Mardi 9 septembre 2008

PARIS JUG

www.parisjug.org

www.parisjug.org

PARIS JUG

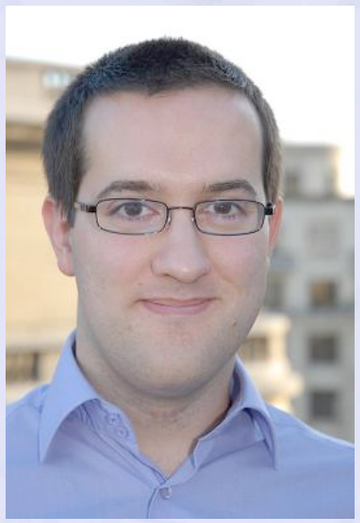
Mardi 9 septembre 2008

Introduction à Groovy

Un langage dynamique pour la JVM

Guillaume Laforge
VP Technology

G2 one



www.parisjug.org

Guillaume Laforge

- Chef de projet de Groovy
- Spec Lead du JSR-241

- Initiateur du framework Grails

- Co-auteur de “Groovy in Action”

- Fondateur de G2One, Inc.

Objectifs

DECOUVRIR :

Qu’est-ce que Groovy ?

Domain-Specific Language

Le framework web Grails

Agenda

- **Qu'est-ce que Groovy ?**
  - Et qu'est-ce que Grails ?
- **Groovy en détail**
  - Sa syntaxe et ses APIs
  - Les nouveautés de Groovy 1.5
  - Cas d'utilisation de Groovy
- **Domain-Specific Languages**

Qu'est-ce que Groovy ?

Décortiquons Groovy

- **Groovy est un langage dynamique pour la machine virtuelle Java**

- **Langage dynamique**
  - Inspiration provenant de langages existants
  - Java pour la syntaxe générale — facile à apprendre
    - Groovy est un langage Orienté Objet
  - Ruby et Smalltalk
    - pour les closures, les aspects de programmation fonctionnelle

- **Machine virtuelle Java**
  - Peut être interprété ou compilé
  - Au final : du bytecode Java est généré

Un projet Open Source

- **Open Source : sous licence Apache**
  - Hébergé chez Codehaus
- **Mature : plus de 5 ans, version 1.5.6**
  - Une petite vingtaine de committers
    - certains à plein temps, ou dédiés au plugin Eclipse, sur Swing...
    - contributions externes — IBM, Oracle, JBoss
- **Un grand succès**
  - 32K-37K downloads / mois
    - comparé à ~30K pour JRuby, ~7K pour Jython et ~5K pour Scala
  - Mailing-listes les plus actives — milliers d'abonnés
- **En cours de standardisation**
  - Au sein du Java Community Process
  - JSR-241 — Java Specification Request

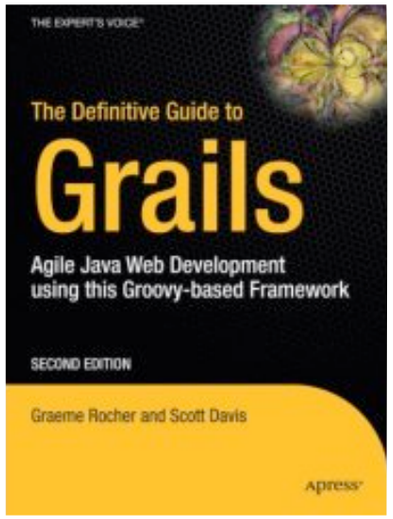
Intégré partout !

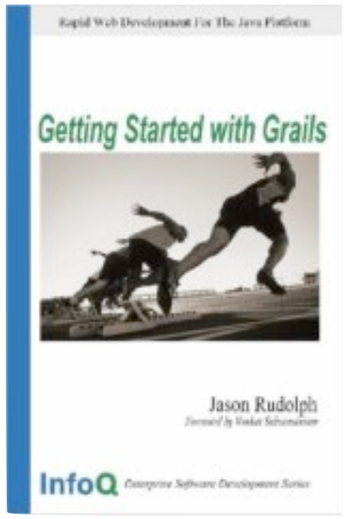
- **Groovy est intégré...**
  - Dans d'autres projets Open Source
    - Spring, JBoss SEAM, ServiceMix, Oracle OC4J, Mule...
  - Dans des applications métier critiques
    - Mutual of Omaha — US Fortune 500
    - National Cancer Institute
  - Dans des produits commerciaux
    - IBM ProjectZero
    - SAP Composition on Grails
    - Oracle Data Integrator
    - Oracle Fusion — ADF Business Components

