Kann ich mit Grails Enterprise Applikationen umsetzen?

Tobias Kraft
exensio GmbH
Karlsruhe

Schlüsselworte

Grails, Enterprise Applikationen, Groovy

Einleitung

Das Framework Grails ist bereits seit einigen Jahren auf dem Markt und es stellt sich die Frage nach den Einsatzbereichen. Insbesondere ist interessant, inwiefern es sich für Applikationen im Unternehmensbereich eignet, oder ob es nur für die „kleine schnelle Lösung“ seine Daseinsberechtigung hat. Nach einem Überblick zu Grails und dessen Schlüsselkonzepte werden Anforderungen sowie Szenarien für den Unternehmenseinsatz beleuchtet und untersucht, wie diese mit Grails abdeckbar sind.

Grails im Überblick

Bereits im Jahr 2005 startete das Projekt Grails, inspiriert im Wesentlichen durch die Konzepte von Ruby on Rails. Zu dieser Zeit war die Entwicklung von JEE-Applikationen schwergewichtig und es floss viel Arbeit in die Programmierung von Glue Code. Mit Hilfe von Spring und Neuerungen im JEE-Standard hat sich dies natürlich in den letzten Jahren massiv geändert. Als Fullstack-Webframework deckt Grails das gesamte Spektrum des Entwickelns, Testens und der Produktivsetzung einer Webapplikation ab. Es hat insbesondere das Ziel, die Komplexität bei der Erstellung von Webapplikationen zu reduzieren, basiert jedoch trotzdem auf etablierten Java-Technologien.
Zwischenzeitlich werden in regelmäßigen Abständen jährlich Minor- und Major-Releases freigegeben. Das nächste Release, die Version 3.0, legt seinen Schwerpunkt auf die Modularisierung und die Verschmelzung mit Gradle als Build-System. Es soll neben Servlet-Containern auch eine bessere Integration für Frameworks wie bspw. Netty geben. Grails ist ein Open-Source-Projekt und steht unter der Apache Lizenz 2.0, so dass keine Lizenzkosten bei der Verwendung anfallen. Die Weiterentwicklung und Pflege wird wesentlich von der Firma Pivotal, eine ausgelagerte Tochterfirma von VMWare, vorangetrieben.

Groovy

Die Basis von Grails bildet die Sprache Groovy. Hierbei handelt es sich um eine nicht typisierte Programmiersprache, die eine JVM benötigt und es ermöglicht, Anweisungen - im Vergleich zu Java - kürzer auszudrücken. Das folgende Code-Beispiel zeigt, wie einfach Listen definiert und wie sie mit Hilfe der Groovy-Closure findAll für Collections verarbeitet werden können. Closures sind das Pendant zu den in Java8 erscheinenden Lambda-Expressions.

```
def names = ['DOAG SIG Security', 'SIG MySQL', 'OpenWorld', 'SIG Database: Migration']

println names

def sigEvents = names.findAll{ it ==~ /.*SIG.*/ }.sort()

println "number of SIG events: ${sigEvents.size()}"

sigEvents.each { println it }
```

Ab Version 2 von Groovy kann bei Bedarf im Code mit der Annotation *@TypeChecked* die Typsicherheit gewährleistet werden. Dies ist beispielsweise für öffentlich verwendete Schnittstellen interessant.
In der Vergangenheit gab es bei Groovy öfters Kritik bzgl. der Performance. Hier wurden in den zurückliegenden Releases wesentliche Optimierungen eingearbeitet und durch explizites Setzen der ebenfalls ab Version 2 verfügbaren Annotation *@CompileStatic* können zusätzliche Performance-Gewinne erzielt werden.

Schlüsselkonzepte
Wesentliche Schlüsselkonzepte von Grails sind DRY (Don't repeat yourself) und CoC (Convention over Configuration). Diese Prinzipien erlauben dem Entwickler eine stärkere Konzentration auf die Business-Logik und auch der Code wird übersichtlicher. Nachfolgend werden einige Beispiele aufgeführt:

- Logging kann auf Anhieb verwendet werden ohne vorherige Instanziierung
- Zugriff auf Services von Controllern erfolgt durch Definition einer Klassen-Variablen, die den gleichen Namen wie der Service hat
- Domain-Klassen, das Pendant zu den Entity Beans, liegen im Verzeichnis *grails-app/domain*. Sie werden automatisch mit der DB gemappt und es kann direkt damit gearbeitet werden, ohne dass ein EntityManager oder Ähnliches benötigt wird

Die Flexibilität geht mit CoC aber nicht verloren. Es bedeutet viel mehr, dass der Standard vordefiniert ist, man bei Bedarf jedoch auch davon abweichen kann.

