Orientation in Objects

Groovy und Grails
Quo vadis?

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.0

Über mich

Orientation in Objects

Falk Sippach

Trainer, Berater, Entwickler

Schwerpunkte
Architektur
Agile Softwareentwicklung
Codequalität

Java, XML und Open Source seit 1998

Orientation in Objects

) Software Factory)

- Schlüsselfertige Realisierung von Java Software
- Individualsoftware
- Pilot- und Migrationsprojekte
- Sanierung von Software
- Software Wartung

) Object Rangers)

- Unterstützung laufender Java Projekte
- Perfect Match
- Rent-a-team
- Coaching on the project
- Inhouse Outsourcing

) Competence Center)

- Schulungen, Coaching, Weiterbildungsberatung, Train & Solve-Programme
- Methoden, Standards und Tools für die Entwicklung von offenen, unternehmens-weiten Systemen

Abstract

Orientation in Objects

Das Jahr 2015 begann turbulent für die beiden bekanntesten Projekte aus dem Groovy Universum. Von der bisherigen "Mutter" Pivotal den Laufpass erhalten, musste sich Groovy auch noch auf die Suche nach einem neuen Zuhause begeben und ist letztlich bei Apache fündig geworden. All diese Unsicherheiten haben die neuen Features der Releases 2.4 (Groovy) bzw. 3.0 (Grails) ziemlich in den Hintergrund gedrängt. Dabei sind die Projekte lebendiger denn je und vor allem schon längst reif für den produktiven Einsatz.

Wir werden uns die wichtigsten und interessantesten Neuerungen der vergangenen Releases anschauen und natürlich auch einen Ausblick auf die Zukunft und Roadmaps wagen.

Gliederung

Orientation in Objects

- **Motivation + Politisches**
- Groovy
- Grails
- Ausblick

Bitte melden!

Orientation in Objects

Wer arbeitet mit Groovy/Grails?

regelmäßig/häufig

selten/früher mal

noch nie

Enthaltungen?

Warum Groovy und Grails?

Orientation in Objects

- Groovy-Fan seit 1.0 (2006)
- 5+ Jahre Projekterfahrung mit Grails

Aber es ist ruhig geworden …

Orientation in Objects

Gefühlt tendiert die öffentliche Aufmerksamkeit **gegen Null**!

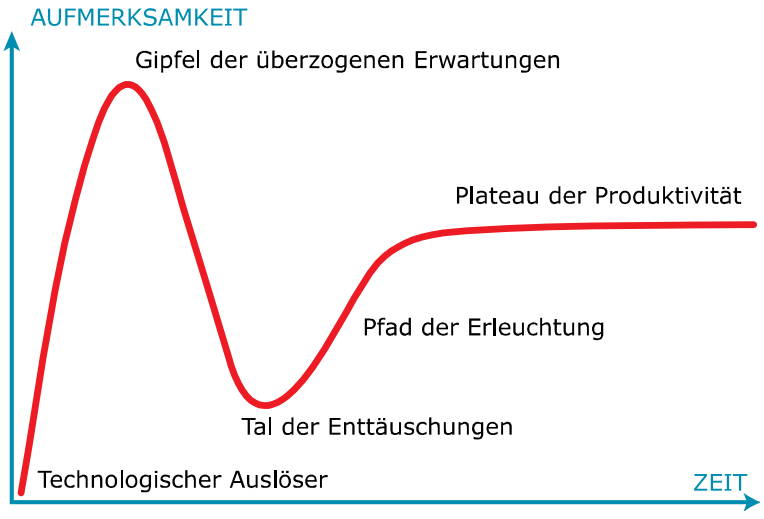
Zu alt?

Orientation in Objects

Sind Groovy und Grails **nicht mehr hip** genug?

Gardners Hype Kurve

Orientation in Objects



Gut Ding will Weile haben: Regina

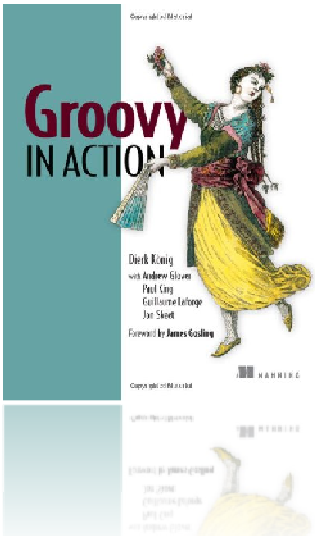
Orientation in Objects

Who the fuck is Regina?

Gehen wir zurück ins Jahr 2007

Orientation in Objects

GinA



Zwei Jahre später: viele Neuerungen mit 1.5 - 1.7

Orientation in Objects

Planung einer zweiten Auflage:

ReGinA war geboren

Und eine lange "Leidensgeschichte" begann …

Anfänglicher Optimismus

Orientation in Objects

Hi all,
announcing the start of MEAP for June (2009)…

Anyway, this summer is the time where we will do the majority of the work on the second edition and you can expect the MEAP progressing rather quickly. …

thanks for you understanding
Dierk

https://forums.manning.com/posts/list/18629.page

Twitter vergisst nichts: #regina #groovy(lang)

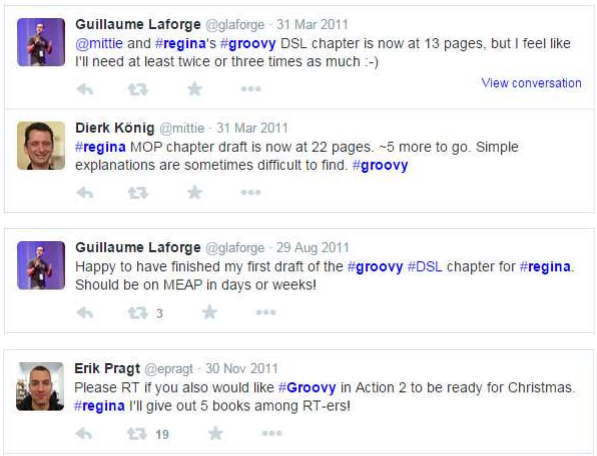
Orientation in Objects

2009/2010

Time flies by …

Orientation in Objects

2011



Passiert hier was?

