



Universidad
Rey Juan Carlos

Programación declarativa

Grado en Inteligencia Artificial
Universidad Rey Juan Carlos



Presentación

- **Profesorado**
- Contexto
- Contenido
- Planificación
- Evaluación
- Material

Profesorado

- **Juan Manuel Serrano:** juanmanuel.serrano@urjc.es

Despacho 024, del edificio departamental II

Tutorías: contactar por correo electrónico

- **Miguel Ángel Rodríguez** (apoyo a prácticas): miguel.rodriquez@urjc.es

Departamental II - Despacho 118

Tutorías: contactar por correo electrónico

HABLA

Architecture Consulting Training Community Team Trusted by Contact

Your software architecture companion

Boosting digital transformation through functional programming & language-driven architectures

Our services

Request your solution

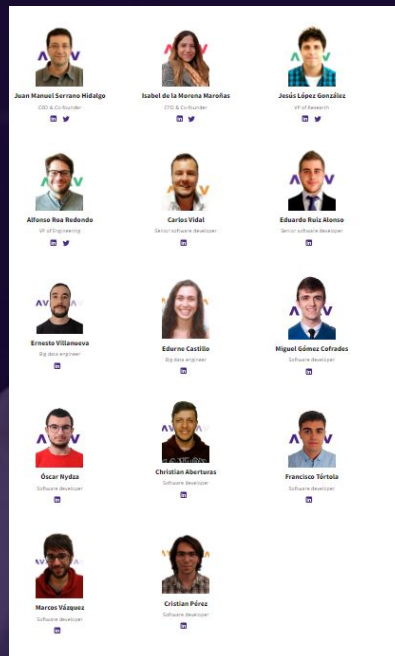


BBVA

Tecsis

MAPFRE

Capgemini





ScalaMAD: Scala Programming @ Madrid

Madrid, España

2287 miembros · Grupo público

Organizado por **Juan Manuel S.** y otras 5 personas

Compartir: [f](#) [t](#) [in](#)

[Sobre nosotros](#)
[Eventos](#)
[Miembros](#)
[Fotos](#)
[Conversaciones](#)
[Unirse a este grupo](#)
[...](#)

Lo que hacemos

Scala es un lenguaje de programación orientado a objetos y, a la vez, un lenguaje funcional. La combinación de estos dos paradigmas hace especialmente atractiva la programación con Scala, y lo convierte en un

Organizadores



Juan Manuel S. y otras 5 personas

[Mensaje](#)

<https://www.meetup.com/es-ES/fp-madrid/>

FP/MADRID



Organized by
HABLA
COMPUTING

Scala **Days**

**The premier Scala conference in
North America and Europe returns in
2023**

SEATTLE

Workshops: June 4th - 5th

Conference: June 5th - 7th

MADRID

Workshops: Sept 11th - 12th

Conference: Sept 12th - 14th

Organized by

HABLA
COMPUTING

EVERYTHING EVERYWHERE ALL WITH **KDB/Q**



Madrid International Lab



30.01.2024



17:00h - 20:30h

Agenda



Javier Sabio. *BBVA*

Data science in q.



Esperanza López. *Squarepoint Capital*

Real-Time Insights: Exploring kdb Use-Cases.



Jesús López. *Habla Computing*

PyKX: a gateway drug into q.



Alfonso Campo. *KX*

KDB.AI: vector databases in action.



Daniel Moreno. *First Derivative*

From Wall Street to Formula 1.

