support, builders and dynamic typing."*

```
Groovy = Java – boiler plate code
               + optional dynamic typing
               + closures
               + domain specific languages
               + builders
               + metaprogramming
```

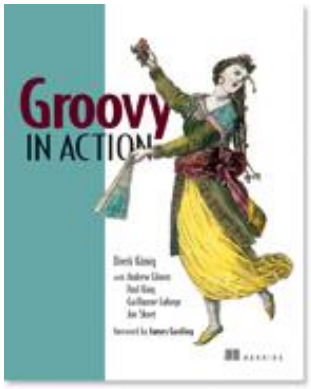
Groovy Goodies Overview

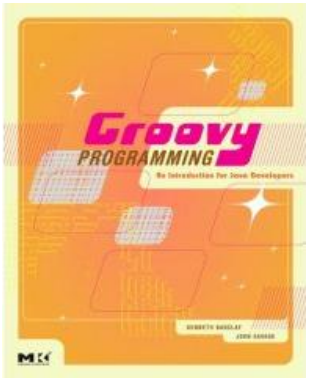
- Fully object oriented
- Closures: reusable and assignable pieces of code
- Operators can be overloaded
- Multimethods
- Literal declaration for lists (arrays), maps, ranges and regular expressions

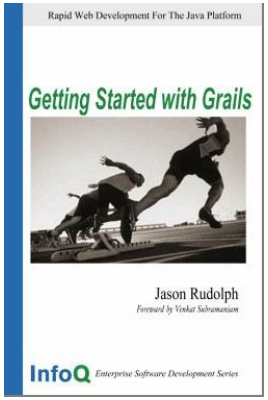
- GPath: efficient object navigation
- GroovyBeans
- grep and switch
- Templates, builder, swing, Ant, markup, XML, SQL, XML-RPC, Scriptom, Grails, tests, Mocks

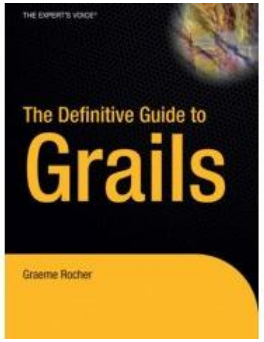
Growing Acceptance

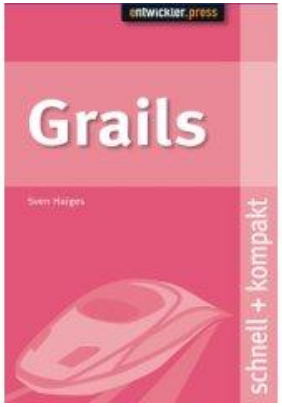
- A slow and steady start but now gaining in momentum; growing in maturity & mindshare

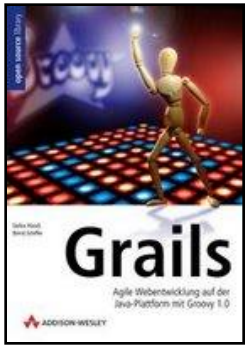








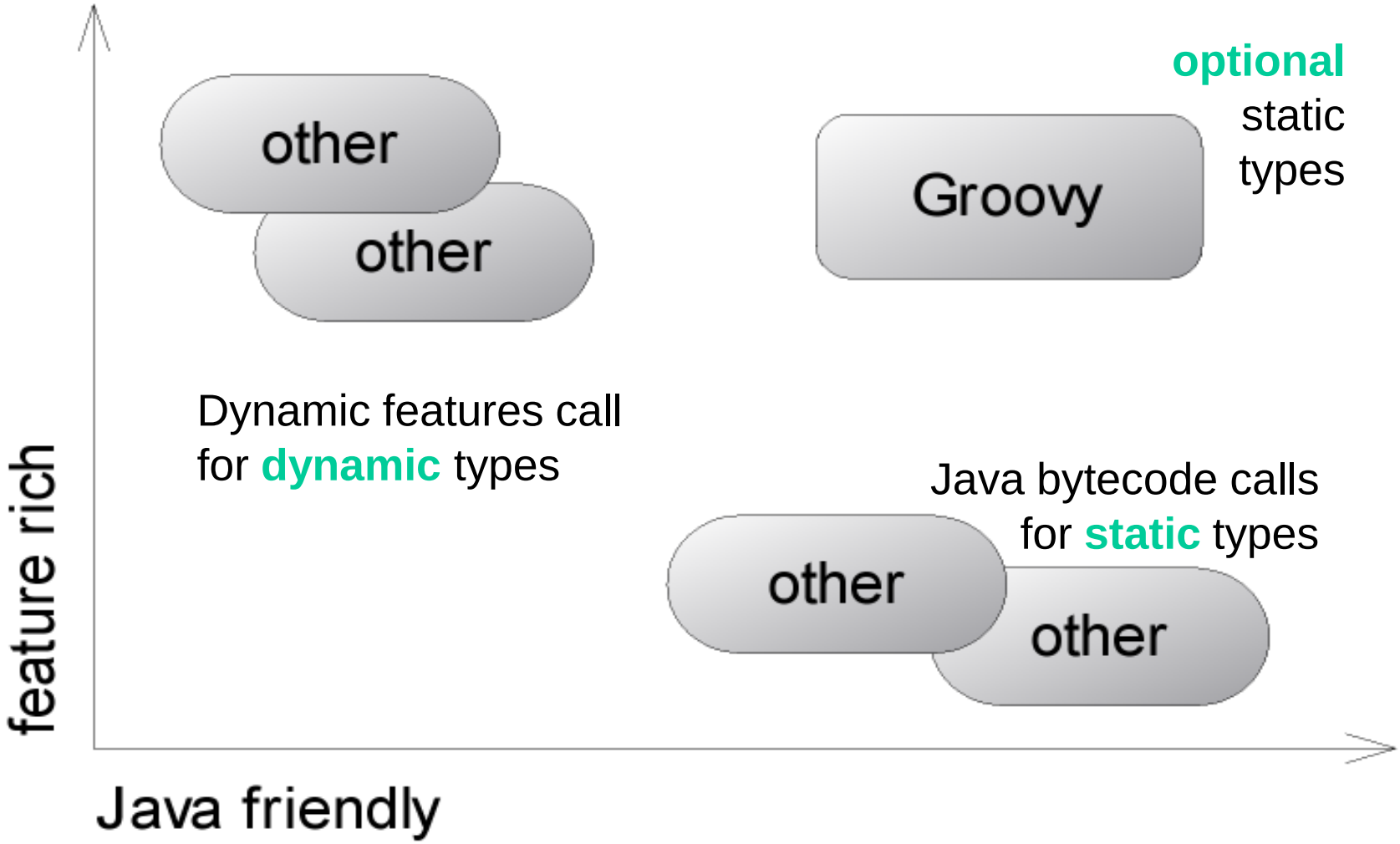




Groovy Recipes

Pragmatic Bookshelf

The Landscape of JVM Languages



The terms “Java Virtual Machine” and “JVM” mean a Virtual Machine for the Java™ platform.

Groovy Starter

```
System.out.println("Hello, World!");   // optional semicolon,
println 'Hello, World!'                // System.out, brackets,
                                       // main() method

def name = 'Guillaume'                 // dynamic typing
println "$name, I'll get the car."     // GString

String longer = """${name}, the car
is in the next row."""                 // multi-line string
                                       // with static typing

assert 0.5 == 1/2                      // BigDecimal equals()

def printSize(obj) {                   // optional duck typing
    print obj?.size()                  // safe dereferencing
}

def animals = ['ant', 'bee', 'cat']    // native list syntax
assert animals.every { pet ->          // closure support
    pet < 'dog'                        // overloading
}
```

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);
        }
    }
}
```

This code is valid Java and valid Groovy

Based on an example by Jim Weirich & Ted Leung

...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);
        }
    }
}
```

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

...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)
        }
    }
}
```

...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)
        }
    }
}
```

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

...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)
}
```

...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?

...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) }
```

...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?

...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
```

...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);
        }
    }
}
```

Better JavaBeans...

```
import java.math.BigDecimal;

public class BikeJavaBean {
    private String manufacturer;
    private String model;
    private int frame;
    private String serialNo;
    private double weight;
    private String status;
    private BigDecimal cost;

    public BikeJavaBean(String manufacturer, String model,
                        int frame, String serialNo,
                        double weight, String status) {
        this.manufacturer = manufacturer;
        this.model = model;
        this.frame = frame;
        this.serialNo = serialNo;
        this.weight = weight;
        this.status = status;
    }

    public String toString() {
        return "Bike:“ +
                “\n    manufacturer -- " + manufacturer +
                "\n    model        -- " + model +
                "\n    frame        -- " + frame +
                "\n    serialNo     -- " + serialNo +
                "\n";
    }

    public String getManufacturer() {
        return manufacturer;
    }

    public void setManufacturer(String manufacturer) {
        this.manufacturer = manufacturer;
    }

    public String getModel() {
        return model;
    }
    ...
```

```
    ...

    public void setModel(String model) {
        this.model = model;
    }

    public int getFrame() {
        return frame;
    }

    public String getSerialNo() {
        return serialNo;
    }

    public void setSerialNo(String serialNo) {
        this.serialNo = serialNo;
    }

    public double getWeight() {
        return weight;
    }

    public void setWeight(double weight) {
        this.weight = weight;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public BigDecimal getCost() {
        return cost;
    }

    public void setCost(BigDecimal cost) {
        this.cost = cost.setScale(3, BigDecimal.ROUND_HALF_UP);
    }
}
```


Modified example from: http://www.ibm.com/developerworks/library/j-pg09196.html

...Better JavaBeans...

```
import java.math.BigDecimal;

public class BikeJavaBean {
    private String manufacturer;
    private String model;
    private int frame;
    private String serialNo;
    private double weight;
    private String status;
    private BigDecimal cost;

    public BikeJavaBean(String manufacturer, String model,
                        int frame, String serialNo,
                        double weight, String status) {
        this.manufacturer = manufacturer;
        this.model = model;
        this.frame = frame;
        this.serialNo = serialNo;
        this.weight = weight;
        this.status = status;
    }

    public String toString() {
        return "Bike:“ +
                “\n    manufacturer -- " + manufacturer +
                "\n    model        -- " + model +
                "\n    frame        -- " + frame +
                "\n    serialNo     -- " + serialNo +
                "\n";
    }

    public String getManufacturer() {
        return manufacturer;
    }

    public void setManufacturer(String manufacturer) {
        this.manufacturer = manufacturer;
    }

    public String getModel() {
        return model;
    }
    ...
```

```
    ...

    public void setModel(String model) {
        this.model = model;
    }

    public int getFrame() {
        return frame;
    }

    public String getSerialNo() {
        return serialNo;
    }

    public void setSerialNo(String serialNo
        this.serialNo = serialNo;
    }

    public double getWeight() {
        return weight;
    }

    public void setWeight(double weight) {
        this.weight = weight;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public BigDecimal getCost() {
        return cost;
    }

    public void setCost(BigDecimal cost) {
        this.cost = cost.setScale(3, BigDecimal.ROUND_HALF_UP);
    }
}
```

Auto getters?
Auto setters?
Auto construction?

