New & cool

Grails is een open source framework voor agile web development met Groovy en Java. Grails combineert het Convention over Configuration concept met de facto Java frameworks en dit vereenvoudigt het bouwen van webapplicaties enorm. Grails reduceert complexiteit, en verhoogt productiviteit. Veel concepten in Grails zijn overgenomen uit dynamische frameworks als Ruby on Rails, Django en TurboGears. Het verschil is echter dat Grails is gebaseerd op bewezen Java frameworks als Spring en Hibernate.

Grails & Groovy

De zoektocht is voorbij

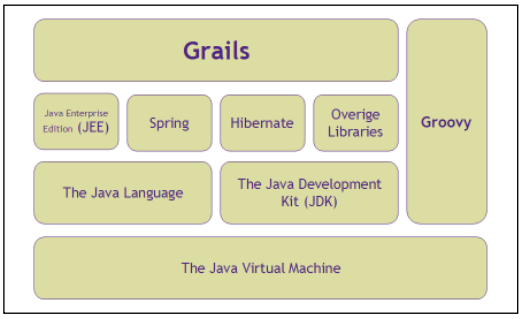
Marcel Overdijk
is Technisch Consultant bij Valid. E-mail: marcel.overdijk@valid.nl
Persoonlijke blog: http://marceloverdijk.blogspot.com

Groovy is een dynamische programmeertaal voor de Java Virtual Machine. De doelstelling van Groovy was om een programmeertaal te ontwikkelen waarbij Java ontwikkelaars eenvoudig kunnen overstappen naar de dynamic scripting wereld. De syntax van Groovy lijkt heel veel op Java en ook zijn dezelfde API's als in de JDK beschikbaar. Daarnaast bevat Groovy extra *power features*, zoals onder andere Closures en Builders, die overgenomen zijn uit programmeertalen als Python, Ruby en Smalltalk. Groovy compileert direct naar Java bytecode en kan dus overal worden gebruikt waar Java wordt gebruikt.

De Grails Stack

Grails is een full stack framework, gebaseerd op het *Model-View-Controller* pattern, en probeert zoveel mogelijk puzzelstukken uit de web development puzzel op te lossen. De Grails stack bestaat uit een aantal lagen:

- Eenvoudige Object Relational Mapping (ORM), gebaseerd op Hibernate
- Expressive view middels Groovy Server Pages
- De Controller, gebaseerd op Spring MVC
- Dependency Injection (DI) middels de Spring Container
- Transactionele service, gebaseerd op Spring's transaction abstraction
- Internationalisatie (i18n) gebaseerd op Spring's MessageSource concept
- Java EE Servlet API



Figuur 1: De Grails stack

Projectstructuur

Bij het creëren van een nieuwe applicatie creëert Grails automatisch de projectstructuur. Convention over Configuration speelt hierbij een grote rol. De naam en de locatie van files worden gebruikt in plaats van expliciete configuratie. Ontwikkelaars weten zo precies waar bepaalde Grails artifacts gevonden kunnen worden. Dit dwingt automatisch standaardisatie af en verhoogt ook de productiviteit.

Automagically

Op basis van conventie voegt Grails runtime *automagically* functionaliteit toe aan classes. Afhankelijk van het Grails artifact type worden runtime automatisch dynamische properties en methodes toegevoegd en Spring beans geconfigureerd. Bij het configureren van de Spring beans worden ook benodigde referentie beans automatisch geïnjecteerd (dependency injection). Hier is geen enkele XML configuratie voor nodig. Dit betekent dus ook veel minder onderhoud.

```
+ boekapp
+ grails-app
  + conf           - configuratie settings
  + controllers    - web controller artifacts
  + domain         - domain class artifacts
  + i18n           - i18n resource bundles
  + services       - service artifacts
  + taglib         - tag library artifacts
  + views          - view en layout templates
+ lib              - additionele libraries
+ scripts          - eigen scripts
+ src              - Groovy en Java sources
+ test             - integration en unit tests
+ web-app          - web resources zoals css, ima-
                     ges en javascrip
```

Figuur 2: Projectstructuur

Development Mode

In development mode maakt Grails de development lifecycle aanzienlijk korter, doordat wijzigingen aan Groovy classes direct worden opgepakt. Er hoeft dus geen nieuwe build te worden gemaakt en er is ook geen applicatie herstart nodig. Grails maakt tijdrovende *compile*, *build*, *deploy* processen tijdens development overbodig. In development mode gebruikt Grails Jetty als embedded servlet container.