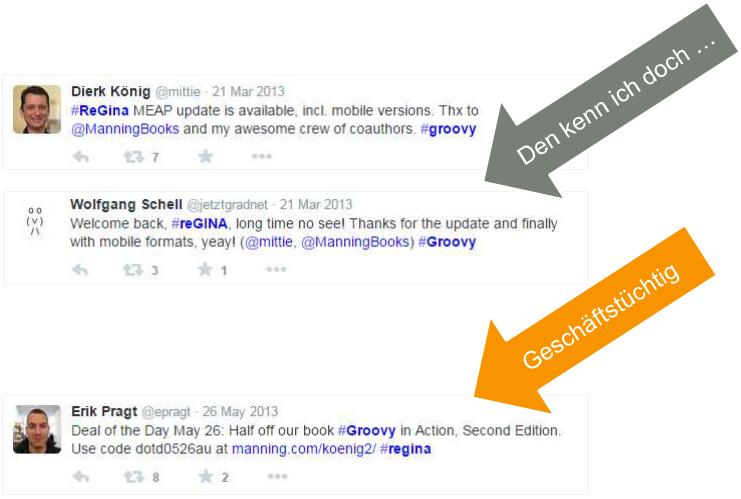
Orientation in Objects

2012

Da passiert was …

Orientation in Objects

2013



Weitere Schaffenspause ...

Orientation in Objects

2014

Hurra – endlich ist es da!

Orientation in Objects

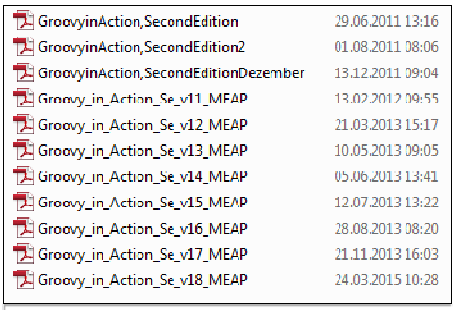
2015

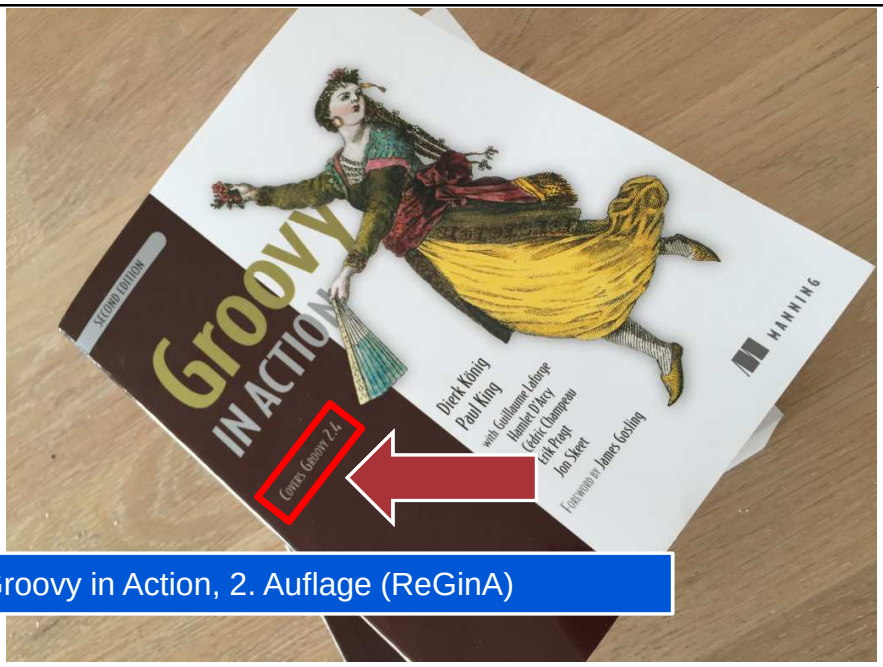
Selfies

Orientation in Objects

MEAP Auslieferungen

Orientation in Objects

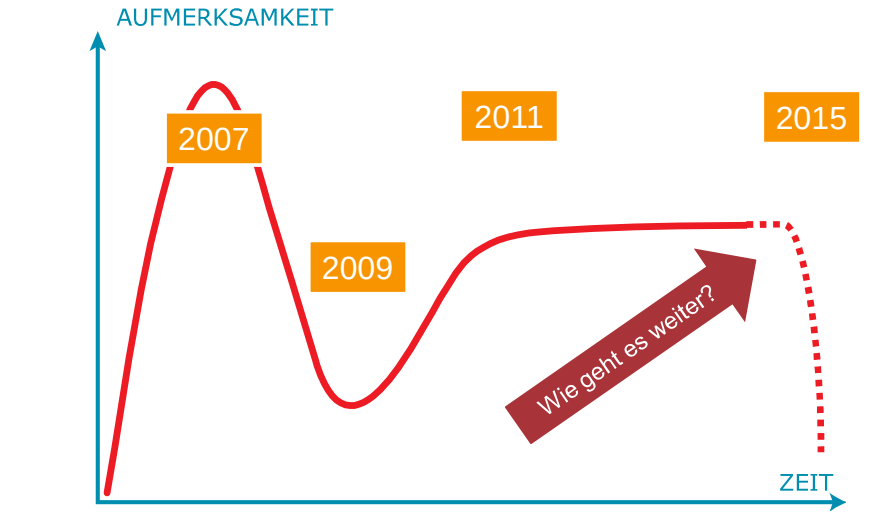




Groovy in Action, 2. Auflage (ReGinA)

Zeitliche Einordnung Groovy/Grails

Orientation in Objects



Historie

Orientation in Objects

	Groovy	Grails
2003	Projektstart	
2005		Projektstart
2006	1.0	
2007	G2One	
2008		1.0
2008	SpringSource	
2009	VMware	
2013	Pivotal	
2015	???	???

About being a paid OSS Developer for Groovy (1)

Orientation in Objects

Groovy is my Child in code you could say.

For over 10 years I helped the project. I was one of the first guys to be paid […] and did so for about 9 years.

[…] nice experience. I mean you do your job, you help the community and work on your hobby at the same time!

Ideal, or not?

Well, not ideal.

http://blackdragsview.blogspot.de/2015/04/about-being-paid-oss-developer-for.html

About being a paid OSS Developer for Groovy (2)

Orientation in Objects

G2One

[…] the workload was high. […] adding feature after feature without testing or documentation.

SpringSource

A few liked us, a few hated us, most ignored us. […] also no real plan of continuation.

http://blackdragsview.blogspot.de/2015/04/about-being-paid-oss-developer-for.html

About being a paid OSS Developer for Groovy (3)

Orientation in Objects

VMware

[…] somewhere deep down in the management chain before, now we have been even deeper down.

[…] pressure management as best […] finally got 2013 Cedric on board.

Pivotal

Still I stayed a bit more happy. […] being now Pivotal and them wanting to be so open-source-oriented […]

http://blackdragsview.blogspot.de/2015/04/about-being-paid-oss-developer-for.html

About being a paid OSS Developer for Groovy (4)

Orientation in Objects

Pivotal

Well, turns out Pivotal cares only about OSS if they can use it. […] seeing people only as cost factors and not as human beings.

I expect several more OSS related problems with Pivotal in the future.

So for the time being it looks like there is no chance for me getting fulltime on Groovy.