... Better JavaBeans

```
class BikeGroovyBean {
    String manufacturer, model, serialNo, status
    final Integer frame
    Double weight
    BigDecimal cost

    public void setCost(BigDecimal newCost) {
        cost = newCost.setScale(3, BigDecimal.ROUND_HALF_UP)
    }

    public String toString() {
        return """Bike:
    manufacturer -- $manufacturer
    model        -- $model
    frame        -- $frame
    serialNo     -- $serialNo
"""
    }
}
```

Topics

- Introduction
- **Language Basics**
- Closures
- Builders
- Data Access
- Other Features
- Testing with Groovy
- Further Integration
- Grails
- More Information

Strings…

- **Several forms**
  - Single quotes for simple strings
  - Double quotes for GStrings which support variable expansion
  - Slashy strings behave like GStrings but preserve backslashes (great for regex and directory names)
  - Multi-line versions

```
// normal strings
def firstname = 'Kate'
def surname= "Bush"
assert firstname * 2 == 'KateKate'

// GString
def fullname = "$firstname $surname"
assert fullname == 'Kate Bush'
assert fullname - firstname == ' Bush'
assert fullname.padLeft(10) ==
                              ' Kate Bush'

// indexing (including ranges)
assert fullname[0..3] == firstname
assert fullname[-4..-1] == surname
assert fullname[5, 3..1] == 'Beta'
```

…Strings…

```
// Multi-line strings
def twister = '''\
She sells, sea shells
By the sea shore'''

def lines =
    twister.split('\n')
assert lines.size() == 2

def address = """
$fullname
123 First Ave
New York
""".trim()

def lines =
    address.split('\n')
assert lines.size() == 3
```

```
// slashy string: (almost) no escaping
def path = /C:\Windows\System32/

def plain = '\n\r\t\b\\\f\$'
assert plain.size() == 7
def slashy = /\n\r\t\b\\\f\$/
assert slashy.size() == 14

// late binding trick with closures
fullname = "${-> firstname} $surname"
assert fullname == 'Kate Bush'
firstname = 'George'
surname = 'Clooney'
assert fullname == 'George Bush'
```

```
println """
 ----------------------
|    $fullname         |
|    123 First Ave     |
|    New York          |
 ----------------------
"""
```

…Strings

```
// more substrings
string = 'hippopotamus'
assert string - 'hippo' - 'mus' + 'to' == 'potato'
assert string.replace('ppopotam','bisc') == 'hibiscus'

// processing characters
assert 'apple'.toList() == ['a', 'p', 'p', 'l', 'e']
//also: 'apple' as String[], 'apple'.split(''), 'apple'.each{}
string = "an apple a day"
assert string.toList().unique().sort().join() == ' adelnpy'

// reversing chars/words
assert 'string'.reverse() == 'gnirts'

string = 'Yoda said, "can you see this?"'
revwords = string.split(' ').toList().reverse().join(' ')
assert revwords == 'this?" see you "can said, Yoda'

words = ['bob', 'alpha', 'rotator', 'omega', 'reviver']
bigPalindromes = words.findAll{w -> w == w.reverse() && w.size() > 5}
assert bigPalindromes == ['rotator', 'reviver']
```

Numbers...

- **Java Approach**
  - **Supports primitive types and object types**
  - **Has wrapper classes to allow conversion**
  - **Java 5+ has autoboxing to hide difference**
- **Groovy Approach**
  - **Treats everything as an object at the language level**
    - ***And does appropriate autoboxing under the covers when integrating with Java***
  - **BigDecimal used as the default type for non-Integers**
  - **Operator overloading applies to all common operations on numbers, e.g. 3 * 4 is the same as 3.multiply(4)**
    - ***You can use these operators for your own types, e.g. Person * 3 calls the multiple method on your person object***

...Numbers

```
def x = 3
def y = 4
assert x + y == 7
assert x.plus(y) == 7
assert x instanceof Integer

assert 0.5 == 1/2   // uses BigDecimal arithmetic as default
def a = 2 / 3       // 0.6666666666
def b = a.setScale(3, BigDecimal.ROUND_HALF_UP)
assert b.toString() == '0.667'

assert 4 + 3 == 7             // 4.plus(3)
assert 4 - 3 == 1             // 4.minus(3)
assert 4 * 3 == 12            // 4.multiply(12)
assert 4 % 3 == 1             // 4.mod(3)
assert 4 ** 3 == 64           // 4.power(3)
assert 4 / 3 == 1.3333333333  // 4.div(3)
assert 4.intdiv(3) == 1       // normal integer division

assert !(4 == 3)    // !(4.equals(3))
assert 4 != 3       // ! 4.equals(3)
assert !(4 < 3)     // 4.compareTo(3) < 0
assert !(4 <= 3)    // 4.compareTo(3) <= 0
assert 4 > 3        // 4.compareTo(3) > 0
assert 4 >= 3       // 4.compareTo(3) >= 0
assert 4 <=> 3 == 1 // 4.compareTo(3)
```

Dates...

- Mostly same support as Java
- Use java.util.Date or java.util.Calendar
  - Or java.sql.Date etc.
  - Can use static imports to help
- Or 3rd party package like Joda Time
- Does have special support for changing times:

```
date2 = date1 + 1.week – 3.days + 6.hours
```

- More utility methods expected for inputting and outputting dates
- Potentially will change with JSR 310

...Dates

```
import static java.util.Calendar.getInstance as now
import org.codehaus.groovy.runtime.TimeCategory
import java.text.SimpleDateFormat

println now().time

def date = new Date() + 1
println date

use(TimeCategory) {
    println new Date() + 1.hour + 3.weeks - 2.days
}

input = "1998-06-03"
df1 = new SimpleDateFormat("yyyy-MM-dd")
date = df1.parse(input)
df2 = new SimpleDateFormat("MMM/dd/yyyy")
println 'Date was ' + df2.format(date)
```

```
Thu Jun 28 10:10:34 EST 2007
Fri Jun 29 10:10:35 EST 2007
Tue Jul 17 11:10:35 EST 2007
Date was Jun/03/1998
```

Lists, Maps, Ranges

- **Lists**
  - Special syntax for list literals
  - Additional common methods (operator overloading)

```
def list = [3, new Date(), 'Jan']
assert list + list == list * 2
```

- **Maps**
  - Special syntax for map literals
  - Additional common methods

```
def map  = [a: 1, b: 2]
assert map['a'] == 1 && map.b == 2
```

- **Ranges**
  - Special syntax for various kinds of ranges

```
def letters = 'a'..'z'
def numbers = 0..<10
```

Lists

```
assert [1,2,3,4]              == (1..4)
assert [1,2,3] + [1]          == [1,2,3,1]
assert [1,2,3] << 1           == [1,2,3,1]
assert [1,2,3,1] - [1]        == [2,3]
assert [1,2,3] * 2            == [1,2,3,1,2,3]
assert [1,[2,3]].flatten()    == [1,2,3]
assert [1,2,3].reverse()      == [3,2,1]
assert [1,2,3].disjoint([4,5,6])
assert [1,2,3].intersect([4,3,1]) == [3,1]
assert [1,2,3].collect{ it+3 }    == [4,5,6]
assert [1,2,3,1].unique().size()  == 3
assert [1,2,3,1].count(1)     == 2
assert [1,2,3,4].min()        == 1
assert [1,2,3,4].max()        == 4
assert [1,2,3,4].sum()        == 10
assert [4,2,1,3].sort()       == [1,2,3,4]
assert [4,2,1,3].findAll{ it%2 == 0 } == [4,2]
```

Maps…

```
def map = [a:1, 'b':2]
println map              // ["a":1, "b":2]
println map.a            // 1
println map['a']         // 1
println map.keySet()     // ["a", "b"]

map = [:]
// extend the map through assignment
map[1] = 'a'; map[2] = 'b'
map[true] = 'p'; map[false] = 'q'
map[null] = 'x'; map['null'] = 'z'
assert map == [ 1:'a', 2:'b', (true):'p',
                (false):'q',  (null):'x', 'null':'z' ]

def sb = new StringBuffer()
[1:'a', 2:'b', 3:'c'].each{ k, v-> sb << "$k:$v, " }
assert sb.toString() == '1:a, 2:b, 3:c, '

map = [1:'a', 2:'b', 3:'c']
def string = map.collect{ k, v -> "$k:$v" }.join(', ')
assert string == '1:a, 2:b, 3:c'
```

...Maps

```
assert [
    [ name: 'Clark',    city: 'London' ],
    [ name: 'Sharma',   city: 'London' ],
    [ name: 'Maradona', city: 'LA'     ],
    [ name: 'Zhang',    city: 'HK'     ],
    [ name: 'Ali',      city: 'HK'     ],
    [ name: 'Liu',      city: 'HK'     ]
].groupBy { it.city } == [
    London: [
        [ name: 'Clark',    city: 'London' ],
        [ name: 'Sharma',   city: 'London' ]
    ], LA: [
        [ name: 'Maradona', city: 'LA'     ]
    ], HK: [
        [ name: 'Zhang',    city: 'HK'     ],
        [ name: 'Ali',      city: 'HK'     ],
        [ name: 'Liu',      city: 'HK'     ]
    ]
]
```

