



DSL á la Groovy - Sprechen Sie Domänisch?

Bernd Schiffer

bernd.schiffer@akquinet.de

Mitarbeit: Stefan Rook

stefan.roock@akquinet.de



Herzlich Willkommen!



Stefan Roock

- akquinet AG
- Coach für agile Methoden, Projektleiter, Softwarearchitekt
- XP, Scrum, Akzeptanztests, TDD, Refactoring ...



Bernd Schiffer

- akquinet AG
- Softwareentwickler
- XP, Scrum, TDD, Groovy, Grails

Was Sie heute erwartet

- Was sind DSLs?
- Beispiele für DSLs
- Von Java zur Groovy-DSL in 8 Schritten
- MOP in Groovy
- Builder in Groovy

Historie: Little Languages

DSL Language

- strukturierte Sprache
- keine Turingvollständigkeit erforderlich

Fachlicher Bezug

DomainSpecificL

- Text oder Grafik (oder Audio)

Gegenteil von DSL

GeneralPurposeLanguage

- „gängige“ Programmiersprachen
(etwa Java, C#, Perl, Ruby, Groovy, Python, Smalltalk)
- turingvollständig
- Datentypen/Objekte (z.B. Listen, Bäume)

Konkret vs. Abstrakt

DSL vs. GPL

DSL-Qualitätsansprüche

- Schreibbarkeit
- Lesbarkeit
- Testbarkeit
- Erweiterbarkeit

Beispiel

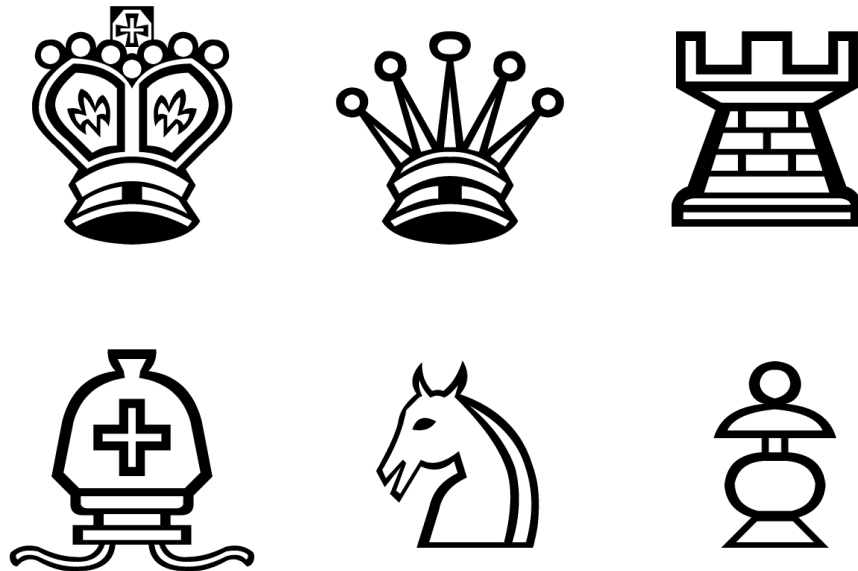
1.e2 – e4

2.e7 – e5

3.Sg1 – f3

4.Sb1 – c3

5.Lg1 – b2



Schach

Beispiel

♥A ♥K ♥Q ♥J ♥T

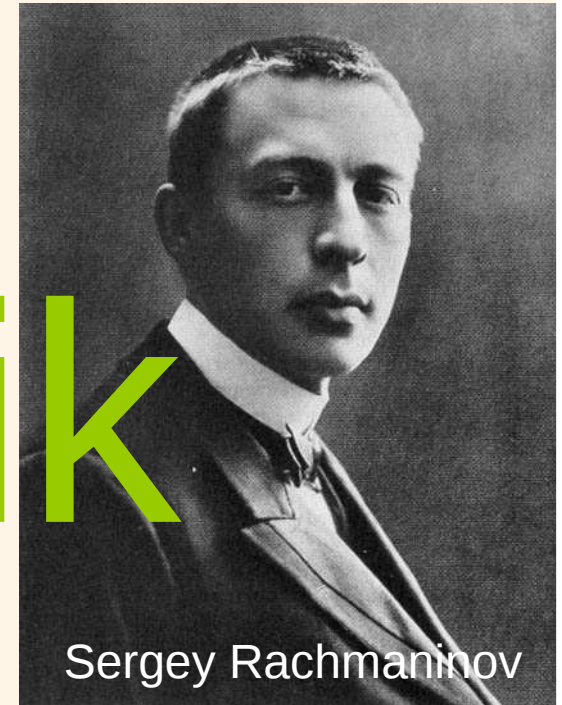


Texas Hold'em Poker

Beispiel



Musik



Sergey Rachmaninov

Beispiel

— · — · · — · ·

K e i n e

· — · · · — · · — — · — · — — ·

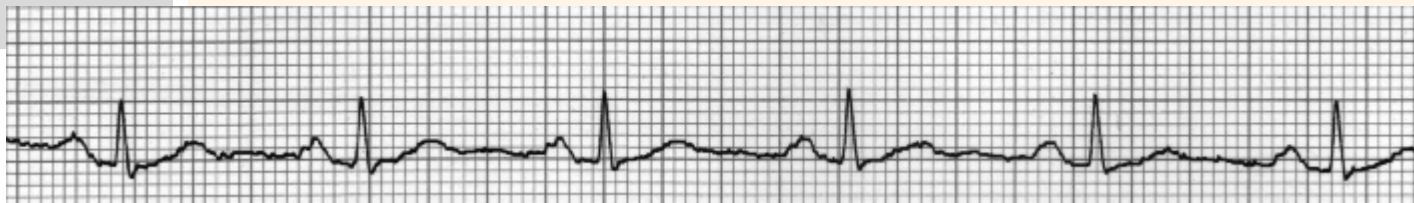
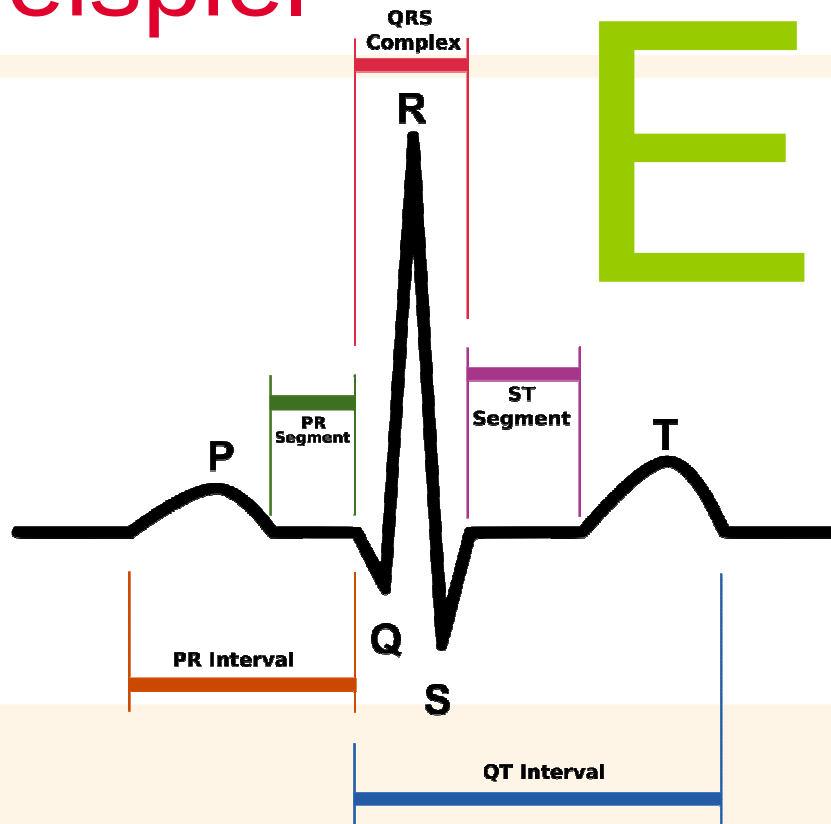
A h n u n g

Morsecode



Beispiel

EKG



Beispiel

```
SELECT *  
FROM adresse  
WHERE ...
```

SQL

Beispiel

[A-Z0-9]?

