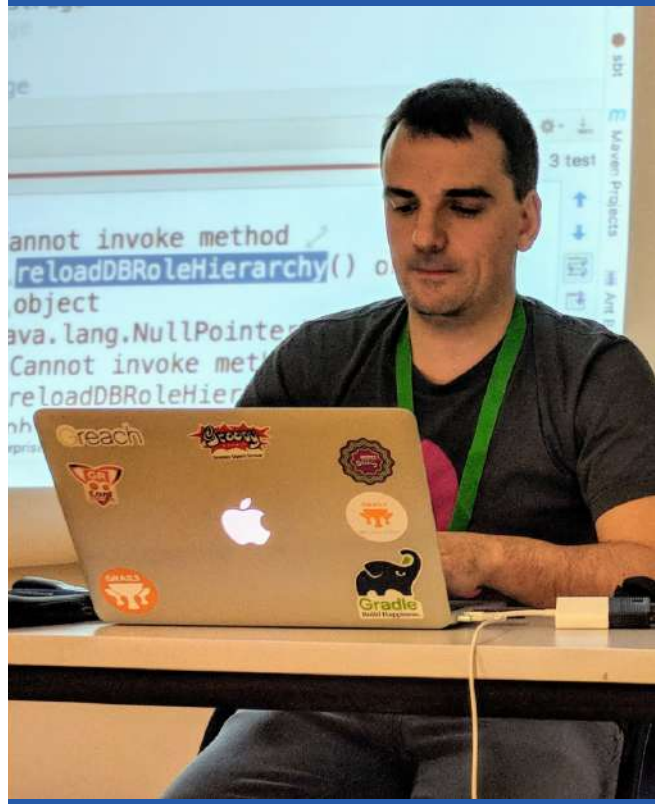
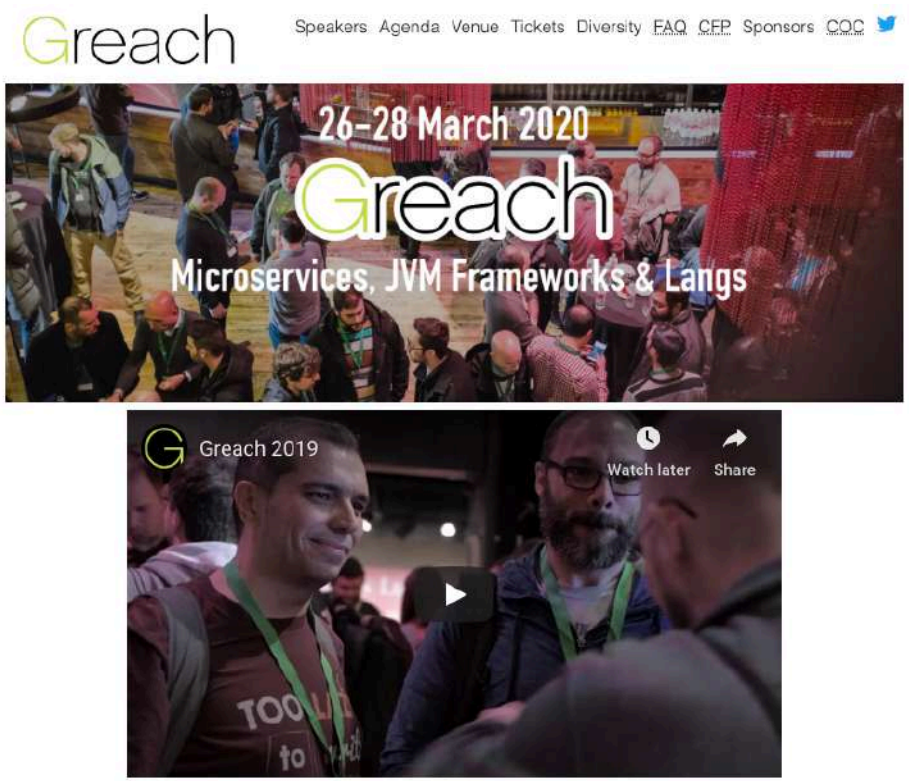
What's new with Grails 4

SERGIO DEL AMO

- SENIOR ENGINEER AT OCI SINCE JANUARY 2017
- MICRONAUT / GRAILS OCI TEAM
- GUADALAJARA, SPAIN
- CURATOR OF GROOVYCALAMARI.COM
- PODCAST HOST OF PODCAST.GROOVYCALAMARI.COM
- GREACH Conference organizer
- @SDELAMO
- HTTP://SERGIODELAMO.ES



https://greachconf.com

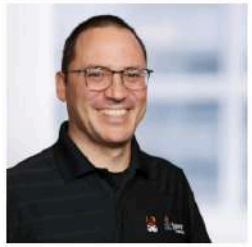


Microservices, JVM Frameworks & JVM Langs

Trisha Gee

Trisha has expertise in Java high performance systems, is passionate about community in tech, and dabbles with Open Source development
Web