Regular Expressions…

```
assert "Hello World!" =~ /Hello/        // Find operator
assert "Hello World!" ==~ /Hello\b.*/   // Match operator
def p = ~/Hello\b.*/                    // Pattern operator
assert p.class.name == 'java.util.regex.Pattern'

// replace matches with calculated values
assert "1.23".replaceAll(/./){ ch ->
    ch.next()
} == '2/34'

assert "1.23".replaceAll(/\d/){ num ->
    num.toInteger() + 1
} == '2.34'

assert "1.23".replaceAll(/\d+/){ num ->
    num.toInteger() + 1
} == '2.24'
```

…Regular Expressions

```
str = 'groovy.codehaus.org and www.aboutgroovy.com'

re = '''(?x)           # to enable whitespace and comments
      (                # capture the hostname in $1
        (?:            # these parens for grouping only
          (?! [-_] )   # neither underscore nor dash lookahead
          [\\w-] +     # hostname component
          \\.          # and the domain dot
        ) +            # now repeat whole thing a few times
        [A-Za-z]       # next must be a letter
        [\\w-] +       # now trailing domain part
      )                # end of $1 capture
'''

finder = str =~ re
out = str
(0..<finder.count).each{
    adr = finder[it][0]
    out = out.replaceAll(adr,
        "$adr [${InetAddress.getByName(adr).hostAddress}]")
}
println out
// => groovy.codehaus.org [63.246.7.187]
// and www.aboutgroovy.com [63.246.7.76]
```

Control Structures

```
if (1) // ...
if (object) // ...
if (list) // ...

for (item in iterable) { }

myMap.each { key, value ->
    println "$key : $value"
}

// if (condition)...
// else if (condition) ...
// else ...
// throw, catch, finally
// for, while
// eachWithIndex, eachLine, ...
```

```
switch (10) {
  case 0              : /* F */ ; break
  case 0..9           : // F
  case [8,9,11]       : // F
  case Float          : // F
  case {it % 3 == 0}  : // F
  case ~/../          : // T
  default             : // F
}

// implement
// boolean isCase(candidate)
```

GroovyBeans and GPath

```
class Dir {
    String name
    List   dirs
}

def root = new Dir (name: '/', dirs: [
    new Dir (name: 'a'),
    new Dir (name: 'b')
])

assert root.dirs[0].name == 'a'
assert root.dirs.name == ['a', 'b']
assert root.dirs.name*.size() == [1, 1]

def expected = ['getName', 'setName', 'getDirs', 'setDirs']
def accessorMethods = Dir.methods.name.grep(~/(g|s)et.*/)
assert accessorMethods.intersect(expected) == expected

// find, findAll, grep, every, any, …
```

Static Imports

Works with Java 1.4
Slightly more powerful than Java equivalent

```
import static java.awt.Color.LIGHT_GRAY
import static Boolean.FALSE as F
import static Calendar.getInstance as now

println LIGHT_GRAY  // => java.awt.Color[r=192,g=192,b=192]
println !F          // => true
println now().time  // => Sun Apr 29 11:12:43 EST 2007

import static Integer.*
println "Integers are between $MIN_VALUE and $MAX_VALUE"
// => Integers are between -2147483648 and 2147483647

def toHexString(int val, boolean upperCase) {
    def hexval = upperCase ?
        toHexString(val).toUpperCase() : toHexString(val)
    return '0x' + hexval
}
println toHexString(15, true)   // => 0xF
println toHexString(15, false)  // => 0xf

import static Math.*
assert cos(2 * PI) == 1.0
```

Topics

- Introduction
- Language Basics
- ➢ **Closures**
- Builders
- Data Access
- Other Features
- Testing with Groovy
- Further Integration
- Grails
- More Information

Using Closures...

- **Traditional mainstream languages**
  - **Data can be stored in variables, passed around, combined in structured ways to form more complex data; code stays put where it is defined**
- **Languages supporting closures**
  - **Data *and code* can be stored in variables, passed around, combined in structured ways to form more complex algorithms and data**

```
doubleNum = { num -> num * 2 }
println doubleNum(3) // => 6

processThenPrint = { num, closure ->
    num = closure(num); println "num is $num"
}
processThenPrint(3, doubleNum)    // => num is 6
processThenPrint(10) { it / 2 }   // => num is 5
```

...Using Closures...

```
import static Math.*

piA = { 22 / 7 }
piB = { 333/106 }
piC = { 355/113 }
piD = { 0.6 * (3 + sqrt(5)) }
piE = { 22/17 + 37/47 + 88/83 }
piF = { sqrt(sqrt(2143/22)) }

howCloseToPI = { abs(it.value() - PI) }

algorithms = [piA:piA, piB:piB, piC:piC,
              piD:piD, piE:piE, piF:piF]

findBestPI(algorithms)

def findBestPI(map) {
    map.entrySet().sort(howCloseToPI).each { entry ->
        def diff = howCloseToPI(entry)
        println "Algorithm $entry.key differs by $diff"
    }
}
```

```
Algorithm piE differs by 1.0206946399193839E-11
Algorithm piF differs by 1.0070735356748628E-9
Algorithm piC differs by 2.668102068170697E-7
Algorithm piD differs by 4.813291008076703E-5
Algorithm piB differs by 8.321958979307098E-5
Algorithm piA differs by 0.001264489310206951
```

...Using Closures...

- **Used for many things in Groovy:**
  - Iterators
  - Callbacks
  - Higher-order functions
  - Specialized control structures
  - Dynamic method definition
  - Resource allocation
  - Threads
  - Continuations

```
new File('/x.txt').eachLine {
    println it
}
```

```
3.times { println 'Hi' }

[0, 1, 2].each { number ->
    println number
}

[0, 1, 2].each { println it}

def printit = { println it }
[0, 1, 2].each printit
```

```
def houston(Closure doit) {
    (10..1).each { count ->
        doit(count)
    }
}
houston { println it }
```

...Using Closures...

```
map = ['a': 1, 'b': 2]
map.each {key, value -> map[key] = value * 2}
assert map == ['a': 2, 'b': 4]

doubler = {key, value -> map[key] = value * 2}
map.each(doubler)
assert map == ['a': 4, 'b': 8]

def doubleMethod(entry) {
    map[entry.key] = entry.value * 2
}
doubler = this.&doubleMethod
map.each(doubler)
assert map == ['a': 8, 'b': 16]
```

...Using Closures...

```
assert [1, 2, 3].grep{ it < 3 } == [1, 2]
assert [1, 2, 3].any{ it % 2 == 0 }
assert [1, 2, 3].every{ it < 4 }
assert (1..9).collect{it}.join() == '123456789'
assert (1..4).collect{it * 2}.join() == '2468'
def add  = { x, y -> x + y }
def mult = { x, y -> x * y }
assert add(1, 3)  == 4
assert mult(1, 3) == 3
def min = { x, y -> [x, y].min() }
def max = { x, y -> [x, y].max() }
def triple = mult.curry(3); assert triple(2) == 6
def atLeastTen = max.curry(10)
assert atLeastTen(5)  == 10
assert atLeastTen(15) == 15
```

...Using Closures

```
def pairWise(list, Closure invoke) {
    if (list.size() < 2) return []
    def next = invoke(list[0], list[1])
    return [next] + pairWise(list[1..-1], invoke)
}

// using min, max, etc. From previous slide
assert pairWise(1..5, add)  == [3, 5, 7, 9]
assert pairWise(1..5, mult) == [2, 6, 12, 20]
assert pairWise(1..5, min)  == [1, 2, 3, 4]
assert pairWise(1..5, max)  == [2, 3, 4, 5]

assert 'cbaxabc' == ['a', 'b', 'c'].inject('x') {
    result, item -> item + result + item
}
```

Topics

- Introduction
- Language Basics
- Closures
- ➢ **Builders**
- Data Access
- Other Features
- Testing with Groovy
- Further Integration
- Grails
- More Information

Builder Pattern Inclusive

- Builder pattern from the GoF at the syntax-level
- Represents easily any **nested tree-structured data**

```
import groovy.xml.*
def page = new MarkupBuilder()
page.html {
  head { title 'Hello' }
  body {
    ul {
      for (count in 1..10) {
        li "world $count"
} } } }
```

NodeBuilder, DomBuilder, SwingBuilder, AntBuilder, …

- Create new builder
- Call pretended methods (html, head, ...)
- Arguments are Closures
- Builder code looks very declarative but is ordinary Groovy program code and can contain any kind of logic

SwingBuilder

```
import java.awt.FlowLayout
builder = new groovy.swing.SwingBuilder()
langs = ["Groovy", "Ruby", "Python", "Pnuts"]

gui = builder.frame(size: [290, 100],
          title: 'Swinging with Groovy!') {
    panel(layout: new FlowLayout()) {
        panel(layout: new FlowLayout()) {
            for (lang in langs) {
                checkBox(text: lang)
            }
        }
        button(text: 'Groovy Button', actionPerformed: {
            builder.optionPane(message: 'Indubitably Groovy!').
                createDialog(null, 'Zen Message').show()
        })
        button(text: 'Groovy Quit',
                actionPerformed: {System.exit(0)})
    }
}
gui.show()
```

Source: http://www.ibm.com/developerworks/java/library/j-pg04125/

AntBuilder

```groovy
def ant = new AntBuilder()

ant.echo("hello") // lets just call one task

// create a block of Ant using the builder pattern
ant.sequential {
    myDir = "target/AntTest/"
    mkdir(dir: myDir)
    copy(todir: myDir) {
        fileset(dir: "src/test") {
            include(name: "**/*.groovy")
        }
    }
    echo("done")
}

// now lets do some normal Groovy again
file = new File("target/test/AntTest.groovy")
assert file.exists()
```