Hauptstraße 42 (no go)

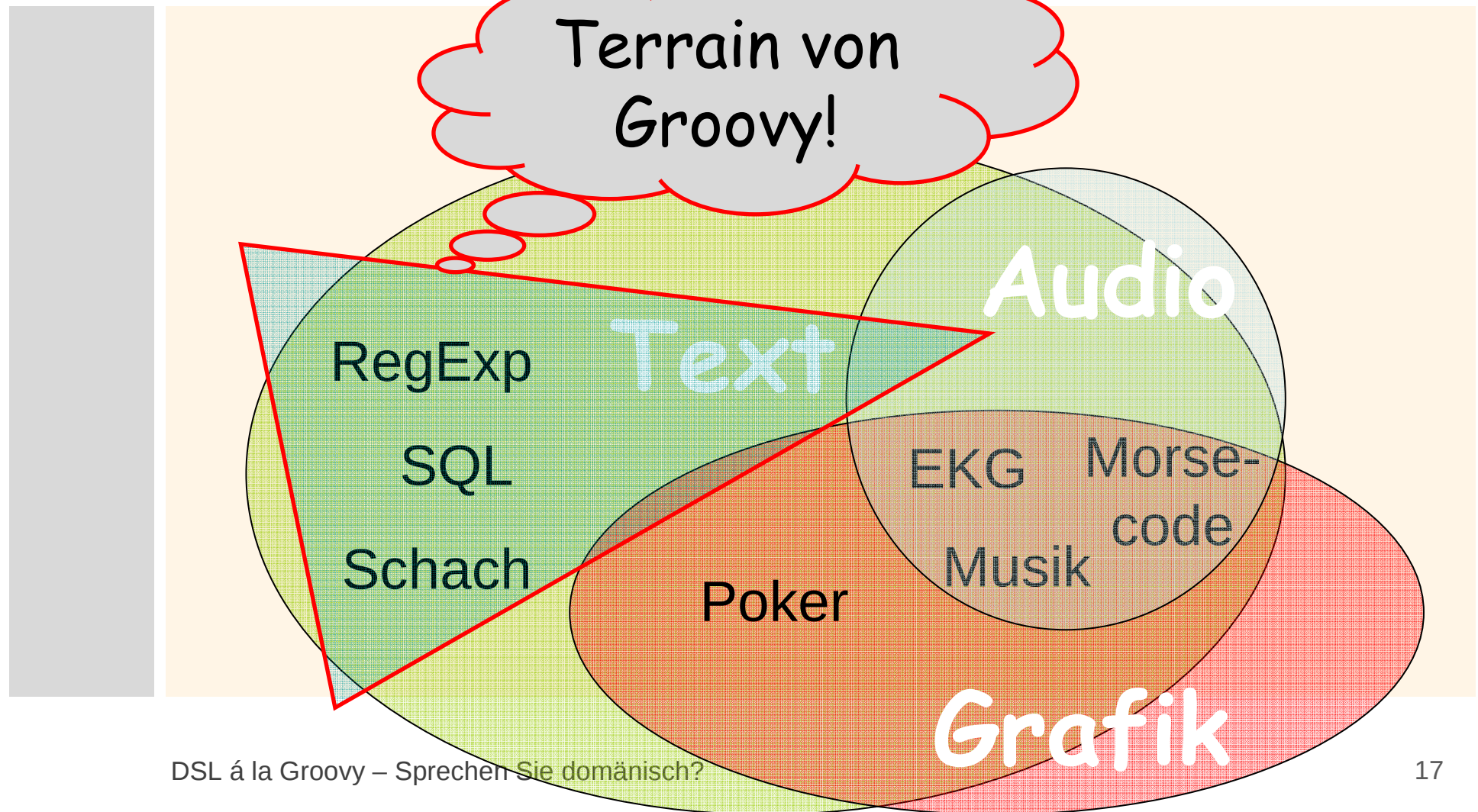
HAUPTSTRASSE 42 (go)

RegExp

Externe vs. Interne

- external DSL
 - yacc
 - lex
 - awk
 - ANTLR
- internal DSL
 - basiert auf General Purpose Language
 - „Jede iDSL ist in einer GPL ausführbar!“

Text vs. Grafik vs. Audio



DSL in Groovy in 8 Schritten (1/8)

```
new Meter(2000)
    .plus(
        new Kilometer(3))
```

DSL in Groovy in 8 Schritten (2/8)

```
new M(2000)
    .plus(
        new KM(3))
```