I dare the community to fill the gaps and show they can do serious stuff in their free time.

http://blackdragsview.blogspot.de/2015/04/about-being-paid-oss-developer-for.html

Was war passiert? (19.01.2015)

Orientation in Objects

Groovy 2.4 And Grails 3.0 To Be Last Major Releases Under Pivotal Sponsorship

Frist bis
31.03.2015

The decision … is part of Pivotal's larger strategy to concentrate resources on … its **growing traction in Platform-as-a-Service, Data, and Agile development**.

Pivotal has determined that the **time is right** to let further development …be led by **other interested parties** … who can best serve the goals ...

http://blog.pivotal.io/pivotal/news-2/groovy-2-4-and-grails-3-0-to-be-last-major-releases-under-pivotal-sponsorship

Pivotal versucht gut aus der Sache rauszukommen

Orientation in Objects

Folge des Vert.x Desasters?

http://www.heise.de/developer/meldung/VMware-beansprucht-Copyright-an-Vert-x-Projekt-1779511.html

Schonfrist von knapp 3 Monaten

Unterstützung bei Sponsorsuche

Hosting von grails.org auch nach 31.03.

Es fehlt was …

Orientation in Objects

https://twitter.com/starbuxman/status/593366929550643200?s=03

One third and one G-Teams

Orientation in Objects

Jeweils 3 Core Committer bei Pivotal

Orientation in Objects

Who is Groovy?

Orientation in Objects

- interessante Statistik der Committer
- insgesamt 100+ Committer seit 2003
- Sieger: **Paul King** (nicht bezahlt)

http://melix.github.io/blog/2015/02/who-is-groovy.html

Rückzug von Pivotal: Auswirkungen auf Tools

Orientation in Objects

- Groovy/Grails Toolsuite (Eclipse) eingestellt
  - weiterhin https://github.com/groovy/groovy-eclipse
  - Grails 3.0 benötigt kein spezielles Eclipse-Plugin mehr
- Gradle entwickelt jetzt eigenes Eclipse-Plugin

Codehaus schließt

Orientation in Objects

- gestartet 2003
- 2015 der Übermacht von Github und Co. gebeugt
  - http://www.codehaus.org/history/
- Groovy braucht einen neuen Hosting Service
  - Source-Repo sowieso schon bei Github
  - aber Jira, Homepage, Wiki, ...

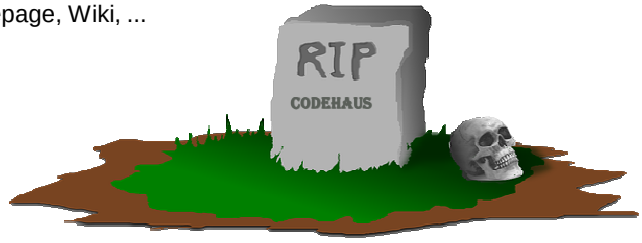


Foto von OpenClipartVectors, available under a CC0 Public Domain license.

Groovy goes Apache

Orientation in Objects

24.03.2015

Aufnahme im Inkubator

- **5 Mentoren**
- **5 initiale Committer**
- **neue Mailinglisten**
- **Jira-Tickets verschoben**
- **neues Git-Repo**
- **weitere Committer**

By kOchstudiO [Public domain], via Wikimedia Commons

Historie

	Groovy	Grails
2003	Projektstart	
2005		Projektstart
2006	1.0	
2007	G2One	
2008		1.0
2008	SpringSource	
2009	VMware	
2013	Pivotal	
2015	2.4 (Apache)	???

Grails has a New Home at OCI

OCI @ObjectComputing · 9. Apr.
We have announced some BIG news! @grailsframework and #Grails have a New Home at OCI!! interact.stltoday.com/pr/business/PR …

OCI

"With the support and backing of OCI, an established IT solutions engineering firm, with deep roots in Open Source, Grails is moving forward with maximum momentum."

Object Computing, Inc.

St. Louis, MO

http://www.ociweb.com

Historie

	Groovy	Grails
2003	Projektstart	
2005		Projektstart
2006	1.0	
2007	G2One	
2008		1.0
2008	SpringSource	
2009	VMware	
2013	Pivotal	
2015	2.4 (Apache)	3.0 (OCI)

Gliederung

- Motivation + Politisches
- **Groovy**
- Grails
- Ausblick

A Groovy kind of love!

Orientation in Objects

PS: I am happily married and NO, I do not have a lover!

I am married to Java, but i have a lover ... Groovy
Java has been there for me many years.
Solid, stable, reliable, mature, old ...
I know java well, i think she knows me too.
Then I met Groovy. Young, sexy, dynamic, agile ...

I am getting to know Groovy,
I am getting butterflies in my stomach again.
The best things is that they get along just fine.
They even talk together in harmony.
What else can I say?

http://www.jroller.com/orrego/entry/i_am_married_to_java

Glück gehabt ...

2003 Projektstart Scala

2008 Programming in Scala

Orientation in Objects

"I can honestly say if someone had shown me the **Programming in Scala** book by Martin Odersky, Lex Spoon & Bill Venners back in **2003** I'd probably have **never created Groovy**."

James Strachan

http://macstrac.blogspot.de/2009/04/scala-as-long-term-replacement-for.html

Dynamische Skriptsprache

Orientation in Objects

=

ausdrucksstarke Syntax

+

mächtige Bibliotheken

+

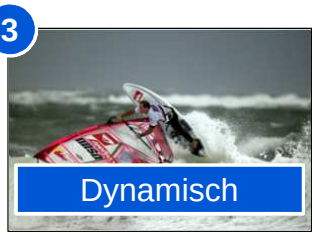
Meta-Programmierung

Hauptprinzipien von Groovy

Orientation in Objects

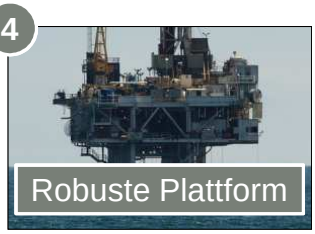
1

3



2

4



Featurereich

Orientation in Objects

- Closures (Lambda++), funktionale Programmierung **light**
- ausdrucksstarke Syntax (Collections, Reguläre Ausdrücke, GString, mehrzeilige Strings, …)
- Out-of-the-box Import/Export von XML, JSON, …
- AST-Transformationen
- Traits (Mehrfachvererbung)
- Skripting

1

Java-freundlich

Orientation in Objects

- Groovy kompiliert zu Java Bytecode
- flache Lernkurve
- leichte Migration (Groovy ist Superset der Java-Syntax)
- gegenseitig Klassen aufrufen, voneinander erben, …
- GDK (Erweiterung des JDK)
- Android-Anwendungen in Groovy schreiben

2

Dynamisch

- Property Support (getter/setter for free)
- Meta-Programmierung (Laufzeit-AOP)
- AST-Transformationen (Compilezeit-AOP)
- Builder (XML, JSON, Swing, …)
- Überladen von Operatoren
- Erstellen von DSLs

3

Foto von WikiImages, available under a CC0 Public Domain license.

