

Language dynamism, scripting and functional programming



Václav Pech

NPRG014 2016/2017

<http://jroller.com/vaclav>

<http://www.vaclavpech.eu>

@vaclav_pech

Today's agenda

- Groovy syntax and interoperability
- Language dynamism
- Scripting
- Functional programming

Aug 2016	Aug 2015	Change	Programming Language	Ratings	Change
1	1		Java	19.010%	-0.26%
2	2		C	11.303%	-3.43%
3	3		C++	5.800%	-1.94%
4	4		C#	4.907%	+0.07%
5	5		Python	4.404%	+0.34%
6	7	⬆	PHP	3.173%	+0.44%
7	9	⬆	JavaScript	2.705%	+0.54%
8	8		Visual Basic .NET	2.518%	-0.19%
9	10	⬆	Perl	2.511%	+0.39%
10	12	⬆	Assembly language	2.364%	+0.60%
11	14	⬆	Delphi/Object Pascal	2.278%	+0.87%
12	13	⬆	Ruby	2.278%	+0.86%
13	11	⬇	Visual Basic	2.046%	+0.26%
14	17	⬆	Swift	1.983%	+0.80%
15	6	⬇	Objective-C	1.884%	-1.31%
16	37	⬆	Groovy	1.637%	+1.27%
17	20	⬆	R	1.605%	+0.60%
18	15	⬇	MATLAB	1.538%	+0.31%
19	19		PL/SQL	1.349%	+0.21%
20	95	⬆	Go	1.270%	+1.19%

Groovy



A JVM programming language

- Dynamic
- Dynamically-typed
- Scripting
- Object-oriented
- Building on Java syntax

★ Groovy



Ecosystem

Grails

Gradle

Spock

GPars

Ratpack

Griffon

SDKMAN!

Groovy in the wild



canoo

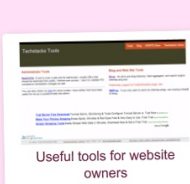
> your provider for business web solutions >

Success Stories and Sites Using Grails

Sites using Grails

A list of sites known to be grails-based:

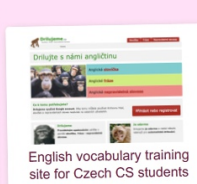
- <http://www.findroomrent.com> - Provides verified listings of rooms for rent in big cities in the US. Uses Twilio for sending text messages and GeoIP module to serve region-related information.
- <http://genxbio.info> - Genxbio introduces biggest biotech product range that have been tested for accuracy, quality, reliable results and consistent performance.
- <http://www.nala.com.cn> - The most famous cosmetics shopping mall in china.
- <http://www.setupmanual.com> - Generate custom PDF manuals for setting up email accounts on various platforms. Built using Grails, Birt and Drools.
- <https://lsp.lexmark.com/lexmark> - Enterprise Cloud Print Release platform allowing mobile, web, driver and email print release.
- <http://www.salesgoals.com> - An online CRM tool with an integrated iPhone application.
- <http://www.welonik.pl/> - Directory of wedding photographers in Poland.
- <http://www.juvamo.de> - Web based kanban tool for personal or professional project management.
- <http://www.chatnearme.com> - A location based real-time chat website, mobile version located @ same url.
- <http://www.nissanusa.com/leaf-electric-car/index> - North American, Nissan Leaf website.
- <http://unsere-regionalen-spezialitaeten.de> - a German portal for collecting regional specialties.
- <http://www.servermeile.com> - Here you can configure and buy your Server.
- <http://manatalks.com> - Magic The Gathering online store and community integrated with WordPress and Magento.
- <http://www.kettlerusa.com> - a retail site for toys, patio furniture, fitness and tennis.
- <http://www.simbo.com.br> - A Real Estate SaaS product to agents and brokers with cloud computing infrastructure and multi-tenant architecture.
- <http://www.bkool.com> - Specialized social network for the sports practice and outdoor. Integrates a 100% Grails web site and backend with a video gallery.
- <http://www.secretescapes.com> - Secret Escapes is a private member site for travel agents in the UK.
- <http://pigink.com> - PigInk - Colour registry and information site
- <http://www.landingSMS.com> - Using services of landingSMS you can integrate your mobile phone numbers from your customers and offer them discounts or different information via SMS. Move easy and without programming knowledge into mobile marketing.



