

Václav Pech

Software Developer and Product Evangelist
JetBrains, Inc.

Groovy for Java experts

Groovy

- Dynamically typed
- Dynamic
- Java-like syntax
- Compiled into Java bytecode



Groovy

- Open source
- Maintained by SpringSource
- IDE support
- Spring, Seam, EJB, ...



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'
```

Groovy 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}
```

Implicit parameter



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

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

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

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)}
```

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()`, ...
- ...

Some operators



- ['Java', 'Groovy']*.`toUpperCase()`
- `customer?.shippingAddress?.street`
- `return user.locale ?: defaultLocale`

Syntax enhancements

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



Enjoy writing tests

- For both Java and Groovy
- Run with Ant, Maven, IDE, ...
- Integrated JUnit support
 - `assert...()`, `shouldFail()`, ...
- Relaxed typing
- Easy mocking



Big Boy Toys

- Scripting
- Builders
- Domain Specific Languages
- Meta-programming



Scripting

- Evaluate custom Groovy code



At run-time!!!

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

Builders

- Construct hierarchies



```
xml.records() {  
    order(id: 'PL19826714', date: '21-01-2008') {  
        item(quantity: 10) {  
            product(id: '76327')  
            price(base: 100) {  
                volumeDiscount(value: 5)  
            }  
        }  
    }  
}
```

Builders - GAnt



```
ant.sequential {
  myDir = "target/AntTest/"
  mkdir(dir: myDir)
  copy(todir: myDir) {
    fileset(dir: "src/test") {
      include(name: "**/*.groovy")
    }
  }
  List dirs = ['core', 'lib', 'engine', 'gui', 'db']
  for(String currentDir:dirs) {
    String targetDir="target/$currentDir"
    mkdir(dir:targetDir)
  }
}
```

Builders – Spring config



```
dataSource(BasicDataSource) {  
    driverClassName = "org.hsqldb.jdbcDriver"  
    url = "jdbc:hsqldb:mem:shopDB"  
}  
  
sessionFactory(ConfigurableLocalSessionFactoryBean) {  
    dataSource = dataSource  
    hibernateProperties = ["hibernate.hbm2ddl.auto": "create-drop",  
        "hibernate.show_sql": true]  
}  
  
calculator(demo.shop.CalculatorImpl) {bean ->  
    bean.singleton = true  
    bean.autowire = 'byType'  
}
```

Categories



```
StringUtils.countMatches(myString, 'Groovy')
```



```
use (StringUtils) {  
    myString.countMatches('Groovy')  
}
```

DSL

- Limited purpose language
- Targeted to a particular domain
- Friendlier API to a framework
 - External
 - SQL, HTML, CSS, ...
 - Internal



DSL – Date manipulation



```
use (org.codehaus.groovy.runtime.TimeCategory) {  
    println "Tomorrow: ${1.day.from.today}"  
    println "A week ago: ${1.week.ago}"  
    println "Date: ${1.month.ago + 1.week + 2.hours - 5.minutes}"  
    println "Date ${ (1.month + 10.days).ago}"  
}
```

DSL – Hibernate criteria



```
def participants = Participant.createCriteria().list {
    gt('age', age)
    or{
        eq('interest', 'Java')
        eq('interest', 'Groovy')
    }
    jug {
        ilike('country', 'de')
    }
    order('lastName', 'asc')
}
```

DSL – Account manipulation

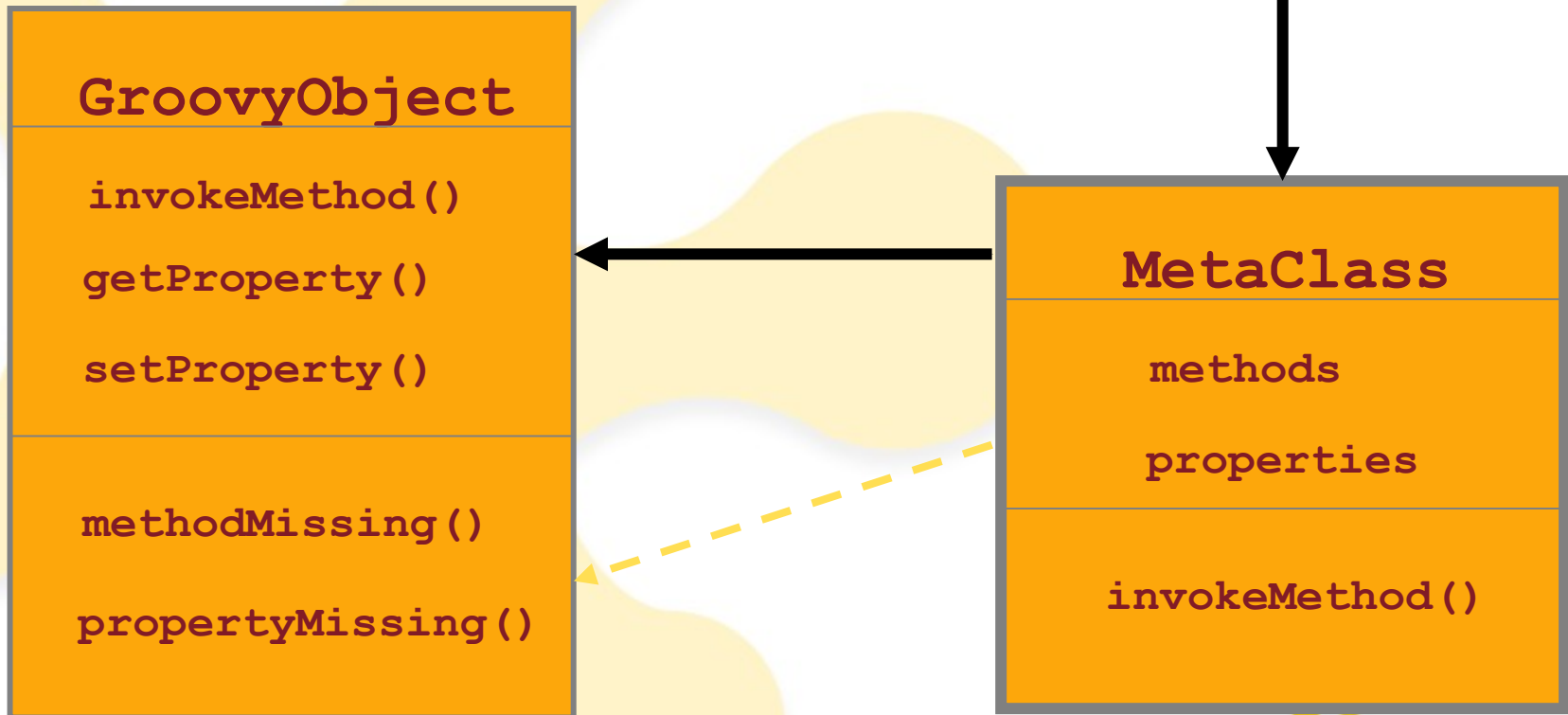


```
Money money = new Money(amount: 350, currency: 'eur')
getAccount('Account1').withDraw money
getAccount('Account3').deposit money
```



```
"Account1" >> 350.eur >> "Account3"
```

Dynamic method invocation



Summary

Groovy



Powerful Java extension

Tests, Builders, Scripting, DSLs

– **Contact me: vaclav@jetbrains.com**

Questions