Dynamické programovací jazyky

Václav Pech
Senior Software Developer
JetBrains, Inc.

Něco o mně

- Václav Pech
  - Programátor
  - Nadšenec do Groovy
- JetBrains
  - Dodavatel nástrojů pro vývojáře
    - IntelliJ IDEA, TeamCity, ReSharper a další

Něco o prezentaci

- Část 1: Dynamické jazyky
  - Charakteristika, vlastnosti
- Část 2: Groovy
  - Nové obzory pro Java vývojáře
- Část 3: Meta-programování v Groovy
  - Skriptování, Buildery, DSLs
- Otázky a odpovědi

Dynamické jazyky

- Smalltalk
- Ruby
- Python
- Groovy
- JavaScript
- Erlang
- Eiffel
- ...

Staticky a dynamicky typované jazyky

- Dynamické x Statické
- Dynamicky typované x Staticky typované

Rysy dynamických jazyků

- Modifikace objektů za běhu
- Funkce eval – vykonání nového kódu
- Funkcionální programování
  - Higher-order funkce
  - Currying
  - Closures
- Reflection
- Continuations – přerušení výpočtu
- Macros

Jeden jazyk to vše nezvládne

- SQL
- XHTML
- JavaScript
- XUL (XML UI Language)
- SVG (Scalable Vector Graphics)
- CSS (Cascading Stylesheets)
- XPath
- Velocity, XSL-T
- …
- **Groovy** – Scripting, DSLs, builders, …

Groovy

- Dynamický, objektově orientovaný skriptovací jazyk
  - Syntaxe podobná Javě
    - Včetně anotací a enumerací
    - Snadné osvojení pro Java vývojáře
  - Překládá se do Java bytecode
  - Velmi snadná integrace s Javou
  - Dynamické typování

Groovy

- Open sourced
- Od 2004 pod JSR 241: The Groovy Programming Language
- V současnosti rozvíjen společností G2one

- Podpora v IDE
  - Pluginy pro všechna IDE
  - JetGroovy plugin pro IntelliJ IDEA
- Runtime podpora
  - Spring, Seam, …

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'
```

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 – implicitní parametr

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

def triple2 = {number -> number * 3}

def triple3 = {it * 3}
```

Iterace

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

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


Closure triple = {it * 3}

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

GDK

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

Operator overloading

```
class Money implements Comparable {
    BigDecimal amount
    String currency

    Money plus(Money other) {
        checkCurrencies other
        return new Money(amount:this.amount
                + other.amount, currency:this.currency)
    }

    Money minus(Money other) {
        checkCurrencies other
        return new Money(amount:this.amount
                - other.amount, currency:this.currency)
    }
```

Stringy a GStringy

```
String greeting1 = 'Hi ' + friend.name

String greeting2 = "Hi $friend.name"
```

Multi-line Stringy a GStringy

```
def emailText="""
Dear ${customer.name},

thank you for your email on ${customer.lastEmail.topic}.
${customer.lastEmail.date < 1.week.ago ?
   "Sorry for not responding for so long" : ''}
...
Yours $currentUser.name
"""
```

Operátory

- Spread
  - def names = customers*.name
  - [‘Java’, *[‘Groovy’, ‘Ruby’, ‘Scala’, ‘Jython’]]
- Safe dereference
  - customer.shippingAddress?.street
- Membership
  - if (code.language in [‘Java’, ‘Groovy’])
- Elvis
  - return user.locale?:defaultLocale

Operátory

- Spaceship (compareTo)
  - a <=> b
- Find and Match - =~, ==~
  - methodName =~ 'findAll(.*)Employees'
- Subscript
  - map['key'] == map.key == map.get('key')
  - 'VeryLongString'[4..-1]
- As
  - new Thread({println 'In parallel'} as Runnable)

Rozšíření syntaxe

- Rozšířené JDK třídy (GDK)
- Dynamické (duck) typování – volitelně!
- Rozšíření syntaxe
  - Properties
  - Closures
  - Named parameters (new Customer(name:'Joe'))
  - Kolekce a mapy
  - Přetěžování operátorů
  - GString ("Name: $name")
  - GPath (employees.findAll{it.age>30}*.name)
  - Datový typ range (e.g. 9..29 or 0..-1)
  - Integrace regulárních výrazů

Testování

- Testy Java i Groovy kódu
- Spouštěné pomocí Ant či Maven

- Integrovaná podpora JUnit
  - assert…(), shouldFail(), …
- Duck typing
- Mocking
  - Integrovaná podpora stubů a mocků

Hlavní zaměření

- Skriptování
  - Přidávání a překlad kódu za běhu aplikace
- Buildery
  - Konstrukce hierarchických datových struktur
- Silná podpora pro DSL
  - Využívá schopnosti meta-programování