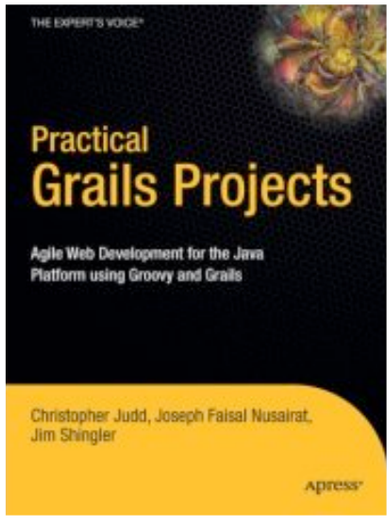
Un projet innovant reconnu

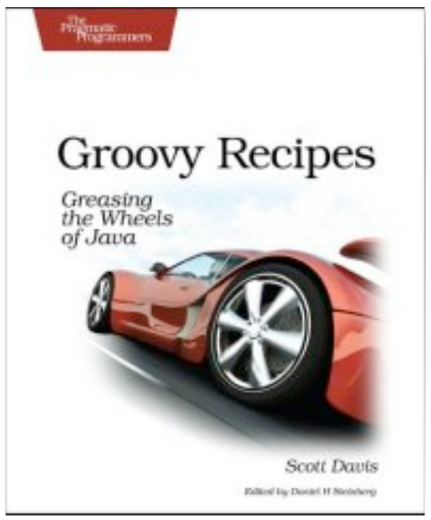
- **40 propositions, 10 nominés**
- **Groovy a gagné le premier prix**
  - Le projet le plus créatif et innovant de 2007
  - Donnation de 10 000 €
- **Gagnant en 2006 et 2008**
  - Spring et Grails

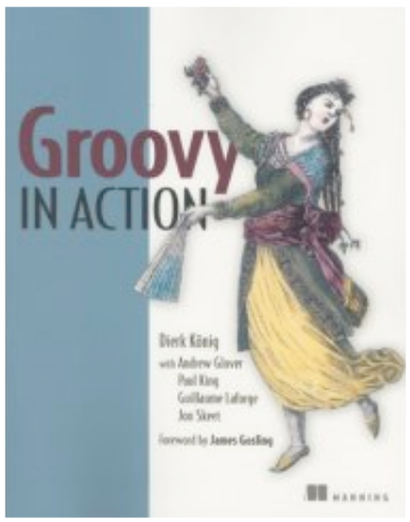
Book Growth


THE EXPERT'S VOICE®
The Definitive Guide to
Grails
Agile Java Web Development using this Groovy-based Framework
SECOND EDITION
Graeme Rocher and Scott Davis
Apress®


Getting Started with Grails
Jason Rudolph
InfoQ


THE EXPERT'S VOICE®
Practical
Grails Projects
Agile Web Development for the Java Platform using Groovy and Grails
Christopher Judd, Joseph Faisal Nusairat, Jim Shingler
Apress®


The Pragmatic Programmers
Groovy Recipes
Greasing the Wheels of Java
Scott Davis


Groovy
IN ACTION
Dierk König
with Andrew Glover
Paul King
Guillaume Laforge
Jon Skeet
Foreword by James Gosling
MANNING


SEARCH INSIDE!™
Groovy
PROGRAMMING


The Pragmatic Programmers
Programming Groovy
Dynamic Productivity for the Java Developer
BETA


dierk KÖNIG
andrew GLOVER
paul KING
guillaume LAFORGE
jon SKEET
GROOVY
IM EINSATZ
MANNING
HANSER


Groovy
für Java-Entwickler
O'REILLY


entwickler.press
Grails
Webanwendungen mit Groovy und Grails
Sven Haiges
schnell + kompakt