GORM

Opslag van data in een relationele database staat centraal in iedere bedrijfsapplicatie. GORM is de Grails Object Relational Mapping implementatie die Grails gebruikt voor het opslaan en opvragen van data. Onder de motorkap gebruikt Grails hier Hibernate voor. In een notendop beschikt GORM over de volgende functionaliteit:

- Domain Modellering
- Associations (1:1, 1:m en m:m relaties)
- Basic Create/Read/Update/Delete (CRUD) methodes
- Dynamic finders voor het vereenvoudigd en snel querien
- Hibernate Criteria en Hibernate Query Language (HQL)
- Constraints voor het valideren van data
- Events en auto timestamping
- Custom ORM mappings voor met name het mappen van domain classes op een legacy database schema.

```
class Boek {
    String titel
    String auteur
    BigDecimal prijs
}
```

Codefragment 1: Boek.groovy

Codefragment 1 is een voorbeeld van een eenvoudige domain class. Voor het configureren van een domain class zijn geen Hibernate Annotations of XML mappings nodig. Door conventie behandelt Grails classes in grails-app/domain als domain classes en worden dynamische properties en methodes runtime automatisch toegevoegd. Automatisch voegt Grails ook de id property als primaire sleutel toe. In codefragment 2 zijn een aantal voorbeelden van de dynamisch toegevoegde methodes te zien.

```
// creëren van een nieuw boek
def b = new Boek(titel:"The Definitive Guide to
Grails", auteur:"Graeme Rocher", prijs:25.99)
b.save()

// ophalen van een boek op basis van het id
def b = Boek.get(1)

// ophalen van alle boeken
def boeken = Boek.list()

// wijzigen van een boek
def b = Boek.get(1)
b.prijs = 19.99
b.save()

// verwijderen van een boek
def b = Boek.get(1)
b.delete()

// enkele dynamic finder voorbeelden (veel meer
variaties mogelijk!)
def b = Boek.findByTitel("The Definitive Guide to
Grails")
b = Boek.findByTitelLike("%Grails%")
b  =  Boek.findByTitelLikeAndPrijsLessThan("%Grail
s%", 30.00)
```

Codefragment 2: Voorbeeld dynamische methodes

Met dynamic finders is heel veel mogelijk maar bij complexe criteria kan de code moeilijk leesbaar worden. In plaats van de dynamic finders kan dan ook Hibernate Criteria of de Hibernate Query Language (HQL) worden gebruikt. Als alternatief kan GORM ook gebruikt worden met Hibernate mapped Java classes. Zowel Hibernate XML mappings als Hibernate Annotations (EJB3/JPA) worden ondersteund. Grails voegt dan runtime alsnog de dynamische properties en methodes toe aan deze domain classes.

Scaffolding

Met scaffolding kan een complete applicatie worden gegenereerd. De gegenereerde applicatie bestaat uit een controller en bijbehorende views (Groovy Server Pages) en beschikt over basis CRUD functionaliteit. Scaffolding is vooral handig voor prototypes en demo's en kan daarnaast worden gebruikt als startpunt voor verdere ontwikkeling.

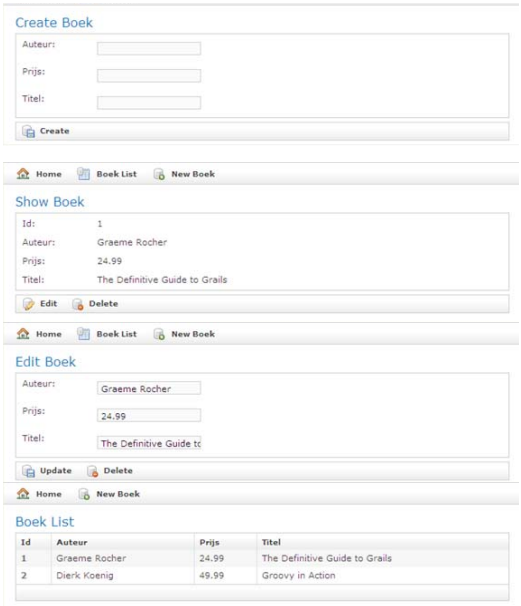
```
class BoekController {
    def scaffold = true
}
```