Using AntLibs: Maven Ant Tasks & AntUnit…

```
def ant = new AntBuilder()

items = [[groupId:'jfree', artifactId:'jfreechart', version:'1.0.5'],
         [groupId:'jfree', artifactId:'jcommon', version:'1.0.9']]

def mvn = new AntLibHelper(ant:ant, namespace:'org.apache.maven.artifact.ant')
def antunit = new AntLibHelper(ant:ant, namespace:'org.apache.ant.antunit')

// download artifacts with Maven Ant tasks
mvn.dependencies(filesetId:'artifacts') { items.each{ dependency(it) } }
// print out what we downloaded
ant.fileScanner { fileset(refid:'artifacts') }.each { println it }

// use AntUnit to confirm expected files were downloaded
def prefix = System.properties.'user.home' + '/.m2/repository'
items.each { item ->
    def g = item.groupId
    def a = item.artifactId
    def v = item.version
    antunit.assertFileExists(file:"$prefix/$g/$a/$v/$a-${v}.jar")
}
```

…Using AntLibs: Maven Ant Tasks & AntUnit

```
...
class AntLibHelper {
    def namespace, ant
    Object invokeMethod(String name, Object params) {
        ant."antlib:$namespace:$name"(*params)
    }
}

/* =>
Downloading: jfree/jfreechart/1.0.5/jfreechart-1.0.5.pom
Transferring 1K
Downloading: jfree/jcommon/1.0.9/jcommon-1.0.9.pom
Transferring 1K
Downloading: jfree/jfreechart/1.0.5/jfreechart-1.0.5.jar
Transferring 1157K
Downloading: jfree/jcommon/1.0.9/jcommon-1.0.9.jar
Transferring 298K
C:\Users\Paul\.m2\repository\jfree\jcommon\1.0.9\jcommon-1.0.9.jar
C:\Users\Paul\.m2\repository\jfree\jfreechart\1.0.5\jfreechart-1.0.5.jar
*/
```

Topics

- Introduction
- Language Basics
- Closures
- Builders
- **Data Access**
  - **Objects**
  - **XML**
  - **Databases**
- Other Features
- Testing with Groovy
- Further Integration
- Grails
- More Information

Uniform Data Access

- **Data Access to Objects, XML, SQL Databases has some differences and some similarities**
  - **Differences in detail about how to set up structures**
  - **Similarities in how to process data within resulting structures**

Object Data Access…

```
import java.text.SimpleDateFormat

class Athlete {
    def firstname, lastname, gender, country, dateOfBirth
}

def asDate(dateStr) {
    new SimpleDateFormat("yyyy-MM-dd").parse(dateStr)
}

def athletes = [
    new Athlete(firstname: 'Paul', lastname: 'Tergat',
                dateOfBirth: '1969-06-17', gender: 'M', country: 'KEN'),
    new Athlete(firstname: 'Khalid', lastname: 'Khannouchi',
                dateOfBirth: '1971-12-22', gender: 'M', country: 'USA'),
    new Athlete(firstname: 'Sammy', lastname: 'Korir',
                dateOfBirth: '1971-12-12', gender: 'M', country: 'KEN'),
    new Athlete(firstname: 'Ronaldo', lastname: 'da Costa',
                dateOfBirth: '1970-06-07', gender: 'M', country: 'BRA'),
    new Athlete(firstname: 'Paula', lastname: 'Radcliffe',
                dateOfBirth: '1973-12-17', gender: 'F', country: 'GBR')
]
```

…Object Data Access

```
def bornSince70 = {
    asDate(it.dateOfBirth) > asDate('1970-1-1') }
def excludingKh = { it.lastname <= 'Kg' ||
                    it.lastname >= 'Ki' }
def byGenderDescThenByCountry = { a, b ->
    a.gender == b.gender ?
        a.country <=> b.country : b.gender <=> a.gender }

def someYoungsters = athletes.
    findAll(bornSince70).
    findAll(excludingKh).
    sort(byGenderDescThenByCountry)

someYoungsters.each {
    def name = "$it.firstname $it.lastname".padRight(25)
    println "$name ($it.gender) $it.country $it.dateOfBirth"
}
```

Xml Data Access…

```
import java.text.SimpleDateFormat
import com.thoughtworks.xstream.XStream

class Athlete {
    String firstname, lastname, gender, country, dateOfBirth
}

def asDate(dateStr) { new SimpleDateFormat("yyyy-MM-dd").parse(dateStr) }

def athleteList = [
    new Athlete(firstname: 'Paul', lastname: 'Tergat‘,
                dateOfBirth: '1969-06-17', gender: 'M', country: 'KEN'),
    new Athlete(firstname: 'Khalid', lastname: 'Khannouchi',
                dateOfBirth: '1971-12-22', gender: 'M', country: 'USA'),
    new Athlete(firstname: 'Sammy', lastname: 'Korir',
                dateOfBirth: '1971-12-12', gender: 'M', country: 'KEN'),
    new Athlete(firstname: 'Ronaldo', lastname: 'da Costa',
                dateOfBirth: '1970-06-07', gender: 'M', country: 'BRA'),
    new Athlete(firstname: 'Paula', lastname: 'Radcliffe',
                dateOfBirth: '1973-12-17', gender: 'F', country: 'GBR')
]

// create XML as input
def input = new XStream().toXML(athleteList)
```

...Xml Data Access

```
def athletes = new XmlSlurper().parseText(input).Athlete

def bornSince70 = {
    asDate(it.dateOfBirth.text()) > asDate('1970-1-1') }
def excludingKh = { it.lastname.text() <= 'Kg' ||
                    it.lastname.text() >= 'Ki' }
def byGenderDescThenByCountry = { a, b ->
    a.gender.text() == b.gender.text() ?
        a.country.text() <=> b.country.text() :
        b.gender.text() <=> a.gender.text() }

def someYoungsters = athletes.
    findAll(bornSince70).findAll(excludingKh).list().
    sort(byGenderDescThenByCountry)

someYoungsters.each {
    def name = "$it.firstname $it.lastname".padRight(25)
    println "$name ($it.gender) $it.country $it.dateOfBirth"
}
```

Sql Data Access…

```
import groovy.sql.Sql

dbHandle = null

def getDb() {
    if (dbHandle) return dbHandle
    def source = new org.hsqldb.jdbc.jdbcDataSource()
    source.database = 'jdbc:hsqldb:mem:GIA'
    source.user = 'sa'
    source.password = ''
    dbHandle = new Sql(source)
    return dbHandle
}

db.execute '''
    DROP   INDEX athleteIdx IF EXISTS;
    DROP   TABLE Athlete IF EXISTS;
    CREATE TABLE Athlete (
        athleteId   INTEGER GENERATED BY DEFAULT AS IDENTITY,
        firstname   VARCHAR(64),
        lastname    VARCHAR(64),
        country     VARCHAR(3),
        gender      CHAR(1),
        dateOfBirth DATE
    );
    CREATE INDEX athleteIdx ON Athlete (athleteId);
'''
```

…Sql Data Access…

```
def athleteList = [
    [firstname: 'Paul', lastname: 'Tergat',
     dateOfBirth: '1969-06-17', gender: 'M', country: 'KEN'],
    [firstname: 'Khalid', lastname: 'Khannouchi',
     dateOfBirth: '1971-12-22', gender: 'M', country: 'USA'],
    [firstname: 'Sammy', lastname: 'Korir',
     dateOfBirth: '1971-12-12', gender: 'M', country: 'KEN'],
    [firstname: 'Ronaldo', lastname: 'da Costa',
     dateOfBirth: '1970-06-07', gender: 'M', country: 'BRA'],
    [firstname: 'Paula', lastname: 'Radcliffe',
     dateOfBirth: '1973-12-17', gender: 'F', country: 'GBR']
]

def athletes = db.dataSet('Athlete')
athleteList.each {a -> athletes.add(a)}
```

…Sql Data Access

```
def bornSince70 = { it.dateOfBirth > '1970-1-1' }
def excludingKh = { it.lastname <= 'Kg' || it.lastname >= 'Ki' }

def someYoungsters = athletes.
    findAll(bornSince70).
    findAll(excludingKh).
    sort { it.gender }.reverse().  // * experimental
    sort { it.country }            // * experimental

println someYoungsters.sql + '\n' + someYoungsters.parameters

someYoungsters.each {
    def name = "$it.firstname $it.lastname".padRight(25)
    println "$name ($it.gender)   $it.country   $it.dateOfBirth"
}
/* =>
select * from Athlete where dateOfBirth > ? and (lastname <= ? or lastname
>= ?) order by gender DESC, country
["1970-1-1", "Kg", "Ki"]
Ronaldo da Costa          (M)   BRA   1970-06-07
Sammy Korir               (M)   KEN   1971-12-12
Paula Radcliffe           (F)   GBR   1973-12-17       */
```

More Details: XmlSlurper…

- **Features**
  - **One-line parsing**
  - **GPath Syntax**
  - **Efficient lazy evaluation**

```
static def CAR_RECORDS = '''
  <records>
    <car name='HSV Maloo' make='Holden' year='2006'>
      <country>Australia</country>
      <record type='speed'>Production Pickup Truck with speed of 271kph</record>
    </car>
    <car name='P50' make='Peel' year='1962'>
      <country>Isle of Man</country>
      <record type='size'>Smallest Street-Legal Car at 99cm wide and 59 kg in weight</record>
    </car>
    <car name='Royale' make='Bugatti' year='1931'>
      <country>France</country>
      <record type='price'>Most Valuable Car at $15 million</record>
    </car>
  </records>
  '''
```