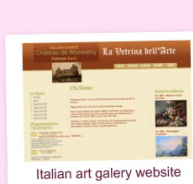
Useful tools for website owners



TwitWinner, comparing keyword popularity on Twitter



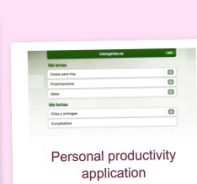
English vocabulary training site for Czech CS students



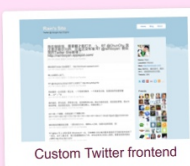
Italian art gallery website



Devovx conference schedule application



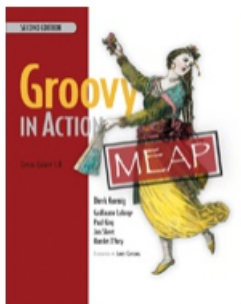
Personal productivity application



Custom Twitter frontend



Renza Vermeulen cake designer website



The 7 usage patterns

- Super Glue
- Liquid Heart
- Keyhole Surgery
- Smart Configuration
- Unlimited Openness
- House-Elf Scripts
- Prototype



Examples in Groovy

canoo

Part 1

Groovy syntax and interoperability

Interoperability

Groovy and Java can **implement**, **extend**, **refer** and **call** each other at will.

groovyc supports mixed mode

Groovy sources compile into *.class* files

IDEs provide cross-reference support

Java

```
public class Person {  
    private final String name;  
    public Person(String name) {  
        this.name = name;  
    }  
    public String getName() {  
        return name;  
    }  
}
```

Groovy

```
public class Person {  
    private final String name;  
    public Person(String name) {  
        this.name = name;  
    }  
    public String getName() {  
        return name;  
    }  
}
```

Groovy

```
public class Person {  
    private final String name  
    public Person(String name) {  
        this.name = name  
    }  
    public String getName() {  
        return name  
    }  
}
```

Groovy

```
public class Person {  
    private final String name  
    public Person(String name) {  
        this.name = name  
    }  
    public String getName() {  
        return name  
    }  
}
```

Groovy

```
public class Person {  
    private final String name  
    public Person(String name) {  
        this.name = name  
    }  
    public String getName() {  
        name  
    }  
}
```

Groovy

```
public class Person {  
    private final String name  
    public Person(String name) {  
        this.name = name  
    }  
    public String getName() {  
        name  
    }  
}
```

Groovy

```
class Person {  
    private final String name  
    Person(String name) {  
        this.name = name  
    }  
    public String getName() {  
        name  
    }  
}
```


Groovy

```
class Person {  
    private final String name  
    Person(String name) {  
        this.name = name  
    }  
    public String getName() {  
        name  
    }  
}
```

Groovy

```
class Person {  
    final String name  
    Person(String name) {  
        this.name = name  
    }  
}
```

Groovy

```
class Person {  
    final String name  
    Person(String name) {  
        this.name = name  
    }  
}
```

Groovy is Java

```
class Person {  
    final String name  
}
```

Variables, constants, params

String a

def a

final a

- Equality `a == b`
- Identity `a.is(b)`
- `()` sometimes optional: `println 'Joe'`

String interpolation

```
final s = 'Hi Joe'
```

```
final s = "Hi Dave"
```

```
final s = "Hi $name"
```

```
final s = "Hi ${user.name}"
```

```
final s = """Hi Dave,
```

```
How are you?
```

```
""")
```

Numbers and primitive types

15 - integer

15G - BigInteger

1.5 - BigDecimal

1.5d - Double

All values are objects: 5.upto(10)

Clever boxing and unboxing

Properties

```
class ProgrammingLanguage {  
    String name  
    String version  
    boolean easy=true  
}  
  
def groovy=new ProgrammingLanguage(  
    name:'Groovy', version:'1.5', easy:true)  
  
def java=new ProgrammingLanguage(name:'Java')  
java.version='1.6'
```


Power assert

assert 5 == customer.score

Exception thrown

17.2.2012 12:30:12 org.codehaus.groovy.runtime.StackTraceUtils sanitize

WARNING: Sanitizing stacktrace:

Assertion failed:

assert 5 == customer.score

```
    | |      |
    | |      4
    | [score:4]
false
```

Closures

```
Closure multiply1 = {int a, int b -> return a * b}
```

```
Closure multiply2 = {int a, int b -> a * b}
```

```
Closure multiply3 = {a, b -> a * b}
```

```
def multiply4 = {a, b -> a * b}
```

Closures – implicit parameter

```
def triple1 = {int number -> number * 3}
```

```
def triple2 = {number -> number * 3}
```

```
def triple3 = {it * 3}
```

Groovy is functional

```
def multiply = {a, b -> a * b}  
def double = multiply.curry(2)  
def triple = multiply.curry(3)
```

```
assert 4 == multiply(2, 2)  
assert 8 == double(4)  
assert 6 == triple(2)
```