Andres Almiray



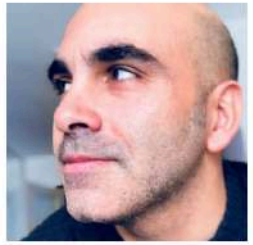
JSR377 Spec lead, Groovy aficionado, Griffon project lead, Java Champion, JCP EC, Hackergarten. Senior Principal Product Manager for Oracle.
Web

Vladimír Oraný

Vladimir is Test Facilitator at Agorapulse. He is interested in code quality, domain-specific languages, cloud computing and writing magical code using AST transformations.
Web

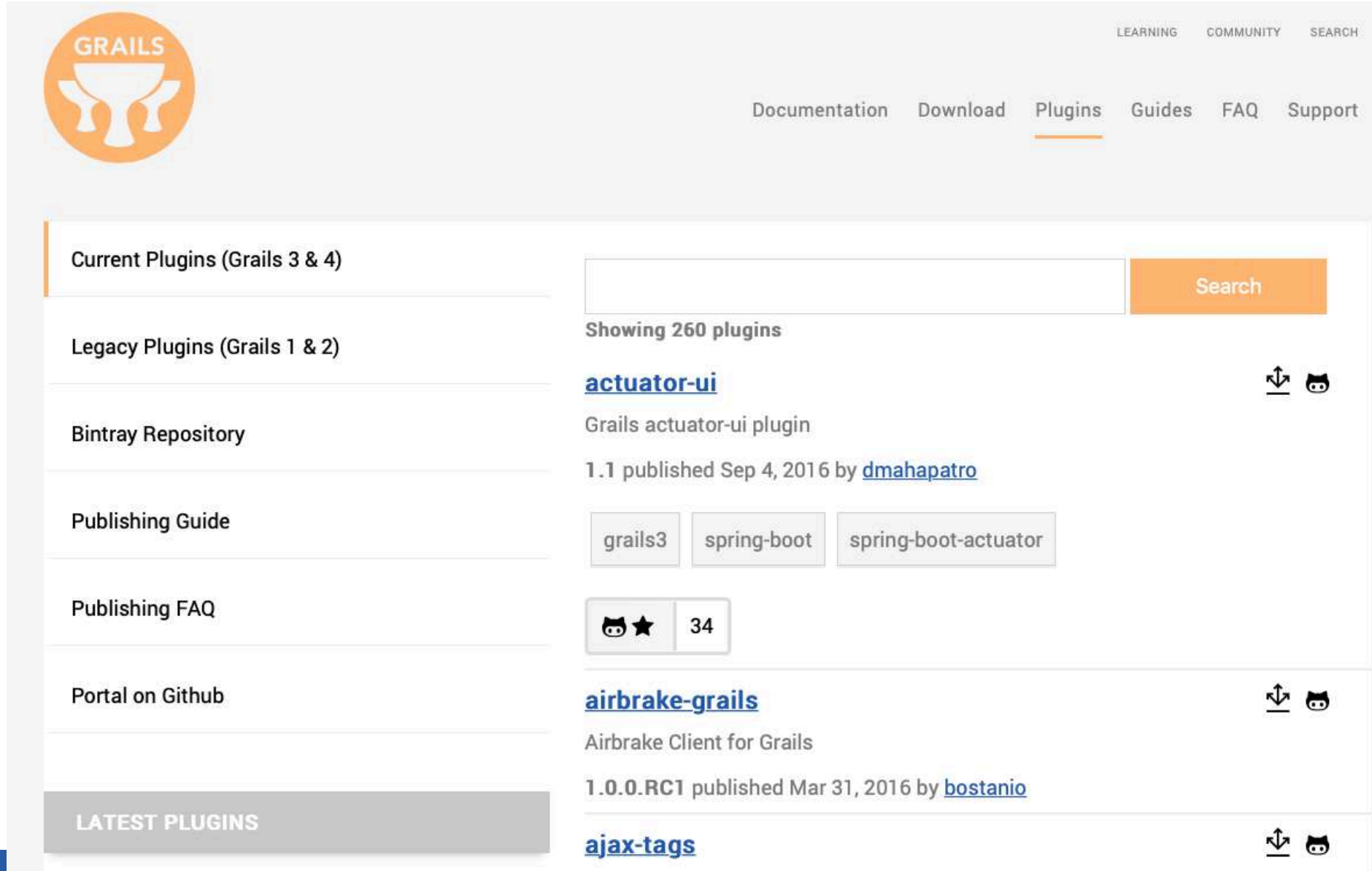
Anton Rodriguez

David Bonilla

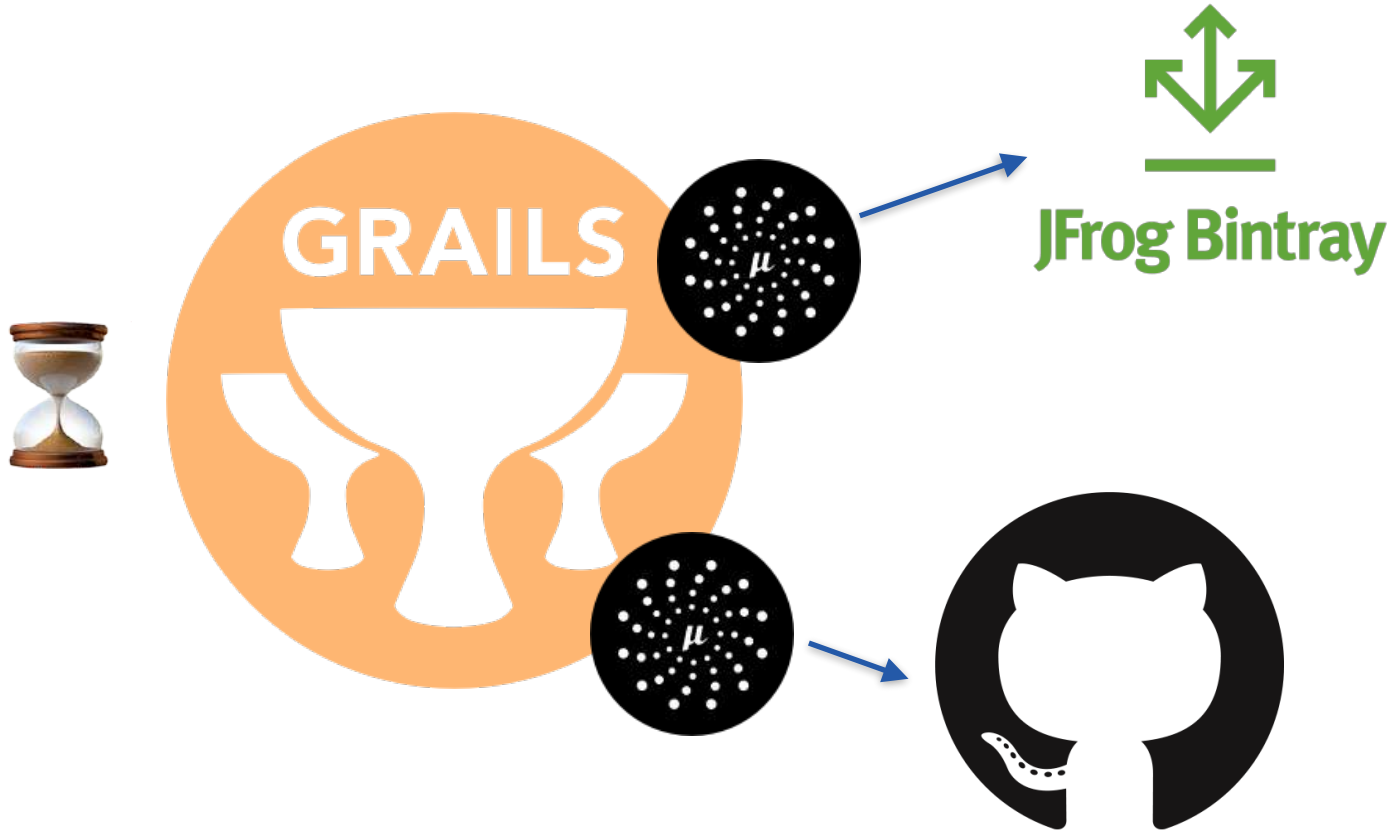


Michael Kutz

plugins.grails.org



plugins.grails.org



Agenda

1. What's New
2. Upgrade Challenges
3. Micronaut Integration

Grails 4 History

OBJECT COMPUTING

Upgrade started October 2017

Then mostly delayed until late 2018

Milestone 1 released in February 19th 2019

Milestone 2 released in March 26th 2019

Release Candidate 1 released in 17th April 2019

Grails 4.0.0 GA released in July 11th 2019

Grails 4.0.1 GA released in Oct 14th 2019

What's New in Grails 4

1. Java 8 Minimum
2. Groovy 2.5.6
3. Spring Boot 2.1.9
4. Spring 5.1.10
5. GORM 7 / Hibernate 5.4.0
6. Gradle 5.1.1
7. Spock 1.2-groovy-2.5
8. Micronaut Integration 1.1.4

