Grails im Überblick und in der Praxis

JUG Karlsruhe - 10.04.2013

Darius Henkelmann - Tobias Kraft

Wer sind wir ?

- Tobias Kraft
  - Senior Consultant bei der exensio GmbH
  - Java Enterprise Technologien und Grails-Ökosystem
  - Grails seit 2010
- Darius Henkelmann
  - Senior Application Engineer bei der Unic GmbH
  - Java Enterprise, Softwarearchitektur, Ecommerce, Hybris und Webtechnologien
  - Grails 2007 – Schulungszentrum und Portierung Frontend Ecommerce-System auf Basis von Hybris

Ziele des Vortrags

- Was ist Grails und was sind die Ziele?
- Auf welchen Technologien baut Grails auf?
- Welches Programmiermodell bietet Grails?
- Wie sind unsere Erfahrungen mit Grails in Projekten?

Auf was werden wir nicht eingehen

- Groovy
- Jedes Detail
- Fehler und Probleme aus anderen Projekten

Wir laden zur weiteren Diskussion am Stammtisch ein!

Agenda

- Einführung in Grails
- Wichtige Grails-Prinzipien
- Integration mit Java
- Grails in der Praxis

Zielgruppe / Einsatzgebiete von Grails

- Java-Entwickler
- Webentwickler ohne Java-Vorkenntnisse
- Newbies und Seniors

- Geeignet von Kleinprojekten bis zu größeren Enterprise-Projekten
- Erstellung Prototypen

Grails Rückblick

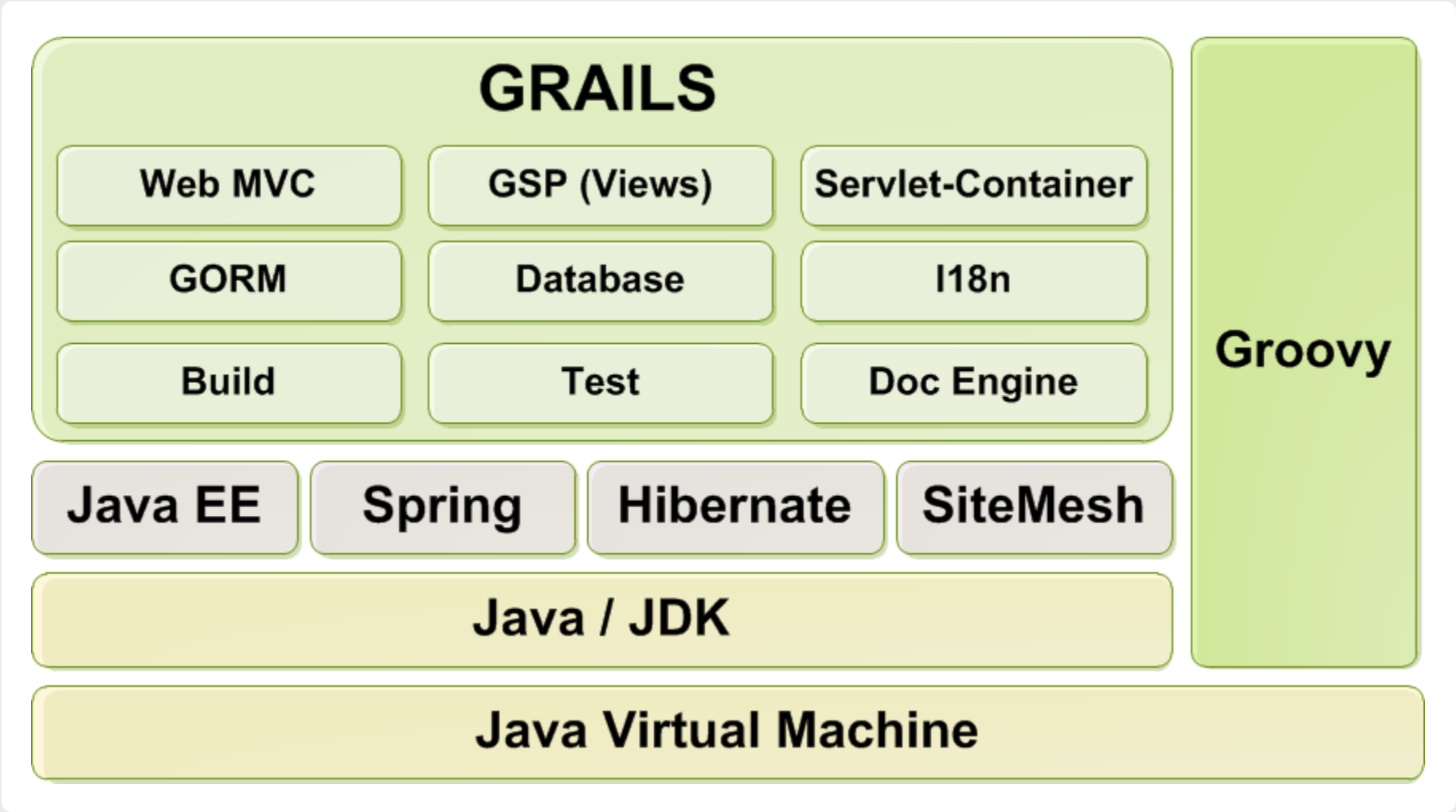
- Juli 2005 - Projektbeginn mit Guillaume LaForge, Steven Devijver und Graeme Rocher
- G2One, November 2008 SpringSource, VMWare
- Grails 0.5 (05.2007) - URL Mappings, Command objects
- Grails 1.0 (02.2008) - ORM DSL, JNDI Data Sources
- Grails 1.3 (05.2010) - Groovy 1.7
- Grails 1.4 (05.2011) - Spring 3.1, Groovy 1.8, Servlet 3.0
- Grails 2.0 (12.2011) - Interactive Mode, Reloading Agent
- Grails 2.1 (05.2012) - Maven Optimierung, Grails Wrapper
- Grails 2.2 (12.2012) - Groovy 2.0

Was ist Grails?

