FWP: AKTUELLE TECHNOLOGIEN ZUR ENTWICKLUNG VERTEILTER JAVA ANWENDUNGEN

Groovy/Grails

Ein Überblick

Julian Wilhelm 03.06.16

Groovy

Ziele

- Dynamische Sprache
- Bestmögliche Komptabilität mit Java
- Integration von Konzepten aus Perl, Python und Ruby
- Effizientere Entwicklung/Wartung durch syntaktischen Zucker

Groovy

Integration mit Java

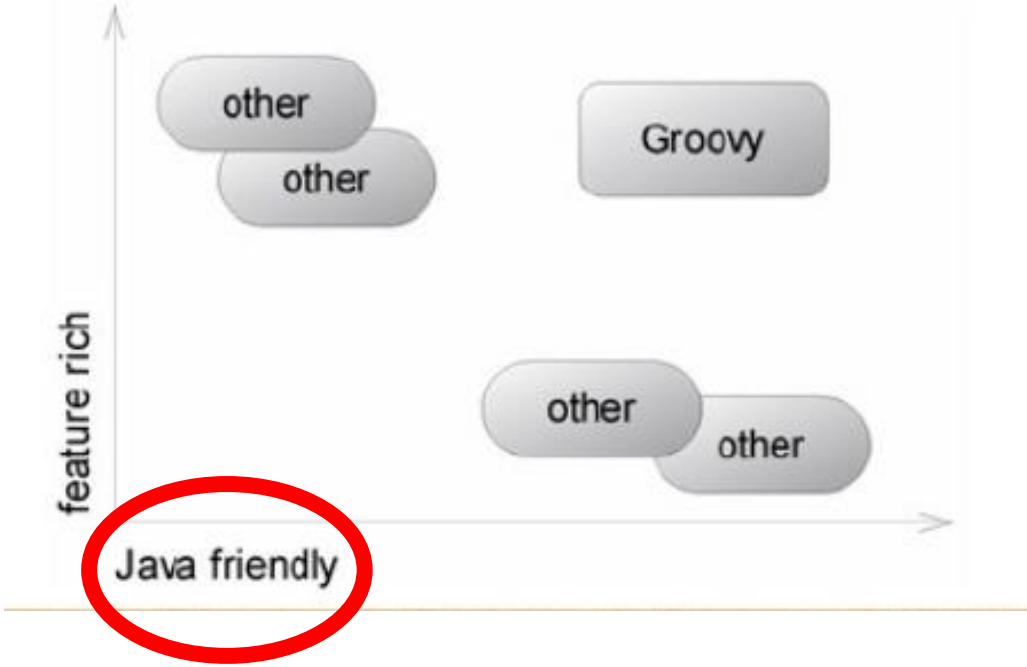
- Läuft auf der Java Virtual Machine (JVM)

