Grails

Adolfo Sanz De Diego

Abril 2013

Contents

1 Acerca de **7**

1.1 Autor 7

1.2 Licencia 7

2 Introducción **9**

2.1 ¿Grails? 9

2.2 Paradigmas 9

2.3 GORM 10

2.4 Plugins 10

2.5 NIH (Not Invented Here) 12

2.6 Arquitectura 12

3 Instalación y configuración **13**

3.1 JDK 13

3.2 Groovy-SDK 13

3.3 Grails 13

3.4 Probando 14

4 Getting Started **15**

4.1 Create Grails Project 15

4.2 Directorios 15

4.3 Create a Domain Class 16

4.4 Create a Controller 16

4.5 Creating Test Data 17

4.6 Start Grails 17

5 Scaffolding **19**

5.1 Definición 19

5.2 Dinámico 19

5.3 Estático 19

5.4 Templates 20

6 Validación **21**

6.1 Clases de dominio 21

6.2 Controladores 21

6.3 Vistas 22

7 CRUD **23**

7.1 Create 23

7.2 Read 23

7.3 Update 23

7.4 Delete 23

8 GORM **25**

8.1 Agregación (unidireccional) 25

8.2 Agregacion (bidireccional) 25

8.3 Uno a uno (Foreign Key) 26

8.4 Uno a muchos 26

8.5 Muchos a muchos 26

9 Quering **29**

9.1 Listados 29

9.2 Por ID 29

9.3 findBy y findAllBy 29

9.4 Comparadores 30

9.5 Ejemplos 30

10 Servicios **31**

10.1 Definición 31

10.2 Creación 31

10.3 Transaccionalidad 31

10.4 Scope 32

10.5 Inyección 32

11 Configuración Log4j **33**

11.1 Logging levels 33

11.2 Artefactos 33

12 Configuración Spring **35**

12.1 Iyección normal 35

12.2 Iyección tests 35

13 Testing **37**

13.1 Unit Test 37

13.2 Integration Test 37

14 Spring Security **39**

14.1 Instalación 39

14.2 Configuración 39

14.3 Uso 39

Chapter 1

Acerca de

1.1 Autor

Adolfo Sanz De Diego

- Correo: asanzdiego@gmail.com
- Twitter: [@asanzdiego](http://twitter.com/asanzdiego)
- Linkedin: http://www.linkedin.com/in/asanzdiego
- Blog: http://asanzdiego.blogspot.com.es

1.2 Licencia

Chapter 2

Introducción

2.1 ¿Grails?

Grails no sólo es un framework de desarrollo web, sino que es una **plataforma completa de desarrollo**:

- Contenedor/servidor web
- Gestor de base de datos
- Scaffolding
- Empaquetado de la aplicación (war)
- Realización de tests (unitarios, de integración, funcionales)
- Extensible con plugins

2.2 Paradigmas

Se basa en los paradigmas:

- **CoC** (Convención sobre Configuración)
- **DRY** (Don't Repeat Yourself)
- **MVC** (Modelo Vista Controlador)

2.3 GORM

GORM (Grails Object Relational Mapping) sirve para el **mapeo objeto-relacional**:

- Uno a uno
- Uno a muchos
- Muchos a muchos

2.4 Plugins

Existen multitud de plugins que **extienden la plataforma**:

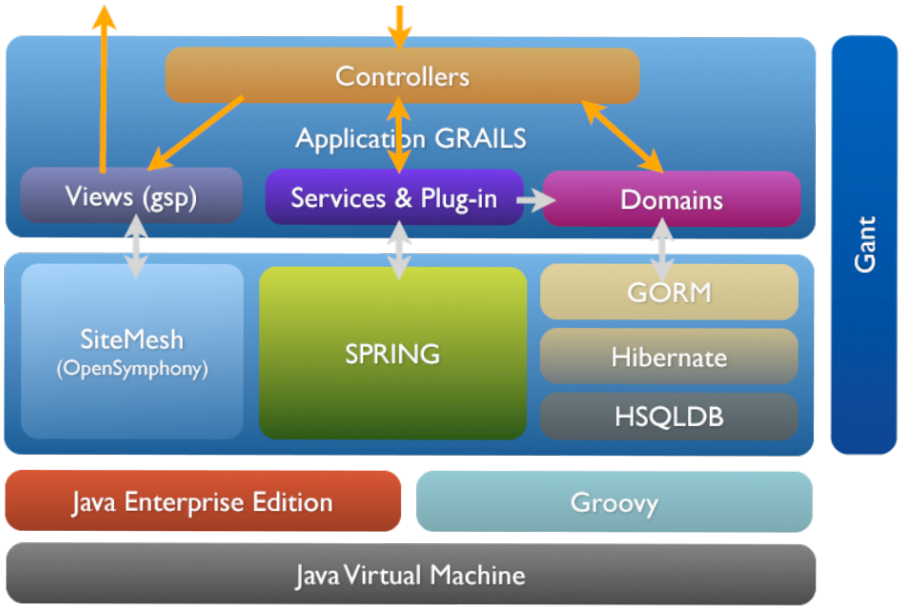
- Seguridad
- AJAX
- Búsqueda
- Informes
- etc.