A Full-Stack Web (MVC) Framework [2]

- Install & GO!
- Nahtlose Java Integration, wird auf der Java Virtual Machine (JVM) ausgeführt
- Rapid development
  - Use a Java-like dynamic language (Groovy)
  - Convention over Configuration
  - Don’t repeat Yourself (DRY)
  - Inspired by Ruby on Rails and others

Grails Aufbau [3]



Installation

- Voraussetzung: Java (JDK) 1.6 oder höher
- Download Grails (http://grails.org/download)
- Umgebungsvariablen JAVA_HOME und PATH setzen
- Terminal öffnen

```
tokr@ex-vi-tk-04 /home $ grails -version

Grails version: 2.2.1
tokr@ex-vi-tk-04 /home $
```

- Alternative: Installation mit GVM (http://gvmtool.net)

```
tokr@ex-vi-tk-04 /home $ curl -s get.gvmtool.net | bash

tokr@ex-vi-tk-04 /home $ gvm install grails
```

Kommandos

grails <Befehl>	Muster
grails help <Befehl>	Zeigt die Hilfe für Grails oder für ein Befehl an
create-app	Erzeugt ein neues Grails Projekt.
create-controller <Name>	Erzeugt einen Controller
create-domain-class <Name>	Erzeugt eine Domain-Klasse (Model)
generate-controller <Name>	Generiert einen Controller für ein Model
generate-views <Name>	Generiert die CRUD-Views für ein Model
create-service <Name>	Erzeugt eine Service-Klasse
create-unit-test <Name>	Erzeugt eine Unit-Test-Klasse
run-app	Startet die Anwendung

Demo-Time

```
tokr@ex-vi-tk-04 /tmp $  grails create-app TwitterDemo
| Created Grails Application at /tmp/TwitterDemo
tokr@ex-vi-tk-04 /tmp $  cd TwitterDemo
tokr@ex-vi-tk-04 /tmp/TwitterDemo $  grails
| Enter a script name to run. Use TAB for completion:
grails>  create-domain-class de.juka.Tweet
| Created file grails-app/domain/de/juka/Tweet.groovy
| Created file test/unit/de/juka/TweetTests.groovy
grails> open grails-app/domain/de/juka/Tweet.groovy
grails> generate-all de.juka.Tweet
| Finished generation for domain class de.juka.Tweet
grails> run-app
| Running Grails application
```

IDE Unterstützung

- Eclipse, STS
- NetBeans
- TextMate
  - grails integrate-with --textmate
- IntelliJ IDEA
  - Ultimate Edition – Voller Support für Grails
    - File -> Import Project-> Create project from existing sources
  - Community Edition
    - Unterstützung von Grails jedoch nicht mit alle Features (GSP-Editor, Classpath-Management, Quick-Acces)
  - grails integrate-with --intellij

Aufbau eines Grails-Projektes

Verzeichnis	Inhalt
grails-app	grails-spezifischer Teil der Anwendung
lib	zusätzliche Java-Archive, JDBC-Treiber
scripts	Gant Skripte
src	Groovy / Java Klassen
target	Build-Verzeichnis
test	Unit- und Integrationstests
web-app	web-app – Root-Verzeichnis für die Anwendung
web-app/WEB-INF	J2EE Metainformationen web.xml/classes
web-app/css	CSS-Dateien für die Anwendung
web-app/images	Bildateien für die Anwendung
web-app/js	JavaScript Dateien für die Anwendung

Aufbau grails-app Verzeichnis

grails-app	
conf	Konfiguration und Startprogramme
conf/hibernate	Eigene Hibernate-Mappings
conf/spring	Spring Konfiguration und Mapping-Dateien
controllers	Die Controller der Anwendung
domain	Die Models (Domain-Klassen) der Anwendung
I18n	Property-Dateien mit Übersetzungen
services	Klassen für Business-Logik
taglib	Eigene tag-Libraries für Grails
utils	Grails spezifische Einstellungen - Codec-Klassen
views	Die Views der Anwendung
views/layout	Die Sitemesh-Layouts (z.B. main.gsp)

Agenda

- Einführung in Grails
- Wichtige Grails-Prinzipien
- Integration mit Java
- Grails in der Praxis

DRY und CoC

Der Standard-Weg ist konfiguriert, aber Änderungen sind möglich

Beispiele:

- Aufbau Projekt-Struktur
- DB-Mapping
- Klassennamen
- Logging
- Autowiring

GORM – DB-Mapping

- Hibernate ist der Standard
- Austausch Persistenzschicht ist möglich

- DB-Operationen erfolgen direkt am Domain-Objekt

```
Tweet.delete()
Tweet.save(flush:true)
```

- Constraints und Mapping in static-Blöcken

```
static constraints = {
    login size: 5..15, blank: false, unique: true
    email (email: true, blank: false, nullable: true)
}
```

GORM – Mehrere Möglichkeiten zum Erstellen von Queries

Dynamic Finders

```
Tweet.findAllByAuthorAndLocation(owner, ‘Karlsruhe’)
```

Where-Queries

```
Tweet.where{ year(published) == 2013 && message!= null}
```

Criteria-Queries

```
Tweet.createCriteria().list(max:20) {
    author { eq('account', 'grails') }
    order('created', 'desc')
}
```

HQL- und Native-Queries

```
User.findAll('from User as u where t.account = ? ', ['grails'])
```

Services

- SpringBeans als einfache POJOs
- Einfacher Zugriff über DI

```
def tweetService
```

- Management von Transaktionen

```
static transactional = false

@Transactional(propagation = Propagation.REQUIRES_NEW)
def someMethod() {…}
```

- Möglichkeit des Scopings
  - singleton (default), request, session, flash, flow, prototype, conversation

Controller und Views

- Controller
  - Direkter Zugriff auf Services
  - Standardmethoden wie render
- Views über Groovy Server Pages (GSP)
  - Ähnlich JSP’s
  - Zusätzliche Tagslibs und Variablen
  - Model-Zugriff
- SiteMesh für Layouting

Controller-Überblick

```groovy
class TweetController {

    def update(Long id, Long version) {
        def tweetInstance = Tweet.get(id)
        if (!tweetInstance) {
            flash.message=message(code:'default.not.found.message',args:['Tweet',id])
            redirect(action: "list")
            return
        }

        ...
        tweetInstance.properties = params

        if (!tweetInstance.save(flush: true)) {
            render(view: "edit", model: [tweetInstance: tweetInstance])
            return
        }

        flash.message=message(code:'default.updated.message',args:['Tweet',tweetInstance.id])
        redirect(action: "show", id: tweetInstance.id)
    }
```