→ Groovy ist Java-Bytecode zur Laufzeit

→ Java (fast) immer gültiger Groovy-Code

→ Java-Bibliotheken zugänglich

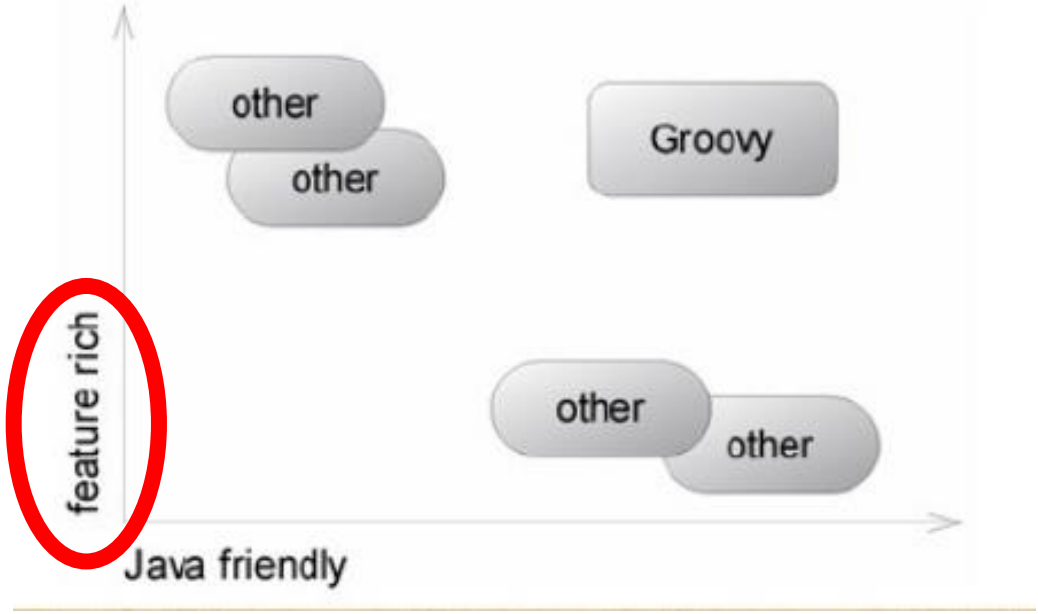
→ Groovy kann in Java Projekten integriert werden



Groovy

Features (einige Beispiele)

- Dynamische Typisierung (duck typing)
- Closures
- Currying
- Elvis Operator
- Safe Navigation



Groovy

Syntaktischer Zucker

Siehe Beispiel

working with Groovy

vs.