DSL in Groovy in 8 Schritten (3/8)

```
new m(2000)
    .plus(
        new km(3))
```

DSL in Groovy in 8 Schritten (4/8)

```
new Distance(2000)
    .getM().plus(
        new Distance(3)
            .getKm())
```

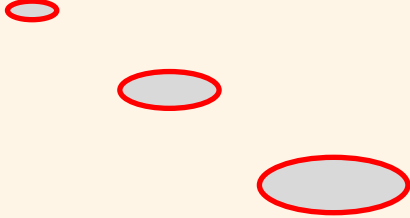
DSL in Groovy in 8 Schritten (5/8)

```
new Distance(2000).m  
    .plus(  
        new Distance(3).km)
```

Ab hier wird's Groovy:
Property-Notation

DSL in Groovy in 8 Schritten (6/8)

2000|m.plus(3|km)



Groovy-MOP:
z.B. Category!

DSL in Groovy in 8 Schritten (7/8)

2000.m + 3.km

Operator-Overloading
mit Groovy!

DSL in Groovy in 8 Schritten (8/8)

2000  m + 3  km

Kein Groovy:
Reguläre Ausdrücke!

Von Drinnen nach Draußen

2000m + 3km

External DSL

Internal DSL

2000m + 3km

Code zum ersten Schritt

```
assert new Meter(5000) ==  
    new Meter(2000).plus(  
        new Kilometer(3))  
  
class Kilometer {  
    def meter  
    Kilometer(kilometer){  
        meter = kilometer * 1000  
    }  
}
```

```
class Meter {  
    def meter  
    Meter(meter) {  
        this.meter = meter  
    }  
    def plus(kilometer) {  
        new Meter(  
            meter + kilometer.meter)  
        }  
    boolean equals(other) {  
        this.meter == other.meter  
    }  
}
```

Code zum zweiten/dritten Schritt

2. `new M(2000).plus(new KM(3))`

3. `new m(2000).plus(km(3))`

Sparen wir uns, oder?! ;-)

Code zum vierten Schritt

```
assert new Meter(5000) ==  
    new Distance(2000).getM()  
        .plus(  
            new Distance(3).getKm())
```

```
class Meter {  
    /* keine Änderungen*/  
}
```

```
class Kilometer {  
    /* keine Änderungen*/  
}
```

```
class Distance {  
    def distance  
    Distance(distance) {  
        this.distance = distance  
    }  
    def getM() {  
        new Meter(distance)  
    }  
    def getKm() {  
        new Kilometer(distance)  
    }  
}
```

Code zum fünften Schritt

```
5. new Distance(2000).m.plus(  
    new Distance(3).km)
```

Keine Änderung am Code
notwendig, da

Property-Notation

Code zum sechsten Schritt

```
use(Distance) {  
    assert 5000.m ==  
        2000.m.plus(3.km)  
}  
class Meter { /*...*/ }  
class Kilometer { /*...*/ }  
class Distance {  
    static def getM(distance) {  
        new Meter(distance)  
    }  
    static def getKm(distance) {  
        new Kilometer(distance)  
    }  
}
```

```
/* VORHER!! */  
class Distance {  
    def distance  
    Distance(distance) {  
        this.distance = distance  
    }  
    def getM() {  
        new Meter(distance)  
    }  
    def getKm() {  
        new Kilometer(distance)  
    }  
}
```

Code zum siebten Schritt

7. 2000.m + 3.km

Keine Änderung am Code
notwendig, da

Operator-Overloading

Code zum finalen achten Schritt

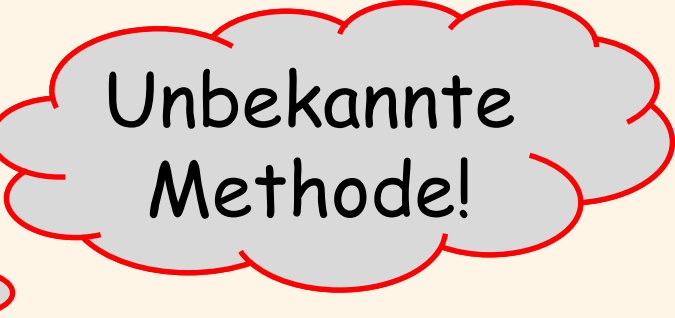
```
use(Distance) {  
    assert wertAus('5000 m == 2000 m + 3 km')  
}  
  
def wertAus(String anweisung) {  
    evaluate(  
        anweisung.replaceAll(/ (k*m)/, {  
            alle, eineit -> ".$${einheit}"  
        })  
    )  
}
```

MOP – Mein ominöses Programm?