Flash Scope

Binding

Validierung Constraints

Standard-Controller-Methoden

Controller-Überblick

```groovy
class TweetController {

    def update(Long id, Long version) {
        def tweetInstance = Tweet.get(id)
        if (!tweetInstance) {
            flash.message=message(code:'default.not.found.message',args:['Tweet',id])
            redirect(action: "list")
```

Flash Scope

```
    <g:if test="${flash.message}">
        <div class="message" role="status">${flash.message}</div>
    </g:if>
    <ol class="property-list tweet">
        <g:if test="${tweetInstance?.message}">
            <li class="fieldcontain">
                <span id="message-label" class="property-label">Message</span>
                <span class="property-value" aria-labelledby="message-label">
                    <g:fieldValue bean="${tweetInstance}" field="message"/>
                </span>
            </li>
        </g:if>
```

```
    }
```

GRAILS

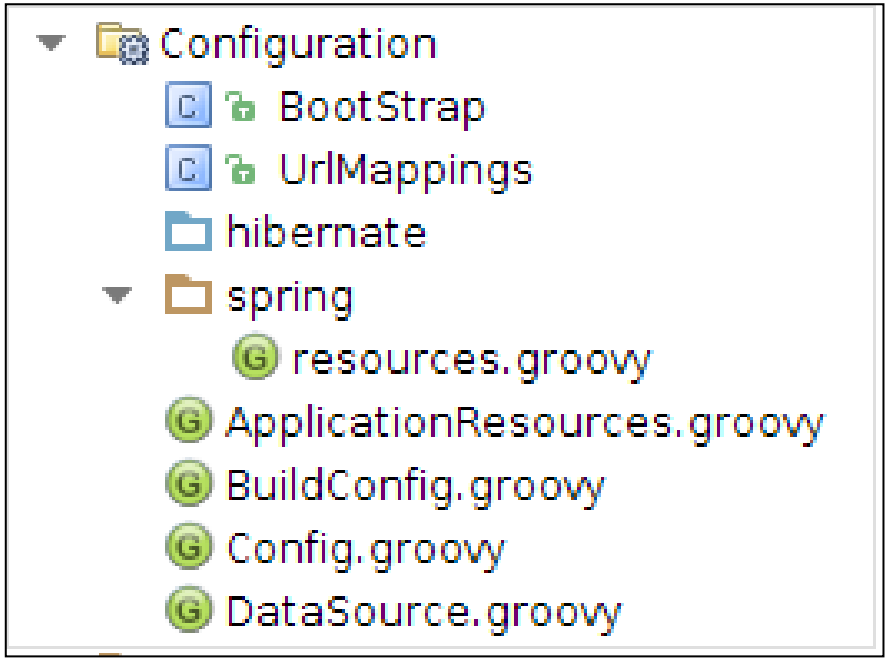
Taglibs

- Einfache Klasse mit Namensendung „TagLib“
- Kann auf Templates zum Rendern verweisen

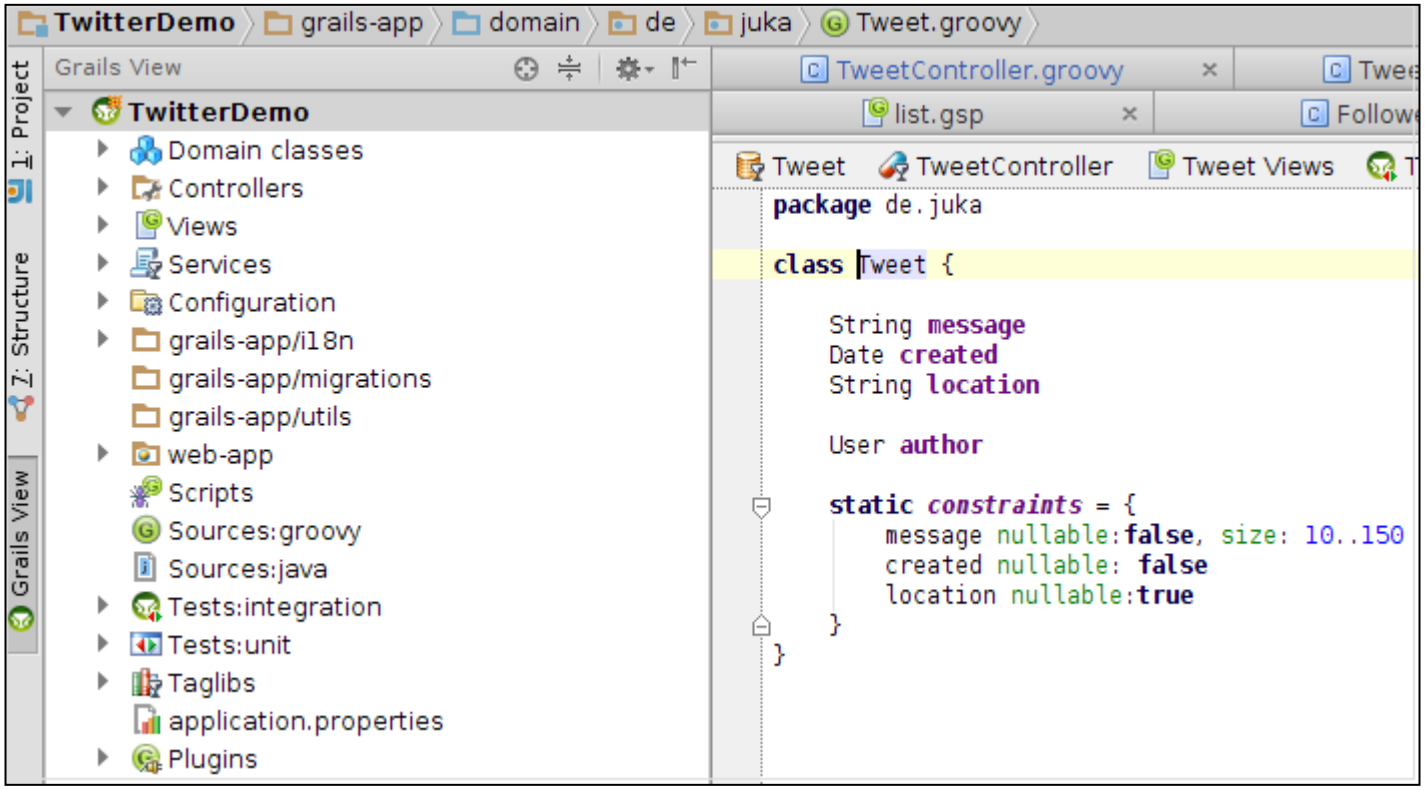
```
class ClaretPortletTagLib {
      static namespace = "claret"

      def portlet = { attrs, body ->
        def title = attrs.remove('title')
        out << render(template: '/claret/portlet', model:[title:title,body:body()])
      }
}
```