SEARCH INSIDE!™
Groovy 1.5

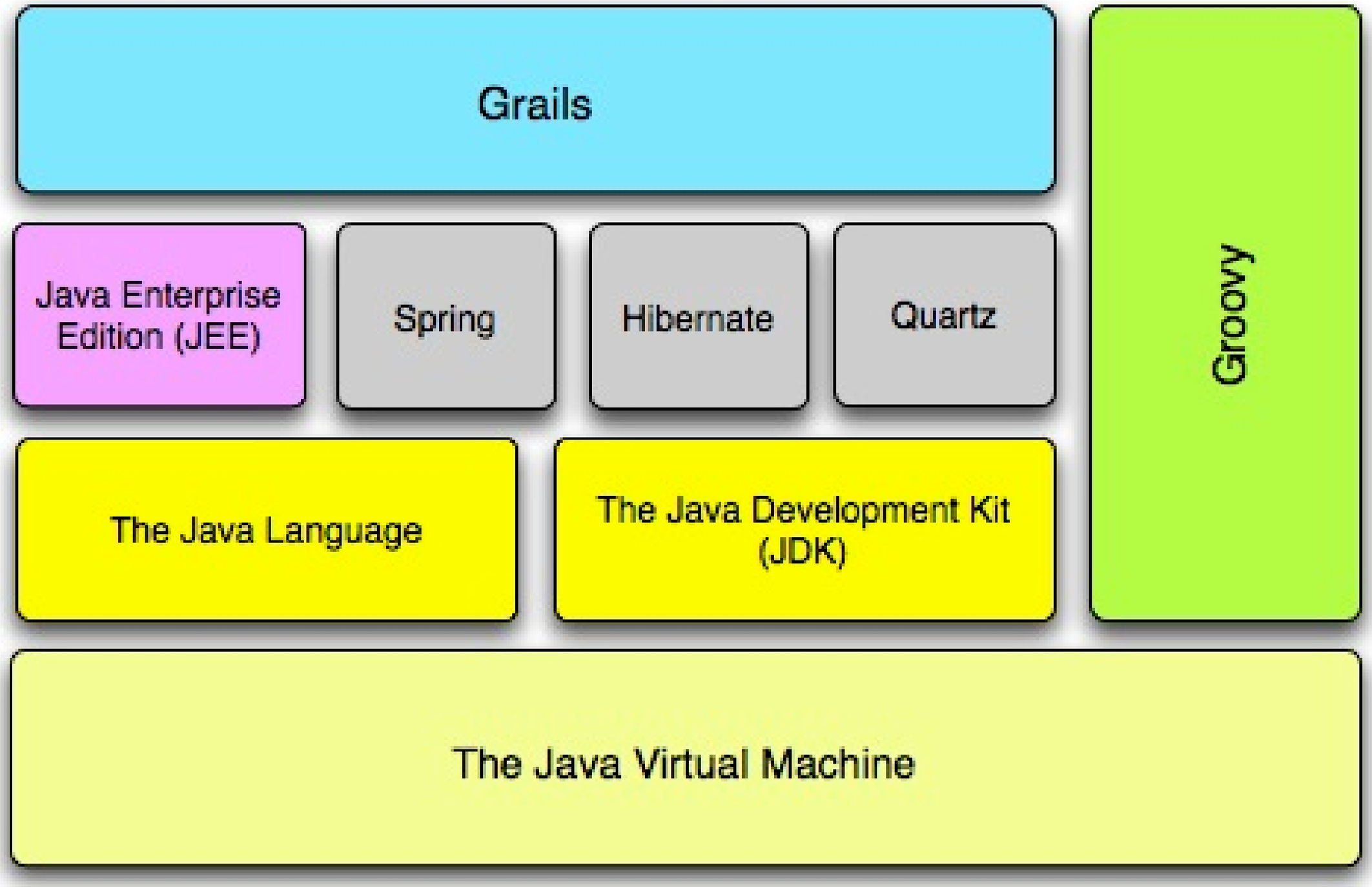

Groovy

Qu'est-ce que Grails ?

- **Grails est un framework de développement d'applications Web**
  - Inspiré par Ruby on Rails, Django, TurboGears
  - Mais s'appuyant sur des briques Open Source reconnues et éprouvées
    - Spring, Hibernate, Quartz, SiteMesh...

- **Avantages**
  - Facile à prendre en main — Conventions over Config.
  - Réutilisation de l'existant
    - Schémas de base existants, configurations Spring
    - Déploiement sur les serveurs d'application Java
    - Protection de l'investissement

Grails : la “Big Picture”



Groovy, plus en détail

Pour commencer

- **Aller sur http://groovy.codehaus.org/Download**
- **Télécharger la dernière version**
  - Groovy 1.5.6 — dernière version officielle
  - Groovy 1.6-beta-1 — prochaine version en béta
- **Dézipper l'archive dans un répertoire**
- **Créer une variable d'environnement**
  - GROOVY_HOME pointant vers ce répertoire
- **Ajouter $GROOVY_HOME/bin au path**

- **Vous êtes prêts !**

En un coup d'oeil

- Totalement Orienté Objet
- Closures : bout de code réutilisable
- Propriétés : pas besoin d'attendre Java 7 ou 8
- Surcharge d'opérateurs : pas le cauchemar de C++
- Syntaxe native pour les listes, maps, regex
- Multi-méthodes
- GPath : navigation à la XPath dans les graphes
- Arithmétique en BigDecimal par défaut
- Parenthèses et points-virgules optionnels
- APIs : SQL, Ant, XML, templates, Swing, JMX, WS
- Idéal pour les Domain-Specific Languages