…More Details: XmlSlurper

```
def records = new XmlSlurper().parseText(XmlExamples.CAR_RECORDS)
// 3 records in total
assert 3 == records.car.size()
// 10 nested nodes
assert 10 == records.depthFirst().collect{ it }.size()
// test properties of the first record
def firstRecord = records.car[0]
assert 'car' == firstRecord.name()
assert 'Holden' == firstRecord.@make.toString()
assert 'Australia' == firstRecord.country.text()
// 2 cars have an 'e' in the make
assert 2 == records.car.findAll{ it.@make.toString().contains('e') }.size()
// 2 cars have an 'e' in the make
assert 2 == records.car.findAll{ it.@make =~ '.*e.*' }.size()
// makes of cars that have an 's' followed by an 'a' in the country
assert ['Holden', 'Peel'] == records.car.findAll{ it.country =~ '.*s.*a.*' }.@make.collect{ it.toString
// types of records
assert ['speed', 'size', 'price'] == records.depthFirst().grep{ it.@type != '' }.'@type'*.toString()
assert ['speed', 'size', 'price'] == records.'**'.grep{ it.@type != '' }.'@type'*.toString()
// check parent() operator
def countryOne = records.car[1].country
assert 'Peel' == countryOne.parent().@make.toString()
assert 'Peel' == countryOne.'..'.@make.toString()
// names of cars with records sorted by year
def names = records.car.list().sort{ it.@year.toInteger() }.'@name'*.toString()
assert ['Royale', 'P50', 'HSV Maloo'] == names
```

More Details: Working with Databases

- **Using standard SQL statements**

```
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}"
}
```

- **Using DataSets**

```
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}" }
```

DataSets and Lazy Evaluation

```
athleteSet = db.dataSet('Athlete')
youngsters = athleteSet.findAll{ it.dateOfBirth > '1970-1-1'}
paula = youngsters.findAll{ it.firstname == 'Paula'}

println paula.sql
// =>
// select * from Athlete where dateOfBirth > ? and firstname = ?

println paula.parameters
// =>
// [1970-1-1, Paula]

paula.each { println it.lastname }    // database called here
// =>
// Radcliffe
```

Topics

- Introduction
- Language Basics
- Closures
- Builders
- Data Access
- ➢ **Other Features**
- Testing with Groovy
- Further Integration
- Grails
- More Information

JMX

```
import javax.management.remote.*
import javax.management.*
import javax.naming.Context

def urlRuntime = '/jndi/weblogic.management.mbeanservers.runtime'
def urlBase = 'service:jmx:t3://localhost:7001'

def serviceURL = new JMXServiceURL(urlBase + urlRuntime)
def h = new Hashtable()
h.put(Context.SECURITY_PRINCIPAL, 'weblogic')
h.put(Context.SECURITY_CREDENTIALS, 'weblogic')
h.put(JMXConnectorFactory.PROTOCOL_PROVIDER_PACKAGES,
        'weblogic.management.remote')
def server = JMXConnectorFactory.connect(serviceURL, h).mBeanServerConnection
def domainName = new ObjectName('com.bea:Name=RuntimeService,' +
        'Type=weblogic.management.mbeanservers.runtime.RuntimeServiceMBean')
def rtName = server.getAttribute(domainName, 'ServerRuntime')
def rt = new GroovyMBean(server, rtName)
println "Server: name=$rt.Name, state=$rt.State, version=$rt.WeblogicVersion"
def destFilter = Query.match(Query.attr('Type'), Query.value(
        'JMSDestinationRuntime'))
server.queryNames(new ObjectName('com.bea:*'), destFilter).each {name ->
    def jms = new GroovyMBean(server, name)
    println "JMS Destination: name=$jms.Name, type=$jms.DestinationType" +
            ", messages=$jms.MessagesReceivedCount"
}
```

ExpandoMetaClass…

```
String.metaClass.swapCase = {->
    def sb = new StringBuffer()
    delegate.each {
        sb << (Character.isUpperCase(it as char) ?
            Character.toLowerCase(it as char) :
            Character.toUpperCase(it as char))
    }
    sb.toString()
}
```

```
List.metaClass.sizeDoubled = {-> delegate.size() * 2 }

LinkedList list = []

list << 1
list << 2

assert 4 == list.sizeDoubled()
```

…ExpandoMetaClass

```
class Person {
    String name
}

class MortgageLender {
   def borrowMoney() {
      "buy house"
   }
}

def lender = new MortgageLender()

Person.metaClass.buyHouse = lender.&borrowMoney

def p = new Person()

assert "buy house" == p.buyHouse()
```

Constraint Programming

```
// require(url:'http://www.alice.unibo.it/tuProlog/', jar:'tuprolog.jar', version:'2.1')
import alice.tuprolog.*

/** Pretty Printing */
def pprint(soln) {
    println soln.isSuccess() ?
    "$soln.query = $soln.solution" :
    'no solution found'
}

/** Prolog clauses */
def getTheory() {
new Theory("""
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

female(pam).
male(tom).
male(bob).
female(liz).
female(pat).
female(ann).
male(jim).

offspring(X,Y) :- parent(Y,X).
...
mother(X,Y) :-
    parent(X,Y), female(X).
father(X,Y) :-
    parent(X,Y), male(X).

grandparent(X,Z) :-
    parent(X,Y),
    parent(Y,Z).
grandmother(X,Y) :-
    grandparent(X,Y),
    female(X).
grandfather(X,Y) :-
    grandparent(X,Y),
    male(X).

sister(X,Y) :-
    parent(Z,X),
    parent(Z,Y),
    female(X).
brother(X,Y) :-
    parent(Z,X),
    parent(Z,Y),
    male(X).

ancestor(X,Y) :- parent(X,Y).
ancestor(X,Z) :-
    parent(X,Y),
    ancestor(Y,Z).
""")
}
...
def engine = new Prolog()
engine.theory = theory
pprint engine.solve('ancestor(tom,X).')
while(engine.hasOpenAlternatives()) {
    pprint engine.solveNext()
}
```

```
ancestor(tom,X) = ancestor(tom,bob)
ancestor(tom,X) = ancestor(tom,liz)
ancestor(tom,X) = ancestor(tom,ann)
ancestor(tom,X) = ancestor(tom,pat)
ancestor(tom,X) = ancestor(tom,jim)
no solution found
```

Functional Programming

```
def fac(n) { n == 0 ? 1 : n * fac(n - 1) }
assert 24 == fac(4)

// now define and use infinite streams
def integers(n) { cons(n, { integers(n+1) }) }
def naturalnumbers = integers(1)
assert '1 2 3 4 5 6 7 8 9 10' ==
    naturalnumbers.take(10).join(' ')
def evennumbers =
    naturalnumbers.filter{ it % 2 == 0 }
assert '2 4 6 8 10 12 14 16 18 20' ==
    evennumbers.take(10).join(' ')
```

Using XStream - Vanilla

```
// require(groupId:'com.thoughtworks.xstream', artifactId:'xstream',
           version:'1.2.2')
// require(groupId:'xpp3', artifactId:'xpp3_min', version:'1.1.3.4.O')

import com.thoughtworks.xstream.*

class Staff {
    String firstname, lastname, position
}

def stream = new XStream()
def msg = new Staff(firstname:'John',
                    lastname:'Connor',
                    position:'Resistance Leader')
println stream.toXML(msg)
```

```
<Staff>
  <firstname class="string">John</firstname>
  <lastname class="string">Connor</lastname>
  <position class="string">Resistance Leader</position>
</Staff>
```

Using XStream - Annotations

```
import com.thoughtworks.xstream.*
import com.thoughtworks.xstream.annotations.*

@XStreamAlias("person")
class Associate {
    @XStreamAsAttribute
    @XStreamAlias('first-name')
    private String firstname

    @XStreamAlias('surname')
    private String lastname

    @XStreamOmitField
    private String position
}

msg = new Associate(firstname:'Sarah',
                    lastname:'Connor',
                    position:'Protector')
Annotations.configureAliases(stream, Associate)
println stream.toXML(msg)
```

```
<person first-name="Sarah">
  <surname>Connor</surname>
</person>
```

Topics

- Introduction
- Language Basics
- Closures
- Builders
- Data Access
- Other Features
- **Testing with Groovy**
  - **Unit Testing**
  - **Mocks**
  - **Acceptance Testing**
- Further Integration
- Grails
- More Information

Built-in JUnit

- **Groovy distribution includes junit (3.8.2)**
- **Automatically invokes text runner**
- **Has some useful extensions**

```
..
Time: 0.092
OK (2 tests)
```

```groovy
class GroovyMultiplierJUnit3Test extends GroovyTestCase {
    void testPositives() {
        def testee = new GroovyMultiplier()
        assertEquals "+ve multiplier error", 9, testee.triple(3)
        assertEquals "+ve multiplier error", 12, testee.triple(4)
    }
    void testNegatives() {
        def testee = new GroovyMultiplier()
        assertEquals "-ve multiplier error", -12, testee.triple(-4)
    }
}
```

JUnit 4.4

```
import org.junit.Test
import org.junit.runner.JUnitCore
import static org.junit.Assert.assertEquals

class ArithmeticTest {
    @Test
    void additionIsWorking() {
        assertEquals 4, 2+2
    }

    @Test(expected=ArithmeticException)
    void divideByZero() {
        println 1/0
    }
}

JUnitCore.main(ArithmeticTest.name)
```

JUnit 4.4 Parameterized Tests

```
import org.junit.Test
import org.junit.Before
import org.junit.runner.RunWith
import org.junit.runner.JUnitCore
import org.junit.runners.Parameterized
import org.junit.runners.Parameterized.*

@RunWith(Parameterized)
class GroovyMultiplierJUnit4Test {
    def testee
    def param
    def expected

    @Parameters static data() {
        return (2..4).collect{
            [it, it * 3] as Integer[]
        }
    }

    GroovyMultiplierJUnit4Test(a, b) {
        param = a
        expected = b
    }

    @Before void setUp() {
        testee = new GroovyMultiplier()
    }

    @Test void positivesFixed() {
        assert testee.triple(1) ==3
    }

    @Test void positivesParameterized() {
        assert testee.triple(param) == expected
    }

    @Test void negativesParameterized() {
        assert testee.triple(-param) == -expected
    }
}

JUnitCore.main('GroovyMultiplierJUnit4Test')
```