Currying vs. Partial application

def multiply = {a, b \rightarrow a * b}

def partial = multiply.curry(3)

def curried = {x \rightarrow multiply.curry(x)}

Memoize

```
def triple = {3 * it}
```

```
def fastTriple = triple.memoize()
```

Closure scope

owner

delegate

this

`closure.resolveStrategy =`

`DELEGATE_FIRST / OWNER_FIRST`

`DELEGATE_ONLY / OWNER_ONLY`

Iterations

```
(1..10).each{number -> println number * 3}
```

```
1.upto(10) {println it * 3}
```

```
Closure triple = {it * 3}
```

```
1.step(11, 1){println triple(it)}
```


(Not exhaustive) list

each (aka for loop)

collect (aka map)

inject (aka reduce)

findAll (aka filter)

sum, size, findFirst, grep, groupBy

any, every, min, max, ...

Collections

```
final emptyList = []
```

```
final list = [1, 2, 3, 4, 5]
```

```
final emptyMap = [:]
```

```
final capitals = [cz : 'Prague', uk : 'London']
```

```
final list = [1, 2, 3, 4, 5] as LinkedList
```

```
final emptyMap = [:] as ConcurrentHashMap
```

map, filter, and reduce explained with emoji 🤔

```
map([🐮, 🍌, 🐔, 🌽], cook)  
=> [🍔, 🍟, 🍗, 🍿]
```

```
filter([🍔, 🍟, 🍗, 🍿], isVegetarian)  
=> [🍟, 🍿]
```

```
reduce([🍔, 🍟, 🍗, 🍿], eat)  
=> 🦌
```

Some operators

```
['Java', 'Groovy']*.toUpperCase()
```

```
customer?.shippingAddress?.street
```

```
return user.locale ?: defaultLocale
```

GDK = JDK + FUN

- `java.util.Collection`
 - `each()`, `find()`, `join()`, `min()`, `max()` ...
- `java.lang.Object`
 - `any()`, `every()`, `print()`, `invokeMethod()`, ...
- `java.lang.Number`
 - `plus()`, `minus()`, `power()`, `upto()`, `times()`, ...

Tip: Ask *DefaultGroovyMethods* for help

Syntax enhancements

- Dynamic (duck) typing – optional!
- GDK
- Syntax enhancements
 - Properties, Named parameters
 - Closures
 - Collections and maps
 - Operator overloading
 - ...

Part 2

Scripting

Agenda

- Scripting
- Script engine customization
- Grabbing libraries

Scripting

Evaluate custom Groovy code

At run-time!!!

```
new GroovyShell().evaluate('println Hi!')
```

<http://groovyconsole.appspot.com/>

Script customization

CompilerConfiguration

CompilationCustomizer

ImportCustomizer

ASTCustomizer

SecureASTCustomizer

Grab

```
1 @Grab(group='org.codehaus.groovy.modules', module='groovyws', version='0.5.2')
2 import groovyx.net.ws.WSClient
3
4 proxy = new WSClient("http://www.w3schools.com/webservices/tempconvert.asmx?WSDL",
5                       this.class.classLoader)
6 proxy.initialize()
7
8 result = proxy.CelsiusToFahrenheit(0)
9 println "You are probably freezing at ${result} degrees Fahrenheit"
```

Part 3

Functional programming

Agenda

- Functors
- Monoids
- Function composition
- Endofunctors
- Monads

Inspired by <http://www.slideshare.net/ScottWlaschin/fp-patterns-buildstuff/>

Functors

Dealing with wrapped data

$\text{map}: ([A], f: A \rightarrow B) \rightarrow [B]$

$\text{map}: (\text{Maybe}\langle A \rangle, f: A \rightarrow B) \rightarrow \text{Maybe}\langle B \rangle$

Functors are *mappable* (they have a **map** operation)

Monoids

Aggregating data and operations

Monoids

Aggregating data and operations

- A set of elements
- An operation that combines two elements
- An 'id' element neutral with respect to the operation
- Closure of the set with respect to the operation

$$1. \ a + id = id + a = a$$

$$2. \ (a + b) + c = a + (b + c)$$

$$3. \ a \in M \ \& \ b \in M \Rightarrow a+b \in M$$

Monoids

Reducible – any set of elements from a monoid can be reduced into a single value

reduce: $([A], f: (A, A) \rightarrow A) \rightarrow A$

Monoids

class Customer {name, address, orders}

vs.

class CustData {orders, totalAmount}

Monoids

class Customer {name, address, orders}

not a monoid

vs.

