Grails

O'Reilly Media on [Safari Books Online](https://www.safaribooksonline.com/)

Ken Kousen

[ken.kousen@kousenit.com](mailto:ken.kousen@kousenit.com)

[@kenkousen](https://twitter.com/kenkousen?lang=en)

Link to this document: <http://bit.ly/2jamVbD>

This document is for notes that come up during class. It's also an easy way to share code and other materials.

It is shared as a Google Doc that is available to anyone with the link. The shortened link is given above.

Set-up info:

* Java 7 or 8, with a full SDK rather than just a JRE
* Groovy (optional; included with Grails, but can be useful separately)

IDEs:

* *IntellIJ IDEA*
  + **Community** edition supports Java (and Gradle projects)
  + **Ultimate** edition supports Spring and Spring Boot and Grails plugins

That's it. Everything else is self contained. I will be using:

* IntelliJ IDEA Ultimate
* Gradle (via a wrapper created as part of the project). You do NOT need to install Gradle separately. The first time you use the included wrapper script, it will download and install Gradle into your home directory under ~/.gradle/wrapper/dists

Groovy home page is at: <http://groovy-lang.org/>

Alternative JVM-based languages:

* Groovy
* Scala → statically typed language with functional characteristics
* Clojure → a Lisp for the JVM
* Kotlin → statically typed, null safe, produced by JetBrains (the company behind IntelliJ IDEA and other IDES)

In Groovy:

* semicolons are optional
* parentheses are usually optional

> println 'Hello, World'

* single-quoted strings are regular java.lang.String instances
* double-quoted strings are Groovy strings

Groovy interactive mode: groovysh

Groovy console: groovyConsole

In Java, java.lang is automatically imported

In Groovy:

* java.lang
* java.util
* java.io
* java.net
* java.math.BigInteger, java.math.BigDecimal
* groovy.lang
* groovy.util

All automatically imported

Groovy POGO examples:

import groovy.transform.\*

//@ToString

//@EqualsAndHashCode

//@TupleConstructor

@Canonical

class Person {

String first

String last

// void setLast(String last) {

// println 'inside setLast method'

// this.last = last

// }

// String toString() { "$first $last" } // last evaluated expression

// returned automatically

}

// POGO --> Plain Old Groovy Object

// class is public by default

// attributes (fields) are private by default

// methods are public by default

// Any attribute without public/private --> getters and setters

// are autogenerated

Person p = new Person()

p.setFirst('Guillaume') // autogenerated

p.last = 'Laforge' // using setLast method

println "${p.getFirst()} ${p.last}"

Person p1 = new Person('Guillaume', 'Laforge')

println p1

Person graeme1 = new Person(first: 'Graeme', last: 'Rocher')

Person graeme2 = new Person(first: 'Graeme', last: 'Rocher')

assert graeme1 == graeme2 // == operator invokes the .equals method

List people = [p, graeme1, graeme2, p1]

println people

println people.class.name

// assert people.class == ArrayList

assert people[0] == p

Map map = [a:1, b:2, c:2] // keys are assumed to be strings

println map

println map.keySet()

println map.values()

println map.getClass().name

assert map['a'] == 1 // overridden getAt method

assert map.a == 1 // overridden dot operator

assert map.get('a') == 1 // method from Java

people.each { person -> println person } // closure == braces + arrow

people.each { println it } // one-arg closures only have an "it" variable

people.eachWithIndex { person, idx ->

println "$idx is the index for $person"

}

map.each { k,v -> println "$k maps to $v" }

map.each { entry -> println "${entry.key} maps to ${entry.value}" }

// Collect method --> creates a new collection by

// applying a closure to each element of a collection

List lastNames = people.collect { it.last }

assert lastNames == ['Laforge', 'Rocher', 'Rocher', 'Laforge']

map.collect { k,v -> "$k=$v" } // converts map to list of key=value

assert map.collect { k,v -> "$k=$v" }.join('&') == 'a=1&b=2&c=2'

// Groovy

[3, 1, 4, 1, 5, 9].collect { it \*2 } // [6, 2, 8, 2, 10, 18]

.findAll { it % 3 == 0 } // [6, 18]

.sum() // 24

// Java

println java.util.stream.IntStream.of (3, 1, 4, 1, 5, 9).map { it \*2 } // [6, 2, 8, 2, 10, 18] like collect

.filter { it % 3 == 0 } // [6, 18] like findAll

.sum() // 24

Geocoder example:

String BASE = 'https://maps.googleapis.com/maps/api/geocode/xml?'