JUnit 4.4 Theories …

```
import org.junit.runner.*
import org.junit.experimental.theories.*
import static org.junit.Assume.assumeTrue

@RunWith(Theories)
class LanguageTheoryTest {
  @DataPoint public static String java = 'Java'
  @DataPoint public static String ruby = 'JRuby'
  @DataPoint public static String python = 'Jython'
  @DataPoint public static String javascript = 'Rhino'
  @DataPoint public static String groovy = 'Groovy'
  @DataPoint public static String scala = 'Scala'
  @DataPoint public static String csharp = 'C#'

  def jvmLanguages = [java, ruby, python, groovy, scala, javascript]

  def teamSkills = [
        tom:   [java, groovy, ruby],
        dick:  [csharp, scala, java, python],
        harry: [javascript, groovy, java]
  ]
...
```

… JUnit 4.4 Theories

```
...

    @Theory void everyoneKnowsJava() {
        teamSkills.each { developer, skills ->
            assert java in skills
        }
    }

    @Theory void teamKnowsEachJvmLanguage(String language) {
        assumeTrue language in jvmLanguages
        assert teamSkills.any { mapEntry ->
            language in mapEntry.value
        }
    }
}

JUnitCore.main(LanguageTheoryTest.name)
```

Popper

```
@RunWith(Theories)
class PopperBetweenTest extends GroovyTheoryContainer {
    private int test, total // for explanatory purposes only

    @Theory void multiplyIsInverseOfDivide(
            @Between(first = -4, last = 2) int amount,
            @Between(first = -2, last = 5) int m
    ) {
        total++
        assume m != 0
        assert new Dollar(amount).times(m).divideBy(m).amount == amount
        test++
    }

    @After void dumpLog() {
        println "$test tests performed out of $total combinations"
    }
}

JUnitCore.main('PopperBetweenTest')
```

```
JUnit version 4.3.1
.49 tests performed out of 56 combinations

Time: 0.234

OK (1 test)
```

Instinct

```
import com.googlecode.instinct.marker.annotate.BeforeSpecification as initially
import com.googlecode.instinct.marker.annotate.Specification as spec
import static com.googlecode.instinct.runner.TextContextRunner.runContexts as check_specs_for

class a_default_storer {
    def storer

    @initially void create_new_storer() {
        storer = new Storer()
    }

    private check_persist_and_reverse(value, reverseValue) {
        storer.put(value)
        assert value == storer.get()
        assert reverseValue == storer.reverse
    }

    @spec def should_reverse_numbers() {
        check_persist_and_reverse 123.456, -123.456
    }

    @spec def should_reverse_strings() {
        check_persist_and_reverse 'hello', 'olleh'
    }

    @spec def should_reverse_lists() {
        check_persist_and_reverse([1, 3, 5], [5, 3, 1])
    }
}

check_specs_for a_default_storer
```

```
a_default_storer
- should_reverse_lists
- should_reverse_strings
- should_reverse_numbers
```

Built-in Mocks for Groovy…

- **Handle statics, explicit constructors, etc.**

```
import groovy.mock.interceptor.MockFor

def mocker = new MockFor(Collaborator.class)  // create the Mock support
mocker.demand.one(1..2) { 1 }      // demand the 'one' method one or two times, returning 1
mocker.demand.two()     { 2 }      // demand the 'two' method exactly once, returning 2
mocker.use {                       // start using the Mock
    def caller = new Caller()          // caller will call Collaborator
    assertEquals 1, caller.collaborateOne()   // will call Collaborator.one
    assertEquals 1, caller.collaborateOne()   // will call Collaborator.one
    assertEquals 2, caller.collaborateTwo()   // will call Collaborator.two
}                                  // implicit verify for strict expectation here
```

…Built-in Mocks for Groovy

```
import groovy.mock.interceptor.MockFor
class MockingTest extends GroovyTestCase {
    def mock1 = new MockFor(Collaborator1)
    def mock2 = new MockFor(Collaborator2)
    def mock3 = new MockFor(Collaborator3)
    private static final Closure DONTCARE = null
    private static final Closure PASS = {}
    private static final Closure FAIL = {
        throw new RuntimeException() }
    void testSuccess() {
        check(PASS, PASS, DONTCARE)
    }
    void testCollaborator1Fails() {
        check(FAIL, DONTCARE, PASS)
    }
    void testCollaborator2Fails() {
        check(PASS, FAIL, PASS)
    }
    private check(expected1, expected2, expected3){
        if (expected1) mock1.demand.method1(expected1)
        if (expected2) mock2.demand.method2(expected2)
        if (expected3) mock3.demand.method3(expected3)
        mock1.use { mock2.use { mock3.use {
            new Mocking().method()
        }}}
    }
}
```

```
class Mocking {
    def method() {
        try {
            new Collaborator1().method1()
            new Collaborator2().method2()
        } catch (Exception e) {
            new Collaborator3().method3()
        }
    }
}
class Collaborator1 {}
class Collaborator2 {}
class Collaborator3 {}
```

JMock 2

```
import org.jmock.integration.junit4.JMock
import org.jmock.Mockery
import org.junit.Test
import org.junit.Before
import org.junit.runner.RunWith
import org.junit.runner.JUnitCore

@RunWith(JMock)
class JMock2Test {
    Mockery context = new JUnit4GroovyMockery()
    def mockReverser, storer

    @Before void setUp() throws Exception {
        mockReverser = context.mock(Reverser.class)
        storer = new JavaStorer(mockReverser)
    }

    @Test void testStorage() {
        expectReverse(123.456, -123.456)
        expectReverse('hello', 'olleh')
        checkReverse(123.456, -123.456)
        checkReverse('hello', 'olleh')
    }

    def expectReverse(input, output) {
        context.checking{
            one(mockReverser).reverse(input); will(returnValue(output))
        }
    }
...
```

```
...
    def checkReverse(value, reverseValue) {
        storer.put(value)
        assert value == storer.get()
        assert reverseValue == storer.getReverse()
    }
}

JUnitCore.main('JMock2Test')
```

```
class Storer {
    ...
    def getReverse() {
        return reverser.reverse(stored)
    }
}
```

WebTest testing Web Sites

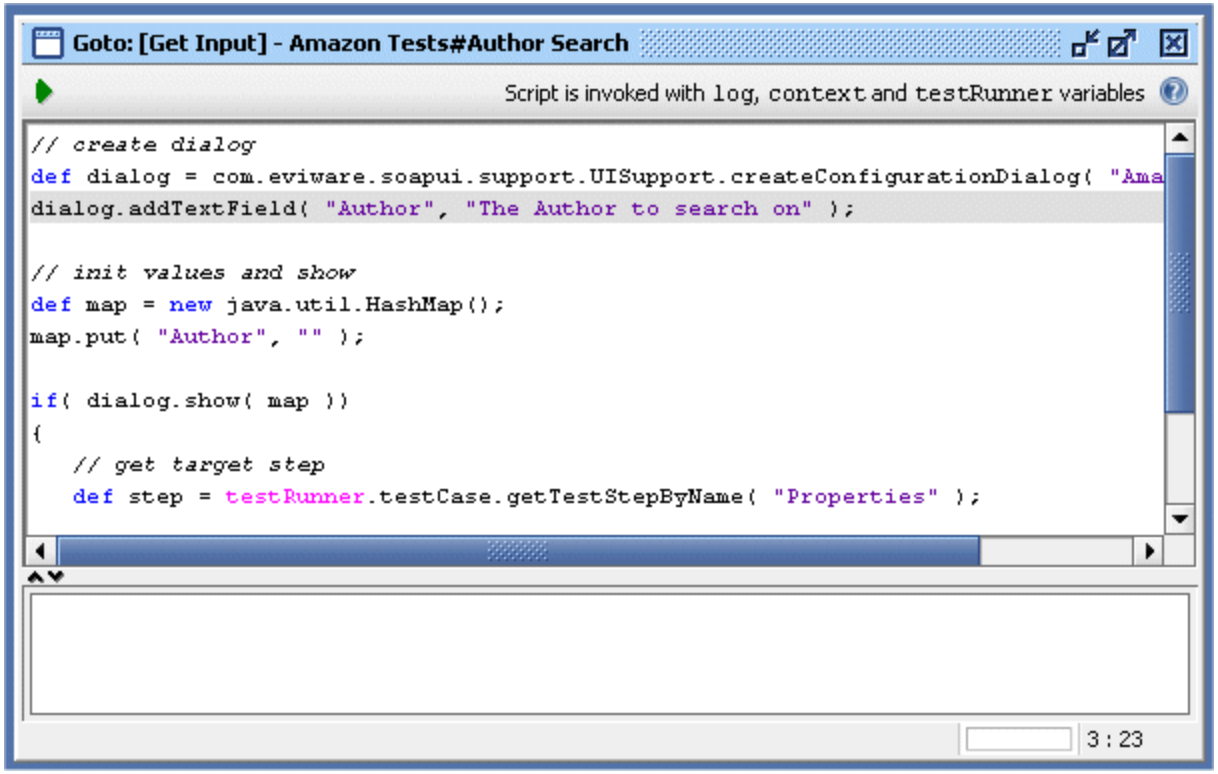
```groovy
def ant = new AntBuilder()

def webtest_home = System.properties.'webtest.home'

ant.taskdef(resource:'webtest.taskdef'){
    classpath(){
        pathelement(location:"$webtest_home/lib")
        fileset(dir:"$webtest_home/lib", includes:"**/*.jar")
    }
}

def config_map = [:]
['protocol','host','port','basepath','resultfile',
'resultpath', 'summary', 'saveresponse','defaultpropertytype'].each{
    config_map[it] = System.properties['webtest.'+it]
}

ant.testSpec(name:'groovy: Test Groovy Scripting at creation time'){
    config(config_map)
    steps(){
        invoke(url:'linkpage.html')
        for (i in 1..10){
            verifyText(description:"verify number ${i} is on pages", text:"${i}")
        }
    }
}
```