working with Java

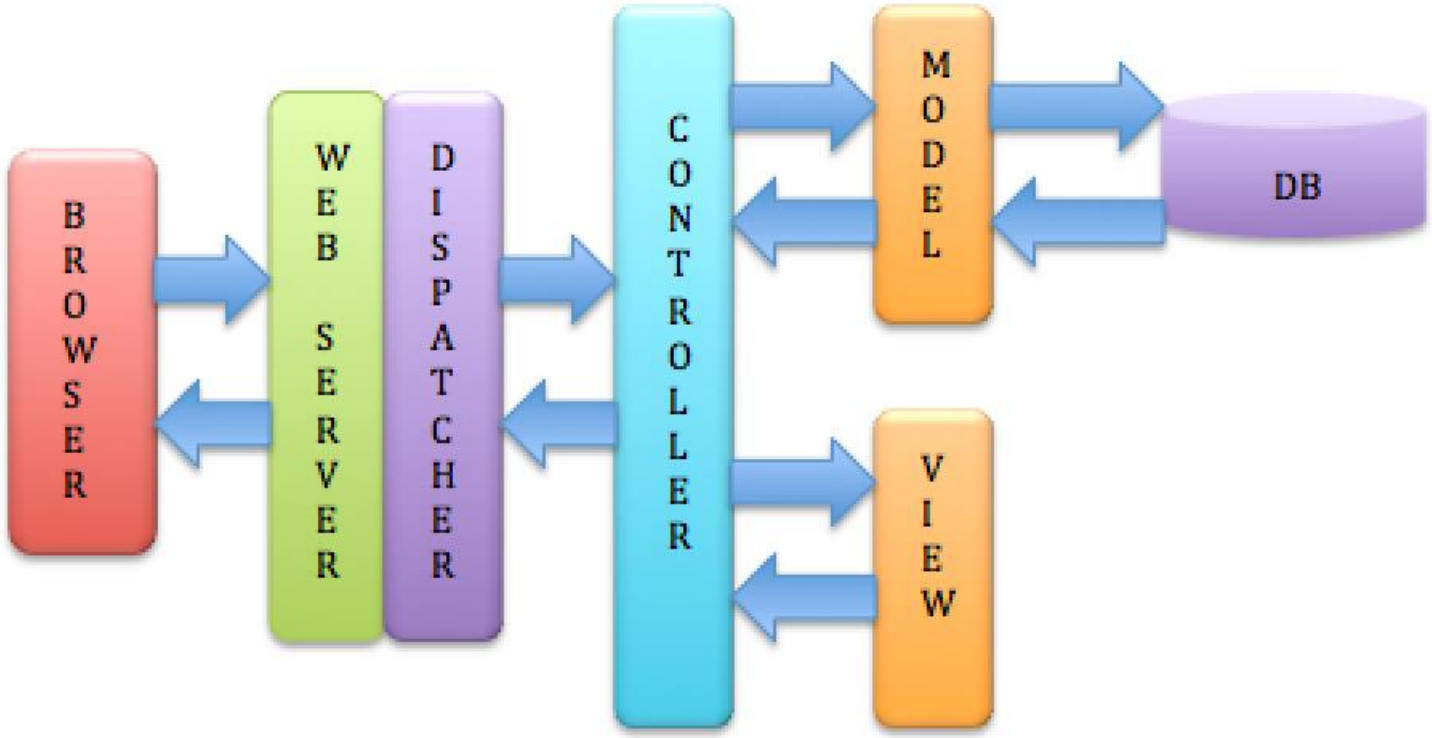
Grails

Was ist Grails?

- Web Application Framework
- Programmiert in Groovy
- „Under the hood“: Spezialisierung des Spring Frameworks
- MVC-Architektur
- Implementiert weitere etablierte Frameworks