Sponsored by

KX

**FIRST
DERIV
ATIVE**

Presentación

- Profesorado
- **Contexto**
- Contenido
- Planificación
- Evaluación
- Material

Contexto

CURSO 1						
Semestre	Materia	Asignatura	Carácter ¹	Créditos	Departamento	Áreas
1	Humanidades	Antecedentes y Desarrollo de la Inteligencia Artificial (HUM)	FBC	6	CELCAHJHLM, CCACLSIEIO	HC, ATC, CCIA, EIO, LSI
1	Matemáticas	Matemática Discreta y Álgebra	FBR	6	MACIMTE, CCACLSIEIO	MA, ATC, CCIA, EIO, LSI
1	Matemáticas	Cálculo	FBR	6	MACIMTE, CCACLSIEIO	MA, ATC, CCIA, EIO, LSI
1	Matemáticas	Lógica	FBR	6	MACIMTE, CCACLSIEIO	MA, ATC, CCIA, EIO, LSI
1	Informática	Programación I	FBC	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
2	Deontología	Ética y Legislación en Inteligencia Artificial (DEONTOLOGÍA)	FBC	6	DPICP, CCACLSIEIO	CPA, DA, DC, DFT, ATC, CCIA, EIO, LSI
2	Estadística	Probabilidad y Estadística	FBR	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
2	Programación	Programación II	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
2	Programación	Programación Declarativa	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
2	Física	Fundamentos de Arquitectura de Computadores	FBR	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
TOTAL DE CURSO: 60 ECTS						

Contexto

CURSO 2						
Semestr e	Materia	Asignatura	Carácter ²	Créditos	Departamento	Áreas
Anual	Idioma	Idioma Moderno	FBC	6		
1	Empresa	Métodos Operativos y Estadísticos de Gestión	FBR	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
1	Programación	Estructuras de Datos I	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
1	Informática	Algoritmos	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
1	Informática	Informática Teórica y Lenguajes Formales	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
2	Programación	Estructuras de Datos II	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
2	Informática	Sistemas Operativos	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
2	Informática	Bases de Datos	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
2	Aprendizaje automático	Aprendizaje Automático I	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
2	Resolución inteligente de problemas	Algoritmos de Búsqueda I	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
TOTAL DE CURSO: 60 ECTS						

Contexto

CURSO 3						
Semestre	Materia	Asignatura	Carácter ³	Créditos	Departament	Áreas
1	Resolución inteligente de problemas	Algoritmos de Búsqueda II	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
1	Informática	Ingeniería del Software	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
1	Aprendizaje automático	Aprendizaje Automático II	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
1	Representación de conocimiento	Representación de Conocimiento y Razonamiento I	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
1	Informática	Inteligencia Ambiental y Computación Ubicua	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
2	Procesamiento de lenguaje natural	Procesamiento de Lenguaje Natural I	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
2	Representación de conocimiento	Representación de Conocimiento y Razonamiento II	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
2	Percepción y actuación computacional	Visión Artificial	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
2	Percepción y actuación computacional	Robótica	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
2	Tecnologías inteligentes	Sistemas Multi-agente	OB	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
TOTAL DE CURSO: 60 ECTS						

Contexto

MÓDULO DE OPTATIVAS						
Curso	Semestr e	Materia	Asignatura	Crédito s ECTS	Departament o	Áreas
4	1	Tecnologías inteligentes	Interfaces de Usuario Inteligentes	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
4	1	Procesamiento de lenguaje natural	Procesamiento de Lenguaje Natural II	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
4	1	Aprendizaje automático	Aprendizaje Automático III	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
4	1	Resolución inteligente de problemas	Planificación	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
4	2	Algoritmia	Algoritmos para la Toma de Decisiones	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
4	2	Aplicaciones de la Inteligencia Artificial	Inteligencia Artificial en Ciberseguridad	6	CCACLSIEIO	ATC, CCIA, EIO, LSI
4	2	Empresa	Técnicas de Organización y Control de Gestión Empresarial	6	EE, CCACLSIEIO	EFC, ATC, CCIA, EIO, LSI
4	2	Aplicaciones de la Inteligencia Artificial	Aplicaciones de la Inteligencia Artificial	6	CCACLSIEIO	ATC, CCIA, EIO, LSI