Codefragment 3: Scaffolding

Controllers

Controllers zijn verantwoordelijk voor het verwerken van requests, die meestal het gevolg zijn van handelingen van gebruikers. Onder water

GSP is flexibeler en intuïtiever dan JSP



Figuur 3: Scaffolding gegenereerde views

gebruikt Grails Spring MVC voor het verwerken van web requests. Grails controllers beschikken over de volgende functionaliteit:

- Eenvoudige en uniforme toegang tot servlet *context, session, request, request parameters* en *flash* variabelen
- Redirects en Chaining
- Interceptors
- Data Binding
- XML en JSON responses
- File uploads
- Flash scope

Codefragment 4 is een voorbeeld van een controller met twee eenvoudige actions. Actions kunnen data doorgeven aan views door het retourneren van een model. Een model is in feite een Map waarbij de key wordt gebruikt als de variabele naam. Door middel van conventie is het niet nodig aan te geven welke view moet worden gebruikt voor een action. De *show* action zal automatisch de show view gebruiken. Uiteraard is het wel mogelijk, indien nodig, om zelf te specificeren welke view moet worden gebruikt. Ook is het mogelijk om een instantie van Spring's ModelAndView class te retourneren.

```
class BoekController {

    def show = {
        def boek = Boek.get(params.id)
        return [boek:boek]
    }

    def save = {
        def boek = new Boek(params)
        boek.save()
          flash.message = "Boek ${boek.id} inge-
voerd"
        redirect(action:show, id:boek.id)
    }
}
```

Codefragment 4: BoekController.groovy

Groovy Server Pages

Groovy Server Pages, of in het kort GSP, zijn de views in Grails. Groovy Server Pages lijken heel erg veel op Java Server Pages (JSP). Echter GSP is flexibeler en intuïtiever. Het grote verschil is dat GSP gebruik maakt van Groovy expressions en dat JSP gebruik maakt van JSTL expressions die beperkter zijn wat betreft functionaliteit. Daarnaast bevat GSP dynamic tag libraries en kunnen tags zowel als normale tag worden aangeroepen en als method call. Grails beschikt standaard ook over een set van logische, iteratieve, form, validatie, layout en ajax tags. Codefragment 5 is een voorbeeld van een view behorende bij de show action uit de vorige paragraaf.

```
<html>
<body>
  <table>
      <tr>
          <td>Titel:</td>
          <%-- JSP-style output --%>
          <td><%=boek.titel%></td>
      </tr>
      <tr>
          <td>Auteur:</td>
          <%-- GSP Expression output --%>
          <td>${boek.auteur}</td>
      </tr>
      <tr>
          <td>Prijs:</td>
        <td><g:formatNumber number="${boek.prijs}"
format="#0.00" /></td>
      </tr>
    </table>
    <g:form>
     <input type="hidden" name="id" value="${boek?.
id}" />
        <g:actionSubmit value="Edit" />
        <g:actionSubmit value="Delete" />
    </g:form>
</body>
</html>
```

Codefragment 5: show.gsp

Dynamic Tag Libraries

Iedere developer is zich ervan bewust dat het geen goede eigenschap is om Java code (scriptlets) op te nemen in een JSP en in het geval van Grails in een GSP. De Servlet specificatie heeft hier custom tag libraries voor geïntroduceerd. Echter, alleen al de gedachte aan custom tag libraries

doet menig developer de tenen krullen. Custom tag libraries zijn een perfect voorbeeld waarom Java web development onnodig moeilijk kan zijn. Custom tag libraries moeten namelijk eerst worden gespecificeerd in een Tag Library Descriptor (TLD). Het probleem zit hier ook in het onderhoud. In Grails is het schrijven van een tag library veel eenvoudiger en productiever. Een Grails tag library is een simpele Groovy class, en door conventie kunnen de tag library tags automatisch gebruikt worden in de views, zonder dat er een TLD nodig is. Codefragment 6 is een voorbeeld van een Grails dynamic tag library en hoe deze wordt aangeroepen vanuit een view.

```
class BoekTagLib {

    static namespace = "boek"

def image = { attrs, body ->
    def baseUrl = "some server path"
    def titel = attrs.boek.titel
    def imageSrc = "${baseUrl}/${titel}.gif"
    out << '<img src="${imageSrc}" />'
}
}


// view code

<html>
<body>
  <boek:image boek="${boek}" />
  .. overige code show.gsp ..
</body>
</html>
```

