DOAG

Deutsche ORACLE-Anwendergruppe e. V.

News

Die Zeitschrift der DOAG-Mitglieder

2/2010

Schwerpunktthema:

Sun & Java

www.doag.org

Sonderdruck
für die exensio GmbH

ISSN 09 36-0360

Web-Applikationen mit Grails und WebLogic

Peter Soth, exensio GmbH

Verbesserte Frameworks haben die Realisierung von Web-Applikationen in den letzten Jahren stark vereinfacht. So hat sich beispielsweise die Komplexität von Enterprise-Java-Applikationen (JEE) ständig reduziert, hinzu kam der Einsatz von Technologien wie Spring und Hibernate. Diese Frameworks eignen sich jedoch nur bedingt für kostensensitive Projekte, da immer noch ein beträchtlicher Aufwand für technische und nicht fachliche Aspekte erforderlich ist. Dieser Artikel stellt das Framework „Grails" vor, das die Produktivität eines Entwicklers durch die Entlastung bei technischen Fragestellungen bedeutend steigert. Der Einsatz des Oracle WebLogic-Servers stellt die Skalierbarkeit und den professionellen IT-Betrieb der Applikation sicher.

Enterprise-Java-Anwendungen sind für ihre technische Komplexität bekannt. Deshalb muss sich ein Entwickler zuerst in sehr viele Technologien einarbeiten, bevor er mit der Umsetzung fachlicher Anforderungen beginnen kann. Dies treibt die Projektkosten in die Höhe. Einige Software-Hersteller haben diese Problematik erkannt und Produkte für eine effizientere Entwicklung auf den Markt gebracht. Dazu zählt auch das Oracle Application Development Framework (ADF), das unter Einsatz einer integrierten Entwicklungsumgebung und fertigen, nur noch zu konfigurierenden Komponenten die Entwicklungsarbeit vereinfacht. Das Grails-Framework geht noch einen Schritt weiter und folgt den Ideen von „Ruby on Rails", wie sich aus dem Namen „Grails" schon vermuten lässt. Grails eignet sich im Besonderen für eine agile Software-Entwicklung.

Grails folgt den Kernprinzipen von Ruby on Rails:

- *Don't Repeat Yourself (DRY)*
  Dieses Prinzip basiert auf den Ideen der Autoren Andy Hunt und Dave Thomas, die im Buch „Der pragmatische Programmierer" beschrieben sind. In Grails werden hierzu Domain-Klassen in allen Schichten (Präsentation, Geschäftslogik und Persistenz) benutzt. JEE verwendet hier zum Vergleich in jeder Schicht separate Klassen, die eine Transformation benötigen. Eine Änderung in einer Schicht bedingt somit Modifikationen in den weiteren Schichten.
- *Convention over Configuration*
  Bei Grails können Konfigurationen vorgenommen werden, falls die Konventionen nicht ausreichen sollten. Eine Domain-Klasse erzeugt demgemäß einen Präsentations-Controller und eine Datenbank-Tabelle als feste Konvention. Dadurch entfällt die von Spring bekannte Konfigurationsproblematik, bei der man die einzelnen Komponenten über Konfigurationsdateien verknüpfen muss, was sehr schnell unübersichtlich wird.

Java, Groovy und Grails

Ein großer Vorteil von Grails – im Vergleich zu Ruby on Rails – ist seine Nähe zur Java-Welt. Hieraus resultieren ein hoher Wiederverwendungsgrad des bestehenden Java-Codes sowie eine sehr gute Skalierbarkeit, beispielsweise durch den Einsatz eines Oracle WebLogic-Servers. Eine Grails-Applikation kann dann sämtliche Funktionalitäten wie Datenbank-Connection-Pool oder das Security-Framework des WebLogic-Servers nutzen. Grails auf einen Blick:

- *Groovy*
  Grails benutzt die Java-basierte Skriptsprache Groovy. Damit erhält der Entwickler eine sehr effiziente Programmiersprache. Groovy-Code wird in Java-Byte-Code kompiliert und innerhalb der Java Virtual Machine (JVM) ausgeführt. Aufgrund der engen Verzahnung von Java und Groovy ist es möglich, vorhandene Java-Klassen in Groovy-Code einzubinden und umgekehrt.
- *Basis-Technologien*
  Grails benutzt Spring und Hibernate. Dies stellt sicher, dass Grails-Applikationen über eine sehr gute Performanz und Skalierbarkeit verfügen.
- *Scaffolding (übersetzt Gerüstbau)*
  Grails generiert aus den Domain-Klassen das Gerüst einer Applikation, bestehend aus der zur Domain-Klasse gehörenden Datenbank-Tabelle und dem User-Interface-Controller – inklusive aller CRUD-Funktionalitäten (Create, Read, Update und Delete). Das Gerüst kann später angepasst werden.
- *Grails Object Relational Mapping (GORM)*
  Domain-Klassen werden mit GORM persistiert. GORM kann selbst komplexe Relationen abbilden, beispielsweise n:m-Relationen oder hierarchische Strukturen zwischen Tabellen.
- *Entwicklungsumgebung*
  Es stehen verschiedene Plug-ins für Standard-Entwicklungsumgebungen zur Verfügung. Ein einfacher Texteditor reicht in der Regel jedoch auch. Grails wird standardmäßig mit der Servlet-Engine „Tomcat" ausgeliefert.
- *Plug-in-System*
  Grails verfügt über ein umfangreiches Plug-in-Angebot: Momentan