Presentación

- Profesorado
- Contexto
- **Contenido**
- Planificación
- Evaluación
- Material

Programación declarativa

- Dado un dominio y una forma de resolver los problemas de dicho dominio, la programación declarativa tiene por objetivo implementar la solución de la forma más cercana a dicho patrón de resolución de problemas
 - Patrones de transformación de datos
 - Resolución lógica de problemas mediante deducción
 - Problemas de satisfacción de restricciones
 - etc.
- Estrechamente relacionado con los lenguajes específicos de dominio
 - SQL: lenguaje de consulta y actualización sobre modelos de datos relacionales
 - Ópticas: lenguajes de consulta sobre modelos de datos algebraicos
 - etc.

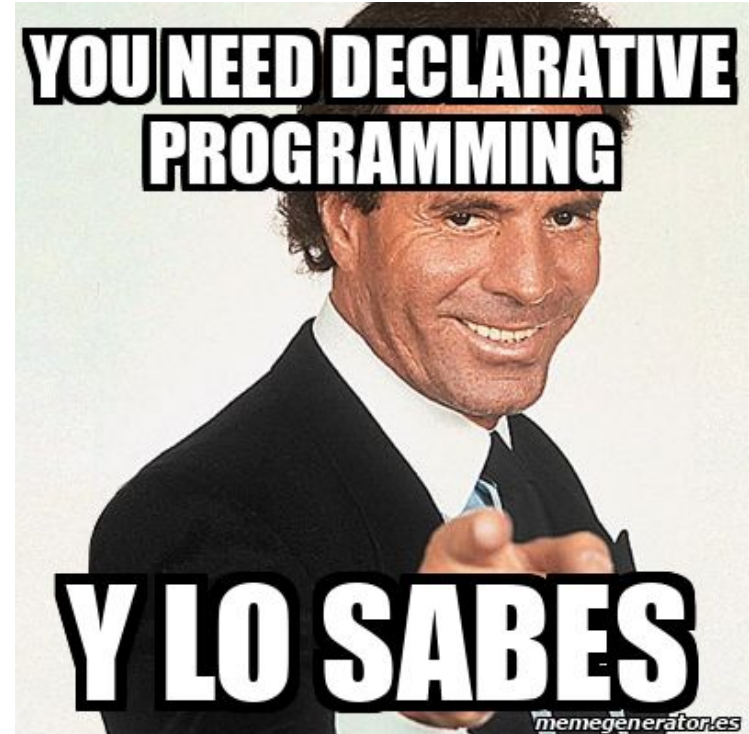
Paradigmas de programación declarativa

- Dos paradigmas principales:
 - Programación funcional
 - Razonamos sobre la forma de resolver un problema en términos de funciones y tipos algebraicos de datos, y un mecanismo computacional de reescritura
 - Programación lógica
 - Razonamos en términos de declaraciones lógicas y mecanismos de deducción automática
- En este curso:
 - Nos centraremos principalmente en la programación funcional
 - Los lenguajes de programación lógica (Prolog) se verán en asignaturas posteriores
 - No obstante, la lógica está íntimamente ligada a la programación funcional, y es parte esencial de este curso

¿Por qué la programación declarativa?

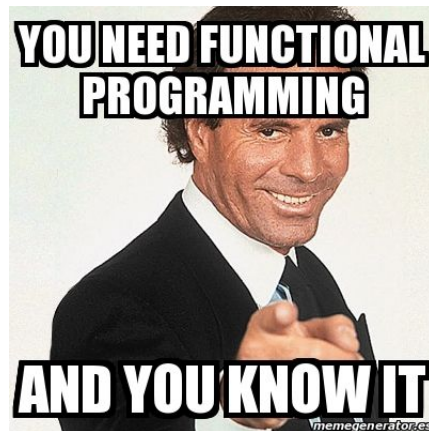
Si quieres que tus programas sean fácilmente

- Comprensibles
- Testables
- Mantenibles
- Reutilizables
- Modificables
- Optimizables
- ...