Upgrading to Grails 4

OBJECT COMPUTING

Grails 2 ↓ Grails 3 != Grails 3 ↓ Grails 4

Upgrading

OBJECT COMPUTING

1. Most deprecations have been removed
2. Some package restructuring
3. No major breaking API changes
4. Most plugins should just work
5. Breaking changes in Spring, Hibernate, Groovy

Migration Steps - Bump up Grails Version

OBJECT COMPUTING

You will need to upgrade your Grails version defined in `gradle.properties`

```
grailsVersion=4.0.1
```

Spring Boot Changes

OBJECT COMPUTING

https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-2.0-Migration-Guide

Notable:

1. Many configuration changes
2. Embedded container API
3. Endpoints Changes

Migration Steps 3 - Spring Boot 2.1 Actuator changes

OBJECT COMPUTING

Actuator have changed substantatially from Spring Boot 1.5 used by Grails 3 to Spring Boot 2.

Grails 3

```
endpoints:
  enabled: false
  jmx:
    enabled: true
    unique-names: true
```

Grails 4

```
spring:
  jmx:
    unique-names: true
management:
  endpoints:
    enabled-by-default: false
```

Migration Steps 4 - Spring Boot Developer Tols

build.gradle

```
.
..
...
configurations {
    developmentOnly
    runtimeClasspath {
        extendsFrom developmentOnly
    }
}

dependencies {
   developmentOnly("org.springframework.boot:spring-boot-devtools")
    ...
```

grails-app/conf/application.yml

```
spring:
    devtools:
        restart:
            additional-exclude:
                - '*.gsp'
                - '**/*.gsp'
                - '*.gson'
                - '**/*.gson'
                - 'logback.groovy'
                - '*.properties'
```

Spring Changes

OBJECT COMPUTING

https://github.com/spring-projects/spring-integration/wiki/Spring-Integration-4.3-to-5.0-Migration-Guide

Nothing that should impact the average Grails application

Gradle Changes

OBJECT COMPUTING

https://docs.gradle.org/current/userguide/upgrading_version_4.html

Gradle 3 no longer supported, 4 not officially supported

Many breaking changes from 3 to 5

```
./gradlew wrapper --gradle-version 5.4.1
```

Gradle - Transitive dependencies not resolved for plugins

https://docs.gradle.org/current/userguide/upgrading_version_4.html#rel5.0:pom_compile_runtime_separation

build.gradle - Grails 3

```
dependencies {
  compile 'org.grails.plugins:rendering:2.0.3'
  …
}
```

build.gradle - Grails 4

```
dependencies {
  compile 'org.grails.plugins:rendering:2.0.3'
  compile("org.xhtmlrenderer:core-renderer:R8")
  compile("com.lowagie:itext:2.1.0"
…
}
```

Migration Steps 5 - Spring Boot Gradle Plugin Changes

OBJECT COMPUTING

Spring Boot Gradle plugin's documentation.

build.gradle - Grails 3

```
bootRun {
   addResources = true
   …
}
```

build.gradle - Grails 4

```
bootRun {
   sourceResources sourceSets.main
   …
}
```

Migration Steps 6 - Building executable jars for Grails Plugins

OBJECT COMPUTING

Spring Boot Gradle plugin's documentation.

build.gradle - Grails 3

```
bootRepackage.enabled=false
```

build.gradle - Grails 4

```
bootJar.enabled=false
```

GORM / Hibernate Changes

OBJECT COMPUTING

http://gorm.grails.org/7.0.x/hibernate/manual/index.html#upgradeNotes

Notable:

1. All operations now require a transaction
2. Proxy behavior has changed
3. No more REST client

GORM / Hibernate Changes

OBJECT COMPUTING

Package Restructuring and Deprecations

Previously deprecated classes have been deleted from this release.

In order to support Java 11 modules in the future some package re-structuring has occurred.

OBJECT COMPUTING

Changes to Proxy Handling

GORM no longer creates custom proxy factories nor automatically unwraps Hibernate proxies.

GORM / Hibernate Proxy Changes

OBJECT COMPUTING

GRAILS 3

```
class Pet {
    String name
}
class Dog extends Pet {}
class Person {
    String name
    Pet pet
}
def person = Person.get(1)
assert person.pet instanceof Dog

assert person.pet.instanceOf(Dog)

assert Pet.get(person.petId).instanceOf(Dog)

assert Pet.get(person.petId) instanceof Dog
```

GORM / Hibernate Proxy Changes

OBJECT COMPUTING

GRAILS 4

```
class Pet {
    String name
}
class Dog extends Pet {}
class Person {
    String name
    Pet pet
}
def person = Person.get(1)
assert person.pet instanceof Dog

assert person.pet.instanceOf(Dog)

assert Pet.get(person.petId).instanceOf(Dog)

assert Pet.get(person.petId) instanceof Dog
```

GORM / Hibernate Changes

OBJECT COMPUTING

grails-validation Deprecated and Removed

Gorm 6.x the *grails-validation* module was deprecated and replaced by *grails-datastore-gorm-validation*.

