

Título del libro

Jorge Bautista

21 de enero de 2014

# Prólogo

Este es un libro que menciona de manera breve las características más importantes del framework Grails<sup>1</sup>. No se detiene en detalles ni explica ningún procedimiento de instalación. Simplemente permite que el lector eche un vistazo a la tecnología para comenzar a conocerla o para *ojearla* y buscar *puntas de hilo* a soluciones cuyo resto encontrará en google.

---

<sup>1</sup>Únicamente las características descritas en el libro “*The Definitive Guide To Grails*”

# Capítulo 1

## Introducción a Grails

### 1.1. Arquitectura

Grails envuelve a muchas tecnologías relacionados con *java*, abstrayendo su complejidad al proporcionarnos configuraciones predeterminadas que cubren los casos de uso más comunes. Estas configuraciones pueden ahorrarnos días o semanas de trabajo gracias a la integración con otras tecnologías.

Aquí se presentan las tecnologías principales sobre las que se Grails está montado:

- Hibernate
- Spring
- Sitemesh
- Tomcat
- H2
- Groovy
- Gant
- JEE

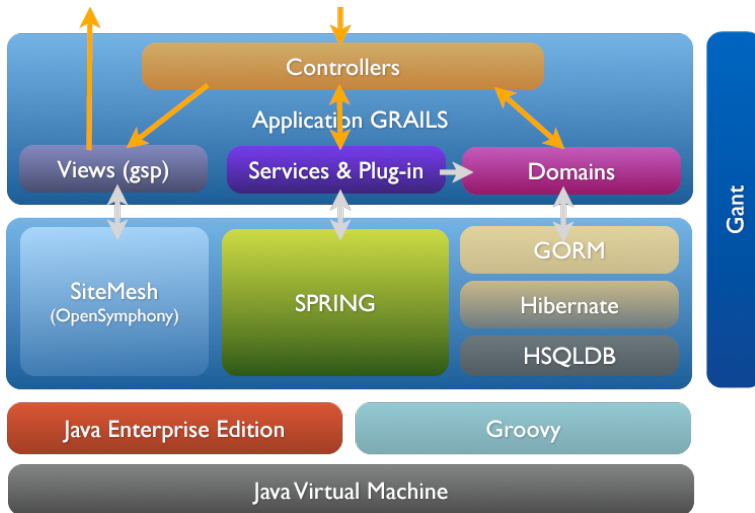


Figura 1.1: Tecnologías que confirman la arquitectura de Grails

## 1.2. Artefactos de Grails

Grails consta de cuatro tipos de artefactos principales:

1. Views
2. Controllers
3. Domains
4. Services

### 1.2.1. Views

Muestran información al usuario de la aplicación. En Grails, las vistas existen en forma de archivos con extensión *gsp*. La información es puesta en los *gsp* a través de plantillas<sup>1</sup> y estas se llenan a través de controllers.

---

<sup>1</sup>Para los más experimentados, llenar estas plantillas usan una sintaxis intermedia entre tags *jstl* y tags de *jsf*

### 1.2.2. Controllers

Se encargan de recibir peticiones desde las vistas, respondiendo con alguna de las siguientes formas:

1. Renderizar una vista *gsp*,
2. renderizar *JSON*,
3. renderizar fragmentos *HTML*
4. etc.

Asimismo los controllers deben lidiar con la lógica de negocio de la aplicación, la cual puede colocarse dentro del mismo controller o en servicios inyectados<sup>2</sup>.

### 1.2.3. Dominios

Se encargan de describir el esquema de la base de datos a través de clases groovy. Grails se refiere al conjunto de estas clases y las funcionalidades agregadas por el framework<sup>3</sup> como *GORM*.

### 1.2.4. Servicios

Se encargan de contener la lógica de la aplicación y se conectan directamente con las clases de dominio.

Debido a la gran simplicidad de Grails, es común que los desarrolladores escriban código de acceso a base de datos y/o reglas de negocio dentro de los controladores. Y está bien si así les funciona, pero pierden las ventajas de algunas propiedades que los servicios tienen, por ejemplo:

- Modularización de código
- Son transaccionales de forma predeterminada, es decir, si hay una operación a base de datos que necesite de varios pasos para completarse, y alguno de estos pasos falla, Grails ejecuta un rollback a todos los cambios.