¿Cómo consigue la programación funcional satisfacer los requisitos no-funcionales de la programación declarativa?

- *Modularity FTW!*
 - functions
 - parametric polymorphism
 - higher-order functions
 - Type classes (ad-hoc polymorphism)
 - Languages (domain-specific languages)
 - datatype generics
 - lazy evaluation
 - ...



¿Qué es la modularidad?

- Código monolítico
 - Diferentes conceptos entre-mezclados
 - Difícil de entender, probar, reutilizar, mantener, etc.
- Código modular
 - Cada aspecto del código se encuentra paquetizado en diferentes módulos
 - Fácilmente comprensible, testable, reutilizable, etc.

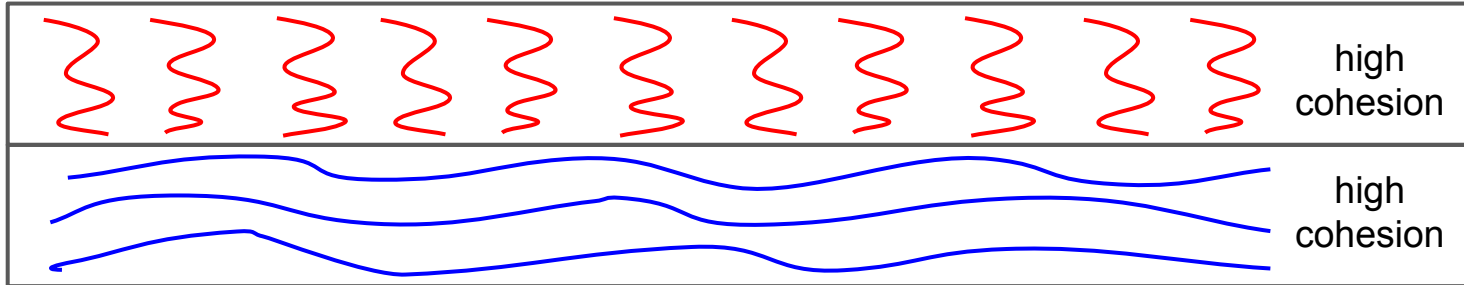
Modularidad: ¡alta cohesión y bajo acoplamiento!

monolithic code



low cohesion
+
high coupling

modular code



high
cohesion

high
cohesion

low coupling

Principales hitos de la programación funcional

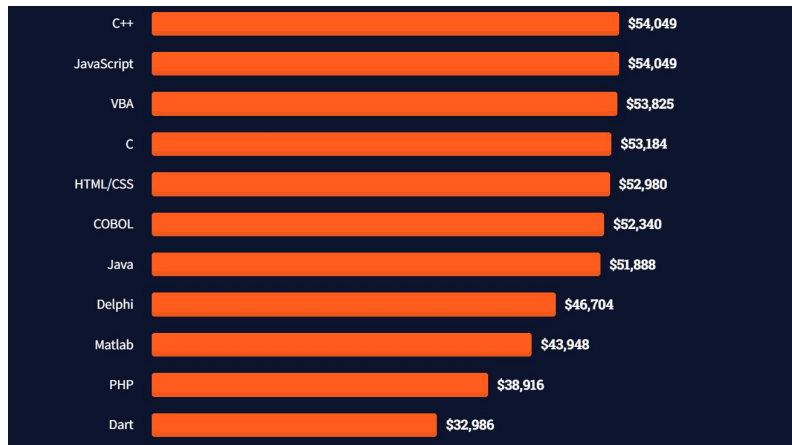
- 1930s- Lambda calculus (Church)
- 1958- LISP (McCarthy)
- 1970s- ML (Milner), HOPE
- 1986- Erlang
- 1987- Haskell
- 1990- Monads in Haskell (Wadler)
- **2004- Scala (Odersky)**
- 2005- F# (Don Syme)
- 2007- Clojure (Hickey)
- 2009- Akka
- 2010 - Spark 0.1
- 2014- Java8, Swift (Apple)
- **2021- Scala 3**



¿Por qué Scala?