Deprecated interfaces were maintained for backwards compatibility. In Gorm 7.0, these depreacted classes have been removed and all dependency on *grails-validation* removed.

GORM / Hibernate Changes

OBJECT COMPUTING

Transactions Now Required for all Operations

Previous versions of Hibernate allowed read operation to be executed without the presence of a declared transaction.

Hibernate 5.2. and above require the presence of an active transaction. If you see a *javax.persistence.TransactionRequiredException* exception, it means your method lacks *@Transactional* annotation around it.

Migration Steps - Bump up GORM Version

OBJECT COMPUTING

You will need to upgrade your GORM version defined in `gradle.properties`

```
gorm.version=7.0.2.RELEASE
```

Migration Steps 5 - Upgrading Hibernate

OBJECT COMPUTING

build.gradle - Grails 3

```
dependencies {
  …
 compile “org.grails.plugins.hibernate5”
 compile “org.hibernate:hibernate-core:5.1.5.Final”
}
```

build.gradle - Grails 4

```
dependencies {
  …
 compile “org.grails.plugins.hibernate5”
 compile “org.hibernate:hibernate-core:5.4.0.Final”
}
```

Rest client Builder Grails Plugin Removal

OBJECT COMPUTING

org.grails:grails-datastore-rest-client deprecated in favor of Micronaut HTTP Client.

Rest client Builder - Grails 3

```
String uri = "http://repo.grails.org/grails/api/security/groups/test-group"
def resp = rest.put(uri) {
    auth System.getProperty("artifactory.user"),System.getProperty("artifactory.pass")
    contentType "application/vnd.org.jfrog.artifactory.security.Group+json"
    json {
        name = "test-group"
        description = "A temporary test group"
    }
}
```

Rest client Builder Grails Plugin Removal

OBJECT COMPUTING

org.grails:grails-datastore-rest-client deprecated in favor of Micronaut HTTP Client.

Micronaut HTTP Client - Grails 4

```
Map<String, Object> payload = [name: "test-group",
                               description: "A temporary test group”]
String uri = "http://repo.grails.org/grails/api/security/groups/test-group"
HttpRequest request = HttpRequest.PUT(uri,payload)
        .basicAuth(System.getProperty("artifactory.user"), System.getProperty("artifactory.pass"))
        .contentType("application/vnd.org.jfrog.artifactory.security.Group+json")
HttpResponse resp = client.toBlocking().exchange(request)
```

Groovy Changes

OBJECT COMPUTING

http://groovy-lang.org/releasenotes/groovy-2.5.html

Notable:

1. No more “all” jar 👍
2. New annotations
3. Date extensions require a new dependency
4. JDK 11 warnings not resolved

Plugins

OBJECT COMPUTING

Plugins should just work unless…

They use an API that has been changed or removed

GrailsDomainClass https://docs.grails.org/latest/guide/upgrading.html#_grails_domain_class_api_deprecated

Spring Boot Embedded Server

Migration Steps - Geb - from Geb 1.x to Geb 2.x

OBJECT COMPUTING

build.gradle - Grails 3

```
dependencies {
 testCompile “org.grails.plugins:geb:1.1.2”
 testRuntime “org.seleniumhq.selenium:selenium-htmlunit-driver:2.47.1”
 testRuntime “net.sourceforge.htmlunit:htmlunit:2.18”
```

build.gradle - Grails 4

```
dependencies {
 testCompile “org.grails.plugins:geb”
 testCompile “org.seleniumhq.selenium:selenium-remote-driver:3.141.59”
 testCompile “org.seleniumhq.selenium:selenium-api:3.141.59”
 testCompile “org.seleniumhq.selenium:selenium-support:3.141.59”
 testRuntime “org.seleniumhq.selenium:selenium-chrome-driver:3.141.59”
 testRuntime “org.seleniumhq.selenium:selenium-firefox-driver:3.141.59”
```

Migration Steps - Geb - Webdriver binaries Gradle plugin

OBJECT COMPUTING

build.gradle - Grails 4

```
buildscript {
    repositories {
        ...
    }
    dependencies {
        ...
        classpath "gradle.plugin.com.github.erdi.webdriver-binaries:webdriver-binaries-gradle-plugin:2.1"
    }
}
...

apply plugin:"com.github.erdi.webdriver-binaries"


webdriverBinaries {
    chromedriver "78.0.3904.105"
    geckodriver "0.24.0"
}
```