Se pueden crear plugins **internos para funcionalidades comunes** entre varias aplicaciones.

Figure 2.1: Grails NIH

2.5 NIH (Not Invented Here)

2.6 Arquitectura



Chapter 3

Instalación y configuración

3.1 JDK

1. Descargar.
2. Instalar/Descomprimir.
3. Variable de entorno y añadir al path.

```
export JAVA_HOME="~/Java/jdk"
export PATH=$PATH":"$JAVA_HOME"/bin"
```

3.2 Groovy-SDK

1. Descargar.
2. Instalar/Descomprimir.
3. Variable de entorno y añadir al path.

```
export GROOVY_HOME="~/Java/groovy"
export PATH=$PATH":"$GROOVY_HOME"/bin"
```

3.3 Grails

1. Descargar.
2. Instalar/Descomprimir.
3. Variable de entorno y añadir al path.

```
export GRAILS_HOME="~/Java/groovy"
export PATH=$PATH":"$GRAILS_HOME"/bin"
```

3.4 Probando

```
$ grails --version
Grails version: 2.2.1
```

Chapter 4

Getting Started

4.1 Create Grails Project

Por línea de comandos:

```
$ grails create-app my-project
```

En el GGTS:

```
File > New > Grails Project
```

Línea de comandos en el GGTS:

```
Control + Alt + Shift + G
```

4.2 Directorios

```
%PROJECT_HOME%
+ grails-app-> ficheros de la aplicación grails
+ lib       -> bibliotecas
+ scripts   -> scripts
+ src
  + groovy  -> otros ficheros groovy opcionales
  + java    -> otros ficheros java opcionales
+ test      -> clases de test
+ web-app
  + css     -> archivos CSS
```

```
  + images  -> archivos de imágenes
  + js      -> archivos JavaScript
  + WEB-INF -> otros ficheros de la aplicación web

%PROJECT_HOME%
+ grails-app
  + conf    -> archivos de configuración
    + hibernate -> archivos de configuración de hibernate
    + spring-> archivos de configuración de spring
  + controllers -> controladores
  + domain  -> clases de dominio
  + i18n    -> ficheros de internacionalización
  + services-> servicios
  + taglib  -> bibliotecas de etiquetas
  + util    -> clases de utilidades
  + views   -> vistas
    + layouts   -> layouts
```

4.3 Create a Domain Class

```
$ grails create-domain-class org.example.Libro
```

```
package org.example

class Libro {
  String titulo
  String author

  static constraints = {
    titulo(blank: false)
    author(blank: false)
  }
}
```

4.4 Create a Controller

```
$ grails create-controller org.example.Libro
```

```
package org.example

class LibroController {
    def scaffold = Libro
}
```

4.5 Creating Test Data

grails-app/conf/BootStrap.groovy

```
import org.example.Libro
class BootStrap {
  def init = { servletContext ->
    // Check whether the test data already exists.
    if (!Libro.count()) {
      new Libro(
        author: "S. King",
        titulo: "The Shining").save(failOnError: true)
      new Libro(
        author: "J. Patterson",
        titulo: "Along Came a Spider").save(failOnError: true)
    }
  }

  def destroy = {
  }
}
```

4.6 Start Grails

```
$ grails run-app
```

Chapter 5

Scaffolding

5.1 Definición

Generación automática de código para las cuatro operaciones básicas de cualquier aplicación (CRUD):