---

<sup>2</sup> Checar Apéndice B

<sup>3</sup> Cualquier funcionalidad relacionada con base de datos como CREATE, READ, UPDATE, DELETE y todas las que extienden de ellas

### 1.2.5. UrlMappings

Es un archivo donde pueden describirse las *URL's* que el sistema ofrece para ser accesibles al cliente. El *UrlMappings* puede o no usarse, ya que las convenciones de Grails permiten que existan *URL's* implícitas que podemos acceder aún si no han sido agregadas a este archivo.

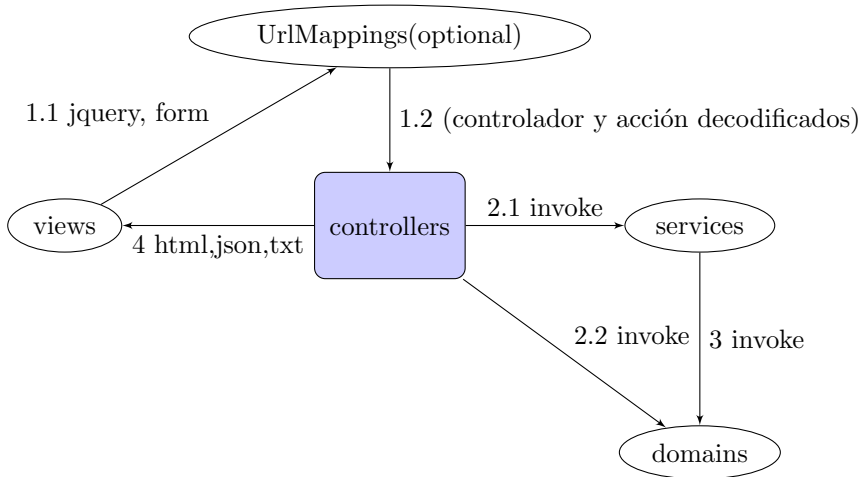


Figura 1.2: Artefactos de Grails funcionando

## Capítulo 2

# Inicio rápido

1. Instala Grails y ponlo en el *Path* de tu sistema operativo
2. Verifica que Grails esté en el *Path* ejecutando el siguiente comando desde una línea de comandos y asegurándote de que devuelva la versión instalada de Grails:

```
grails -version
```

3. Navega desde una línea de comandos hacia el folder dónde generarás tu aplicación
4. Para autogenerar una aplicación, ejecuta desde una línea de comandos:

```
grails create-app NOMBRE_DE_APP
```

5. Para correr tu aplicación en el servidor web integrado, ejecuta desde una línea de comandos y dentro del folder autogenerado por Grails:

```
grails run-app
```

6. Accede a la dirección:

```
http://localhost:8080/NOMBRE_DE_APP
```

## Capítulo 3

# Métodos de renderizado

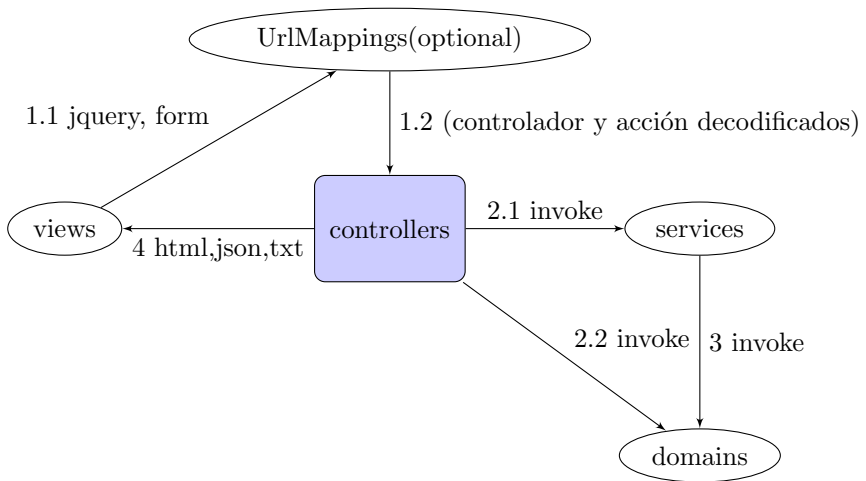


Figura 3.1: Artefactos de Grails funcionando

El resultado enviado de los controllers a las vistas puede ir en varios formatos. Siempre se hace a través de la instrucción *render*:

- **Renderizar texto**

```
render 'Welcome to the gTunes store!'
```

- **Solicitar ayuda de un comando**



```
grails help NOMBREDECOMANDO
```

- **Crear aplicación de Grails**

```
grails create-app NOMBREDEAPP
```

- **Crear un controller + sus pruebas + sus vistas**

```
grails create-controller [PAQUETE]NOMBRE
```

- **Ejecutar pruebas de la aplicación**

```
grails test-app
```

- **Ejecutar la aplicación**

```
grails run-app
```

# Capítulo 4

## Pruebas

### 4.1. Tipos de pruebas en Grails

Grails divide las pruebas en 2 tipos:

**Unitarias** La ejecución de estas pruebas es rápida, comparada con las de integración. La desventaja es que requieren de un uso extensivo de *mocks* y *stubs* para imitar el comportamiento de algunas clases y regresar valores predeterminados para la prueba.