<https://insights.stackoverflow.com/survey/2021>



¿Por qué Scala?



Flink

Tema 1. Introducción.

PARTE I

Tema 2. Lenguajes fuertemente tipados

Tema 3. Tipos algebraicos de datos

Tema 4. Programación lógica: Curry-Howard

PARTE II

Tema 5. Funciones y tipos de datos recursivos

Tema 6. Programación modular: funciones de orden superior

Tema 7. Aplicaciones

Parte I: dar cera, pulir cera

Dedución Natural: Ejemplo

$T[s \wedge (p \vee q), p \rightarrow \neg r, q \rightarrow \neg r] \vdash s \wedge \neg r$

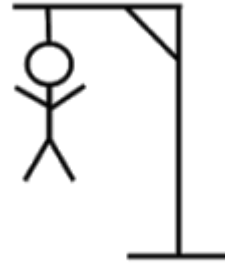
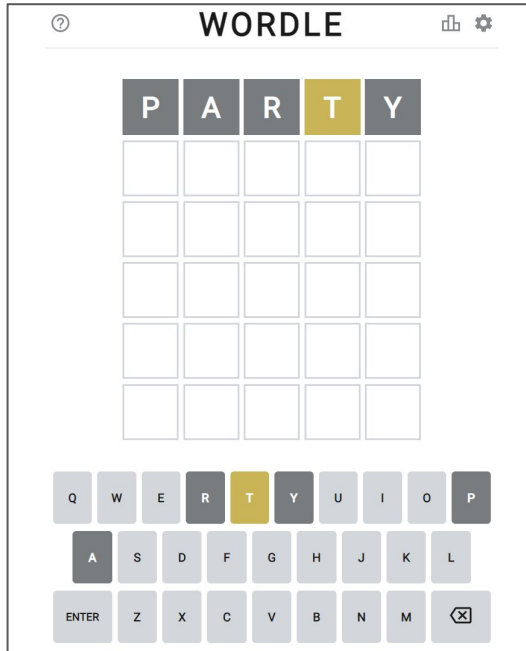
1. $s \wedge (p \vee q)$ *premisa*
2. $p \vee q$ $E_{\wedge}(1)$
3. $p \rightarrow \neg r$ *premisa*
4. $q \rightarrow \neg r$ *premisa*
5. $\neg r$ $E_{\vee}(2, 3, 4)$
6. s $E_{\wedge}(1)$
7. $s \wedge \neg r$ $I_{\wedge}(5, 6)$

```
1 type and[p, q] = (p, q)
2 type or[p, q] = Either[p, q]
3 type implies[p, q] = p => q
4 type not[p] = implies[p, Nothing]
```

```
defined type and
defined type or
defined type implies
defined type not
```

```
1 def proof[p, q, r, s](p1: s and (p or q),
2                        p2: p implies (not[r]),
3                        p3: q implies (not[r])):
4                        s and (not[r]) =
5   (p1._1 : s,
6    p1._2 match {
7      case Left(p) => p2(p) : not[r]
8      case Right(q) => p3(q) : not[r]
9    })
```

Parte II: aplicaciones



....

Presentación

- Profesorado
- Contexto
- Contenido
- **Evaluación**
- Planificación
- Material

Evaluación

- Dos convocatorias: ordinaria (mayo) y extraordinaria (junio)
- En cada convocatoria la evaluación se divide en dos exámenes:
 - PRUEBA 1: Temas 1-4
 - PRUEBA 2: Temas 5-7
- Para aprobar la asignatura es necesario compensar los dos exámenes (≥ 4) y sacar una nota media ≥ 5 (cada examen cuenta un 50% en la nota final)
- Los exámenes compensados en la convocatoria ordinaria se guardan para la convocatoria de junio
- Las pruebas se realizarán en el aula de informática