stehen rund 300 Plug-ins für die unterschiedlichsten Anforderungen zur Verfügung.

Einsatzmöglichkeiten

Der Autor setzt das Grails-Framework bereits seit dessen Veröffentlichung erfolgreich ein. Grund dafür war ursprünglich ein Kundenauftrag, ein Excel-Sheet in eine Web-Applikation zu migrieren. Diese hat im Vergleich zu einem Excel-Sheet folgende Vorteile:

- Die Daten werden zentral verwaltet:
  - Es ist keine Verteilung per E-Mail oder File-Share nötig.
  - Eine Bearbeitung der Daten ist durch mehrere Benutzer gleichzeitig möglich und alle Benutzer haben immer den aktuellen Stand zur Hand.
- Man kann Dritt-Systeme anbinden. Durch Anbindung eines LDAP-Servers muss beispielsweise eine E-Mail-Adresse nicht mehr manuell eingegeben werden.

Positive Erfahrungen aus verschiedenen Grails-Projekten zeigen, dass ein Entwickler drei- bis viermal produktiver ist als bei der Nutzung eines herkömmlichen Frameworks. Eine Realisierung mit JEE beziehungsweise Spring und Hibernate wäre in einer so kurzen Zeit nicht möglich gewesen und hätte demnach den Budget-Rahmen gesprengt.

Grails nur für kleine Projekte?

Die Frage nach der Projekt-Größe wird in den Grails-Foren oft gestellt. Dabei muss man genau unterscheiden, was unter einem großen Projekt zu verstehen ist. Grails eignet sich auch für diese unter folgenden Gesichtspunkten:

- Grails skaliert sehr gut, da es auf Spring und Hibernate basiert. Diese Frameworks haben das bereits in sehr vielen Projekten unter Beweis gestellt. Im Anhang findet sich ein Link auf Grails Success Stories. Unter diesen findet sich beispielsweise auch ein schwedischer Rundfunksender, dessen Seite mehrere Millionen Anfragen pro Monat bewältigt.
- Ein großes Entwicklungsteam stellt keine Einschränkung für den Einsatz von Grails dar. Es bietet eher den Vorteil, dass eine einheitliche Projekt-Struktur (per Konvention) vorliegt, an die sich alle Teammitglieder halten müssen.

Grails eignet sich hingegen nur bedingt für hochkomplexe Applikationen, bei denen für die Entwicklung alle Freiheitsgrade notwendig sind. So wird beispielsweise per Konvention eine Transaktion beim Verlassen eines Dialogs abgeschlossen. Für eine Dialogmaske, die verschachtelte Transaktionen benötigt, reicht dies nicht aus.

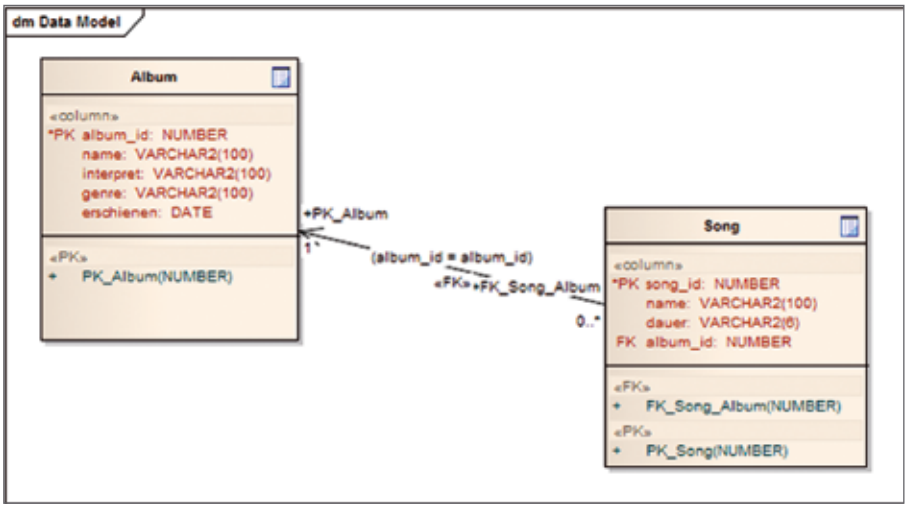

Abbildung 1: Datenmodell der Beispiel-Applikation

Durch eine entsprechende Konfiguration kann man dieses Verhalten ändern. Jedoch wird durch zu viele Konfigurationsänderungen irgendwann das Grundprinzip „Convention over Configuration“ verletzt.