Konfiguration

- Konfiguration von
  - DB
  - Spring
  - Build / Dependencies
  - URL's
  - Filter
  - …
- umgebungsabhängig
  - dev, prod, test, …



Demo-Time



GRAILS

Lifecycle / Build

- Turnaround-Zeiten minimiert
- Automatisches Reloading für:
  - Controller, Views, TagLibs, Domain-Klassen
  - Services auch mit typisierter Referenzen
  - Klassen: src/groovy, src/Java
- Interactive Mode und Integrationstest
- Restart und clean selten notwendig

- WAR-File Generierung und Ausführung

Plugins - Einführung

Ein Grails Plugin ist ein in sich abgeschlossenes Bundle von Funktionalitäten.

- Kern-Plugins: hibernate, resources, jquery
- Zusätzliche Plugins http://grails.org/plugins

```
G:\grails_jug\TwitterDemo>grails list-plugins
| Environment set to development.....

Plugins available in the grailsCentral repository are listed below:
------------------------------------------------------------
DjangoTemplates Plugin<>                --
None                <>                  --
acegi               <0.5.3.2>           --  Acegi Plugin
activemq            <0.4.1>             --  Grails ActiveMQ Plugin
G:\grails_jug\TwitterDemo>grails install-plugin mongodb
| Environment set to development.....
| Warning The install-plugin command is deprecated and may b
sed in grails-app/conf/BuildConfig.groovy.  See http://grail
| Plugin installed.
```

- Plugins können selbst entwickelt werden
- Binary Plugins – Kommerzielle Plugins

Wichtige Plugins

- Resources
  - Optimiert für JavaScript
  - Optimiert CSS-Dateien
  - Zipping und Caching
  - Minimization und Bundling
- Spring Security Core
- Searchable
- ElasticSearch
- Cache
- Database Migration
- Database Reverse Engineering
- MongoDB GORM
- Quartz
- Spock
- Geb
- Code Coverage

Testing

- Tests werden bei jedem create-* Kommandos erzeugt
- grails create-controller Simple
  - SimpleController.groovy
  - SimpleControllerTests.groovy

```
G:\grails_jug\TwitterDemo>grails test-app
| Compiling 120 source files

| Running 9 unit tests... 4 of 9
```

```
| Completed 9 unit tests, 5 failed in 4781ms
| Compiling 1 source files.....
| Tests FAILED  - view reports in G:\grails_jug\TwitterDemo\target\test-reports
```

```
G:\grails_jug\TwitterDemo>grails
| Enter a script name to run. Use TAB for completion:
grails> open test-report
```

Testing ab Grails 2.0

- Keine Basis-Testklassen: GrailsUnitTestCase
- Keine doppelte Vererbung: Einsatz von Spock nicht möglich: spock.lang.Specification
- Unit Test Mixin: grails.test.mixin.TestFor , Mock

```
@TestFor(BookController)
@Mock([Book, Author, BookService])
```

- Mock und Stubs für manuelles Mocking

```
void testSearch() {
      def control = mockFor(SearchService)
      control.demand.searchWeb { String q -> ['mock results'] }
      control.demand.static.logResults { List results ->  }
      controller.searchService = control.createMock()
      controller.search()

      assert controller.response.text.contains "Found 1 results"
}
```

Testing Controller

```
1  package hellograils
2
3  import grails.test.mixin.*
4
5  @TestFor(HelloController)
6  class HelloControllerTests {
7
8      void testHello() {
9          controller.index()
10          assert response.text == 'Hello Grails!'
11     }
12 }
```

Testing mit Spock

- Testframework für Groovy und Java
- Ausdrucksstarke DSL für Spezifizierung der Tests
- Unterstützung von
  - Data Driven Testing
  - Interactions Based Testing