Robuste Plattform

- kompiliert zu Bytecode, läuft auf der JVM
- JDK und Java-Bibliotheken nutzen
- gleiches Memory-, Threading-, Vererbungs- und Sicherheitsmodell
- Werkzeuge wiederverwendbar (Build-Tools, IDEs, Profiler, Debugger, …)

4

Foto von PublicDomainPictures, available under a CC0 Public Domain license.

Kritik an Groovy

Orientation in Objects

- ~~schwache~~ dynamische Typisierung
- fehlende Tool-Unterstützung (Refactoring)
- Fehler erst zur Laufzeit
- Performance
- public Fields

Dynamische Typisierung ist ein Feature

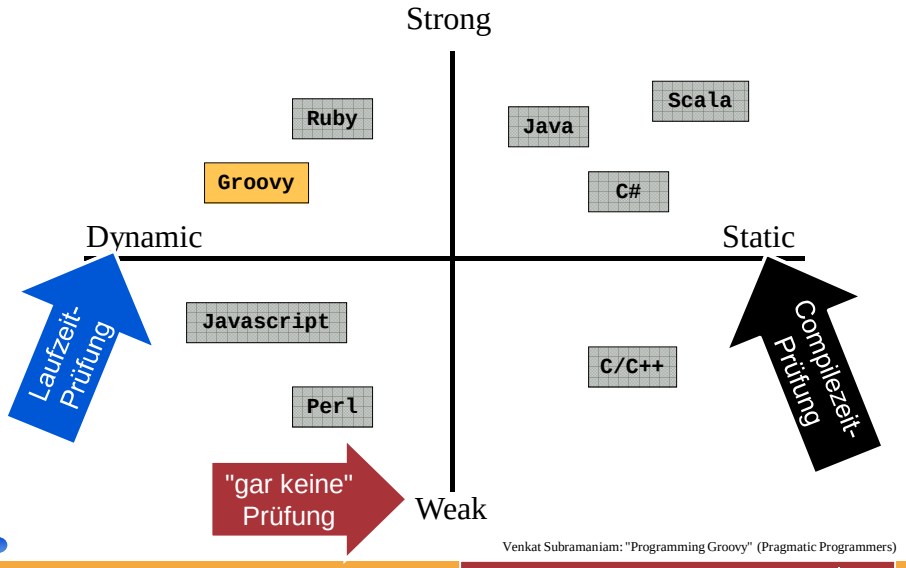
Orientation in Objects

- ermöglicht Runtime-Metaprogrammierung

Alternativen:

- @TypeChecked und @CompileStatic
- AST-Transformation: Compiletime-Metaprogrammierung
- Traits

Weakly != Dynamic Typing



Venkat Subramaniam: "Programming Groovy" (Pragmatic Programmers)

An die Kette nehmen - Statische Typüberprüfung

```
@TypeChecked
class MeineKlasse {
  def meineMethode1() {
     // nur statisch getypter Code erlaubt
  }

  @TypeChecked(TypeCheckingMode.SKIP)
  def meineMethode2() {
     // dynamisch getypter Code möglich
  }
}
```

Fehler erst zur Laufzeit

Orientation in Objects

- Erinnerung: dynamische Typisierung ist ein Feature für die Runtime-Metaprogrammierung
- Sicherheitsnetz durch Testen
  - agile Softwareentwicklung, TDD, BDD, …
  - sowieso notwendig fürs Refactoring
- Syntax eignet sich wunderbar zum Schreiben von Tests
- Mocking-Framework eingebaut
  - Unit-Tests einfach isolieren dank Metaprogrammierung
- DSLs: Spock, Geb, …

Performance: stetige Verbesserungen

Orientation in Objects

```
@TypeChecked
@CompileStatic
class MeineKlasse {
   [..]
}
```

	Fibonacci	Pi quadrature	Binary trees
Java	191 ms	97 ms	3.6 s
Groovy 2.x: Static Compilation	197 ms	101 ms	4.3 s
Groovy 1.8: Primitive optimizations	360 ms	111 ms	23,7 s
Groovy 1.7: No primitive optimizations	2590 ms	3220 ms	50 s

Public Fields

Orientation in Objects

- Lange Diskussion, bisher keine Lösung:
  - It's not a bug, it's a feature.
  - Groovy nutzt das intern für seine Unit-Tests

It's not clear if it is a bug or by design.
Groovy treats visibility keywords as a hint for a programmer.

- https://issues.apache.org/jira/browse/GROOVY-1875
- http://refaktor.blogspot.de/2012/07/private-fields-and-methods-are-not.html

Coole Groovy Features - Top 5

Orientation in Objects

Konzentration auf kleine, nützliche Funktionen

Keine Reihenfolge

Rein subjektiv

Warum nur 5?

Meine Top 5 Groovy Features

Orientation in Objects

1. Multiline Strings/GStrings
2. Elvis Operator
3. Objektnavigation/Dereferenzieren
4. XyzSlurper/Parser
5. Power Asserts

GStrings

Orientation in Objects

- können zusätzlich Variablen enthalten
- Lazy Evaluation


```
String s1 = "Das ist GString!"

String ersetzung = 'Ersetzungen'
String s2 = "Ein GString kann $ersetzung
  enthalten!"

println "Hallo ${user.name}!"
```

1

Slashy Syntax

Orientation in Objects

```
String s1 = /\d{1,2}\.\d{1,2}\.\d{2,4}/

java.util.regex.Pattern.compile(/\s\w+\s/)

def dir = /c:\Schulung\workspace/
```

1

Mehrzeilige Strings

Orientation in Objects

```
String s1 = '''Das ist
ein besonders
langer String!'''

String s2 = """Dieser GString
reicht ebenfalls
über mehrere Zeilen"""
```

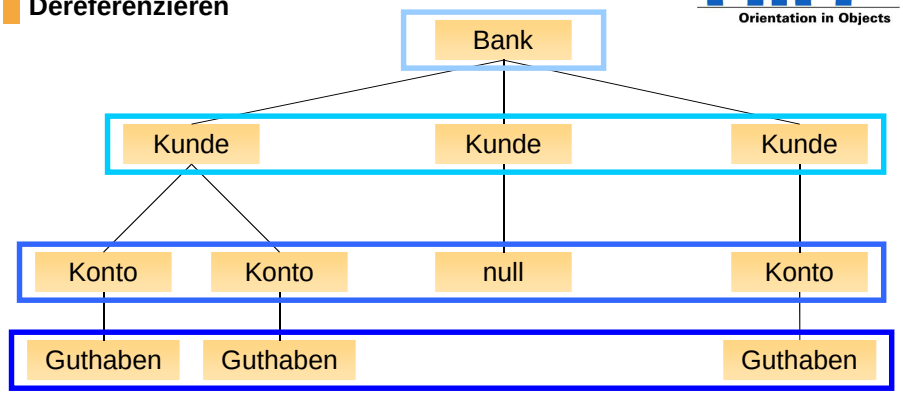
1

Von if/else zum ternären zum Elvis Operator

```
def username = getUsernameFromAnywhere()

if (username) { username } else { 'Unknown' }

username ? username : 'Unknown'

username ?: 'Unknown'
```