Eine kleine Beispiel-Applikation

Die Vorzüge von Grails lassen sich am besten mit einer kleinen Applikation aufzeigen. Dazu als Beispiel eine CD-Verwaltung (siehe Abbildung 1). Listing 1 zeigt die Domain-Klasse für die Entität „Album“.

```
class Album {
  String interpret
  String name
  String genre
  Date erschienen

  static hasMany = [ songs :
Song ]

  static constraints = {
    interpret(nullable: false)
    name(nullable: false)
    genre(nullable: true)
  }

  String toString() {„${name} -
${genre}“}
}
```

Listing 1: Album.groovy

Interessant bei der Domain-Klasse „Album“ ist der Ausdruck „static hasMany = [songs : Song]“. Mit diesem wird die 1:n-Relation zwischen Album und Song definiert. Listing 2 zeigt die Domain-Klasse „Song“.

```
class Song {
  String name
  String dauer

  Album album

  static constraints = {
    name(nullable: false)
    dauer(nullable: false)
  }

  String toString() {„${name}“}
}
```

Listing 2: Song.groovy

Für die Erstellung der Applikation sind folgende Schritte nötig:

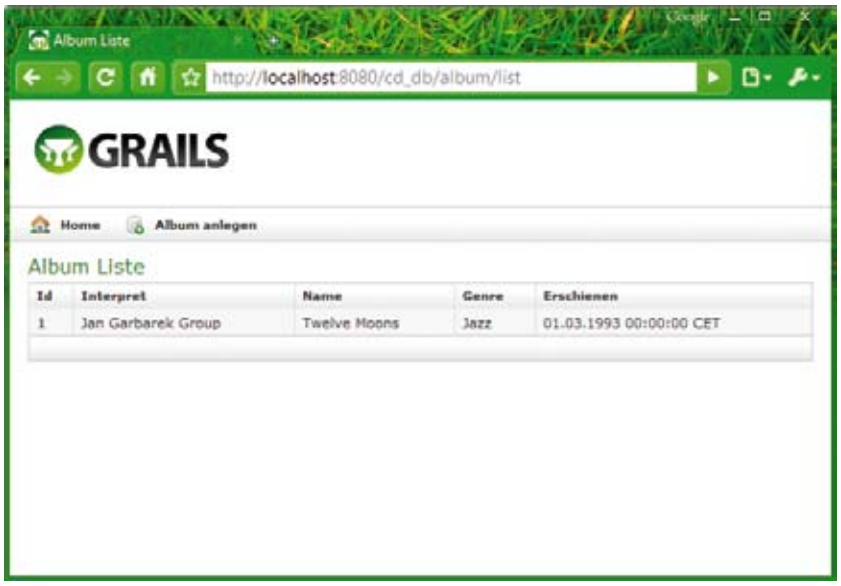

Abbildung 2: Dialog „Album Liste“

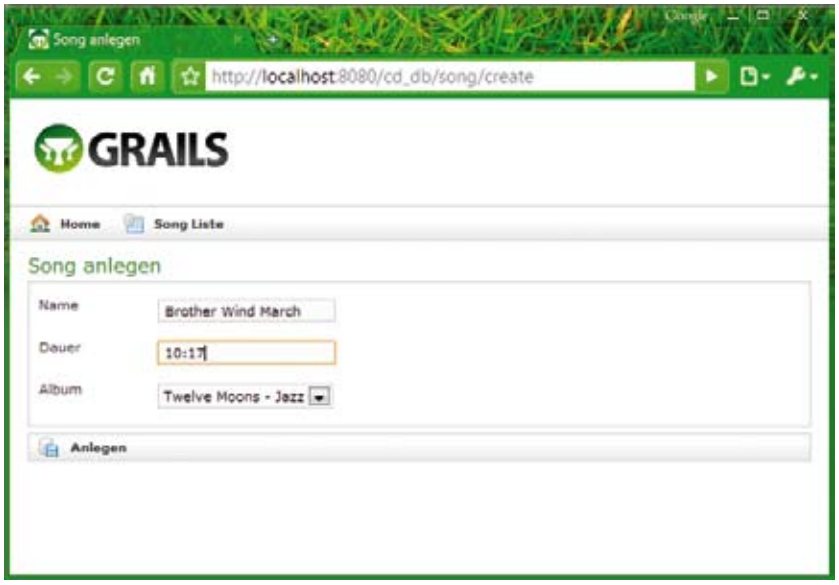

Abbildung 3: Dialog „Song anlegen“

- Erstellen des Grails-Applikations-Gerüsts mit dem Befehl „C:\Grails\projects>grails create-app cd_db“. Aus Platzgründen werden bei diesem Beispiel keine Package-Namen verwendet.
- Ablegen der beiden Domain-Klassen „Album“ und „Song“ im Verzeichnis „C:\Grails\projects\cd_db\grails-app\domain“
- Erzeugen der Domain-Klassen „Album“ und „Song“ mit den Befehlen:
  - C:\Grails\projects\cd_db>grails create-domain-class Album
  - C:\Grails\projects\cd_db>grails create-domain-class Song
- Erzeugen der User-Interface-Controller mit den Befehlen:
  - C:\Grails\projects\cd_db>grails generate-controller Album
  - C:\Grails\projects\cd_db>grails generate-controller Song
- Erzeugen der Views mit den Befehlen:
  - C:\Grails\projects\cd_db>grails generate-views Album
  - C:\Grails\projects\cd_db>grails generate-views Song
- Starten der Applikation mit dem Befehl:
  - C:\Grails\projects\cd_db>grails run-app“

Die Applikation wird nun automatisch kompiliert und auf den mitgelieferten Tomcat-Server aufgespielt. In der Konsole ist die generierte URL zu sehen. Abbildung 2 zeigt die Übersichtstabelle und Abbildung 3 den Dialog zum Anlegen eines Songs. Das Look-and-Feel lässt sich jetzt noch eigenen Anforderungen anpassen.