String encoded = ['1600 Ampitheatre Pkwy', 'Mountain View', 'CA'].collect {

URLEncoder.encode(it, 'UTF-8')

}.join(',')

String qs = "address=$encoded"

def root = new XmlSlurper().parse("$BASE$qs")

def loc = root.result[0].geometry.location

println "(${loc.lat},${loc.lng})"

Home page for Grails: <https://grails.org/>

### **Proxy configuration for Gradle (in case you need it):**

gradle.properties

systemProp.http.proxyHost=www.somehost.org  
systemProp.http.proxyPort=8080  
systemProp.http.proxyUser=userid  
systemProp.http.proxyPassword=password  
systemProp.http.nonProxyHosts=\*.nonproxyrepos.com|localhost

systemProp.https.proxyHost=www.somehost.org  
systemProp.https.proxyPort=8080  
systemProp.https.proxyUser=userid  
systemProp.https.proxyPassword=password  
systemProp.https.nonProxyHosts=\*.nonproxyrepos.com|localhost

Note: gradle.properties goes in either the project root (at which point it only affects the current project), or in ~/.gradle, where it would affect all projects on your system

Grails history:

* In 2005, Ruby on Rails
  + David Heinemeier Hansson (DHH)
  + Convention over configuration
  + Complete stack framework
  + Dry → Don't Repeat Yourself
* In 2007, Grails was released
  + Based on Spring and Hibernate
* Several of the original team members for both Groovy and Grails formed a company called G2One
* Bought by SpringSource
* SpringSource bought by VMware
* VMware split off their software development capabilities into Pivotal
  + All of the web sites for both Groovy and Grails branded with the name Pivotal
* In 2013, Pivotal announced they were no longer going to support Groovy or Grails financially
  + By end of March, Grails 3.0 was released
  + This only affected six developers (3 in Groovy, 3 in Grails)
* Groovy moved to the Apache Software Foundation
* Grails people joined OCI (based in St. Louis)
  + Team has rapidly expanded; currently about 15 devs
* The only casualty of the "Pivotal divorce" was GGTS (Groovy/Grails Tool Suite). Eclipse support is lacking

i18n → internationalization

hasMany → assign to a map

the key becomes the name of a Set<contained>

the key also is now an attribute of the domain class

Grails adds methods: addTo and removeFrom

Tells Hibernate to establish a one-to-many association

One the "contained" side, there are four options:

1. bi-directional with a cascade delete

static belongsTo = [quest: Quest]

"quest" attribute means you can navigate from task to quest

cascade delete means if you delete the Quest, you delete all the tasks

Now a Task can belong to only a single Quest at a time

Hibernate generates only two tables (Quest and Task)

1. uni-directional with a cascade delete

static belongsTo = Quest

The task could belong to more than one Quest

Hibernate will generate a join table

1. bi-directional without a cascade delete

Quest quest

Task has an attribute of type Quest

Remember, all attribute values are non-null by default

1. uni-directional without a cascade delete

We don't add anything about the Quest to the Task

3 tables instead of 2 (Quest, Task, Quest\_Task)

If you prefer not to do that:

**For IntelliJ IDEA:**

* clone repo or download zip and extract
* Use *File → Open* or if not project is open, *Import*
* Navigate to the ***build.gradle*** file inside the project
* Click enter and accept all the defaults

Actions in a controller are any public methods

URLs are mapped to controller actions, according to:

localhost:8080/controller/action?/id?

Three ways to return from a controller action:

* *render* → writes response directly to the output stream
* *redirect* → create a new HTTP request and direct the browser to a new URL (all existing request parameters are lost unless you add a params property)
* *return* → if the last expression is a map, then the entries are added to the request and you are forwarded to a GSP with the same name as the action (in a folder under grails-app/views/controller\_name)
* respond (front end for the others)

Default action for a controller: (URL without an action)

* if you have only one action, it's the default
* if you have an action called "index", it's the default
* if you set a property called "defaultAction", its value is the default

Source for the "Hello" app:

grails-app/controllers/hello/WelcomeController.groovy

package hello

class WelcomeController {

def index() {

redirect(action: 'greet', params: params)

}

def greet(String name) {

String n = name ?: 'World'

render "<h2>Hello, ${n}!</h2>"

}

def hi() {

String n = params.name ?: 'World'

[user:n]

}

}

grails-app/views/welcome/hi.gsp

<html>

<head>

<meta name="layout" content="main"/>

<title>Welcome Page</title>

</head>

<body>

<h2>Hello, ${user}!</h2>

</body>

</html>

src/test/groovy/hello/WelcomeControllerSpec.groovy

package hello

import grails.testing.web.controllers.ControllerUnitTest

import spock.lang.Specification

class WelcomeControllerSpec extends Specification implements ControllerUnitTest<WelcomeController> {

void 'greet without name returns Hello, World!'() {

when:

controller.greet()

then:

response.text == '<h2>Hello, World!</h2>'