2

Effiziente Objektnavigation und sicheres Dereferenzieren



- Summe der Guthaben aller Konten der Bank:

  bank.kunden.konten?.guthaben.sum()

3

JSON einlesen

Orientation in Objects

```
def jsonSlurper = new JsonSlurper()

def object = jsonSlurper.parseText(
  '{ "name": "John Doe" } /* comment */')

assert object instanceof Map

assert object.name == 'John Doe'
```

4

JSON erzeugen

Orientation in Objects

```
class Author {
      String name
}
def authors = [new Author (name: "Guillaume"),
      new Author (name: "Jochen"),
      new Author (name: "Paul")]

def json = new JsonBuilder()
json authors, { Author author ->
     name author.name
}
assert json.toString() ==
  '[{"name":"Guillaume"},{"name":"Jochen"},{"name":"
  Paul"}]'
```

4

Java Asserts

Orientation in Objects

```
a = 10
b = 9

assert 91 == a * b


Exception thrown: Expression: (92 == (a * b)).
  Values: a = 10, b = 9

java.lang.AssertionError: Expression: (92 == (a *
  b)). Values: a = 10, b = 9
    at ConsoleScript2.run(ConsoleScript2:4)
```

5

Power Asserts

Orientation in Objects

```
a = 10
b = 9

assert 91 == a * b

Exception thrown
Assertion failed:
assert 91 == a * b
          |  | | |
          |  10| 9
          |    90
          false

    at ConsoleScript2.run(ConsoleScript2:4)
```

5

Coole Features von Groovy - viele, viele mehr ...

Orientation in Objects

- Multiple Assignments
- NullObjectPattern
- Spaceship-Operator
- diverse GDK-Methoden und -Operatoren
- diverse AST-Transformationen
- dynamische Method-Interception
- Default-Parameter in Methoden
- ....

- siehe: http://stackoverflow.com/questions/303512/hidden-features-of-groovy

Releases Groovy

Orientation in Objects

- …
- Januar 2013: 2.1
- November 2013: 2.2
- Mai 2014: 2.3
- Februar 2015: 2.4

Groovy Neuerungen

Orientation in Objects

- diverse neue AST-Transformationen (2.3, 2.4)
- Traits (Groovy 2.3)
- Android Support (Groovy 2.4)

AST Transformationen

Orientation in Objects

- "Build-in" Lombok
- erweiterbar durch eigene AST-Transformationen
- viele fertige
  - @Singleton, @Immutable, @Lazy
  - @Delegate, @Newify, @Category
  - @PackageScope
  - @Bindable, @Vetoable
  - …

AST Transformation @Builder

Orientation in Objects

```
@Builder
class Person {
    String firstName, lastName
    int age
}
def person = Person.builder()
                   .firstName("Dieter")
                   .lastName("Develop")
                   .age(21)
                   .build()

assert person.firstName == "Dieter"
assert person.lastName == "Develop"
```


Traits

Orientation in Objects

- Interfaces mit Default Implementierungen und Zustand

```
trait Fahrbar {
    int geschwindigkeit
    void fahren() {
        println "Fahren mit " +
            "${geschwindigkeit} km/h!"}
}

class Bobbycar implements Fahrbar {}
new Bobbycar(geschwindigkeit:100).fahren()
```

Abstrakte Methoden

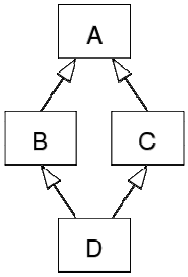
Orientation in Objects

```
trait Greetable {
  abstract String name()
  String greeting() {
    "Hello, ${name()}!"
  }
}

class Person implements Greetable {
   String name() { 'Duke' }
}

def p = new Person()
assert p.greeting() == 'Hello, Duke!'
```


Konflikte bei Mehrfachvererbung

Orientation in Objects

```
trait A {
    String exec() { 'A' }
}

trait B extends A {
    String exec() { 'B' }
}

trait C extends A {
    String exec() { 'C' }
}

class D implements B, C {}

def d = new D()
assert d.exec() == 'C'
```



Last wins!

Manuelles Auflösen Mehrfachvererbung

Orientation in Objects

```
class D implements B, C {
    String exec() { B.super.exec() }
}
def d = new D()
assert d.exec() == 'B'
```


Implementierung von Traits zur Laufzeit

Orientation in Objects

```
trait Extra {
    String extra() { "I'm an extra method" }
}
class Something {
    String doSomething() { 'Something' }
}

def s = new Something() as Extra
s.extra()
s.doSomething()
```

SAM Type Coercion

Orientation in Objects

```
trait Greeter {
    String greet() { "Hello $name" }
    abstract String getName()
}

Greeter greeter = { 'Alice' }
println greeter.greet()

// oder

void greet(Greeter g) { println g.greet() }
println greet { 'Alice' }
```

Single Abstract Method

Closure erzeugt SAM Type.

Bauen von Android Apps mit Groovy

Orientation in Objects

- schlanker
- prägnantere Syntax als Java
- SwissKnife-Bibliothek minimiert den Android-Boilerplate-Code
  - durch AST-Transformationen
  - z. B. Behandeln von UI-Events, Background-Threads

Gliederung

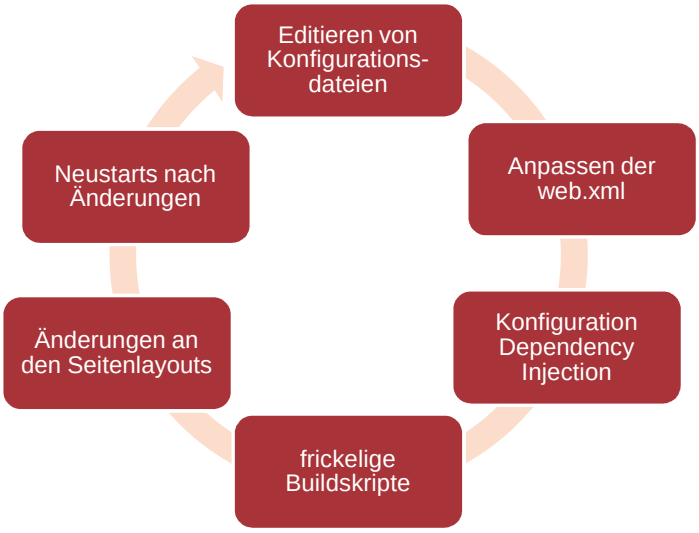
- Motivation + Politisches
- Groovy
- **Grails**
- Ausblick

Groovy auf Schienen

Aber mittlerweile eigenständiges, gestandenes Framework!