class CustData {orders, totalAmount}

a monoid

Monoids

class Customer {name, address, orders}

not a monoid

map

vs.

class CustData {orders, totalAmount}

a monoid

Composing functions

$f: A \rightarrow B$

$g: B \rightarrow C$

$f \gg g: A \rightarrow C$

Composing functions

$f: A \rightarrow B$

$g: B \rightarrow C$

$f \gg g: A \rightarrow C$

```
def f = {String s → s.size()}
```

```
def g = {Integer i → i%2==0 ? true : false}
```

```
def h = f >> g
```

Composing functions

$f: A \rightarrow B$

$g: B \rightarrow C$

$f \gg g: A \rightarrow C$

Not a monoid

Endofunctors

$f: A \rightarrow A$

with composition ($>>$) and an **id()** function
form a monoid

`[f1, f2, f3, f4, f5, ...].reduce(id, >>)`

Other monoids of functions

Elements: $f: \text{String} \rightarrow \text{Boolean}$

Other monoids of functions

Elements: $f: \text{String} \rightarrow \text{Boolean}$

`id()` – returns *true/false*

Operation: logical AND/OR

Monads

A monad is just a monoid in the category of endofunctors!

Monads

A monad is just a **monoid** in the **category** of **endofunctors**!

Monads

- Control side-effects – logging, IO, tx
- Manage containment of some sort
- Handle errors

Monads

Chaining functions that return values wrapped in some context

```
def f = {Integer v → new LoggedValue(v+1, 'Incremented')}
```

```
def g = {Integer v → new LoggedValue(v*2, 'Doubled')}
```

```
...
```

Signature: $\text{Integer} \rightarrow \text{LoggedValue}\langle\text{Integer}\rangle$

Monads

Error handling

```
try {  
  def y = f(x)  
  try {  
    def z = g(y)  
    try {  
      result = g(y)  
    } catch (...) {}  
  } catch (...) {}  
} catch (...) {}
```

Monads

Null-checking

```
if (x!=null) {  
    def y = f(x)  
    if (y!=null) {  
        def z = g(y)  
        if (z!=null)  
            result = g(y)  
    }  
}
```


Monads

class Maybe, subclasses Some, None

```
if (x.isSome()) {  
    def y = f(x)  
    if (y.isSome) {  
        def z = g(y)  
        if (z.isSome())  
            result = g(y)  
    }  
}  
}
```

Monads

class Maybe, subclasses Some, None

```
result = x >> f >> g >> h
```

Monads

Error handling

class *Maybe*, subclasses *Some*, *None*

def f = {Integer v \rightarrow new *Some*(v+1)}

def g = {Integer v \rightarrow v!=0 ? new *Some*(100/v) : new *None*()}

Signature of the functions: *Integer* \rightarrow *Maybe*<*Integer*>

Generalization

Integer \rightarrow *LoggedValue*<*Integer*>

Integer \rightarrow *Maybe*<*Integer*>

A \rightarrow *LoggedValue*<*A*>

A \rightarrow *ContextFor*<*A*>

A \rightarrow *ContextFor*<*B*>

Generalization

$A \rightarrow \text{ContextFor}\langle B \rangle$

$(\text{Integer} \rightarrow \text{LoggedValue}\langle \text{Integer} \rangle)$

Such functions do not form a monoid – not composable!

We need to convert them into

$\text{ContextFor}\langle A \rangle \rightarrow \text{ContextFor}\langle B \rangle$

$(\text{LoggedValue}\langle \text{Integer} \rangle \rightarrow \text{LoggedValue}\langle \text{Integer} \rangle)$

Monads' functions

Turn A into contextual A

$\text{unit}: A \rightarrow [A]$

Monads' functions

Turn A into contextual A

$\text{unit}: A \rightarrow [A]$

Turn a contextual function into a chainable contextual function

$\text{bind}: ([A], f: A \rightarrow [B]) \rightarrow [B]$

Monads' functions

Turn A into contextual A

$\text{unit}: A \rightarrow [A]$

Turn a contextual function into a chainable contextual function

$\text{bind}: ([A], f: A \rightarrow [B]) \rightarrow [B]$

Turn a plain function into a contextual function

$\text{lift}: (A \rightarrow B) \rightarrow (A \rightarrow [B])$

Summary



The joy of Ruby for Java programmers

<http://jroller.com/vaclav>
vaclav@vaclavpech.eu



References

<http://www.groovy.cz>

<http://groovy.codehaus.org>

<http://grails.org>

<http://groovyconsole.appspot.com/>

<http://www.manning.com/coenig2/>