Migration Steps - Geb - Webdriver binaries

OBJECT COMPUTING

build.gradle - Grails 4

```
tasks.withType(Test) {
    systemProperty "geb.env", System.getProperty('geb.env')
    systemProperty "geb.build.reportsDir", reporting.file("geb/integrationTest")
    systemProperty "webdriver.chrome.driver", System.getProperty('webdriver.chrome.driver')
    systemProperty "webdriver.gecko.driver", System.getProperty('webdriver.gecko.driver')
}
```

Migration Steps - GebConfig

OBJECT COMPUTING

src/integration-test/resources/GebConfig.groovy - Grails 4

```
import org.openqa.selenium.chrome.ChromeDriver
import org.openqa.selenium.chrome.ChromeOptions
import org.openqa.selenium.firefox.FirefoxDriver

environments {

    // run via “./gradlew -Dgeb.env=chrome iT”
    chrome {
        driver = { new ChromeDriver() }
    }

    // run via “./gradlew -Dgeb.env=chromeHeadless iT”
    chromeHeadless {
        driver = {
            ChromeOptions o = new ChromeOptions()
            o.addArguments('headless')
            new ChromeDriver(o)
        }
    }

    // run via “./gradlew -Dgeb.env=firefox iT”
    firefox {
        driver = { new FirefoxDriver() }
    }
}
```

Migration Steps - Asset Pipeline

OBJECT COMPUTING

build.gradle - Grails 3

http://www.asset-pipeline.com/manual/index.html

```
buildscript {
  dependencies {
    ..
    classpath: “com.bertramlabs.plugins:asset-pipeline-grails:2.14.1”
  }
}
apply plugin “asset-pipeline”
dependencies {
  ..
  runtime “com.bertramlabs.plugins:asset-pipeline-grails:2.14.1”
```

build.gradle - Grails 4

```
buildscript {
 ..
 dependencies {
    classpath: “com.bertramlabs.plugins:asset-pipeline-grails:3.0.11”
    …
}
…
apply plugin “com.bertramlabs.asset-pipeline”
…
dependencies {
  …
  runtime “com.bertramlabs.plugins:asset-pipeline-grails:3.0.11”
```

Migration Steps - Spring Security Core

OBJECT COMPUTING

https://grails-plugins.github.io/grails-spring-security-core/

build.gradle - Grails 3

```
dependencies {
  …
  compile “org.grails.plugins:spring-security-core:3.2.0”
}
```

build.gradle - Grails 4

```
dependencies {
  …
  compile “org.grails.plugins:spring-security-core:4.0.0.RC2”
}
```

Migration Steps - Spring Security Core

OBJECT COMPUTING

grails-app/domain/example/User.groovy - Grails 3

```
class User {
  SpringSecurityService springSecurityService
  …
  …
  def beforeInsert() { encodePassword() }
  def beforeUpdate() {
    if (isDirty('password')) { encodePassword() }
  }
  protected void encodePassword() {
    password = springSecurityService?.passwordEncoder ? springSecurityService.encodePassword(password)
  }
}
```

Migration Steps - Spring Security Core

OBJECT COMPUTING

grails-app/domain/example/User.groovy - Grails 4

```
class User {
  SpringSecurityService springSecurityService
  …
  …
  def beforeInsert() { encodePassword() }
  def beforeUpdate() {
    if (isDirty('password')) { encodePassword() }
  }
  protected void encodePassword() {
    password = springSecurityService?.passwordEncoder ? springSecurityService.encodePassword(password)
  }
}
```

Migration Steps - Spring Security Core

src/main/groovy/example/UserPasswordEncoderListener.groovy - Grails 4

```
import grails.plugin.springsecurity.SpringSecurityService
import org.grails.datastore.mapping.engine.event.*
import org.springframework.beans.factory.annotation.Autowired
import grails.events.annotation.gorm.Listener
@CompileStatic
class UserPasswordEncoderListener {
    @Autowired
    SpringSecurityService springSecurityService

    @Listener(User) void onPreInsertEvent(PreInsertEvent event) { encodePasswordForEvent(event) }

    @Listener(User) void onPreUpdateEvent(PreUpdateEvent event) { encodePasswordForEvent(event) }

    private void encodePasswordForEvent(AbstractPersistenceEvent event) {
        if (event.entityObject instanceof User) {
            User u = event.entityObject as User
            if (u.password && ((event instanceof  PreInsertEvent) || (event instanceof PreUpdateEvent &&
u.isDirty('password')))) {
                event.getEntityAccess().setProperty('password', encodePassword(u.password))
            }
        }
    }
    private String encodePassword(String password) {
        springSecurityService?.passwordEncoder ? springSecurityService.encodePassword(password) : password
    }
}
```

Migration Steps - Spring Security Core

grails-app/conf/spring/resources.groovy - Grails 4

```
import example.UserPasswordEncoderListener
beans = {
    userPasswordEncoderListener(UserPasswordEncoderListener)
}
```

Migration Steps - Spring Security Core

OBJECT COMPUTING

Spring Security 5 changed the way password are encoded and compared for matches.