Inspiriert durch Ruby on Rails

Schmerzen mit klassischen Java Web Frameworks



Hauptprinzipien von Grails

Produktivität

Orientation in Objects

Convention over Configuration

Don't repeat yourself

Prototyping

1

Einfachheit

Orientation in Objects

- Groovy
- GORM – groovyfied Hibernate
- Spring (MVC) unter der Haube
- wohldefinierte Projektstruktur
- keine unnötige Konfiguration
- KISS (Design Komplexität)
- Vermeiden von Boilerplate Code

2

Erweiterbarkeit

- Grails modular aufgebaut, alles austauschbar
  - Hibernate/RDBMS ⇔ NoSQL
  - Tomcat ⇔ Jetty
- Funktionalität über Plugins hinzufügbar
  - Core-Plugins
  - Community-Plugins
- modularer Aufbau von Grails-Applikationen über Plugins
  - bessere Wiederverwendung

3

Stabile Basis

basiert auf Java EE Standards

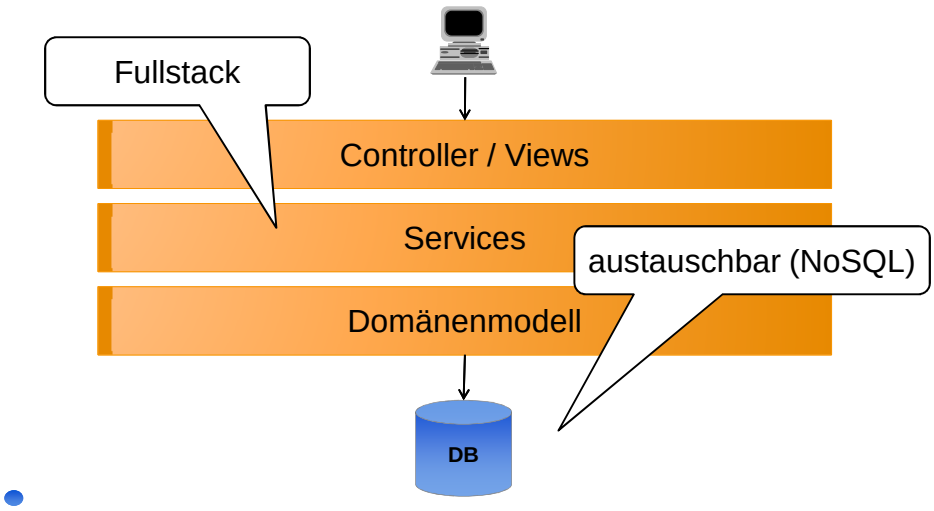
Integration von Spring, Hibernate, Sitemesh

Ausführungsumgebung dabei

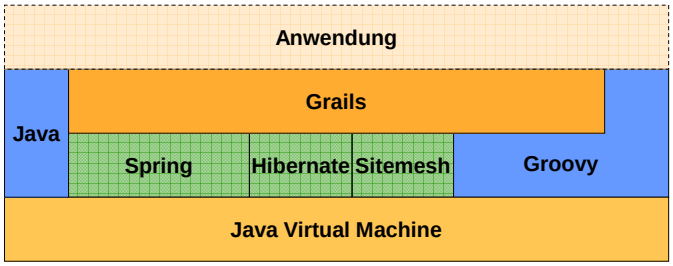
Fullstack

4

Grails Schichtenmodell



Technologien in Grails



Kritik an Grails

Orientation in Objects

- problematisch in großen Projekten
- Abwärtskompatibilität, hoher Aufwand bei Upgrades
- Aktualität der Plugins
- Stacktraces

Große Projekte

Orientation in Objects

- schwierig bei Misch-Masch von Java und Groovy-Entwicklern
- zwingend projektübergreifende Code-Conventions festlegen
  - werden Typen ausgeschrieben?
  - benutzen wir return
  - ...
- gute Testabdeckung!
  - sonst gibt es wenig Chancen auf Refactoring
- 80/20 Regel
  - 80 % funktionieren in Grails super (CRUD, ...)
  - 20 % sind umständlicher, aber nicht unlösbar

Aufwand bei Upgrades

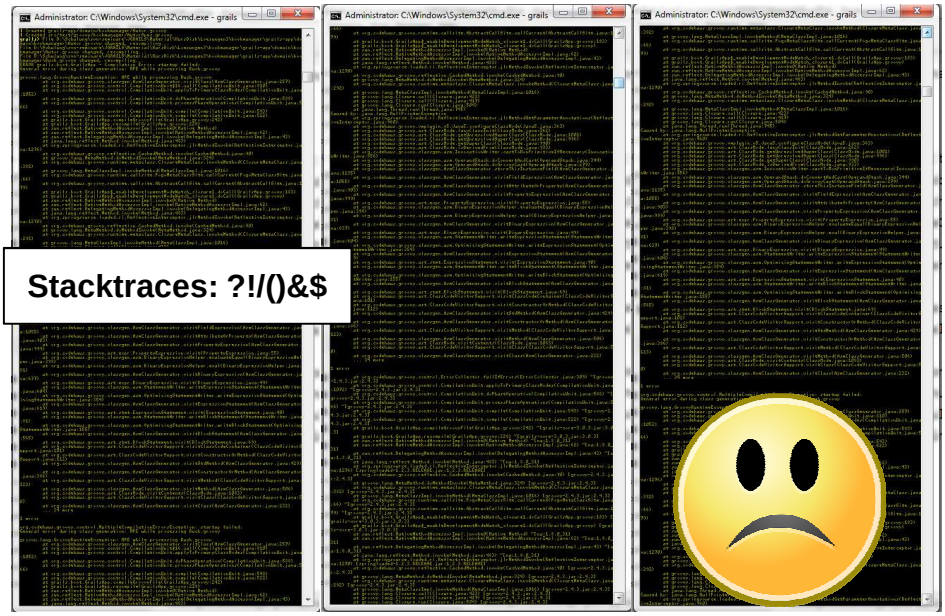
Orientation in Objects

- leider ja
- aber es gibt einen Migrationsguide in der Grails-Dokumentation
- möglichst alle neuen Release-Sprünge sofort upgraden, nicht zu lange warten

Aktualität der Plugins

Orientation in Objects

- Plugins der Core-Entwickler sind super
  - werden gewartet
  - bei neuen Grails-Versionen erfolgt schnell ein Upgrade
- Vorsicht bei kleinen Plugins ohne nennenswerte Aktivitäten mit vielen offenen Bugs, die vor 3 Jahren eingestellt wurden
- mit Grails 3 sind einige Plugins obsolet geworden



Meine Top 5 Grails Features

1. Automatische DI
2. Validation
3. Tag Libs
4. CRUD-Methoden
5. Where Queries

Foto von geralt, available under a CC0 Public Domain license.