Hello World en Java

```
public class HelloWorld {
    private String name;

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public String greet() {
        return "Hello "+ name;
    }

    public static void main(String[] args) {
        HelloWorld helloWorld = new HelloWorld();
        helloWorld.setName("Groovy");
        System.out.println( helloWorld.greet() );
    }
}
```

Hello World en Groovy

```groovy
public class HelloWorld {
    private String name;

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public String greet() {
        return "Hello "+ name;
    }

    public static void main(String[] args) {
        HelloWorld helloWorld = new HelloWorld();
        helloWorld.setName("Groovy");
        System.out.println( helloWorld.greet() );
    }
}
```

Démonstration

www.parisjug.org

Un peu plus Groovy

```
class HelloWorld {
      String name
      String greet() { "Hello $name" }
}

def helloWorld = new HelloWorld(name: "Groovy")
println helloWorld.greet()
```

Listes, maps, regex, ranges...

- Listes
  - `def list = [“Groovy”, “Grails”]`
- Maps — dictionnaires
  - `def states = [CA: “California”, TX: “Texas”]`
- Expressions régulières
  - `assert “foo” ==~ /fo*/`
- Ranges
  - `def range = 1..10`
- Chaînes multilignes
  - `def multiligne = ””” plusieurs lignes ”””`
- GString : `println “My name is $name”`

Les closures

- **Pas besoin d'attendre Java 7/8/9**
- **Les closures**
  - block de code assignable et réutilisable
  - combiner, stocker, partager données et logique

- **Exemples**

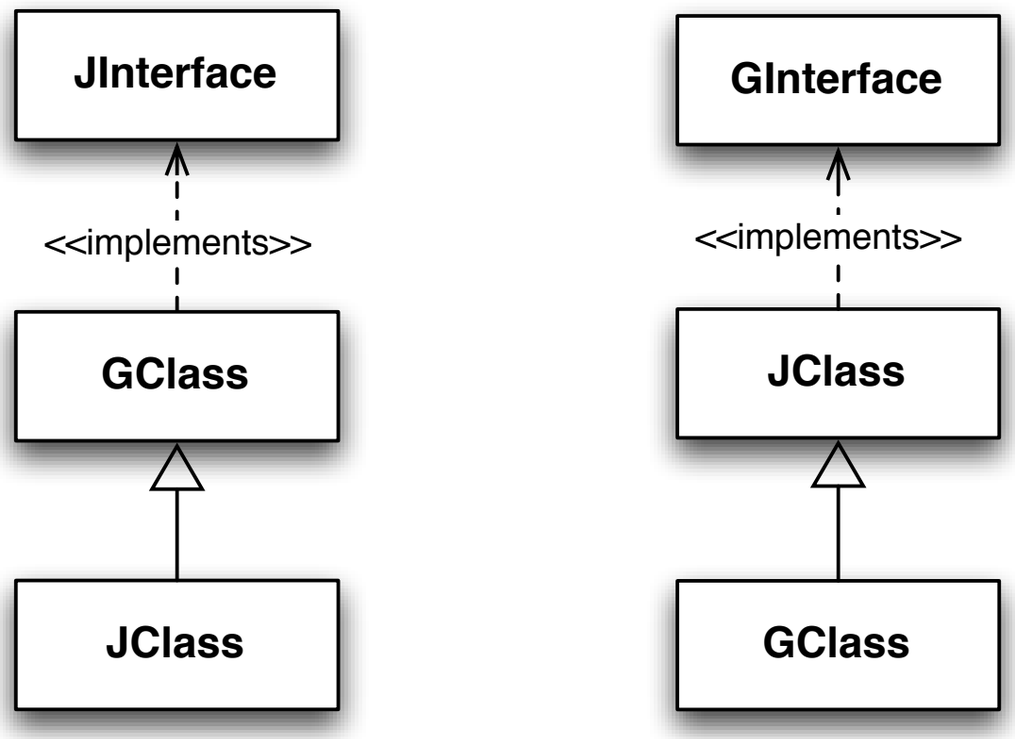
```
new File('x.txt').eachLine{ println it }
[0, 1, 2].each { println it }

def numbers = 1..100
numbers.findAll { it > 90 }
def odd = { it % 2 == 1 }
numbers.findAll( odd )

Thread.start { /* do something */ }
```