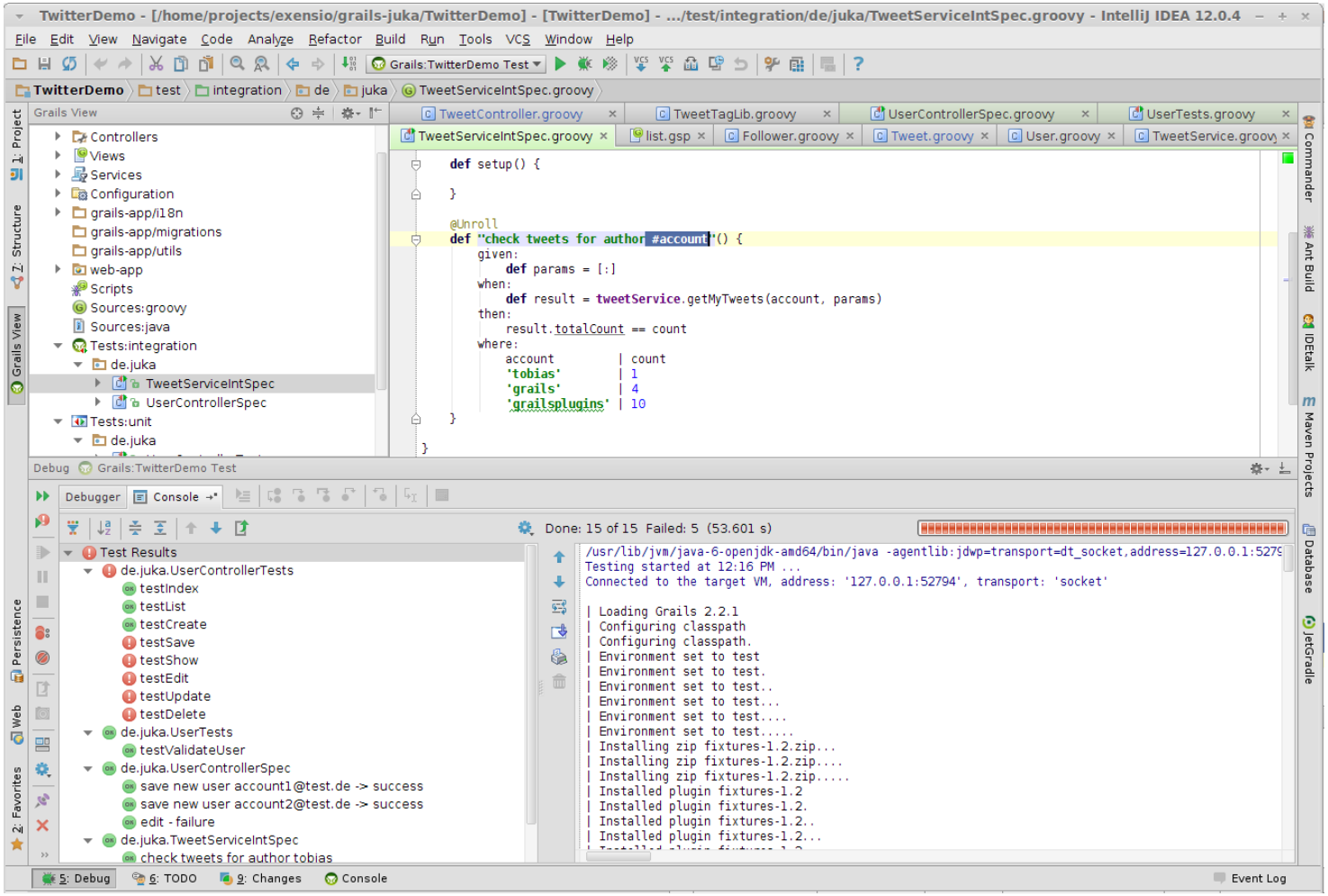
```
(1..3) * tweetService.message("hello")
```

- Ausprobieren mit Spock Web Console: http://meetspock.appspot.com/

Testing mit Spock

```
1   package de.juka
2   import grails.test.mixin.*
3
4   @TestFor(PersonController)
5   class PersonControllerSpec extends spock.lang.Specification {
6       def "Index action should redirect to list page"() {
7
8           when: "The index action is hit"
9           controller.index()
10
11          then: "The user should be redirected to the list action"
12          response.redirectUrl.endsWith "list"
13      }
14  }
```

Demo-Time



GRAILS

Agenda

- Einführung in Grails
- Wichtige Grails-Prinzipien
- Integration mit Java
- Grails in der Praxis

Integration mit Java

- Java Klassen
  - Klassen in src/java ablegen
  - JAR erzeugen und im Lib-Verzeichnis ablegen
  - Lokales Maven Repository und Abhängigkeit mit Ivy
- Spring Beans
  - grails-app/conf/spring/resources.groovy
  - Konfiguration mit Spring DSL (Closure in Property)
  - Konfiguration mit XML - resources.xml
- EJB-Integration
- Legacy DB
- JNDI

Spring Bean DSL

```
beans = {
    switch(Environment.current) {
        case Environment.PRODUCTION:
            myBean(MyBeanImpl) {
                someProperty = 42
                otherProperty = "blue"
            }
            break

        case Environment.DEVELOPMENT:
            myBean(MockImpl) {
                someProperty = 42
                otherProperty = "blue"
            }
            break
    }
}
```

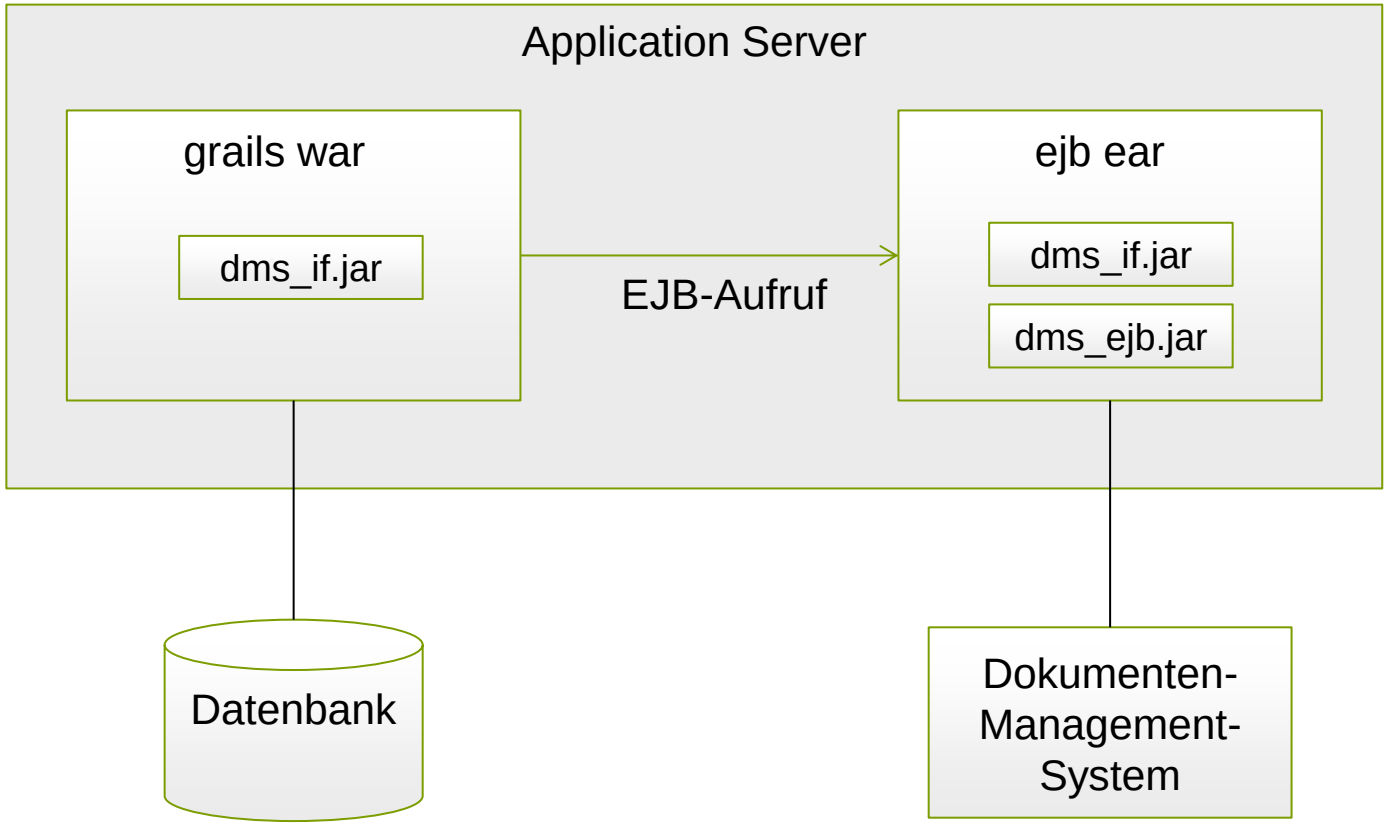
```
class ExampleController {

    def myBean
    ""
}
```