Automatische Dependency Injection von …

- … Service in Services, Controller, TagLib, Domain-Klassen
- … Logger in Controller, Service, TagLib

- über Spring-Konfiguration manuell steuerbar

```
class BuchController {
      BuchService buchService
}

Class User {
      def securityService
      static transients = ['securityService']
}
```

Implizite Attribute im …

- … Controller
- ... View
- ... TagLib
- ... Service

```
class BuchController {
  String action() {
    log.info(params)
    servletContext.getResourceAsStream("..")
  }
}
```

Validierung

Orientation in Objects

```
class User {
    ...

    static constraints = {
        login size: 5..15, blank: false, unique: true
        password size: 5..15, blank: false
        email email: true, blank: false
        age min: 18
    }
}
```

2

Übersicht Validatoren

Orientation in Objects

Name	Beschreibung
blank	String darf nicht leer ein
email	String muss eine syntaktisch korrekte Email-Adresse sein
inList	Wert muss in angegebener Collection enthalten sein
matches	String muss auf regulären Ausdruck matchen
max	Wert darf Maximalwert nicht überschreiten
maxSize	size des Werts darf Maximalgröße nicht überschreiten
nullable	Wert darf null sein. **Default: false**
range	Wert muss ist in angegebener Range enthalten sein
size	size des Werts muss ist in angegebener Range enthalten sein
unique	Wert muss in Datenbank einzigartig sein
url	String muss eine syntaktisch korrekte URL sein
validator	Eigene Validierungslogik anwenden

2

Beispiel für eigenen Validator

Orientation in Objects

```
class KalenderEintrag {
   Date von
   Date bis

   static constraints = {
       von(nullable: true)
       bis(nullable: true, validator: bisDatumValidator)
   }

   private static Closure bisDatumValidator = { val, obj ->
       return (!obj.von || val?.after(obj.von))
   }
}
```

2

Dynamic Tag Libraries

Orientation in Objects

```
class MyTagLib {
   static namespace = "oio"

   def greeting = {attrs, body ->
       out << "Hallo ${attrs.name}"
   }

   def isAdmin = { attrs, body ->
       def user = attrs['user']
       if(user != null && checkUserPrivs(user)) {
             out << body()
       }
   }
}
```

3

Dynamic Tag Libraries – Beispiel in GSP

Orientation in Objects

```
<oio:greeting name="Fritz"/>

<oio:isAdmin user="${myUser}">
  // Nur fuer Admin sichtbar
</oio:isAdmin>
```

3

CRUD-Methoden von Domänenklassen

Orientation in Objects

- Dynamisch hinzugefügte Methoden für CRUD-Operationen
  - get(id) lädt ein Objekt aus der DB
  - list() lädt alle/viele Objekte aus der DB
  - save() speichert das Objekt in die DB (initial oder update)
  - delete() löscht ein Objekt aus der DB
  - exists(id) prüft, ob Objekt existiert
  - count() zählt die Objekte in der DB

4

CRUD-Methoden von Domänenklassen - Beispiel

Orientation in Objects

```
Buch buch = new Buch(titel:'Groovy')
buch.save()
assert 1 == Buch.count()

buch.titel = 'Groovy in Action'
buch.save()

buch.delete()
assert !Buch.exists(buch.id)
```

4

Where Queries

Orientation in Objects

- Setzt auf Detached Criteria auf
- Compile-time checked query DSL
- „Entwickler-freundliche“ Syntax
- IDE support / code completion
- Groovy Operatoren werden auf Restrictions gemappt

```
def query = Buch.where {
   titel == "Grails in Action"
}
Buch grailsInAction = query.find()
```

5

Gegenüberstellung
Criteria Queries vs. Where Queries

Orientation in Objects

```
// select * from Buch where titel like '%grails%'

// Criteria Builder
Buch.createCriteria().list {
  ilike('titel', '%grails%,)
  authors {
      ilike('nachname', 'r%')
  }
}

// Where Query
Book.findAll {
  title =~ '%grails%' && authors.lastName =~ 'r%'
}
```

5

Gegenüberstellung
Criteria Queries vs. Where Queries II

Orientation in Objects

```
// Criteria Builder
Book.createCriteria().list {
  ilike('title', '%grails%')
  maxResults(1)
  firstResult(1)
  order('title', 'asc')
}

// Where Query
Book.findAll(max:1,offset:1,sort:'title',order:'asc'){
  title =~ '%grails%'
}
```

5

Releases Grails

- …
- Dezember 2012: 2.2
- September 2013: 2.3
- Mai 2014: 2.4
- März 2015: 2.5, 3.0

Grails Neuerungen

- Basis Spring Boot (3.0)
- Interceptor API (3.0)
- Anwendungsprofile (3.0)
- Gradle als Buildsystem (3.0)
- API Redesign mit Traits (3.0)

Basis Spring Boot

Orientation in Objects

- Spring 4.1
- Spring Boot 1.2

- Grails-Projekte enthalten nun eine Klasse mit main-Methode
  - lauffähige JAR-Datei (eingebetter Jetty, Tomcat, Undertow)
  - ohne Container start-/debugbar
  - keine spezielle IDE-Unterstützung mehr notwendig

Interceptor API löst Grails Filter ab

Orientation in Objects

- Eigener Interceptor implementiert Interceptor Trait

- 3 Methoden: before, after, afterView
  - vor der Controller-Action
  - nach Aufruf der Action
  - nach dem die View gerendert wurde

- Convention over Configuration: Namenskonvention
  - BookInterceptor für BookController

Applikationsprofile

Orientation in Objects

- ähnlich Java EE Profilen (Web, Full, ...)
- Profil kapselt die Anwendungsstruktur
  - Kommandos
  - Plugins
  - Skeletons, Templates
  - Ressourcen

```
grails create-app myapp --profile=web-plugin
```

- Default: Web-Profil
  - Projektstruktur für Webanwendung

Applikationsprofile

Orientation in Objects

- Verwaltung in einem Repository (USER_HOME/.grails/repository)
- Profil = Verzeichnis mit folgender Struktur

```
web
    * commands
        * create-controller.yml
        * run-app.groovy
        …
    * skeleton
        * grails-app
            * controllers
            …
        * build.gradle
    * templates
        * artifacts
            * Controller.groovy
    * profile.yml
```

Gradle als Build-System

Orientation in Objects

- Build-Management-Integration eines Grails-Projekts in eine Projektlandschaft war ziemliche Qual
- properitäres und fehleranfälliges Gant ist Geschichte
- Ivy ist Geschichte (eigener Dependency Resolver)
- keine IDE mit speziellen Grails-Plugins mehr nötig
  - nur Gradle-Support notwendig
  - theoretisch reichen die Commandline + Sublime/Atom/Vi/Emacs/...