Interopérabilité avec Java

- **Groovy génère des .class — bytecode Java**
  - Mêmes chaînes de caractères et expressions régulières
  - Mêmes APIs — JDK, collections, 3rd party
  - Même modèle de sécurité, même threading
  - Même modèle de programmation Orienté Objet
  - Compilation “jointe”



Nouveautés Groovy 1.5

- **Groovy 1.0 (jan. 2007) et 1.5 (déc. 2007)**
  - performances, fonctionnalités dynamiques, outillage
- **Ajout de fonctionnalités Java 5**
  - annotations, generics
  - enums, import static, varargs
- **Groovy**
  - Le seul langage dynamique alternatif pour la JVM qui supporte les spécificités Java 5
    - EJB 3, JPA, TestNG, Hibernate annotations, Spring annotations
  - Contribution de JBoss pour la mise au point

JBoss SEAM

- Exemple pris de la documentation SEAM

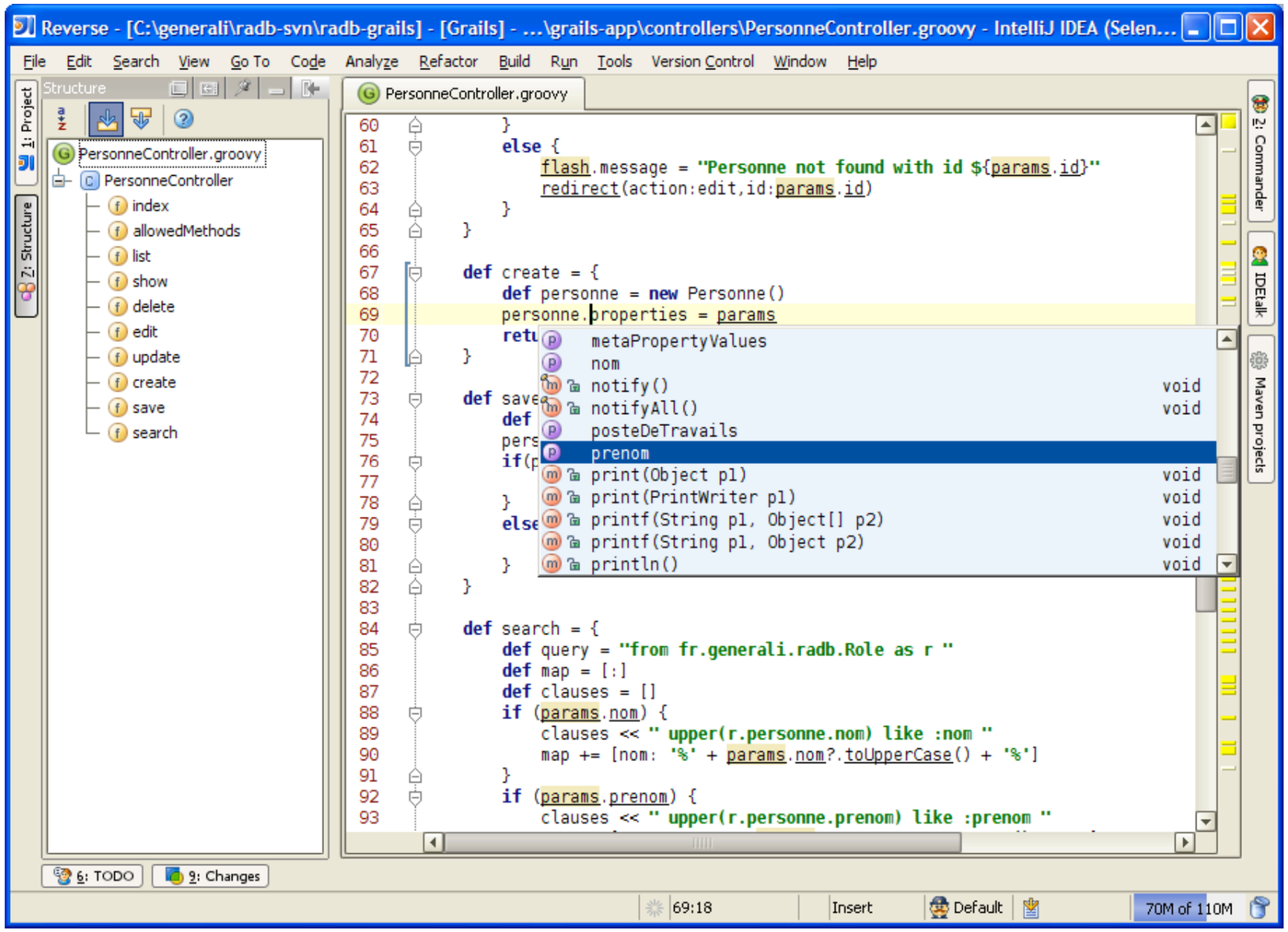
```
@Entity
@Name("hotel")
class Hotel implements Serializable {
    @Id @GeneratedValue
    Long id

    @Length(max=50) @NotNull
    String name

    @Override String toString() {
        "Hotel ${name}"
    }
}
```