Codefragment 6: BoekTagLib.groovy

Services

Grails beschikt over een aparte service laag. De service laag bevat de applicatie logica van de applicatie. Het is niet verstandig om applicatie logica op te nemen in controllers omdat dit hergebruik tegenhoud en geen goede eigenschap is van het splitsen van verantwoordelijkheden. Grails services zijn standaard transactioneel. Dit wil zeggen dat alle methodes van de service in een transactie worden gewrapped en dat er automatisch een rollback plaatsvindt in het geval een exceptie in een van methodes optreedt. Services maken ook gebruik van Spring's dependency injection. Sterker nog, Grails beschikt over *dependency injection by convention*. Op basis van de property naam kunnen services automatisch worden geïnjecteerd in bijvoorbeeld controllers en tag libraries. Hier is dus geen XML configuratie voor nodig. Codefragment 7 geeft hier een voorbeeld van.

AJAX

Grails heeft built-in ondersteuning voor *Asynchronous Javascript and XML* via de Grails AJAX tags. Standaard gebruikt Grails de Prototype library, maar het is mogelijk om andere libraries (bijvoorbeeld via de Dojo of Yahoo UI plugins) te gebruiken. Grails beschikt ook over zogenaamde

```
class BoekService {

def buyBoek() {
    ..
}
}


class BoekController {

    def boekService

    def buy = {
        def boek = Boek.get(params.id)
        boekService.buyBook(boek)
        ..
    }
}
```

Codefragment 7: Dependency injection van services

Automatic XML en *JSON Marshalling* functionaliteit via speciale converters. Hiermee kan een domain class, of een lijst van domain classes, automatisch worden geconverteerd naar XML of JSON. Deze functionaliteit is zeer handig in combinatie met AJAX zoals te zien is in codefragment 8. In het voorbeeld wordt de *remoteLink* tag gebruikt. Dit is een van de AJAX tags beschikbaar in Grails.

```
// controller action

import grails.converters.*
def show = {
def boek = Boek.get(params.id)
// return xml response
    render boek as XML
}


// view code

<g:javascript>
function updateBoek(e) {
// javascript code voor het updaten van de client
// uitlezen van de ontvangen xml
}
</g:javascript>
<g:remoteLink  action="show"  id="  ${boek.id}"
onSuccess="updateBoek(e)">
    Ververs Boek
</g:remoteLink>
```

Codefragment 8: AJAX voorbeeld

Testing

Geautomatiseerd testen is een belangrijk onderdeel van Grails. Standaard heeft Grails twee smaken van testen, namelijk unit tests en integratie tests. Met unit tests kunnen afzonderlijk methodes of code blokken worden getest zonder rekening te houden met de omliggende infrastructuur. In integratie tests is de gehele Grails applicatie beschikbaar. Het is dus mogelijk om in een controller test te verifiëren of er daadwerkelijk een record in de database is aangemaakt. Grails maakt hiervoor gebruik van een in-memory HSQLDB database, zodat tussen iedere test de database automatisch wordt opgeschoond. Grails gebruikt de Spring Mock Library en configureert bijvoorbeeld voor iedere controller test een *MockHttpServletRequest*,

MockHttpServletRequest en *MockHttpSession*. Ook is het mogelijk om functional tests te schrijven via de Canoo WebTest en Selenium plugins.

Plugins

Naast de core functionaliteit beschikt Grails ook over een flexibele plugin architectuur. Hiermee is de Grails functionaliteit volledig uitbreidbaar. De mogelijkheden van plugins zijn oneindig. Plugins kunnen basic Grails artifacts bevatten om bijvoorbeeld domain classes te hergebruiken in meerdere Grails applicaties. Ook kunnen plugins nieuwe artifact types toevoegen aan Grails. De Quartz plugin introduceert bijvoorbeeld het Job artifact, waarmee het mogelijk is op basis van conventie Jobs te definiëren. Daarnaast kunnen plugins participeren in runtime configuratie en dynamic methods runtime toevoegen. Er zijn momenteel al tal van plugins ontwikkeld door de Grails community waaronder testing, security, rich client, webservice en database migration plugins.