Spring Bean XML

```xml
<bean id="myBean" class="my.company.MyBeanImpl">
    <property name="someProperty" value="42" />
    <property name="otherProperty" value="blue" />
    <property name="bookService" ref="bookService" />
</bean>
```

```
package my.company;

class MyBeanImpl {

    private BookService bookService;
    private Integer someProperty;
    private String otherProperty;

    public void setBookService(BookService theBookService) {
        this.bookService = theBookService;
    }

    public void setSomeProperty(Integer someProperty) {
        this.someProperty = someProperty;
    }

    public void setOtherProperty(String otherProperty) {
        this.otherProperty = otherProperty;
    }
}
```

Integration Enterprise Java Beans



Integration Enterprise Java Beans

resources.groovy

```groovy
beans = {
    ejbJndi(org.springframework.jndi.JndiTemplate) {
        environment = [
        "java.naming.factory.initial":"weblogic.jndi.WLInitialContextFactory",
        "java.naming.provider.url":"t3://localhost:7001"
        ]
    }

    dctmService(org.springframework.jndi.JndiObjectFactoryBean) {
        lookupOnStartup = false
        jndiName = "DCTMService#de.exensio.dctm.facade.DCTMService"
        jndiTemplate = ref("ejbJndi")
        proxyInterface = de.exensio.dctm.facade.DCTMService.class as Class
    }
```

Aufruf im Service

```groovy
class DocumentumTransferBatchService extends BatchService {
    def dctmService

    def testCall() {
        def serviceName = dctmService.getServiceName()
        log.info("serviceName: ${serviceName}")
        return serviceName
    }
```

GRAILS

Legacy DB einbinden mit Mapping über Hibernate XML

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- Example mapping file inclusion -->
        <mapping resource="org.example.Book.hbm.xml"/>
        ...
    </session-factory>
</hibernate-configuration>
```

Mapping Domain-Klassen:
grails-app/conf/hibernate/hibernate.cfg.xml

Legacy DB einbinden mit Mapping über Hibernate Annotations

```
<!DOCTYPE hibernate-configuration SYSTEM
  "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <mapping package="com.books" />
        <mapping class="com.books.Book" />
    </session-factory>
</hibernate-configuration>
```

- Klasse in der SessionFactory registrieren: grails-app/conf/hibernate/hibernate.cfg.xml
- Alle dynamischen Methoden von Gorm verwendbar
- Constraints src/java/com/books/BookConstraints.groovy

Nutzung von JNDI-Resourcen

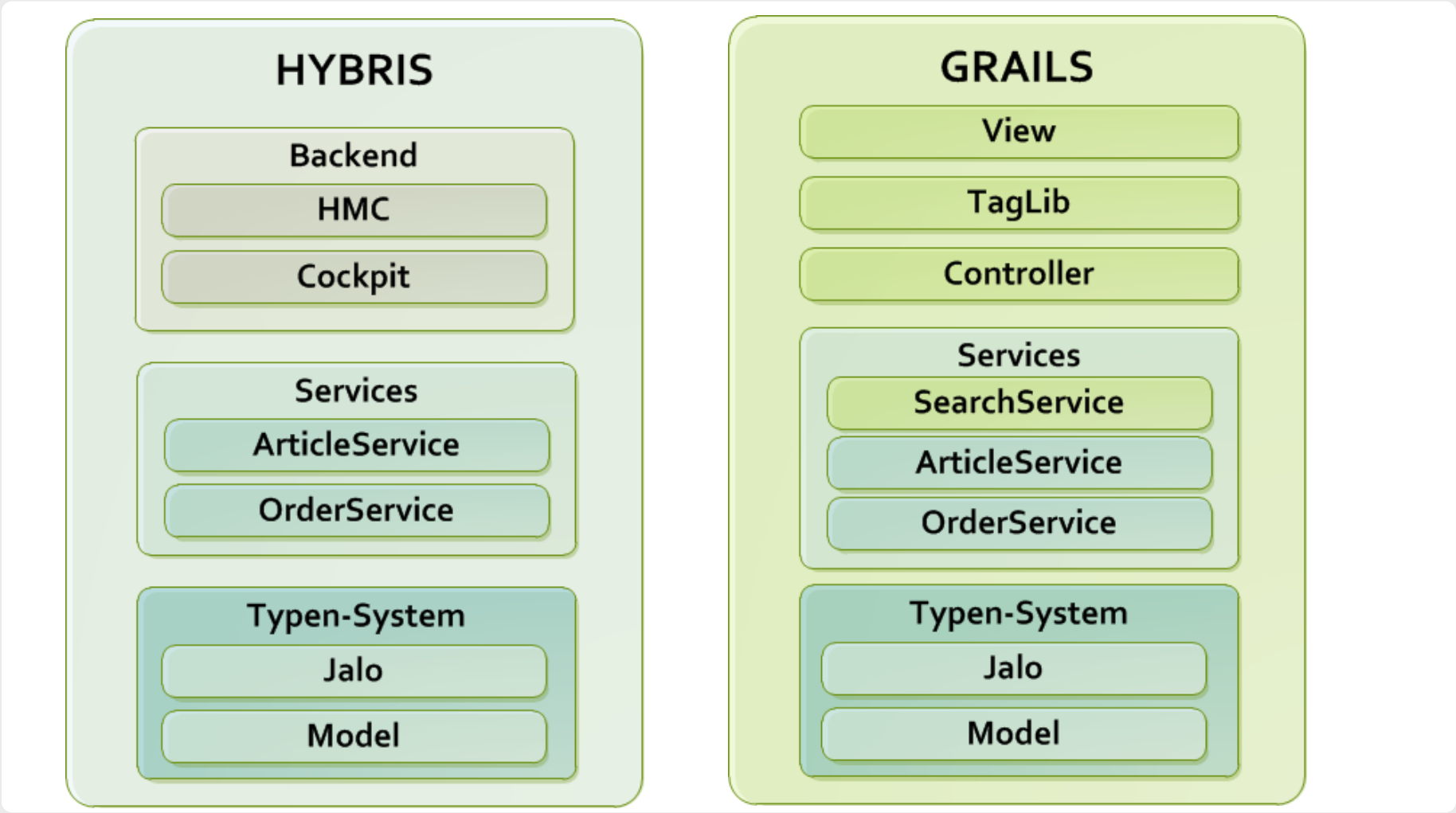
- DB-Anbindung