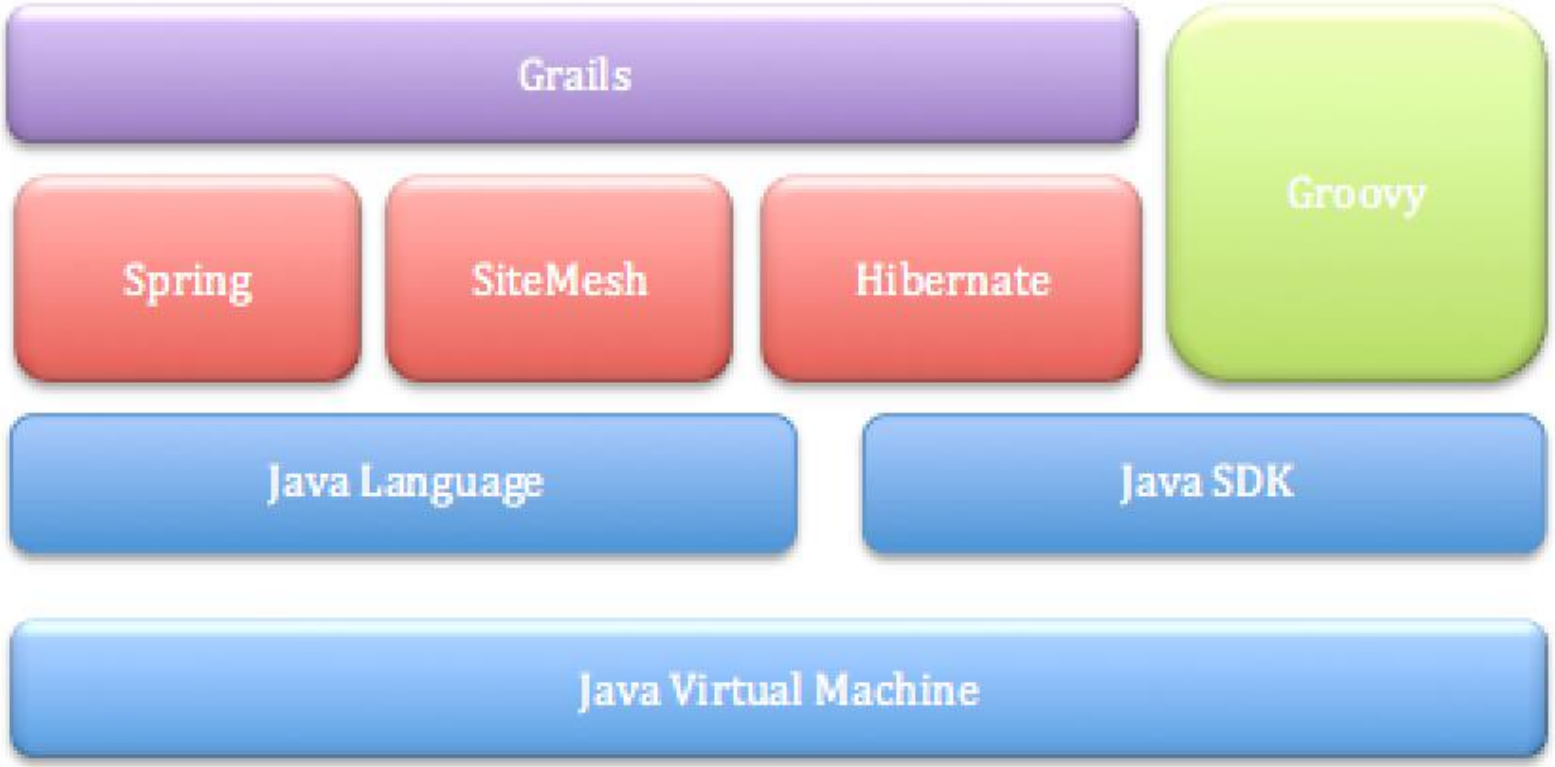
Grails

Grails‘ MVC Architektur



Grails

Grails und sein Fundament in der Java Welt



Grails

Deployment

- Build erfolgt standardmäßig als WAR-File
- Deployment händisch oder per Tool (z.B. Jenkins)
    - Gradle – vergleichbar mit Ant/Maven
- Server nach JavaEE Spezifikation
- Gant (Groovy Ant) ermöglicht Deployment als EAR.

Grails

Inversion of Control

- Hollywood-Prinzip

„Don‘t call us, we‘ll call you!“

- Bekannt: JavaEE EJB und CDI
- Bei Grails: Spring Beans und Spring Dependency Injection
- Unterschied: Groovy Spring Bean DSL für bean descriptors

Grails

Groovy Spring Bean DSL

```
beans {
    framework String, 'Grails'
    foo String, 'hello'
    bar(Bar,s:'hello',i:123)
}
```

- beans{ …} ist ein Closure
- GroovyBeanDefinitionReader verarbeitet DSL (kompatibel zu XML)
- Syntax: beanName(Typ, Konstruktor-Args) //Klammern optional
- Groovy Objekte verfügen immer über Default-Konstruktor (Map mit allen Klassen-Attributen)
- Letzte Zeile zeigt Instanz von „Bar“ mit 2 Feldern

Grails

Vergleich: XML und Java Annotations

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.spr
ingframework.org/schema/beans/spring-beans.xsd">
      <bean id="myBean" class="my.company.MyBeanImpl">
          <property name="someProperty" value="42"/>
          <property name="anotherProperty" value="blue"/>
     </bean>
</beans>
```

```java
import my.company.MyBeanImpl;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Bean;
@Configuration
public class MyConfig() {
    @Bean myBean() {
          MyBeanImpl myBean = new MyBeanImpl();
          myBean.setSomeProperty(42);
          myBean.setAnotherProperty("blue");
          return myBean;
    }
}
```

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN · FH MÜNCHEN

Grails

Kontrollstrukturen in Groovy DSL

```
beans {
    // org.springframework.beans.factory.groovy.GroovyBeanDefinitionReader
    if (environment.activeProfiles.contains("prof1")) {
        foo String, 'hello'
    } else {
        foo String, 'world'
    }
}
```

Grails

Convention over Configuration

- Konfigurationen in extra Files oder Annotationen nur im Sonderfall
- Entscheidungen in Grails basieren auf Defaults
- Bestes Beispiel: Projektstruktur und Scaffolding
    - Verknüpfungen zwischen Model, Controller und View müssen (normalerweise) nicht definiert werden

Grails

Scaffolding

- Automatisches Generieren ganzer (einfacher) Anwendungen für Domain-Klassen
- Controller und Views für CRUD-Operationen
- Statisches Scaffolding (Code liegt vor Kompilierung vor)
- Dynamisches Scaffolding (Code wird zur Laufzeit generiert)
- Voraussetzung Domain-Klasse mit definierten Constraints