Plugins
Ein zusätzlicher wichtiger Baustein sind Plugins, über die weitere Funktionalitäten hinzugefügt werden können. Derzeit stehen fast 800 Plugins für Grails bereit. Plugins bieten hauptsächlich Erweiterungen für folgende Bereiche:

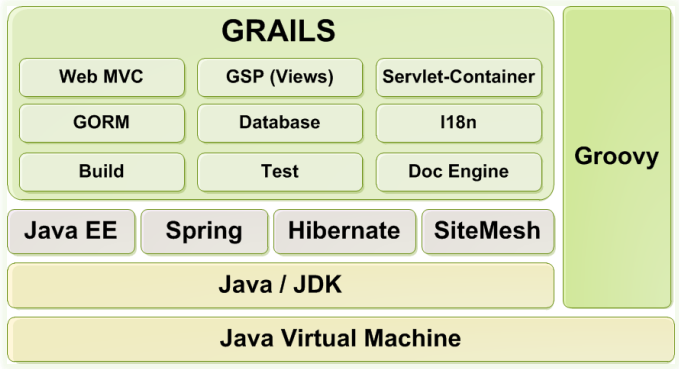
- Frontend (z.B. JQuery, AngularJS)
- Integration von Drittsystemen (z.B. Suchmaschinen, BPMN-Maschinen)
- Fachliche Funktionalitäten (z.B. Tagging, Voting)
- Persistenzschicht (z.B. neo4j, MongoDB)
- Nichtfunktionale Erweiterungen (z.B. Auditing zum Messen von Antwortzeiten)

Neben der Wiederverwendbarkeit sind als wesentliche Vorteile von Plugins eine einfache Installation und Möglichkeiten zur Modularisierung anzusehen.

Einfacher Start mit Grails
Besonders der Einstieg in Grails gestaltet sich einfach. Für den Start mit Grails wird von der Homepage die aktuelle Version heruntergeladen und entpackt. Da Grails auf der JVM aufsetzt, muss nichts als das JDK vorhanden, sowie über die Umgebungsvariable eingebunden sein. Es ist möglich, mit einem gewöhnlichen Texteditor loszulegen. Das Basisprojekt wie auch einzelne Code-Fragmente

können mit der Anweisung *grails* und daran angehängten Optionen erstellt werden. Der Befehl *grails create-domain-class com.exensio.Project* erzeugt die entsprechende Domain-Klasse *Project* für die Interaktion mit der Persistenzschicht. Die Applikation wird mit dem Befehl *grails run-app* auf dem eingebetteten Tomcat-Server gestartet. Alternativ kann ebenfalls mit den später vorgestellten Plugins Jetty als Laufzeitumgebung ausgewählt werden. Während der gesamten Entwicklungszeit und auch beim Ausführen von Tests wird gegen den eingebauten Tomcat-Server entwickelt.
Ein weiteres Feature, das den Einstieg erleichtert, ist das sogenannte Scaffolding. Über Scaffolding können aus Domain-Klassen Controller und Views für die CRUD-Funktionalitäten erzeugt werden. Dieser Ansatz eignet sich hervorragend für das Rapid Prototyping, um bspw. dem Kunden zeitnah erste Ergebnisse zeigen und besprechen zu können. Es wird zwischen dynamischen und statischen Scaffolding unterschieden. Ersteres generiert den Code der Views und Controller immer zur Laufzeit aus Templates. Die statische Methode erstellt beide Komponenten einmalig und speichert sie in den entsprechenden Dateien. Damit kann der Source-Code im Nachhinein an die eigenen Bedürfnisse angepasst werden. Ändern sich Attribute in der Domain-Klasse, so müssen diese manuell in den erzeugten Dateien nachgezogen werden.

Bewährte Technologien
Grails setzt auf bewährte Technologien in seinem Technologiestack, wie die nachfolgende Grafik verdeutlicht. Neben der JVM ist Spring eine wesentliche Basistechnologie, die in allen Bereichen eingesetzt wird. Außer JEE-Standards und Hibernate für die Persistenz kommen mit JUnit und Ivy auch im Test- und Build-Zyklus weit verbreitete Technologien zum Einsatz.