**Integración** Estas pruebas necesitan que la aplicación esté cargada por completo para poder realizarse, incluyendo la base de datos, por lo que son más lentas.

### 4.2. Objetos implícitos en clases de prueba

Dentro de las clases de prueba existen varios objetos con los que puedes contar para apoyarte durante su implantación:

**controller** Es una referencia al controller definido en la anotación *@TestFor*, lo que permite el acceso a los métodos del controller.

**params** Es una referencia a los parámetros enviados al controller para usarlos en su lógica. Al llenar este objeto los *params* serán enviados al objeto *controller* cuando se invoque alguno de sus métodos.

**response** Una vez que alguno de los métodos del *controller* sea invocado, el objeto *response* se llenará con los detalles de la respuesta (estatus, texto de respuesta, headers, etc).

Algunas muestras de código para realizar pruebas en Grails:

- **Clase para probar un controller. Autogenerada junto con el controlador *StoreController***

```
@TestFor(StoreController)
class StoreControllerTests {
    void testSomething() {
        fail "Implement me"
    }
}
```

- **Clase para probar el texto de respuesta de un controller al ejecutar su método *index()*:**

```
@TestFor(StoreController)
class StoreControllerTests {
    void testSomething() {
        controller.index()
        assert 'Welcome to the gTunes store!'
            == response.text
    }
}
```

### 4.3. Comandos útiles en el desarrollo de pruebas

Los resultados de la ejecución de algún comando de pruebas generan archivos<sup>1</sup> con los resultados en los formatos *html*, *txt* y *xml*.

- **Comando que ejecuta todas las pruebas del sistema**

```
grails test-app
```

- **Clase para probar el texto de respuesta de un controller al ejecutar su método *index()*:**

---

<sup>1</sup>Los resultados se generan en el folder *test/reports*

```
@TestFor(StoreController)
class StoreControllerTests {
    void testSomething() {
        controller.index()
        assert 'Welcome to the gTunes store!'
            == response.text
    }
}
```

# Apéndice A

## Comandos más comunes

- **Consultar versión de Grails**

Devolverá una línea de texto describiendo la versión de grails.

```
grails -version
```

- **Solicitar ayuda de un comando**

Devolverá información (no tan detallada) sobre el comando y sus argumentos.

```
grails help NOMBRE_DE_COMANDO
```

- **Crear aplicación de Grails**

Genera toda la estructura de archivos de una aplicación de Grails.

```
grails create-app NOMBRE_DE_APP
```

- **Crear un controller + sus pruebas + sus vistas**

Genera un controlador de Grails así como archivos de prueba y *gsp's*. Tanto los archivos de prueba como los *gsp's* se generan vacíos.

```
grails create-controller [PAQUETE]NOMBRE
```

- **Ejecutar pruebas de la aplicación**

Ejecuta todas las pruebas de la aplicación, poniendo los resultados en *test/reports*

```
grails test-app
```

- **Ejecutar la aplicación**

Ejecuta la aplicación. Si ENVIRONMENT no se especifica, el default a usar será el de desarrollo (development).

```
grails [ENVIRONMENT] run-app
```

- **Crear clase de dominio**

Se encarga de crear una nueva clase de dominio y además genera un archivo donde pueden implantarse sus pruebas.

```
grails create-domain-class [PAQUETE]NOMBRE
```

- **Abrir reporte de pruebas**

Se encarga de abrir el reporte con los resultados de las pruebas con formato HTML en el navegador predeterminado.

```
grails open test-report
```

## Apéndice B

# Inyección de dependencias en Grails

# Índice de figuras

- 1.1. Tecnologías que confirman la arquitectura de Grails . . . 2
- 1.2. Artefactos de Grails funcionando . . . . . 4
- 3.1. Artefactos de Grails funcionando . . . . . 6



# Índice de cuadros

# Bibliografía

[1] article-description1

[2] article-description2