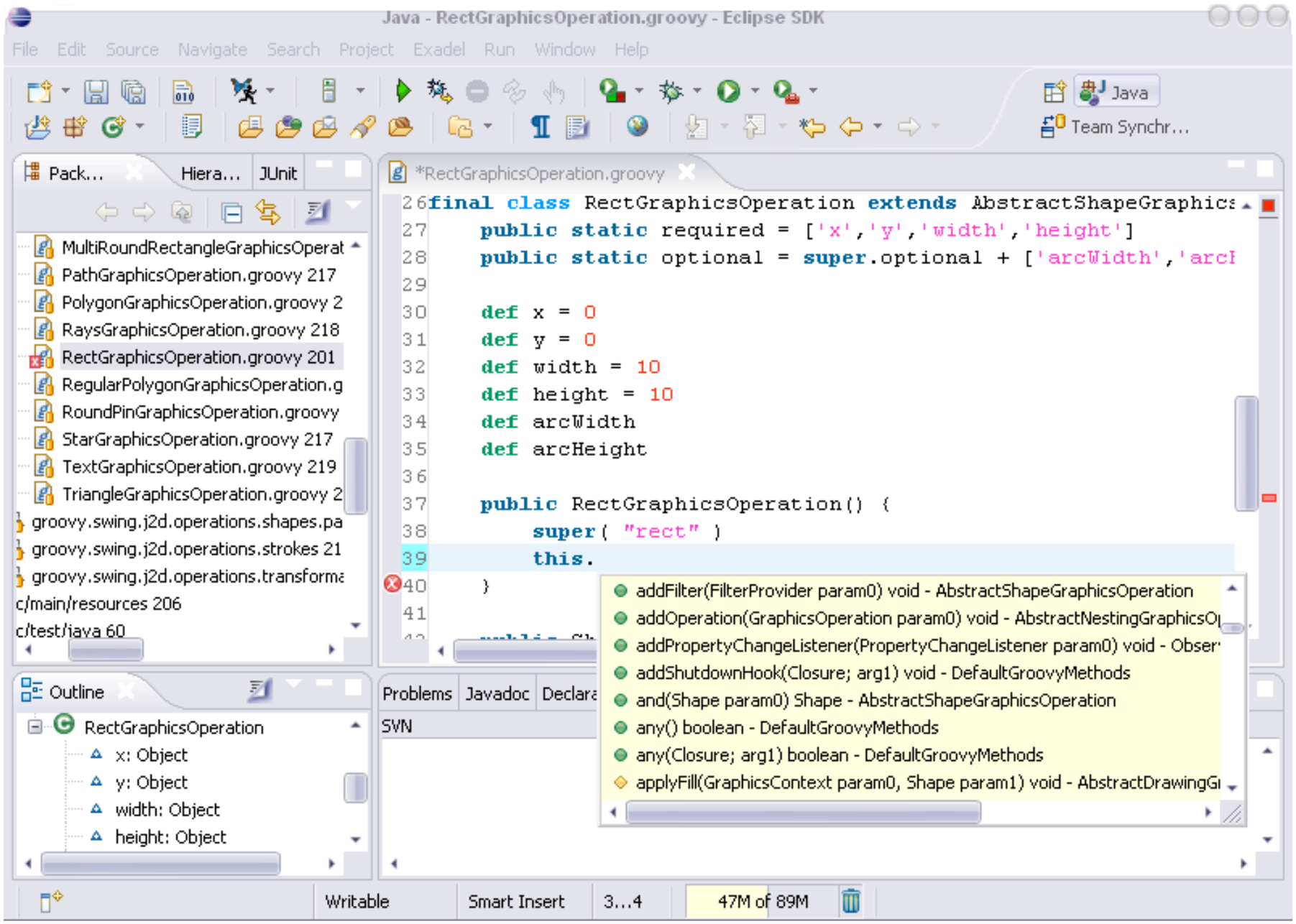
Outillé : IntelliJ IDEA

- **IntelliJ IDEA**, Eclipse, NetBeans, JDeveloper



Outillé : Eclipse

- IntelliJ IDEA, Eclipse, NetBeans, JDeveloper



Manipuler du XML

- Avec ce bout d'XML
  - ```
    def xml = """
    <languages>
      <language name="Groovy">
        <feature coolness="low">SQL</feature>
        <feature coolness="high">Template</feature>
      </language>
      <language name="Perl"/>
    </languages>"""
    ```

- Naviguer ce graphe XML
  - ```
    def root = new XmlParser().parseText(xml)
    println root.language.feature[1].text()

    root.language.feature
        .findAll{ it['@coolness'] == "low" }
                  .each{ println it.text() }
    ```