- **C**reate
- **R**ead
- **U**pdate
- **D**elete

5.2 Dinámico

En el controlador:

```
def scaffold = true // si se sigue la convención de nombrado

def scaffold = DomainClass // si no se sigue la convención de nombrado
```

5.3 Estático

Genera el controlador:

```
grails generate-controller org.example.Libro
```

Genera la vista:

```
grails generate-views org.example.Libro
```

Genera el controlador y la vista:

```
grails generate-all org.example.Libro
```

5.4 Templates

Podemos extraer las templates de generación de código con el siguiente comando:

```
grails install-templates
```

Se pueden ver y modificar para su uso en la carpeta:

```
%PROJECT_HOME%
+ src
  + templates
    + artifacts
    + scaffolding
    + testing
    + war
```

Chapter 6

Validación

6.1 Clases de dominio

```
class User {

  String email
  String password
  Integer age
  String twitter

  static constraints = {
    email(email:true, blank:false, unique:true)
    password(size:5..15, blank:false)
    age(range:18..99)
    twitter(url:true, nullable:true)
  }
}
```

6.2 Controladores

```
def user = new User(params)
if(user.validate()) {
  // do something with user
}
else {
  user.errors.allErrors.each {
    println it
```

```
  }
}
```

6.3 Vistas

```
<g:hasErrors bean="${user}">
  <ul>
    <g:eachError var="err" bean="${user}">
      <li>${err}</li>
    </g:eachError>
  </ul>
</g:hasErrors>
```

Chapter 7

CRUD

7.1 Create

```
def p = new Persona(nombre: "Fred", edad: 40)
p.save()
```

7.2 Read

```
def p = Persona.get(unaPersona.id)
assert 1 == p.id
```

7.3 Update

```
def p = Persona.get(unaPersona.id)
p.nombre = "Bob"
p.save()
```

7.4 Delete

```
def p = Persona.get(unaPersona.id)
p.delete()
```

Chapter 8

GORM

8.1 Agregación (unidireccional)

```
class Cara {
  Nariz nariz
}

class Nariz {
  ...
}
```

8.2 Agregacion (bidireccional)

```
class Cara {
  Nariz nariz
}

class Nariz {
  static belongsTo = [face:Face]
}


new Cara(nose:new Nariz()).save() // guarda ambos: Cara y Nariz
new Nariz(face:new Cara()).save() // da error
Face.get(faceId).delete() // borra ambos: Cara y Nariz
```

8.3 Uno a uno (Foreign Key)

```
class Cara {
  static hasOne = [nose:Nose]

  // opcional, pero buena práctica
  static constraints = {
    nariz unique: true
  }
}
class Nariz {
  Cara cara // crea una FK en la tabla de Nariz
}
```

8.4 Uno a muchos

```
class Autor {
  static hasMany = [ libros : Libro ]
  String nombre
}
class Libro {
  String titulo
}
```

Cascada al salvar y al actualizar pero no al borrar.

```
class Autor {
  static hasMany = [ libros : Libro ]
  String nombre
}
class Libro {
  static belongsTo = [ author: Autor ]
  String titulo
}
```

Con belongsTo cascada al salvar, al actualizar y al borrar.

8.5 Muchos a muchos

```
class Autor {
  static hasMany = [libros:Libro]
```

```
  String nombre
}

class Libro {
  static belongsTo = Autor
  static hasMany = [authors:Autor]
  String titulo
}
```

El belongsTo marca el "propietario" de la relación, en este caso el Autor.

Al salvar un Autor, se salvarán sus Libros, pero no al revés.

Recomiendan usar 2 relaciones uno a muchos, mejor que una relación muchos a muchos.

Chapter 9

Quering

9.1 Listados

Todos los elementos:

```
def libros = Libro.list()
```

Paginación:

```
def libros = Libro.list(offset:10, max:20)
```

Ordenación

```
def libros = Libro.list(sort:"title", order:"asc")
```

9.2 Por ID

```
def libro = Libro.get(23)
```

```
def libros = Libro.getAll(23, 93, 81)
```

9.3 findBy y findAllBy

```
.find[All]By([Property][Comparator][And|Or])?[Property][Comparator]
```

9.4 Comparadores

InList - Busca el valor dentro de la lista pasada por parámetro.

LessThan - Menor que el valor pasado por parámetro.