```
dataSource {
    jndiName = ' TW_DEMO_DS '
    dialect = 'org.hibernate.dialect.Oracle10gDialect'
    driverClassName = 'oracle.jdbc.OracleDriver'
    dbCreate = "create-drop"
}
```

- Mail-Service

```
grails {
    mail {
        jndiName = "TW_DEMO_MAIL_SESSION"
        disabled = false
    }
    mail.default.from="noReply@twitter-demo.de"
}
```

Integration mit Hybris



Fazit

- Java Know-How und Code Basis kann vollständig weiter verwendet werden
- Alle API‘s konnten integriert werden
- Installation mit anderen Spring Anwendungen mit Tomcat ohne Probleme möglich.
- Probleme mit älteren Server Oracle OC4J wegen unterschiedlichen API Version

Agenda

- Einführung in Grails
- Wichtige Grails-Prinzipien
- Integration mit Java
- Grails in der Praxis

In der Praxis hat sich gezeigt

- Know-How Bedarf bei Administration, Monitoring, Deployment und Infrastruktur bleibt gleich
- Interactive Mode rocks !
- Standard Vorgehen bei Grails-Entwicklung
  - Domain-Driven-Design
  - Prüfen, ob es ein Plugin gibt
  - DB-Modus create und Daten über Fixtures laden

Fallstricke in Grails

- Produzieren von unsauberem Code
  - Schichtentrennung
  - Schwergewichtige Taglibs
- Zu viel Logik in Controllern
- Vernachlässigung der Tests (Nicht typisiert)
- Dokumentation
- Viele Plugins existieren, aber nicht alle sind gepflegt und fehlerfrei
- DB-Änderungen

Was ist cool an Grails

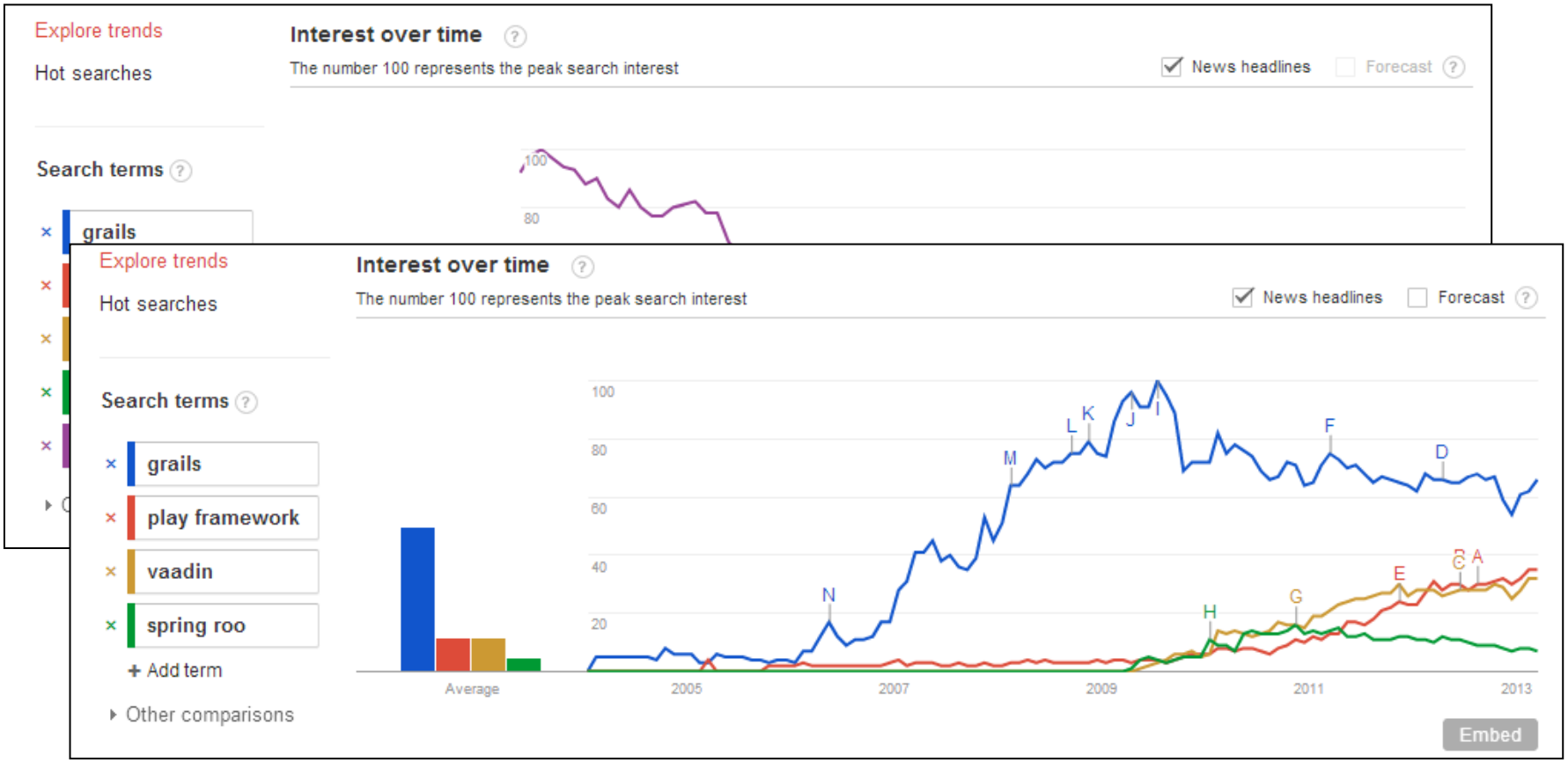