API-Redesign mit Traits

Orientation in Objects

- Grails Metaprogrammierung hat auch manche Probleme gemacht
- Traits werden vom Compiler verarbeitet (keine Runtime-Metaprogrammierung)
- vieles der Grails-Magie mittlerweile über Traits => Stabilität
  - Domain-Klassen
  - Controller
  - Services
  - TagLibs
- Schnittstellen aufgeräumt
  - öffentliche unter grails.*
  - interne jetzt unter org.grails.*

Gliederung

- Motivation + Politisches
- Groovy
- Grails
- **Ausblick**

Orientation in Objects

Aussage vom Groovy-Projektleiter

https://speakerdeck.com/glaforge/groovy-state-of-the-union-gr8conf-europe-2015

Groovy ist z. B. geeignet für …

Orientation in Objects

- Scripting (Shell-Skripte, Jira/Confluence Script Runner, …)
- Schreiben von Tests (Spock, Geb, …)
- Buildmanagement (Gradle)
- Admin-Konsole in Java EE Anwendungen
- und natürlich in Kombination mit Grails

Grails ist z. B. geeignet für …

Orientation in Objects

- Prototypen
- kleine Intranet-Anwendungen (keine Bange, falls es wächst)
- Microservices

Großes Ecosystem

Große Community

Konferenzen

- GR8Conf Europe (Kopenhagen) + GR8Conf US
- Greach (Spanien)

Podcasts

- http://groovypodcast.podbean.com/

Weekly Newsletter

- http://www.groovy-lang.org/groovy-weekly.html

Stackoverflow und aktive Mailinglisten

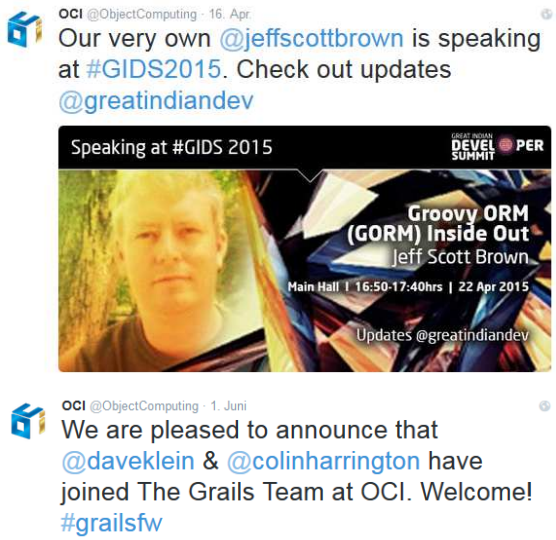
22,350
questions tagged
grails

Synonyms
groovygrails

Roadmap Grails

Roadmap Grails - Last update

OCI gibt Gas



Sponsor und Auftritte auf Konferenzen

Anstellung von weiteren Leuten

Ab auf die Überholspur …

Sie suchen weiter Leute

And the winner is Grails …

Orientation in Objects

Studie von ZeroTurnaround/RebelLabs von 2013/2014 ?

IN THE END
GRAILS & VAADIN
ARE THE BIG WINNERS,
FOLLOWED REASONABLY CLOSELY BY GWT, JSF, AND PLAY.
THERE YOU HAVE IT. NOW YOU CAN GO HOME ;-)

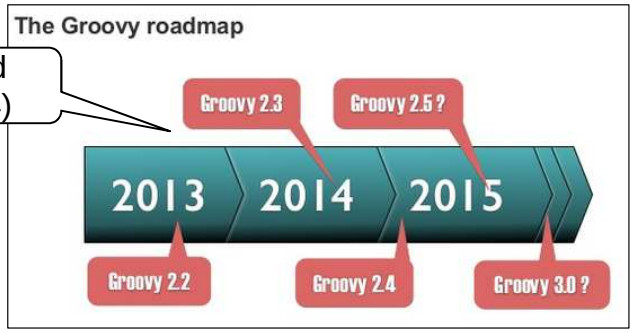
http://de.slideshare.net/hamedhatami2012/curious-coders-java-web-frameworks-comparison

Roadmap Groovy

Orientation in Objects

- neues Meta-Object Protokoll
- Laufzeit auf Basis von Invoke Dynamic
- Sprachgrammatik neu in Antlr v4

Plan vor Abschied von Pivotal (2014)



http://de.slideshare.net/SpringCentral/groovy-in-2014andbeyond

Prioritäten haben sich geändert

Orientation in Objects

erstes Release nach Apache Richtlinien

- **2.4.4 vom 16.07.2015** 

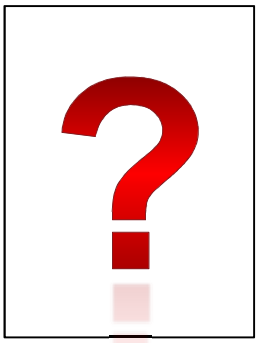
(enthält wichtiges Sicherheitsupdate)

kleinere Änderungen für 2.5 geplant
- **AST-Transformationen**
- **GDK-Verbesserungen**

Was passiert mit größeren Vorhaben?

Orientation in Objects

Rewrite MOP in 3.0

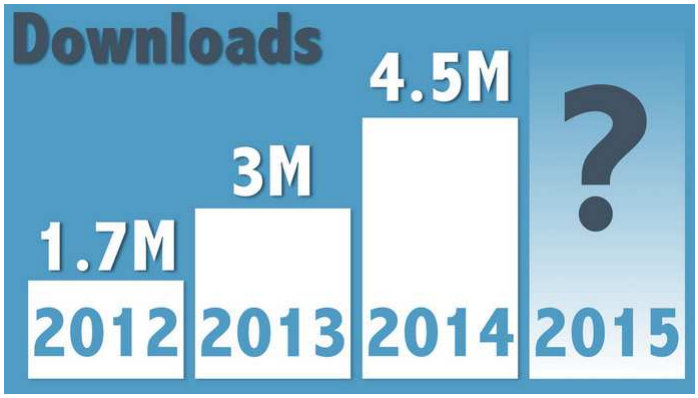


Können sich diese Firmen irren?

Orientation in Objects

http://groovy-lang.org/

Zudem steigende ...



https://speakerdeck.com/glaforge/groovys-history-and-current-status

... Downloadzahlen

- nur Maven Central
- es fehlen Codehaus + Bintray

2.1 MILLIONS
FOR FIRST 4 MONTHS
OF 2015

https://speakerdeck.com/glaforge/groovys-history-and-current-status

Explodierende Downloadzahlen

Orientation in Objects

Orientation in Objects

Fragen ?

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Orientation in Objects

Vielen Dank für ihre Aufmerksamkeit !

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de