- MOP heißt
 - Meta-Object Protocol
 - Meta(-Object) Programming
- Verarbeitet unbekannte Signale
 - Methoden
- In Groovy auf vielerlei Weise erreichbar
 - Category
 - invokeMethod()
 - Custom Metaclass
 - Injected Metaclass
 - ExpandoMetaclass

Beispiel: ExpandoMetaClass

```
groovy> class Stefan {}  
groovy> stefan = new Stefan()  
groovy> stefan.sagWas()
```



Unbekannte
Methode!

Exception thrown:

```
groovy.lang.MissingMethodException: No signature  
of method: Stefan.sagWas() is applicable for  
argument types: () values: {}
```

Beispiel: ExpandoMetaClass

```
groovy> class Stefan {}  
groovy> stefan = new Stefan()  
groovy> stefan.metaClass.sagWas =  
           {println 'Sorry, konnte heute leider nicht...'}  
groovy> stefan.sagWas()
```

Sorry, konnte heute leider nicht...

DSLs für Bäume

- Beispiele für Bäume
 - XML, HTML
 - Datenabfragen (SQL, HQL, ...)
 - Jason, YAML, ODGL
 - Swing
 - Ant
 - proprietär

Groovy-Bäume – noch ein Beispiel

```
<vorträge>
  <vortrag nummer="1" titel="Groovy - eine Einführung">
    <sprecher name="Joachim Baumann" />
  </vortrag>
  <vortrag nummer="2" titel="Groovy für Fortgeschrittene">
    <sprecher name="Dierk König" />
  </vortrag>
  <vortrag nummer="3" titel="GTDD">
    <sprecher name="Stefan Roock" />
    <sprecher name="Bernd Schiffer" />
  </vortrag>
</vorträge>
```

XML in Groovy-Notation

```
vorträge {  
    vortrag(nummer:1, titel: 'Groovy - eine Einführung') {  
        sprecher(name: 'Joachim Baumann')  
    }  
    vortrag(nummer:2, titel: 'Groovy für Fortgeschrittene') {  
        sprecher(name: 'Dierk König')  
    }  
    vortrag(nummer:3, titel: 'GTDD') {  
        sprecher(name: 'Stefan Roock')  
    }  
}
```

interne
DSL

Embedded internal DSL

```
dsl = '''vorträge {  
    vortrag(nummer:1,  
            titel:'Groovy - eine Einführung') {  
        sprecher(name:'Joachim Baumann')  
    }  
}'''  
  
writer = new StringWriter()  
builder = new groovy.xml.MarkupBuilder(writer)  
evaluate("builder.$dsl")  
result = writer.toString()
```

Kann aus
externem
File
geladen
werden!

Der Datenzugriff – Voll Groovy!

```
dsl = '''vorträge {  
    vortrag(nummer:1, titel:'Groovy - eine Einführung') {  
        sprecher(name:'Joachim Baumann')  
    }  
}'''
```

GPathResult

```
result = ... // vom MarkupBuilder bekommen  
vorträge = new XmlSlurper().parseText(result)  
assert 'Joachim Baumann' ==  
    vorträge.vortrag.sprecher.@name.text()
```

Voll Groovy in DSL – aber Vorsicht!

```
vorträge {  
    def alleTitel = ['Groovy - eine Einführung',  
                    'Groovy für Fortgeschrittene',  
                    'GTDD']  
    def alleSprecher = ['Joachim Baumann',  
                        'Dierk König',  
                        'Stefan Roock']  
    alleTitel.eachWithIndex{ titel, index ->  
        vortrag(nummer:index + 1, titel:titel) {  
            sprecher(name:alleSprecher[index])  
        }  
    }  
}}
```

Ob das
der Kunde
noch
versteht?

DSL: Kunde-Entwickler-Kanal

Domäne



Schlecht, wenn
der Kunde das
nicht mehr
versteht!