- Runterladen und loslegen
- Scaffolding
- Plugins wie Resources und Security steigern die Produktivität
- Full-Stack Unterstützung Continous Integration
  - Testing
  - Coverage
  - Build

Wer setzt Grails ein?

- http://grails.org/websites
- http://grails.org/Success+Stories

→ Grails wird bei großen Firmen und auch in großen Projekten eingesetzt

Grails und andere Frameworks [6]

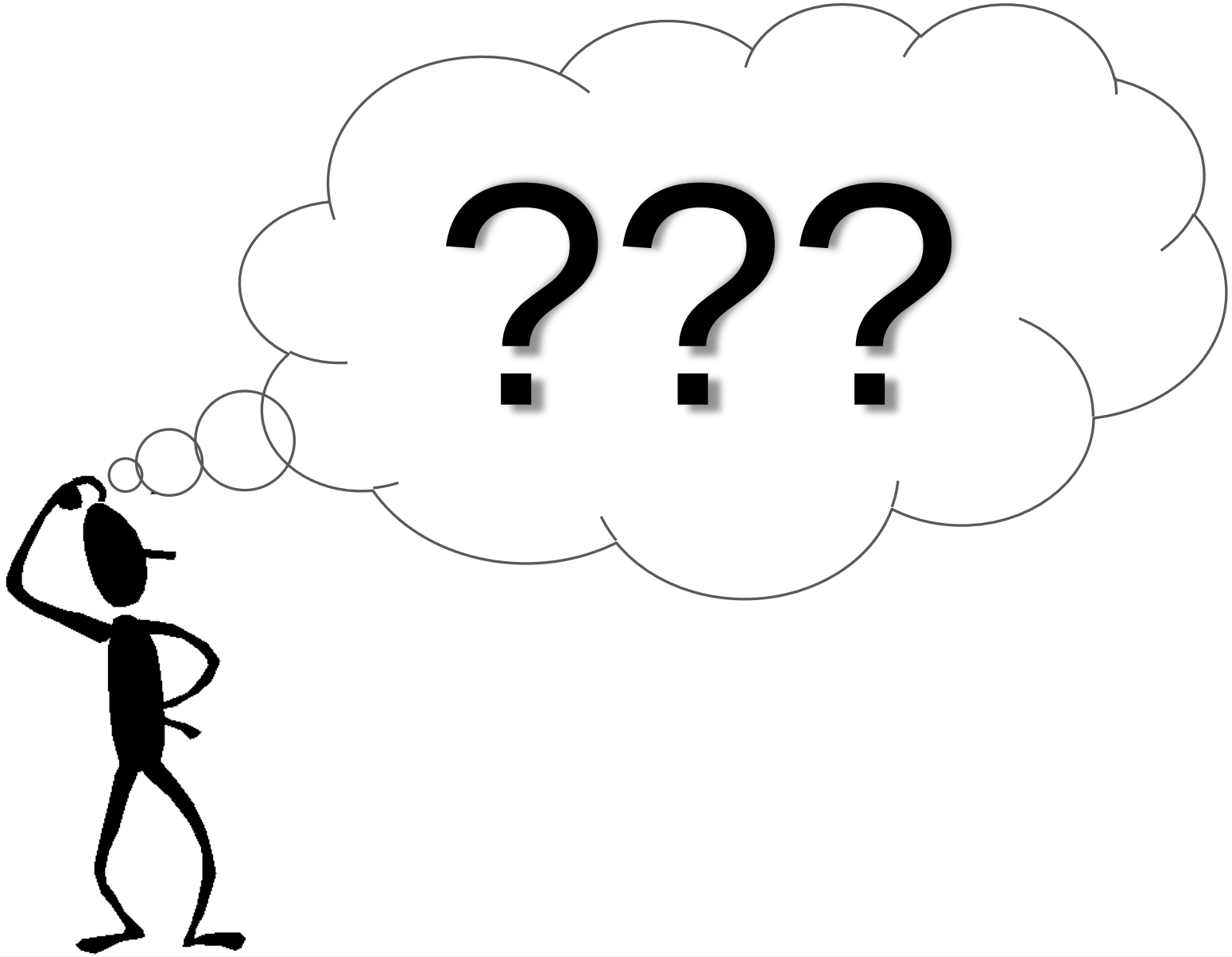


Warum haben wir uns für Grails entschieden?

- Sinnvoll strukturierte und vorkonfigurierte Anwendungsstruktur
- Open Source, bindet moderne und bewährte Technologien ein
- Steigert die Produktivität
- Einfach einzusetzen, fokussiert auf die Entwicklung von Features, keine Ablenkung durch Konfiguration
- Gute IDE-Unterstützung
- Gute Community
- Viele nützliche Plugins
- Ausdruckstarke Syntax von Groovy

Buchempfehlungen

- The Definitive Guide to Grails 2
- Grails in Action, Second Edition
- Programming Grails



Auch wir suchen neue Kollegen …

exensio
intelligente informationssysteme

mehr Informationen unter

http:www.unic.com

http:www.exensio.de

Referenzen

[1] http://grails.org

[2] http://grails.org/start#modal-pres1

[3] http://www.youtube.com/watch?v=PXHxo43hn34 Getting Started with Grails - Part 1 (Peter Ledbrook)

[4] http://grails.org/start#modal-pres1 - Play presentation

[5] GR8Conf US 2012, Static Type Checking and Compilation – Cédric Champeau

[6] http://www.google.com/trends