{bcrypt}someencryptedpassword // using bcrypt

{noop}planintextpassword

OBJECT COMPUTING

MICRONAUT

Micronaut Integration

- Micronaut a Foundational Library for building applications of any type
- Focuses on Small Memory Footprint and Speed
- Eliminates Reflection, Runtime Proxies and Runtime Analysis

Micronaut Integration

OBJECT COMPUTING

Micronaut has also been used to improve startup and reduce overall memory consumption of Grails applications (along associated improvements in Spring Boot 2.1)

Micronaut Messaging

OBJECT COMPUTING

- Micronaut Supports Message-Driven Applications
- Declarative Clients for Kafka & RabbitMQ
- Use @RabbitListener for RabbitMQ
- Use @KafkaListener for Kafka
- Planned Support For Other Messaging Systems

EXAMPLE

Micronaut HTTP Client used by Grails

Micronaut Declarative Http Client in Grails 4


```
curl start.grails.org/versions

["3.1.13","3.1.14","3.1.15","3.1.16","3.1.17.BUILD-
SNAPSHOT","3.2.2","3.2.3","3.2.4","3.2.5","3.2.6","3.2.7","3.2.8","3.2.9","3.2.
10","3.2.11","3.2.12","3.2.13","3.2.14.BUILD-
SNAPSHOT","3.3.0","3.3.1","3.3.2","3.3.3","3.3.4","3.3.5","3.3.6","3.3.7","3.3.
8","3.3.9","3.3.10.BUILD-SNAPSHOT"]`
```

Using Micronaut Declarative Client from a Grails 4 app

src/main/groovy/example/grails/GrailsClient.groovy

```
package example.grails

interface GrailsClient {
    List<String> versions()
}
```

build.gradle

```
…
dependencies {
  …
  compile 'io.micronaut:micronaut-http-client'
}
```

src/main/groovy/example/grails/GrailsApplicationForge.groovy

```
package example.grails

import io.micronaut.http.annotation.Get
import io.micronaut.http.client.annotation.Client

@Client("https://start.grails.org")
interface GrailsApplicationForge extends GrailsClient {

    @Override
    @Get("/versions")
    List<String> versions();
}
```

Using Micronaut Declarative Client from a Grails 4 app

grails-app/controllers/example/grails/VersionsController.groovy

```
package example.grails

import groovy.transform.CompileStatic
import org.springframework.beans.factory.annotation.Autowired

class VersionsController {

    @Autowired
    GrailsClient grailsClient

    def index() {
        render grailsClient.versions()
    }
}
```

Micronaut Http Client in integration test of Grails 4

src/integration-test/groovy/example/grails/VersionsControllerSpec.groovy

```
package example.grails

import grails.testing.mixin.integration.Integration
import io.micronaut.http.HttpRequest
import io.micronaut.http.client.BlockingHttpClient
import io.micronaut.http.client.HttpClient
import spock.lang.Specification

@Integration
class VersionsControllerSpec extends Specification {

    void "/versions/index returns Grails versions"() {
        given:
        BlockingHttpClient client = HttpClient.create(new URL("http://localhost:$serverPort".toString())).toBlocking()

        when:
        String versions = client.retrieve(HttpRequest.GET('/versions/index'), String)

        then:
        versions.contains('3.2.10')
        versions.contains('3.3.1')
    }
}
```

Using Micronaut Declarative Client from a Grails 4 app

grails-app/conf/application.yml

```
micronaut:
  http:
    services:
      appforge:
        url: "https://start.grails.org"
```

src/main/groovy/example/grails/GrailsApplicationForge.groovy

```
package example.grails

import io.micronaut.http.annotation.Get
import io.micronaut.http.client.annotation.Client

@Client("appforge")
interface GrailsApplicationForge extends GrailsClient {

    @Override
    @Get("/versions")
    List<String> versions();
}
```

Micronaut or Grails?

OBJECT COMPUTING

- Consider Building Configurations instead of Plugins
- Work with Micronaut, Spring (with `micronaut-spring`) and Grails
- Plugins only work with Grails
- … although some things only possible with Plugins (Views, taglibs etc.)