Deployment auf dem Oracle WebLogic-Server

Für das Deployment erweitert man das von Grails gebaute WAR-File um eine weblogic.xml-Datei. Hierzu sollte der Build-Prozess auf ein eigenes ANT-Skript umgestellt werden. Ein auf diese Weise erweitertes WAR-File kann dann auf alle Funktionalitäten des WebLogic-Servers zugreifen.

Fazit

Die über die letzten Projekte gesammelten Erfahrungen haben gezeigt, dass Grails ein Framework für eine hochproduktive Entwicklung von Web-Applikationen ist. Meistens hat man als Entwickler das Gefühl, dass Grails den eigenen Ideen folgt und man nicht den Ideen eines Tools hinterherläuft. Ein anschließendes Deployment auf dem Oracle WebLogic-Server stellt sicher, dass die erstellte Web-Applikation sehr gut skaliert. Darüber hinaus kann man auch auf alle Infrastruktur-Komponenten wie beispielsweise das Security Framework oder den Connection Pool zugreifen. Zudem bettet sich solch eine Lösung reibungslos in den IT-Betrieb ein.

Weitere Informationen

Neben den Standard-Links zu Grails ist vor allem die Artikelserie „Mastering Grails“ von Scott Davis zu empfehlen:

http://www.grails.org/
http://www.grails.org/GORM
http://www.grails.org/Success+Stories
http://www.ibm.com/developerworks/views/java/libraryview.jsp?search_by=mastering+grails

Kontakt:
Peter Soth
peter.soth@exensio.de

2010
DOAG
Konferenz + Ausstellung

16. – 18. November 2010 in Nürnberg: DOAG 2010 Konferenz + Ausstellung

Das Treffen der Oracle-Community

Mehr als 300 Fachvorträge • Keynotes bekannter Manager • Ausstellung • Networking

Impressum

Herausgeber:
DOAG Deutsche ORACLE-Anwendergruppe e.V.
Tempelhofer Weg 64
12347 Berlin
Tel.: 0700 11 36 24 38
www.doag.org

www.doag2010.org

Verlag:
DOAG Dienstleistungen GmbH
Fried Saacke, Geschäftsführer
info@doag-dienstleistungen.de

Chefredakteur (VisdP):
Wolfgang Taschner,
redaktion@doag.org

Chefin von Dienst (CvD):
Carmen Al-Youssef,
office@doag.org

Anzeigen:
Carmen Al-Youssef, office@doag.org
Mediadaten und Preise unter:
www.doag.org/publikationen/

Gestaltung und Satz:
Claudia Wagner,
DOAG Dienstleistungen GmbH

Druck:
adame Advertising and Media GmbH Berlin,
www.adame.de