```
static constraints = {
  name(blank:false)
  createdDate()
  priority()
  status()
  note(maxSize:1000, nullable:true)
  completedDate(nullable:true)
  dueDate(nullable:true)
}
```

Grails

Object Relational Mapping (GORM)

- „Under the Hood“: Hibernate
- Dynamic Finders
    - Aus Ruby übernommen
    - Zur Laufzeit dynamisch generiert
    - „findBy“/“findAllBy“ + Attribut d. Domain + Komparator (insgesamt 13) + Parameter (Wert)
    - Mehrere Vergleiche können durch And und Or verknüpft werden
    - Unterteilung in Seiten und Sortierung möglich

Grails

Dynamic Finders Beispiele

```
def book = Book.findByTitle("The Stand")

book = Book.findByTitleLike("Harry Pot%")

book = Book.findByReleaseDateBetween(firstDate, secondDate)

book = Book.findByReleaseDateGreaterThan(someDate)

book = Book.findByTitleLikeOrReleaseDateLessThan("%Something%", someDate)
```

Grails
Web Layer - Controller

- Controller und Views (klassisch nach MVC)
- Model kann von Controller an View übergeben werden
- Per Default wird *show*-Action zuerst gerendert
- Gewünschte View kann manuell bestimmt werden
- Render kann auch genutzt werden um JSON oder XML auszugeben
- Redirect auf eine URL mittels HttpServletResponse.sendRedirect

```
def show() {
def map = [book: Book.get(params.id)
render(view: "display", model: map)
}
```

Grails
Web Layer - Views

- Groovy Server Pages (GSP) nach Vorlage der JSP
- Flexibler und intuitiver
- Syntax:
- <%%> - eingebetteter Code (Bad Practice!)
- <%= %> - Werteausgabe
- <%-- --%> - Serverseitige Kommentare
- Alternative Werte-Ausgabe:

```
<html>
  <body>
      Hello ${params.name}
  </body>
</html>
```

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN · FH MÜNCHEN

Grails
Web Layer - Views

- GSP-Tags
- Custom Tag Libraries möglich
- GSP-Tags können Body und Attribute enthalten
- Übersicht Tags:
  - If/else
  - Each-Schleifen
  - While schleifen
  - FindAll
  - grep
  - actionSubmit

```
<g:example attr="${new Date()}">
Hello world
</g:example>
```

Fazit Pro und Contra Grails

- Fokus auf effiziente Entwicklung

→ Relativ kurze Entwicklungszyklen

- Bereits für kleine Projekte geeignet
- Gute Skalierbarkeit
- Lebendige Community
- Solides Fundament aus Frameworks/Plug-Ins etc.

- Viel Bewegung zur Laufzeit

→ Fehlerquelle und potentiell schwieriges Debugging

- Teils Schwierigkeiten mit Multi-Threaded Applikationen (GORM)
- Performance-Einbußen durch Übersetzung des Code in Java-Bytecode
- Etablierte Frameworks haben teilweise aufgeholt

Ursprung der Grafiken

- Von Zorak1103 - Eigenes Werk, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=13358930
- Groovy in Action, Seite 3, Dierk König
- https://fbflex.wordpress.com/2010/04/22/working-with-groovy-vs-working-with-java/
- http://www.unixstickers.com/stickers/coding_stickers/grails-JVM-web-application-framework-shaped-sticker
- http://www.bhaskarpundkar.com/introduction-to-grails/
- https://spring.io/blog/2014/03/03/groovy-bean-configuration-in-spring-framework-4
- http://www.ociweb.com/products/grails/
- https://zeroturnaround.com/rebellabs/get-groovy-with-jrebel-and-grails/
- https://kousenit.wordpress.com/2013/10/02/making-java-groovy-ratpack-mongodb-and-chuck-norris/

Danke für die Aufmerksamkeit!

HOCHSCHULE
FÜR ANGEWANDTE
WISSENSCHAFTEN · FH
MÜNCHEN