LessThanEquals - Menor o igual que el valor pasado por parámetro.

GreaterThan - Mayor que el valor pasado por parámetro.

GreaterThanEquals - Mayor o igual que el valor pasado por parámetro.

Like - Equivalente al like de SQL.

Ilike - Similar a Like sólo que no es sensible a las mayúsculas.

NotEqual - No es igual al valor pasado por parámetro.

Between - Entre dos valores (necesita dos parámetros).

IsNotNull - Valor no nulo (no requiere ningún parámetro).

IsNull - Valor nulo (no requiere ningún parámetro)

9.5 Ejemplos

```
class Libro {
    String titulo
    Date fecha
}

def libro = Libro.findByTitulo("The Stand")
def libros = Libro.findAllByTituloLike("Harry Pot%")
libros = Libro.findAllByFechaBetween(primeraFecha, segundaFecha)
libros = Libro.findAllByFechaGreaterThan(someDate)
libros = Libro.findAllByTituloOrFechaLessThan("%titulo buscado%", fechaBuscada)
```

Chapter 10

Servicios

10.1 Definición

Se utilizan cuando necesitamos **transacciones** o cuando utilizamos varias **clases de dominio**

10.2 Creación

```
$ grails create-service org.example.Libro
```

```
package org.example

class LibroService {

  def doSomething() {

    // do domething
  }
}
```

10.3 Transaccionalidad

Por defecto son transaccionales, para deshabilitarlo:

```
static transactional = false
```

10.4 Scope

Por defecto son singleton, pero podemos usar otros scopes:

- **prototype** - Una instancia por cada inyección.
- **request** - Una instancia por cada request.
- **flash** - Una instancia para la request actual y la siguiente.
- **flow** - Una instancia por cada webflow.
- **conversation** - Una instancia por cada conversacion de un webflow.
- **session** - Una instancia por cada sesión.
- **singleton** - Una única instancia (por defecto).

```
static scope = "flow"
```

10.5 Inyección

Los servicios se pueden inyectar en los controladores.

```
class LibroController {

  def libroService
  ...
}
```

Chapter 11

Configuración Log4j

11.1 Logging levels

1. off
2. fatal
3. error
4. warn
5. info
6. debug
7. trace
8. all

11.2 Artefactos

conf/Config.groovy

```
log4j = {

    // warn a todos los artefactos de nuestra aplicacion
    warn "grails.app"

    // debug a un controlador específico alojado en el paquete por defecto
    debug "grails.app.controllers.YourController"

    // debug a una clase de dominio específica
    debug "grails.app.domain.org.example.Book"
```

```
    // error a todos los taglibs
    error "grails.app.taglib"

    // info a tos los servicios
    info "grails.app.services"
}
```

Chapter 12

Configuración Spring

12.1 Iyección normal

conf/spring/resources.groovy

```
beans = {
    rules(org.example.Rules) {
        deltaAge = 5
        deltaHeight = 0.1
    }
}
```

12.2 Iyección tests

conf/spring/resources.groovy

```
defineBeans {
    rules(org.example.Rules) {
        deltaAge = 5
        deltaHeight = 0.1
    }
}
```

Chapter 13

Testing

13.1 Unit Test

Tienen que ser rápidos, no se ejecutan en el servidor, utilizan mocks.

Utilizan las anotaciones @TestFor y @Mock

13.2 Integration Test

Se ejecutan en el servidor con datos reales.

Podemos utilizar el BootStrap para meter datos en la base de datos.

Chapter 14

Spring Security

14.1 Instalación

conf/BuildConfig.groovy

```
...
plugins {
  ...
  compile ':spring-security-core:1.2.7.3'
  ...
}
```

14.2 Configuración

```
grails refresh-dependencies
```

```
grails s2-quickstart org.example User Role
```

14.3 Uso

Se usa la anotación @Secured(['ROLE_NAME']) tanto a nivel de clase como a nivel de método.

Se pueden usar tambien las siguientes reglas:

- **IS_AUTHENTICATED_ANONYMOUSLY**: cualquiera puede entrar, incluso sin hacer login

- **IS_AUTHENTICATED_REMEMBERED**: sólo usuarios con login pueden entrar
- **IS_AUTHENTICATED_FULLY**: obliga a hacer login aunque tengas la cookie de remember