EXAMPLE

Micronaut Configuration used by Grails, Micronaut and Spring Boot

Building configurations instead of plugins


```
├── build.gradle
├── gradle
│   └── wrapper
│       ├── gradle-wrapper.jar
│       └── gradle-wrapper.properties
├── gradle.properties
├── gradlew
├── gradlew.bat
├── grails-app
│   ├── conf
│   │   ├── application.yml
│   │   └── logback.groovy
│   ├── services
│   │   └── eu
│   │       └── vies
│   │           ├── VatService.groovy
│   ├── init
│   │   └── eu
│   │       └── vies
│   │           ├── Application.groovy
│   │           └── BootStrap.groovy
│   │
├── grails-wrapper.jar
├── grailsw
├── grailsw.bat
…
```

Building configurations instead of plugins

OBJECT COMPUTING

grails-plugin: grails-app/services/eu/vies/VatService.groovy

```
package eu.vies

import groovy.transform.CompileDynamic
import groovy.transform.CompileStatic
import wslite.soap.SOAPClient
import wslite.soap.SOAPResponse

@CompileStatic
class VatService {
    String url = 'http://ec.europa.eu/taxation_customs/vies/services/checkVatService'
    SOAPClient client = new SOAPClient("${url}.wsdl")

    @CompileDynamic
    Boolean validateVat(String memberStateCode, String vatNumberCode) {
        SOAPResponse response = client.send(SOAPAction: url) {
            body('xmlns': 'urn:ec.europa.eu:taxud:vies:services:checkVat:types') {
                checkVat {
                    countryCode(memberStateCode)
                    vatNumber(vatNumberCode)
                }
            }
        }
        response.checkVatResponse.valid.text() == 'true'
    }
}
```

Building configurations instead of plugins


```
.
├── build.gradle
├── gradle.properties
└── src
    ├── main
    │   └── groovy
    │       └── eu
    │           └── vies
    │               └── VatService.groovy
    └── test
        └── groovy
            └── eu
                └── vies
                    └── VatServiceSpec.groovy
```

Building configurations instead of plugins

OBJECT COMPUTING

lib: build.gradle

```
plugins {
    id 'groovy'
}

repositories {
    jcenter()
}

dependencies {
    compileOnly "io.micronaut:micronaut-inject-groovy:$micronautVersion"
    compile 'org.codehaus.groovy:groovy-xml:2.5.7'
    compile 'com.github.groovy-wslite:groovy-wslite:1.1.2'
    testCompile("org.spockframework:spock-core:${spockVersion}") {
        exclude module: 'groovy-all'
    }
}
```

Building configurations instead of plugins

OBJECT COMPUTING

lib: src/main/groovy/eu/vies/VatService.groovy

```
package eu.vies

import groovy.transform.CompileDynamic
import groovy.transform.CompileStatic
import wslite.soap.SOAPClient
import wslite.soap.SOAPResponse
import javax.inject.Singleton

@CompileStatic
@Singleton
class VatService {
    String url = 'http://ec.europa.eu/taxation_customs/vies/services/checkVatService'
    SOAPClient client = new SOAPClient("${url}.wsdl")

    @CompileDynamic
    Boolean validateVat(String memberStateCode, String vatNumberCode) {
        SOAPResponse response = client.send(SOAPAction: url) {
            body('xmlns': 'urn:ec.europa.eu:taxud:vies:services:checkVat:types') {
                checkVat {
                    countryCode(memberStateCode)
                    vatNumber(vatNumberCode)
                }
            }
        }
        response.checkVatResponse.valid.text() == 'true'
    }
}
```

Building configurations instead of plugins

OBJECT COMPUTING

grails: grails-app/controllers/example/grails/ViesController.groovy

```
package example.grails

import eu.vies.VatService
import groovy.transform.CompileStatic
import org.springframework.beans.factory.annotation.Autowired

@CompileStatic
class ViesController {

    @Autowired
    VatService vatService

    def valid(String memberStateCode, String vatNumberCode) {
        render vatService.validateVat(memberStateCode, vatNumberCode)
    }
}
```

Building configurations instead of plugins

OBJECT COMPUTING

micronaut: src/main/example/micronaut/ViesController.groovy

```groovy
package example.micronaut

import eu.vies.VatService
import groovy.transform.CompileStatic
import io.micronaut.http.annotation.Controller
import io.micronaut.http.annotation.Get
import javax.inject.Inject

@CompileStatic
@Controller('/vies')
class ViesController {

    @Inject
    VatService vatService

    @Get('/valid')
    Boolean valid(String memberStateCode, String vatNumberCode) {
        vatService.validateVat(memberStateCode, vatNumberCode)
    }
}
```

Micronaut Configurations

OBJECT COMPUTING

- Configuration with @ConfigurationProperties
- Beans with @Singleton, @Factory etc.
- Conditional Behavior with @Requires
- Customization with @Replaces

In Summary

- Upgrading very different compared to 2 -> 3
- Micronaut Provides an Awesome Foundation
- Building Blocks to Create Libraries, Configurations and Clients
- Most Micronaut Features Available in Grails
- Build Micronaut Libraries not Plugins

Questions?

OBJECT
COMPUTING

CONNECT WITH US

1+ (314) 579-0066

@objectcomputing

objectcomputing.com

Greach 2020?