Tooling

De populaire IDE's Eclipse, NetBeans en IntelliJ IDEA beschikken alle drie over plugins voor Groovy en/of Grails development. Op dit moment beschikt IntelliJ over de meest geavanceerde plugin met wizards voor het creëren van een Grails project en Grails artifacts. Maar nog belangrijker, IntelliJ bevat code completion voor Groovy classes, Groovy server Pages en Grails dynamic methods. NetBeans 6.5 zal standaard uitgerust zijn met een plugin met ongeveer dezelfde functionaliteit. NewBeans 6.5 zal deze maand officieel worden gereleased. Eclipse heeft op dit moment alleen een Groovy plugin, maar deze is helaas nog niet vergelijkbaar wat betreft functionaliteit.

Conclusie

Grails is een vrij nieuw framework, maar de fundamenten (onder andere Spring en Hibernate) hebben zich al volop bewezen en zijn niet meer weg te denken uit hedendaagse webapplicaties. De combinatie van deze bewezen fundamenten en het Convention over Configuration concept dat Grails naar Java brengt, maakt van Grails een echt killer framework. Welke project verlangt immers niet naar minder complexiteit en een hogere productiviteit! «

Referenties

Grails Home: http://www.grails.org
Groovy Home: http://groovy.codehaus.org
The Definitive Guide to Grails: http://www.apress.com/book/view/1590597583
Getting Started with Grails: http://www.infoq.com/minibooks/grails
Groovy in Action: http://www.manning.com/koenig
Nederlandse Groovy & Grails usergroup: http://grupes.google.com/group/nlgug

Patches Patches Patches Patches Patches Patches Patches Pa

JSR 311 vrijgegeven

JAX-RS is een annotation-gebaseerde API voor het implementeren van RESTful web services , gebaseerd op HTTP, in Java. Klassen en methoden worden geannoteerd met informatie, die het mogelijk maakt ze runtime als resources aan te bieden.

SpringSource: nieuwe COO en meer binaries

De discussie over Spring over de Serverside is op 26 september opnieuw begonnen, nadat SpringSource Rob Bearden tot COO had benoemd. Bearden is tevens werkzaam is voor Benchmark Capital, een bedrijf dat tien miljoen Dollar in SpringSource heeft geïnvesteerd. Bearden is overigens voormalig COO bij JBoss. Omdat dit via theregister.co.uk bekend werd voordat SpringSource het op de eigen website had aangekondigd, was het aanleiding tot nogal wat speculaties. Mede daardoor ontstond de ellenlange discussie op theserverside waarin Rod Johnson zich waarschijnlijk veel vaker moest mengen dan hem lief was. Uiteindelijk schreef Johnson op 8 oktober een nieuwe verklaring die hij op de SpringSource-website zette, waarin deels tegemoet gekomen werd aan de eisen van de opensource-gemeenschap. Zo belooft SpringSource onder meer ook tussentijdse binaries beschikbaar te stellen.

GRails Application Generator (GRAG) v1.0 vrijgegeven

GRAG is een free domain object generator voor de Grails framework. Met GRAG kun je een bestaande database reverse engineeren om er je eigen Grails domain objects uit te generen.Grag zal Grails domain objecten met constraints en relations genereren. Zie verder http://grag.sourceforge.net/

MV Flex Expression Language (MVEL) 2.0 Bèta 1 vrijgegeven

Na bijna een jaar is MVEL 2.0 bijna af, en de bèta helemaal.
MVEL 2.0 schijnt duidelijk verbeterd te zijn op het gebied van performance, flexibiliteit en integratiemogelijkheden. Verder zijn er vele nieuwe features, vooral op scripting-gebied.

SpringSource dm Server vrijgegeven

SpringSource dm Server is een gehele modulaire, OSGi-gebaseerde Java server voor JEE- en Springapplicaties. Volgens SpringSource is het de enige applicatieserver die exclusief ontwikkeld is om Spring-applicaties te draaien, en daar is waarschijnlijk weinig tegenin te brengen.
De server is gebaseerd op standaarden als OSGi, Java-standaarden en uiteraard op Spring. De SpringSource dm Server biedt onder meer gemakkelijke applicatie- en resourcebibliotheek-upgrades, naast elkaar bestaande versie-deployments, en applicatie-monitoring en analyse van URL verkeer om er query-, cache- en transactie-statistieken op te leveren. De server is de oplossing voor applicatie dependency-problemen. Bovendien maakt de server incrementele deployments mogelijk zonder dat hij opnieuw opgestart hoeft te worden.