}

void 'greet with a name returns Hello, name!'() {

when:

controller.greet('Dolly')

then:

response.text == '<h2>Hello, Dolly!</h2>'

}

void 'index without a name redirects to greet without a name'() {

when:

controller.index()

then:

response.redirectedUrl == '/welcome/greet'

}

void 'index with a name redirects to greet with a name'() {

when:

params.name = 'Dolly'

controller.index()

then:

response.redirectedUrl == '/welcome/greet?name=Dolly'

}

void 'hi without a name returns map with user=World'() {

when:

Map model = controller.hi()

then:

model.user == 'World'

}

void 'hi with a name returns map with user=name'() {

when:

params.name = 'Dolly'

Map model = controller.hi()

then:

model.user == 'Dolly'

}

}

Current version of HolyGrails app (first break, second day)

grails-app/domain/com/oreilly/Quest.groovy

package com.oreilly

class Quest {

String name

String toString() { name }

static hasMany = [tasks: Task]

static constraints = {

name(blank: false)

}

}

grails-app/domain/com/oreilly/Task.groovy

package com.oreilly

import groovy.transform.ToString

@ToString(includes = ['name', 'priority', 'completed'])

class Task {

String name

int priority = 3

Date startDate = new Date()

Date endDate = new Date()

boolean completed

static belongsTo = [quest: Quest]

static constraints = {

name(blank: false)

priority range: 1..5

startDate()

endDate validator: { val, task ->

val >= task.startDate

}

completed()

}

}

Added to i18n messages.properties:

quest.name.nullable=Quests must have a non-null name

If you want to change the "context" (i.e., the URL for the app)

In application.yml:

server:

context-path: /holygrails

// contextPath also works

Transient properties are not saved (or even mapped) to the DB

static transients = ['duration']

Hibernate generated and managed properties:

Long id // not shown, but added by default as primary key

Integer version // used for "optimistic locking"

Date dateCreated → Hibernate will set to first saved date

Date lastUpdated → Hibernate will set whenever tx commits

Pessimistic locking → Grails adds a lock() method to each instance

Optimistic locking → Grails adds a column (default "version" of int type) that is incremented by Hibernate whenever a transaction commits

Some sample queries:

dynamic finders

Quest.findAllByNameLike('%e%')

Task.findAllByNameLikeAndPriorityGreaterThanEquals('%e%', 2)

criteria queries

Task.withCriteria {

like('name', '%e%')

gte('priority', 2)

between('startDate', new Date() - 1, new Date() + 1)

}

Task.withCriteria {

priority >= avg(priority)

}

Quest.withCriteria {

ilike('name', '%e%')

tasks { // inner join

ilike('name', '%e%')

gte('priority', 3)

}

}

where queries

Task.findAll {

priority >= 2 && completed

}

Many more examples in the User Manual section on GORM

<http://gorm.grails.org>

BootStrap.groovy → used to initialize your app

init closure executed on startup

not usually needed if you are working with an existing database

Final version of the Holy Grails app:

Quest.groovy

package com.oreilly

class Quest {

String name

Date dateCreated

Date lastUpdated

String toString() { name }

static hasMany = [tasks: Task, knights: Knight]

static constraints = {

name(blank: false)

}

}

Task.groovy

package com.oreilly

import groovy.transform.ToString

@ToString(includes = ['name', 'priority', 'completed'])

class Task {

String name

int priority = 3

Date startDate = new Date()

Date endDate = new Date()

boolean completed

int getDuration() { endDate - startDate + 1 }

static transients = ['duration']

static belongsTo = [quest: Quest]

static constraints = {

name(blank: false)

priority range: 1..5

startDate()

endDate validator: { val, task ->

val >= task.startDate

}

completed()

}

}

Knight.groovy

package com.oreilly

class Knight {

String title

String name

Quest quest

Castle castle

String toString() { "$title $name" }

static constraints = {

title inList: ['Sir', 'Lord', 'Lady', 'King', 'Queen']

name blank: false

quest nullable: true

castle nullable: true

}

}

Castle.groovy

package com.oreilly

class Castle {

String name

String city

String state

double latitude

double longitude

String toString() { "$name Castle" }

static hasMany = [knights: Knight]

static constraints = {

name blank: false

city blank: false

state blank: false

latitude min: -90d, max: 90d

longitude()

}

}

GeocoderService.groovy

package com.oreilly

import grails.gorm.transactions.Transactional

@Transactional

class GeocoderService {

Castle fillInLatLng(Castle castle) {

String base = 'http://maps.googleapis.com/maps/api/geocode/xml?'