Skriptování

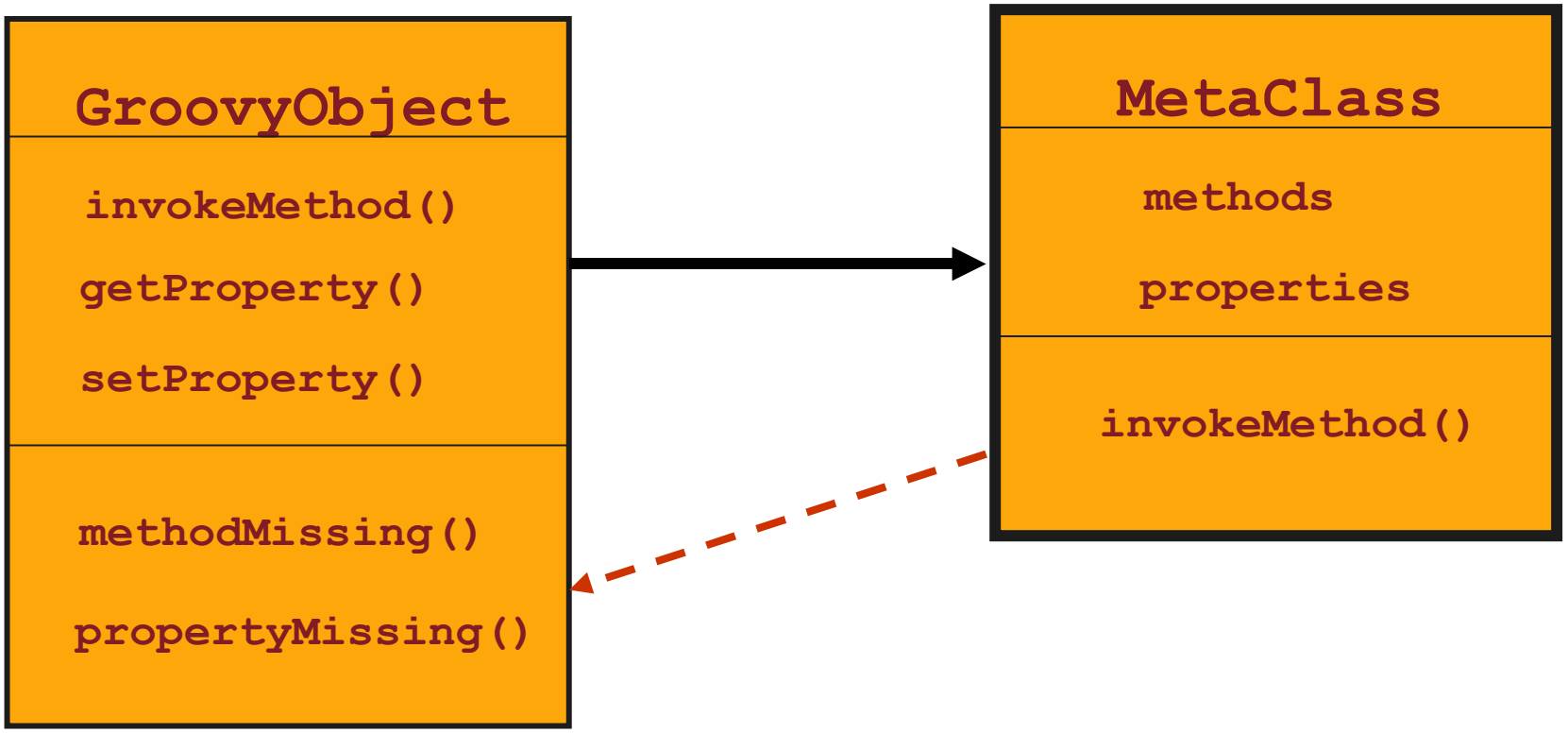
- Vyhodnocení Groovy kódu za běhu

```
def classDefinition = new GroovyShell().evaluate(codePane.text)
Runnable task=classDefinition.newInstance()
new Thread(task).start()
```

Meta-programování

- Změna postupu při vyvolání metod
  - invokeMethod()
  - methodMissing()
  - propertyMissing()
  - Přidat metodu nebo property k meta-třídě
  - Předefinovat invokeMethod() na meta-třídě

Dynamické vyvolání metod



Dynamické vyvolání metod v Groovy

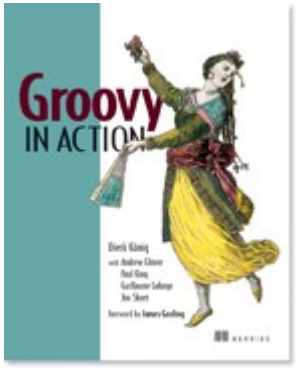
- Meta-class
  - Každá třída, včetně Java tříd
  - Metoda se nejdřív hledá v meta-class
- invokeMethod()
  - Není zřejmé, jaké metody se na objektu budou volat – buildery, parsery
  - Lze také provádět aspekty kolem existujících metod
  - Pomalé, protože interceptuje vždy
- Missing method – interceptuje jen neexistující metody – ze smalltalku
  - Caching of missing methods – vytvořit metodu v metodě methodMissing()
- Mix-ins – statické, dynamické
  - Kategorie

Komunita

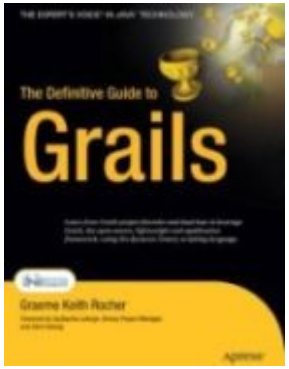
- G2One Inc. – G2One.com
  - Guillaume Laforge (Groovy Project Lead)
  - Graeme Rocher (Grails Project Lead)
- http://groovy.codehaus.org/
- http://grails.org/
- Novinky ze světa Groovy
  - http://aboutgroovy.com
  - http://groovyblogs.org
  - http://groovy.dzone.com

Literatura

- Groovy in Action



- The Definitive Guide to Grails



Shrnutí

- Dynamické jazyky
- Groovy
- Koncepty
  - Testy
  - Skriptování
  - Meta-programování
  - **Kontakt: vaclav@jetbrains.com**

Otázky