WebTest testing Emails

```
def ant = new AntBuilder()

def webtest_home = System.properties.'webtest.home'

ant.taskdef(resource:'webtest.taskdef'){
    classpath(){
       pathelement(location:"$webtest_home/lib")
       fileset(dir:"$webtest_home/lib", includes:"**/*.jar")
    }
}

ant.testSpec(name:'Email Test'){
    steps {
        emailSetConfig(server:'localhost', password:'password',
            username:'devteam@mycompany.org', type:'pop3')
        emailStoreMessageId(subject:'/Build notification/',
            property:'msg')
        emailStoreHeader(property:'subject',
            messageId:'#{msg}', headerName:'Subject')
        groovy('''def subject = step.webtestProperties.subject
            assert subject.startsWith('Build notification')''')
        emailMessageContentFilter(messageId:'#{msg}')
        verifyText(text:'Failed build')
    }
}
```

SoapUI

- **Tool for testing Web Services has a built-in Groovy editor for custom steps**



Topics

- Introduction
- Language Basics
- Closures
- Builders
- Data Access
- Other Features
- Testing with Groovy
- ➢ **Further Integration**
- Grails
- More Information

Integration With Existing Native Apps

- **Scriptom allows you to script any ActiveX or COM Windows component from within your Groovy scripts**

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

def outlook = new ActiveXProxy("Outlook.Application")
def message = outlook.CreateItem(0)
def emails = "galleon@codehaus.org;glaforge@codehaus.org"
def rec = message.Recipients.add(emails)
rec.Type = 1
message.Display(true)
```

Integration With Existing Services

- **WS or XML/RPC allow seamless connection to existing services …**

```
import groovy.net.soap.SoapClient
proxy = new SoapClient(
   "http://www.webservicex.net/CurrencyConvertor.asmx?WSDL")
rate = proxy.ConversionRate("USD", "EUR")
println rate
```

  - **No need to generate stubs,**
  - **Complex types are supported**

SOAP Client and Server

```
class MathService {
    double add(double a, double b) {
        a + b
    }
    double square(double c) {
        c * c
    }
}
```

```
import groovy.net.soap.SoapServer

def server = new SoapServer('localhost', 6789)
server.setNode('MathService')
server.start()
```

```
import groovy.net.soap.SoapClient

def math = new SoapClient('http://localhost:6789/MathServiceInterface?wsdl')
assert math.add(1.0, 2.0) == 3.0
assert math.square(3.0) == 9.0
```

Integration with Spring...

```
// traditional approach using a beans xml file
import org.springframework.context.support.ClassPathXmlApplicationContext

def ctx = new ClassPathXmlApplicationContext('calcbeans.xml')
def calc = ctx.getBean('calcBean')
println calc.doAdd(3, 4) // => 7


// using BeanBuilder
def bb = new grails.spring.BeanBuilder()
bb.beans {
    adder(AdderImpl)
    calcBean(CalcImpl2) { adder = adder }
}
def ctx = bb.createApplicationContext()
def calc = ctx.getBean('calcBean')
println calc.doAdd(3, 4) // => 7
```

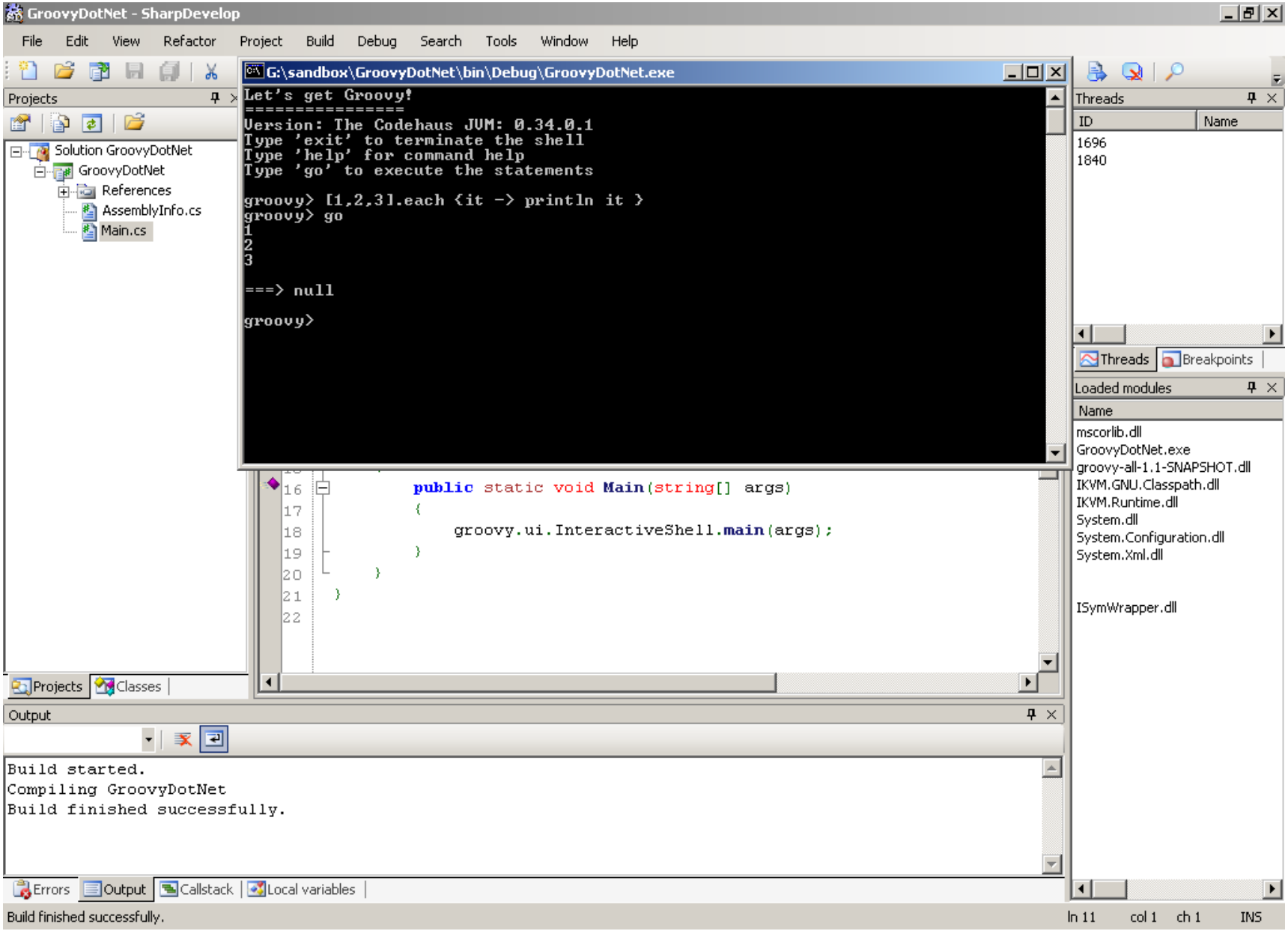
...Integration with Spring

```
// using annotations
import org.springframework.stereotype.Component

@Component class AdderImpl {
    def add(x, y) { x + y }
}

import org.springframework.beans.factory.annotation.Autowired
import org.springframework.stereotype.Component

@Component class CalcImpl3 {
    @Autowired private AdderImpl adder
    def doAdd(x, y) { adder.add(x, y) }
}

import org.springframework.context.support.GenericApplicationContext
import org.springframework.context.annotation.ClassPathBeanDefinitionScanner

def ctx = new GenericApplicationContext()
new ClassPathBeanDefinitionScanner(ctx).scan('') // scan root package for components
ctx.refresh()
def calc = ctx.getBean('calcImpl3')
println calc.doAdd(3, 4) // => 7
```

Groovy.Net...

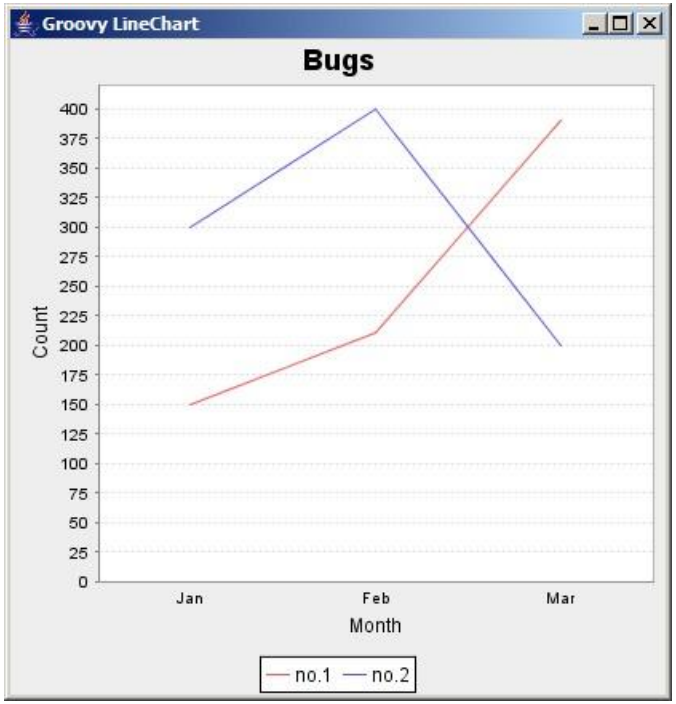
```
// To create dlls:
// ikvmc -target:library groovy-all-1.1-SNAPSHOT.jar

// Invoking groovy shell via GroovyDotNet.exe
using System;
using System.Collections.Generic;

namespace GroovyDotNet {
  class MainClass {
    public static void Main(string[] args) {
      groovy.ui.InteractiveShell.main(args);
    }
  }
}