Produire du XML

- On peut également en produire facilement

```
new MarkupBuilder().languages {
    language(name: "Groovy") {
        feature(coolness: "low", "SQL")
        feature(coolness: "high", "Template")
    }
    language(name: "Perl")
}
```

Interfaces Swing

```
import javax.swing.WindowConstants as WC
import groovy.swing.SwingBuilder
def theMap = [color: "green", object: "pencil"]
    def swing = new SwingBuilder()
    def frame = swing.frame(
        title: 'A Groovy Swing', location: [240,240],
        defaultCloseOperation:WC.EXIT_ON_CLOSE) {
        panel {
        for (entry in theMap) {
            label(text: entry.key)
            textField(text: entry.value)
        }
        button(text: 'About', actionPerformed: {
            def pane = swing.optionPane(message: 'SwingBuilder')
            def dialog = pane.createDialog(null, 'About')
            dialog.show()
        })
        button(text:'Quit', actionPerformed:{ System.exit(0) })
    }
}
frame.pack()
frame.show()
```

Démonstration

www.parisjug.org

Support JDBC et SQL

- JDBC facile
  - ```
    def sql = Sql.newInstance(url, usr, pwd, driver)
    sql.execute("insert into table values ($foo, $bar)")
    sql.execute("insert into table values(?,?)", [a, b])
    sql.eachRow("select * from USER") { print it.name }
    def list = sql.rows("select * from USER")
    ```

- DataSet : Object Relational Mapping léger
  - ```
    def set = sql.dataSet("USER")
    set.add(name: "Johnny", age: 33)
    set.each { user -> println user.name }
    set.findAll { it.age > 22 && it.age < 42 }
    ```

COM / ActiveX automation

- Des souvenirs de VBScript ?

```
def outlook = new
    ActiveXObject("Outlook.Application")

def message = outlook.CreateItem(0)
def emails = "glaforge@gmail.com"
def rec =
   message.Recipients.add(emails)
rec.Type = 1
message.Display(true)
```

Pourquoi intégrer Groovy ?

- **Groovy est idéal pour le test unitaire**
  - mocks, stubs, GroovyTestCase
  - tests plus concis et plus lisibles
- **Un outil pratique et utile dans son escarcelle**

- **Externaliser la logique métier**
  - rêgles métier pouvant être écrites par les experts
  - les rêgles peuvent suivre leur propre cycle de vie
- **Proposer des points d'extension**
  - pour des plugins additionels
  - pour personnaliser une application selon les besoins du client

Quelques cas d'utilisation

- **Mutual of Omaha**
  - 50K LOC, moitié code métier / moitié tests
  - Calcul de risque de polices d'assurance
- **Le ministère de la Justice**
  - Approche MDA
  - UML, XMI, templating et génération de code
- **EADS**
  - Interface de soumission de calcul distribué
- **Institutions financières**
  - Personnalisation d'outil de traders
  - Gestion de "hedge funds" traitant des millions de dollars
- **CISCO, Institut américain du cancer...**

Différents mécanismes

- **javax.script.* / JSR-223**
- **Apache Bean Scripting Framework**
  - vieillisant : préférer JSR-223 ou mécanismes Groovy
- **Spring : support de Groovy intégré**
  - <lang:groovy id='bean' script-source='...'/>
- **Mécanismes spécifiques de Groovy**
  - Eval, GroovyShell, GroovyClassLoader, GroovyScriptEngine

JDK 6 : javax.script.* / JSR-223

- API de scripting agnostique pour s'intégrer à des langages différents de Java
- scripting.dev.java.net propose un moteur de script enrobant Groovy
  - Ajouter les jars sur le classpath et c'est parti !

