

Grails

Adolfo Sanz De Diego

Abril 2013

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

- **Este obra está bajo una licencia:**
 - Creative Commons Reconocimiento-CompartirIgual 3.0
- **El código fuente de los programas están bajo una licencia:**
 - GPL 3.0

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

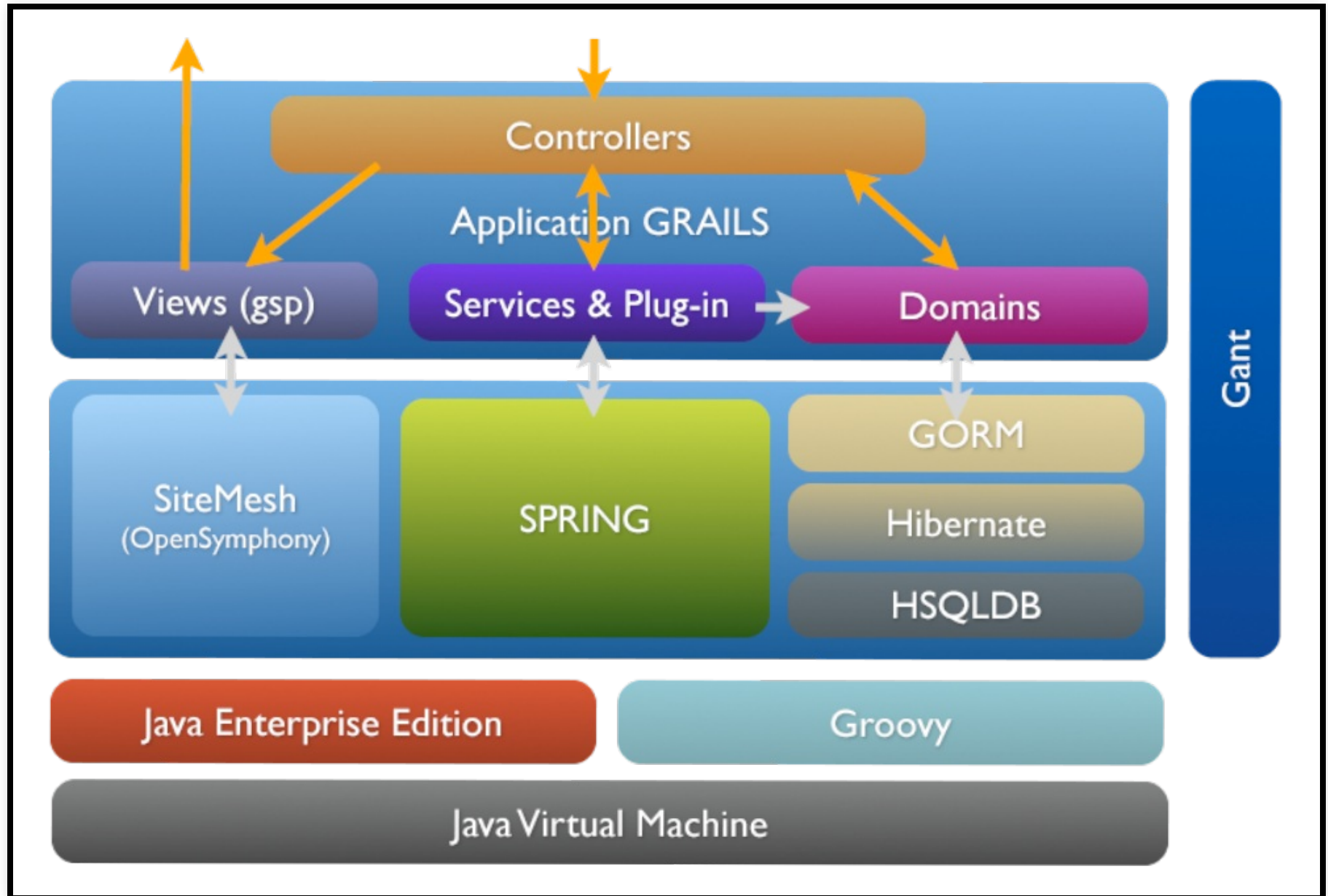
- GORM (Grails Object Relational Mapping) sirve para el **mapeo objeto-relacional**:
 - Uno a uno
 - Uno a muchos
 - Muchos a muchos

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.

2.5 NIH (Not Invented Here)

2.6 Arquitectura



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
```


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 (I)

```
%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
```

4.3 Directorios (II)

%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

4.4 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)  
    }
```

```
}
```

4.5 Create a Controller

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

```
package org.example
```

```
class LibroController {  
    def scaffold = Libro  
}
```

4.6 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 = {
    }
}
```

4.7 Start Grails

```
$ grails run-app
```


5 Scaffolding

5.1 Definición

- **Generación automática de código** para las cuatro operaciones básicas de cualquier aplicación (CRUD):
 - **C**reate
 - **R**ead
 - **U**ppdate
 - **D**eleate

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
```

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>
```

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()
```

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 {  
  statichasOne = [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 (I)

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

- Cascada al salvar y al actualizar pero no al borrar.

8.5 Uno a muchos (II)

```
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.

8.6 Muchos a muchos

```
class Autor {  
  static hasMany = [libros:Libro]  
  String nombre  
}  
  
class Libro {  
  static belongsTo = Autor  
  static hasMany = [authors:Author]  
  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.
- Recomendán usar 2 relaciones uno a muchos, mejor que una relación muchos a muchos.

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 (I)

- **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.

9.5 Comparadores (II)

- **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)

9.6 Ejemplos

```
class Libro {  
  String titulo  
  Date fecha  
}
```

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

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 something
```

```
    }
```

```
}
```

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  
    ...  
}
```

11 Configuración

$\text{Log}_4 j$

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 todos los servicios  
    info "grails.app.services"  
}
```

12 Configuración Spring

12.1 Inyección normal

- conf/spring/resources.groovy

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

12.2 Inyección tests

- conf/spring/resources.groovy

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


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.

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 también 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