// you can invoke normal groovy
[1, 2, 3].each { println it }

// you can also invoke a .NET method
cli.System.Console.WriteLine('hello world {0}', 'from Groovy')
```

...Groovy.Net

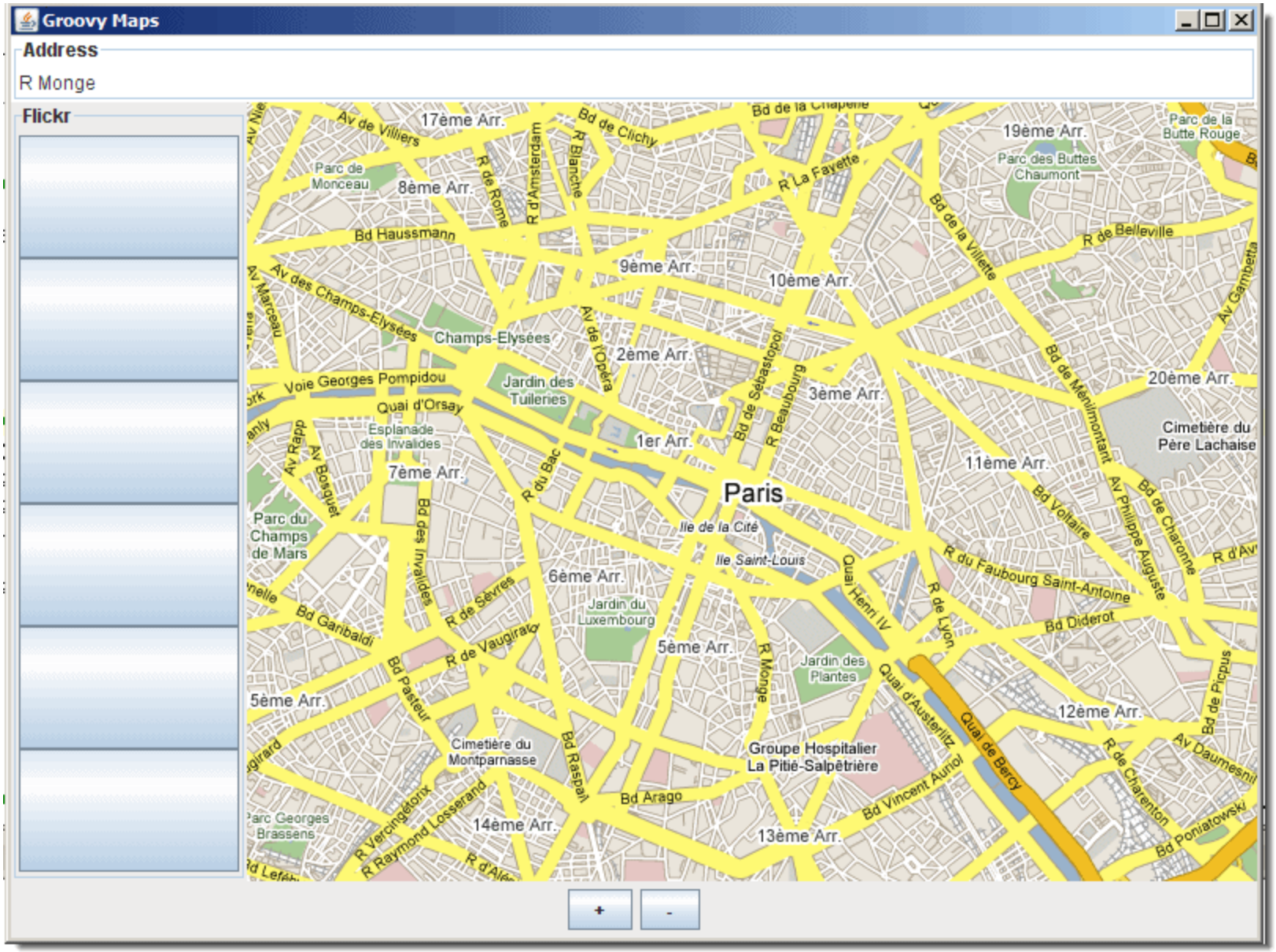


JFreeChart

```
import org.jfree.chart.ChartFactory
import org.jfree.data.category.DefaultCategoryDataset
import org.jfree.chart.plot.PlotOrientation as Orientation
import groovy.swing.SwingBuilder
import javax.swing.WindowConstants as WC

def dataset = new DefaultCategoryDataset()
dataset.addValue 150, "no.1", "Jan"
dataset.addValue 210, "no.1", "Feb"
dataset.addValue 390, "no.1", "Mar"
dataset.addValue 300, "no.2", "Jan"
dataset.addValue 400, "no.2", "Feb"
dataset.addValue 200, "no.2", "Mar"

def labels = ["Bugs", "Month", "Count"]
def options = [true, true, true]
def chart = ChartFactory.createLineChart(*labels, dataset,
            Orientation.VERTICAL, *options)
def swing = new SwingBuilder()
def frame = swing.frame(title:'Groovy LineChart',
      defaultCloseOperation:WC.EXIT_ON_CLOSE) {
   panel(id:'canvas') { rigidArea(width:400, height:400) }
}
frame.pack()
frame.show()
chart.draw(swing.canvas.graphics, swing.canvas.bounds)
```



Mashups



Topics

- Introduction
- Language Basics
- Closures
- Builders
- Data Access
- Other Features
- Testing with Groovy
- Further Integration
- ➢ **Grails**
- More Information

Wikipedia says:

- **Grails has been developed with a number of goals in mind:**
  - **Provide a high-productivity web framework for the Java platform**
  - **Offer a consistent framework that takes away confusion and is easy to learn**
  - **Offer documentation for those parts of the framework that matter for its users**
  - **Provide what users expect in areas that are often complex and inconsistent:**
    - ***Powerful and consistent persistence framework***
    - ***Powerful and easy to use view templates using GSP (Groovy Server Pages)***
    - ***Dynamic tag libraries to easily create web page components***
    - ***Good Ajax support that is easy to extend and customize***
  - **Provide sample applications that demonstrate the power of the framework**
  - **Provide a complete development mode, including web server and automatic reload of resources**

Goals

- **Grails has three properties that increase developer productivity significantly when compared to traditional Java web frameworks:**
  - **No XML configuration**
  - **Ready-to-use development environment**
  - **Functionality available through mixins**

Web framework

- **Controllers**

```
class BookController {
   def list = {
      [ books: Book.findAll() ]
   }
}
```

```
grails create-controller
```

- **Views**

```
<html>
  <head>
    <title>Our books</title>
  </head>
  <body>
    <ul>
      <g:each it="books">
        <li>${it.title} (${it.author.name})</li>
      </g:each>
    </ul>
  </body>
</html>
```

Persistence

- **Model**

```
grails create-domain-class
```

```
class Book {
    Long id
    Long version

    String title
    Person author
}
```

- **Methods**

```
def book = new Book(title:"The Da Vinci Code", author:Author.findByName("Dan Brown"))
book.save()
```

```
def book = Book.find("from Book b where b.title = ?", [ 'The Da Vinci Code' ])
```

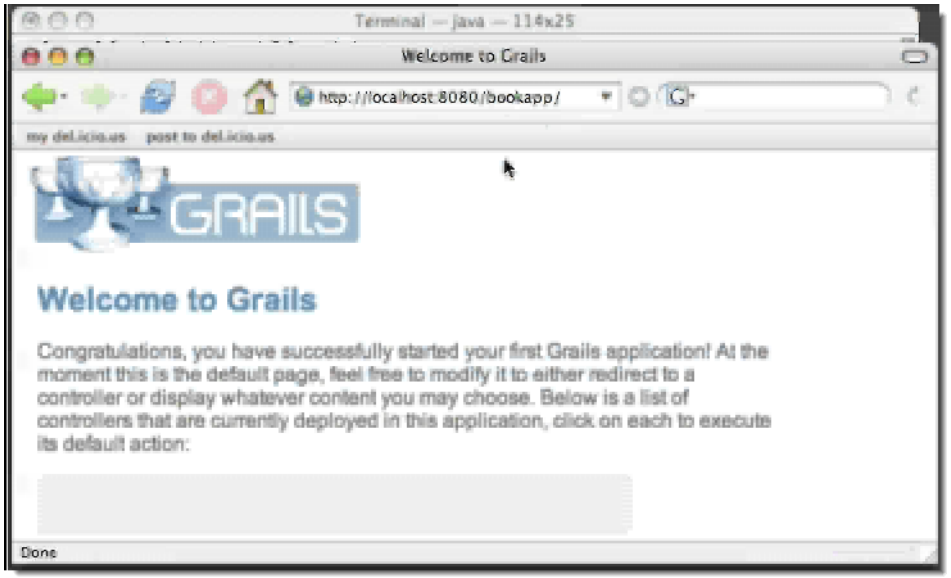
```
def books = Book.findAll()
```

```
def book = Book.findByTitleLike("%Da Vinci%")
```

```
def book = Book.findWhere(title:"The Da Vinci Code")
```

Project Structure

```
%PROJECT_HOME%
   + grails-app
      + conf            ---> location of configuration artifacts like data sources
      + controllers     ---> location of controller artifacts
      + domain          ---> location of domain classes
      + i18n            ---> location of message bundles for i18n
      + services        ---> location of services
      + taglib          ---> location of tag libraries
      + util            ---> location of special utility classes (e.g., codecs, etc.)
      + views           ---> location of views
         + layouts      ---> location of layouts
   + hibernate          ---> optional hibernate config
   + lib
   + spring             ---> optional spring config
   + src
      + groovy          ---> optional; location for Groovy source files
                             (of types other than those in grails-app/*)
      + java            ---> optional; location for Java source files
   + war
      + WEB-INF
```



Topics

- Introduction
- Language Basics
- Closures
- Builders
- Data Access
- Other Features
- Testing with Groovy
- Further Integration
- Grails

➢More Information

More Information: on the web

- **Web sites**
  - http://groovy.codehaus.org
  - http://grails.codehaus.org
  - http://pleac.sourceforge.net/pleac_groovy (many examples)
  - http://www.asert.com.au/training/java/GV110.htm (workshop)
- **Mailing list for users**
  - user@groovy.codehaus.org
- **Information portals**
  - http://www.aboutgroovy.org
  - http://www.groovyblogs.org
- **Documentation (600+ pages)**
  - Getting Started Guide, User Guide, Developer Guide, Testing Guide, Cookbook Examples, Advanced Usage Guide
- **Books**
  - Several to choose from ...

More Information: Groovy in Action