



# Groovy & Grails: Desarrollo rápido de aplicaciones

## Sesión 1: Introducción a Groovy

# ¿Qué es Groovy?

- Lenguaje de programación ágil y dinámico
- Plataforma Java
- Características típicas de Python, Ruby o Smalltalk
- Sintaxis típica de Java
- Superclase de Java
- Evita la ceremonia de Java

# ¿A quién va dirigido Groovy?

- A los programadores Java
- A los programadores de scripts
- A los programadores ágiles y extremos

# Editores Groovy

- Plugin IntelliJ IDEA
- Plugin para Eclipse
- NetBeans
- UltraEdit
- Plugin para Jedit

# Instalación

- Descargar desde http://groovy.codehaus.org
- Descomprimir
- Establecer variable de entorno *GROOVY_HOME*
- Añadir GROOVY_HOME/bin al PATH
- Establecer variable de entorno JAVA_HOME

# Hola Mundo!

Tres formas de ejecutar programas en Groovy

- groovysh
- groovyConsole
- groovy

# Características de Groovy

- Comentarios
  - //, comentarios de una línea
  - /*…*/, comentarios multilínea
  - /**….*/, comentarios estilo Javadoc
  - #!, comentarios estilo *shebang* sólo en la primera línea

# Características de Groovy

- Comparando la sintaxis de Java y Groovy. En común:
  - Mecanismo de paquetes
  - Sentencias
  - Definición de clases y métodos
  - Estructuras de control

# Características de Groovy

- Comparando la sintaxis de Java y Groovy. En común:
  - Operadores, asignaciones y expresiones
  - Manejo de excepciones
  - Declaración de literales
  - Instanciación de objetos y llamadas a métodos

# Características de Groovy

- Comparando la sintaxis de Java y Groovy. Valor añadido en Groovy
  - Nuevas expresiones y operadores
  - Nuevas formas de declarar objetos
  - Nuevas estructuras de control
  - Nuevos tipos de datos con sus operadores y expresiones
  - Todo es un objeto

# Características de Groovy

- Brevedad del lenguaje
  - Groovy evita la ceremonía que acompaña a Java
  - Aumenta expresividad al lenguaje
  - Importa automáticamente varios paquetes

# Características de Groovy

- Aserciones
  - Desde Java 1.4
  - Aseguran la corrección de nuestro programa
  - Nuevo debug

# El código de Groovy

- Declaración de clases

```
class Libro {
        private String titulo

        Libro (String elTitulo){
                titulo = elTitulo
        }
        String getTitulo(){
                return titulo
        }
}
```

# El código de Groovy

- Scripts en Groovy

```
Libro cgg = new Libro('Curso GroovyGrails')

assert cgg.getTitulo() == 'Curso GroovyGrails'
assert getTituloAlReves(cgg) == 'sliarGyvoorG osruC'

String getTituloAlReves(libro) {
        titulo = libro.getTitulo()
        return titulo.reverse()
}
```

# El código de Groovy

- GroovyBeans

```
/*Un Bean en Java*/
class Libro{
        String titulo;

        String getTitulo(){
                return this.titulo;
        }
        void setTitulo(String str){
                this.titulo = new String(str);
        }
}
```

# El código de Groovy

- GroovyBeans

```
/*Un Bean en Groovy*/
class Libro{
        String titulo;
}
```

# El código de Groovy

- Cadenas de texto
  - GString

```
def part1 = 'groovy'
def part2 = 'grails'
assert "me gusta groovy grails" == "me gusta $part1 $part2"
```

# El código de Groovy

- Los números son objetos

```
def x = 1
def y = 2
assert x + y == 3
assert x.plus(y) == 3
assert x instanceOf Integer
```

# El código de Groovy

- Listas, mapas y rangos
  - Groovy facilita el trabajo con este tipo de colecciones de datos

# El código de Groovy

- **Listas**, mapas y rangos

```
def sesiones = [
'Introducción a Groovy',
'El lenguaje Groovy',
'Aspectos avanzados en Groovy']

assert sesiones[1] == 'El lenguaje Groovy'

sesiones[3] = 'Librerías propias de Groovy'
```

# El código de Groovy

- Listas, **mapas** y rangos

```
def http = [
            100 : 'Continue',
            200 : 'OK',
            400 : 'Bad Request'
]

assert http[200] == 'OK'
```

# El código de Groovy

- Listas, mapas y **rangos**

```
def x = 1..10
assert x.contains(2)
assert x.size() == 10
assert x.reverse() == 10..1
```

# Closures

- Bloques de código anónimo definido entre llaves

```
def ayer = {Date dia -> dia - 1}

ayer.call(new Date())
```

# Closures

- Nos permiten ser más ágiles programando

```
['Pedro','Lola','Juan'].each { it -> println(it) }
```

# Estructuras de control en Groovy

- Los típicos *if-else, while, switch y try-catch-finally*
- El bloque *for* utiliza la notacion *for (i in x) { cuerpo}*

```
for(i in 1..10)
        println i

for(i in [1,2,3,4,5,6,7,8,9,10])
        println i
```

# Estructuras de control en Groovy

- O mediante closures

```
def alumnos = ['Pedro','Miguel','Alejandro','Elena']
alumnos.each{nombre -> println nombre}
```

# Groovy en el entorno Java

- Groovy puede ser ejecutado en la JVM:
  - Compilando con groovyc
  - Sin compilar

# GDK: la librería Groovy

- GDK es una extensión a la librería JDK
- Facilita el acceso a base de datos y procesamiento de XML
- Extiende funcionalidades de Java

# GDK: la librería Groovy

| Tipo | En Java | En Groovy |
|---|---|---|
| Array | Propiedad length | Método size() |
| String | Método length() | Método size() |
| StringBuffer | Método length() | Método size() |
| Collection | Método size() | Método size() |
| Map | Método size() | Método size() |
| File | Método length() | Método size() |
| Matcher | Método groupCount() | Método size() |
| | | |