Kurz eingegangen werden soll hier auf GORM, das die Persistenzschicht kapselt und entsprechende Funktionalitäten für einen schnellen Zugriff auf diese bereitstellt. Hibernate ist der standardmäßig konfigurierte OR-Mapper. Es ist aber auch möglich GORM mit NOSQL-Datenbanken wie MongoDB zu verwenden.
Insbesondere das Erstellen von Datenbankabfragen ist mit GORM sehr komfortabel möglich. Über die Domain-Klasse können direkt Finder-Methoden mit bis zu drei Abfrage-Kriterien erstellt werden, wie das nachfolgende Beispiel zeigt. Ein Zweites zeigt die sogenannten Where-Queries, die Abfragen erlauben mit in Groovy-Code formulierten Bedingungen.

```
// Finde Methode mit zwei AND verknüpften Attributen

Book.findAllByTitleAndAuthor('Grails in Action', authorInstance)

// Where-Query

def query = Book.where{

  year(published) == 2013 && title =~ 'grails'

}

def results = query.list(sort: 'title')
```

Anforderungen an Enterprise Applikationen

In der Regel wird unter Enterprise Applikationen Software verstanden, die bzgl. der Datenmengen, der Verarbeitungslogik, der Integration von Drittsystemen oder der Oberfläche komplex ist. Außerdem müssen Enterprise Applikationen in der Lage sein, nicht funktionale Anforderungen zu erfüllen. Eine Auswahl von relevanten Kriterien sind:

- Betrieb und Monitoring
- Verfügbarkeit
- Ausfallsicherheit
- Wartbarkeit und Erweiterbarkeit
- Security
- Kein Daten-Verlust bzw. Korruption bei Fehlern
- Testbarkeit

Enterprise Applikationen mit Grails
In den nachfolgenden Abschnitten werden einige Bereiche, die im Enterprise-Umfeld relevant sind im Zusammenhang mit Grails beleuchtet.

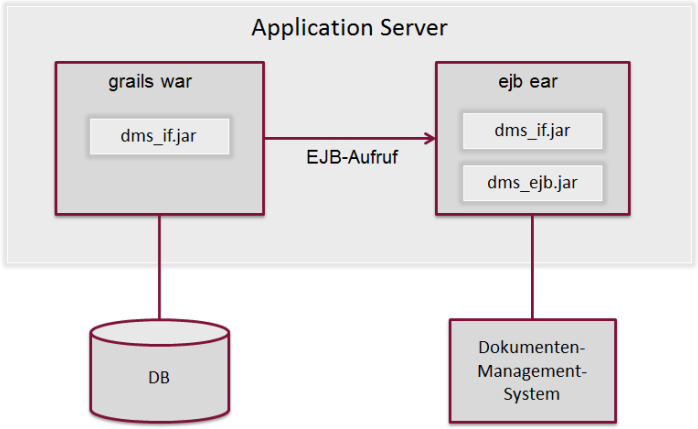
Produktivbetrieb
Für die Entwicklung wird, wie bereits erwähnt, Tomcat benutzt. In größeren Unternehmen ist in der Regel der Laufzeit-Container jedoch vorgegeben und es kommen vor allem kommerzielle Server wie bspw. Websphere oder Weblogic zum Einsatz. Da für den Wirkbetrieb ein WAR-Archiv erzeugt wird, kann Grails auf sämtlichen Servlet-Engines und Applikations-Servern betrieben werden. Clustering zur Erhöhung der Ausfallsicherheit ist dank der Verteilung als WAR-Archiv mit Grails problemlos möglich. Abhängig von den Anforderungen mag Load-Balancing mit und ohne Session-Replication eingesetzt werden [CLUST].
Im Produktivbetrieb sind Monitoring-Aktivitäten unumgänglich. Zusätzlich zu den Standardmonitoring-Möglichkeiten von WAR-Archiven können von der Applikation über MBeans Informationen exportiert werden und diese mittels JMX abgefragt werden.

Konfigurationsmöglichkeiten
Existieren mehrere Umgebungen, wie Test-, Integrations-, Produktionsumgebung, so ist das Auslagern von Konfigurationsparametern sinnvoll. Außerdem sind damit Anpassungen ohne ein neues Deployment möglich.

Hierfür unterstützt Grails JNDI. Hiermit können bspw. die Werte für den Datenbankzugriff oder auch des Mail-Servers aus dem Laufzeit-Container gelesen und ebenso dort angepasst werden. Eine weitere Alternative sind Konfigurationsdateien, die über den Klassenpfad oder ein relatives Verzeichnis eingelesen werden.

Security
In Unternehmen erfolgt die Benutzer- und Rollen-Verwaltung in vielen Fällen über LDAP- Server. Zum Zwecke der zentralen und einmaligen Pflege ist es sinnvoll, dass Software-Applikationen diese Services nutzen. Das Spring-Security-Plugin deckt für Grails-Anwendungen den kompletten Bereich der Authentifizierung und der Autorisierung ab. Neben der Datenbank können genauso LDAP-Server angebunden werden. Wird ein Microsoft IIS der Grails-Applikation vorgeschaltet, ist eine Anbindung der passwortfreien integrierten Windows-Authentifizierung (ehemals NTLM) problemlos umsetzbar. Mit den diversen verfügbaren Aufsätzen für das Spring Security Plugin sind auch Authentifizierungsmechanismen für Twitter, Facebook und weitere andere Provider möglich. Durch die Erweiterbarkeit können gleichfalls eigene Authentifizierungs- und Autorisierungs-Mechanismen umgesetzt werden.