- Exemple :
  - ```
    ScriptEngineManager manager =
            new ScriptEngineManager();
    ScriptEngine gEngine =
            manager.getEngineByName("groovy");
    String result = (String) gEngine.eval("'Foo' * 2");
    ```
    ```

Intégration Spring 2.x

- **Depuis Spring 2.0, ajout d'un namespace spécial pour les langages alternatifs**
  - pour la configuration / définitions de beans
  - on peut injecter un POJO dans un POGO et vice versa
    - mélange complètement transparent de Groovy et Java !

- **Exemple :**
  -
```
<lang:groovy id="events"
        script-source="classpath:dsl/eventsChart.groovy"
        customizer-ref="eventsMetaClass" />
```

Mécanismes Groovy

- **Groovy peut-être compilé ou “interprété”**
  - Compiler en .class et mettre dans un JAR

- **Mais Groovy offre différents mécanismes pour intégrer du code Groovy à la volée (“interpreté”)**
  - Eval
    - pour évaluer des expressions simples
  - GroovyShell
    - similaire à l’API javax.script
  - GroovyScriptEngine
    - pour des scripts interdépendants avec rechargement sur modif.
  - GroovyClassLoader
    - pour les utilisations plus avancées, avec manipulation d’AST...

Les Domain-Specific Languages

Qu’est-ce qu’un DSL ?

- **DSL : un “Domain-Specific Language”**
  - parfois appelé “little language”

- **Wikipedia**
  - “A DSL is a programming language designed to be useful for a specific set of tasks”

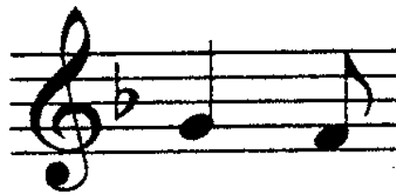
- **C’est un langage qui modèlise un certain domaine de connaissance ou d’expertise, où les concepts du domaine sont rattachés aux éléments du langage**

Caractéristiques d'un DSL

- **Un Domain-Specific Language est**
  - un langage
    - qui n'est pas nécessairement Turing complet
  - qui couvre un certain domaine de connaissance
  - qui a une forme
    - textuelle ou visuelle
  - qui produit au final un résultat
    - configure des objets, représente des données, une notation
  - qui peut être interne ou externe
    - Embedded DSL se reposant sur un langage hôte
    - ou indépendant — avec un lexer / parser maison
  - qui a certaines propriétés qualitatives
    - facilité d'écriture, utilisable, testabilité, extensible

Exemples de DSLs

- **Technique**
  - SELECT * FROM USER WHERE NAME LIKE ‘Guil%’
  - ^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$
- **Notations**
  - 1. e4 e5 2. Nf3 Nc6 3. Bb5 a6
  - U R’ U2 R U R’ U R



- **Langages plus “métier”**
  - Description des rêgles de calcul de risque sur des polices d’assurance
  - Rêgle comptables d’une banque
  - Description de facturation, de calcul de gestion de fonds des particularités techniques d’un produit...

Pourquoi créer un DSL ?

- Utiliser un langage plus expressif qu'un langage de programmation généraliste
- Partager une métaphore commune entre développeurs et experts métier
- Mettre à contribution les expers métier pour le développement de la logique métier
- Eviter de noyer le code métier dans du code technique sans intérêt
- Séparer proprement la logique métier du code applicatif technique

Syntaxes et fonctionnalités offertes par Groovy

- Parenthèses optionnelles, map, propriété sur les entiers
  - `move left`
  - `compare indicator: 'Nikei', withFund: 'Aberdeen'`
  - `account.debit amount: 30.euros, in: 3.days`
- Structures de contrôle personnalisées à base de closures
  - ```
    unless ( account.balance < 0 ) {
        account.debit 10.dollars
    }
    ```
  - `execute( withTimeOutOf: 50.seconds ) { ... }`
- Surcharge d'opérateurs
  - `a + b` équivalent à `a.plus(b)`
  - `taskA | taskB` equivalent à `taskA.or(taskB)`
    ```

Conclusion

Pour finir...

- Langage dynamique parfaitement intégré avec l'environnement Java
- Simplifie la vie des développeurs
- Perspectives intéressantes pour étendre une application existante
- Parfait pour les Domain-Specific Languages
- Propose un framework de développement Web intégré réutilisant l'état de l'art OSS
- Protège son investissement
  - en compétences, en développements existants en matériel, et en logiciel

Quelques liens

- Groovy : http://groovy.codehaus.org
- Grails : http://grails.org

- Groovy blogs: http://groovyblogs.org
- AboutGroovy: http://aboutgroovy.com

- G2One: http://www.g2one.com

Questions / Réponses

www.parisjug.org

Sponsors

Merci de votre attention!

www.parisjug.org

Licence

http://creativecommons.org/licenses/by-nc-sa/2.0/fr/