Entwickler



- e2 – e4
- e7 – e5
- Sg1 – f3
- Sb1 – c3
- Lg1 – h2

Kunde



DSL

DSL á la Groovy – Sprechen Sie domänisch?

Wie baue ich einen Builder?

- Implementiere `groovy.util.BuilderSupport` – und fertig!

```
class XBuilder extends BuilderSupport {  
    void setParent(parent, node) {}  
    def createNode(Object name) {}  
    def createNode(Object name, Object value) {}  
    def createNode(Object name, Map attributes) {}  
    def createNode(Object name, Map attributes, Object value) {}  
}
```

Andere Formate

Das Ziel

vorträge

```
vortrag["titel":"Groovy - eine Einführung", "nummer":1]  
sprecher["name":"Joachim Baumann"]
```

Andere Formate

```
class XBuilder extends BuilderSupport {  
    int level  
    StringBuffer markup =  
        new StringBuffer()  
    void setParent(parent, node) {}  
    void nodeCompleted(parent, node) {  
        level++  
    }  
    def createNode(Object name) {  
        write(name + attributes)  
    }  
    String toString() { markup }
```

```
def createNode(Object name,  
                Map attributes) {  
    write(name + attributes)  
}  
def createNode(Object name,  
                Map attributes,  
                Object value) {}  
def write(string) {  
    markup << indent() + string + '\n'  
    level++  
    markup  
}  
def indent() { (' ' * level) }
```

AntBuilder

```
def vortragende = ['Stefan', 'Joachim', 'Dierk']  
def ant = new AntBuilder()  
vortragende.each{  
    ant.mail(mailhost: 'my.email.server',  
            subject : 'build ist fertig'){  
        from(address: 'bernd@jfs.de')  
        to(address: "$it@jfs.de")  
        message("Hey $it, denk' an Deinen Vortrag heute! Gruß, Bernd!")  
    }  
}
```



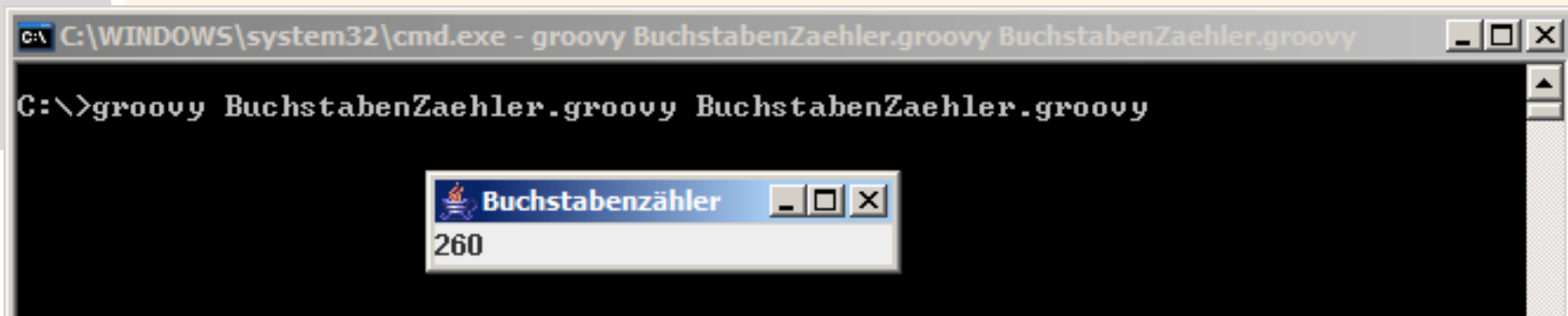
SwingBuilder

```
import groovy.swing.SwingBuilder

frame =
    new SwingBuilder().frame(
        title: 'Buchstabenzähler') {
    label = label(
        text: 'Buchstaben werden' +
            'gezählt...')
    }

frame.pack()
frame.show()

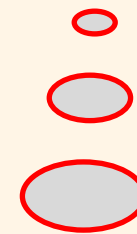
while(true) {
    label.text =
        new File(args[0]).text.size()
    sleep 200
}
```



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe - groovy BuchstabenZaehler.groovy BuchstabenZaehler.groovy". The command prompt displays the command "C:\>groovy BuchstabenZaehler.groovy BuchstabenZaehler.groovy". Below the command prompt, a Java Swing window titled "Buchstabenzähler" is visible, showing the number "260".