String encoded = [castle.city, castle.state].collect {

URLEncoder.encode(it, 'UTF-8')

}.join(',')

String qs = "address=$encoded"

def root = new XmlSlurper().parse("$base$qs")

def loc = root.result[0].geometry.location

castle.latitude = loc.lat.toDouble()

castle.longitude = loc.lng.toDouble()

castle

}

}

GeocoderServiceSpec.groovy

package com.oreilly

import grails.testing.gorm.DomainUnitTest

import grails.testing.services.ServiceUnitTest

import spock.lang.Specification

class GeocoderServiceSpec extends Specification implements ServiceUnitTest<GeocoderService>,

DomainUnitTest<Castle> {

void 'verify lat,lng from Mountain View, CA'() {

given:

Castle google = new Castle(name: 'Google', city: 'Mountain View', state: 'CA')

when:

service.fillInLatLng(google)

then:

(google.latitude - 37.4).abs() < 0.1

(google.longitude - -122.1).abs() < 0.1

}

}

CastleController.groovy

GeocoderService geocoderService

In save and update methods:

geocoderService.fillInLatLng(castle)

BootStrap.groovy

package holygrails

import com.oreilly.Castle

import com.oreilly.GeocoderService

import com.oreilly.Quest

class BootStrap {

GeocoderService geocoderService

def init = { servletContext ->

Quest q = new Quest(name: 'Seek the grail')

.addToTasks(name: 'Run away from killer rabbit')

.addToTasks(name: 'Answer the Bridge Keeper', priority: 4)

.addToTasks(name: 'Defeat the Black Knight', completed: true)

.save()

Castle camelot = new Castle(name: 'Camelot', city: 'Boston', state: 'MA')

.addToKnights(title: 'King', name: 'Arthur', quest: q)

.addToKnights(title: 'Sir', name: 'Lancelot', quest: q)

.addToKnights(title: 'Sir', name: 'Robin', quest: q)

geocoderService.fillInLatLng(camelot)

.save()

Castle swamp = new Castle(name: 'Swamp', city: 'Framingham', state: 'MA')

geocoderService.fillInLatLng(swamp).save()

}

def destroy = {

}

}

grails-app/views/castle/index.gsp

In header:

<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>

<script type="text/javascript"

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCfjf9Il5PFGctk1DUwXK83Vdvng5M71cg"></script>

In body, before table:

<script type="text/javascript">

google.charts.load('current', { 'packages': ['map'] });

google.charts.setOnLoadCallback(drawVisualization);

function drawVisualization() {

var map = new google.visualization.Map(document.getElementById('map\_div'));

var table = new google.visualization.DataTable();

table.addColumn('number', 'Lat');

table.addColumn('number', 'Lng');

table.addColumn('string', 'Name');

$.getJSON('/holygrails/castle/index.json').done(function (data) {

$.each(data, function (i, castle) {

table.addRow([castle.latitude, castle.longitude, castle.name]);

});

map.draw(table, {showTip: true});

});

}

</script>

<div id="map\_div"></div>

Note: generate-all takes a domain class as an argument and generates both the controller and views

Use the argument \* to generate all for all existing domain classes

grails-app/views/castle/create.gsp

In form:

<f:with bean="castle">

<f:field property="name"/>

<f:field property="city"/>

<f:field property="state"/>

</f:with>

%{-- <fieldset class="form"> <f:all bean="castle"/> </fieldset> --}%

References:

Grails in Action, 2nd edition, Ledbrook and Smith (Manning)

<https://www.safaribooksonline.com/library/view/grails-in-action/9781617290961/>

Grails 2: A Quick-Start Guide, Klein and Klein (Pragmatic Programmers)

<https://www.safaribooksonline.com/library/view/grails-2-a/9781941222652/>

Definitive Guide to Grails 2, Brown and Rocher, Apress

<https://www.safaribooksonline.com/library/view/the-definitive-guide/9781430243779/>

Also videos:

Starting a Grails 3 Project

<https://www.safaribooksonline.com/library/view/starting-a-grails/9781491988046/>

Understanding the Grails 3 Domain Model

<https://www.safaribooksonline.com/library/view/understanding-the-grails/9781491988060/>

Working with Grails 3 Controllers and Services

<https://www.safaribooksonline.com/library/view/working-with-grails/9781491988084/>

Advanced Database and REST Principles in Grails 3

<https://www.safaribooksonline.com/library/view/advanced-databases-and/9781491988107/>