Integration mit Java und Drittsystemen
Mit der JVM als Basisplattform kann in Grails sämtlicher Java-Code ausgeführt und auch entsprechende JAR-Archive integriert werden.
Sehr interessant ist die Integration von EJBs, um mit bestehenden Java-Systemen zu kommunizieren. Im nachfolgenden ist ein Szenario für den Zugriff auf ein Dokumenten-Management-System über EJB's dargestellt.



Hierbei wird in Grails via Spring-Beans der JNDI-Zugriff in weniger als 5 Zeilen konfiguriert. Das JAR-Archiv mit den Java-Interfaces ist in beiden Applikationen vorhanden. Über die Spring-Bean können anschließend die in den Schnittstellen definierten Service-Methoden genutzt werden.
Häufig findet man in der Industrie Messaging-Systeme zur Interkommunikation und asynchronen Abarbeitung vor.
Eine unkomplizierte Integrationsmöglichkeit für Drittsysteme bieten ebenfalls JSON und XML. Mit Hilfe von Groovy ist die Verarbeitung dieser Formate absolut einfach.

Robustheit des Systems
Die aus Java bekannten Exception-Mechanismen werden auch bei Grails eingesetzt und es können zentrale Fehlerseiten abhängig vom Fehler-Typ angezeigt werden. In den Services ist ein besonders feingranulares Transaktionshandling denkbar, um bedingt vom Anwendungsfall entsprechende Rollbacks zu initiieren. Es ist zu beachten, dass nur Unchecked Exceptions einen Rollback auslösen. Sind 2 Phase-Commits notwendig - falls z.B. mit Messaging und der DB innerhalb einer Transkation

gearbeitet wird, gibt es - wie in Grails üblich - wiederum ein Plugin, das eingebunden werden kann. [ATOM].

Tool- und Fullstack-Unterstützung
Unumgänglich ist im Enterprise-Umfeld eine ausreichende Tool-Unterstützung. Im Bereich der Entwicklungsumgebung ist die kommerzielle Version von IntelliJ ungemein fortschrittlich und unterstützt immer die aktuellsten Grails-Versionen. Mit den weit verbreiteten Open-Source Produkten Jenkins und Hudson stehen für Continuous Integration sehr ausgereifte Werkzeuge zur Verfügung. Für die in professionellen Projekten unumgänglichen automatisierten Tests bieten Spock und Geb die ideale Basis für Grails. Über die Continuous Integration Server wird auch mit den entsprechenden Grails-Plugins die Testabdeckung [COB] oder die Codequalität [COD] analysiert.

Große Code-Basis
Wie verhält sich Grails bei einer großen bzw. immer weiter wachsenden Code-Basis? Die bis heute von uns umgesetzten Projekte - mit bis zu 200 Domain-Klassen - zeigten keine negativen Performance-Einbußen. Bei steigender Code-Basis verlängert sich natürlich die Zeit für Neu-Kompilierungen.
Im Zusammenspiel mit Datenbanken und Datenmengen konnten bei den bisher in Produktion befindlichen Applikationen Erfahrungen mit bis zu 20 GB gesammelt werden und auch hier gab es keinerlei Anzeichen von Problemen.
Aus Architektur-Sicht ist es bei umfangreicheren Vorhaben sinnvoll, nicht alles in einem einzigen Grails-Projekt umzusetzen, sondern vielmehr nach fachlichen Gesichtspunkten die Anwendung zu schneiden und Teile über den Plugin-Mechanismus auszulagern.

Fazit

Da die Anforderungs-Szenarien an Enterprise-Anwendungen breitgefächert und umfangreich sind, kann der Artikel nur einige wesentlichen Bereiche beleuchten. Es ist jedoch deutlich geworden, dass Grails auf Grund der engen Beziehung zu Java Ansprüche an Enterprise-Applikationen in vielen Fällen erfüllen kann. Durch die zahlreichen Plugins können weitere Funktionalitäten abgedeckt werden oder bei Bedarf selbst mit einem eigenen Plugin implementiert werden.

Referenzen

[CLUST] http://blog.exensio.de/2012/05/teil-3-grails-in-produktion-mit-apache.html
[ATOM] http://grails.org/plugin/atomikos
[COB] http://grails.org/plugin/code-coverage
[COD] http://grails.org/plugin/codenarc

Kontaktadresse:
Tobias Kraft
Exensio GmbH
Am Rüppurrer Schloß 12
D-76199 Karlsruhe

Telefon:	+49 (721) 989 647 84
Fax:	+49 (721) 598 41 67
E-Mail	tobias.kraft@exensio.de
Internet:	http://www.exensio.de
Twitter:	http://twitter.com/exensio