HibernateCriteriaBuilder

```
def c = Account.createCriteria()
def results = c {
    like("holderFirstName", "Fred%")
    and {
        between("balance", 500, 1000)
        eq("branch", "London")
    }
    maxResults(10)
    order("holderLastName", "desc")
}
```

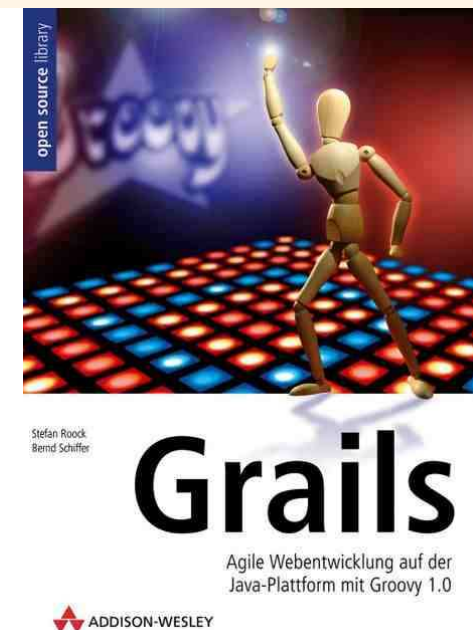
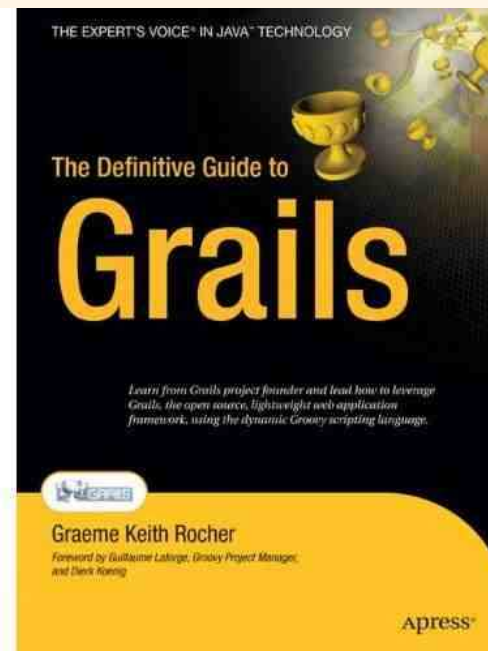
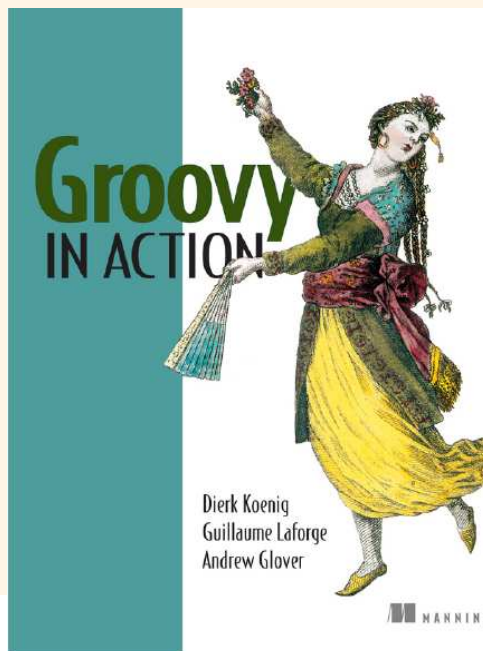


Fundgrube an
MOP- & DSL-
Beispielen!

Quelle: <http://grails.codehaus.org/Hibernate+Criteria+Builder>

Lust bekommen auf mehr?

- Groovy: <http://groovy.codehaus.org>
- Groovy-Mailinglisten: <http://groovy.codehaus.org/Mailing+Lists>
- Grails: <http://grails.org/>



+49 172 44 256 68

Vielen Dank für die
Aufmerksamkeit



Noch Fragen?



Schulung verlängerte Werkbank
agile Softwareentwicklung
Festpreisprojekte Coaching
RCP **Systemintegration** Eclipse
h3270 Hostintegration
Scrum Refactoring testgetriebene Entwicklung
Hibernate SAP-Netweaver **OpenSource**
Ajax JBoss/JEMS Groovy
it-agile eXtreme Programming