Presentación

- Profesorado
- Contexto
- Contenido
- Evaluación
- **Planificación**
- Material

Planificación

Enero						
L	M	X	J	V	S	D
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	¹ 24	25	² 26	27	28
29	30	² 31				

Febrero						
L	M	X	J	V	S	D
			1	³ 2	3	4
⁵ 5	6	³ 7	8	³ 9	10	11
12	13	³ 14	15	³ 16	17	18
19	20	³ 21	22	⁴ 23	24	25
26	27	⁴ 28	29			

Planificación

Marzo							Abril						
L	M	X	J	V	S	D	L	M	X	J	V	S	D
				⁴ 1	2	3	1	2	⁶ 3	4	⁶ 5	6	7
4	5	⁴ 6	7	⁴ 8	9	10	8	9	⁶ 10	11	⁶ 12	13	14
11	12	⁴ 13	14	15 EXAMEN P1	16	17	15	16	⁶ 17	18	⁶ 19	20	21
18	19	⁵ 20	21	⁵ 22	23	24	22	23	⁶ 24	25	⁶ 26	27	28
25	26	27	28	29	30	31	29	30					

Planificación

Mayo						
L	M	X	J	V	S	D
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Junio						
L	M	X	J	V	S	D
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Presentación

- Profesorado
- Contexto
- Contenido
- Evaluación
- Planificación
- **Material**

<https://github.com/jserranohidalgo/urjc-gia-pd>

Files

master + 🔍

Go to file t

> exams

> images

> topic1

- Intro.pdf

> topic2

- 2.1 DynamicTyping-Template.ipynb
- 2.1 DynamicTyping.ipynb
- 2.2 StaticTyping-Template.ipynb
- 2.2 StaticTyping.ipynb
- 2.3 Scala-Template.ipynb
- 2.3 Scala.ipynb
- 2.4 Exercises-Sols.ipynb
- 2.4 Exercises.ipynb

> topic3

> topic4

urjc-gia-pd / topic2 / 2.1 DynamicTyping.ipynb



jserranohidalgo Add topic 2

df83665 · 2 years ago

History

Preview

Code

Blame

247 lines (247 loc) · 38.3 KB

Raw



Topic 2. Strongly-typed vs. Dynamic languages

2.1 Dynamically-typed language

Types and values

Values are literals like `1`, `"hola"`, `true`, `'a'`, etc.; types allow us to classify values into different collections. For instance:

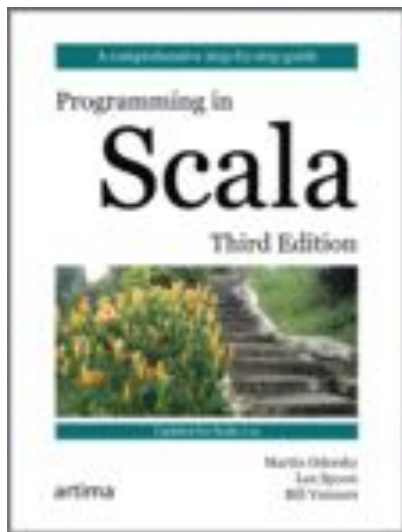
- `0`, `1`, `2` and `3` belong to the collection, or type of, *natural* numbers
- `"hola"`, `"hi"`, `"bye"`, `"adios"` and `""` are *string* values
- *images*, *vectors*, *binary numbers*, *week days*, etc., are also types.

Why do we need types in computing? For one thing, types allow us to explain why we can't apply a given operation to certain arguments. For instance, we can't (i.e. it's meaningless) to multiply *boolean* values with *strings*. It only makes sense if both arguments are values of *numeric types*, such as *integers*, *reals*, etc. But types also serve of great help from a methodological point of view, as we will see when we study type-driven development in the forthcoming lectures.

Bibliografía

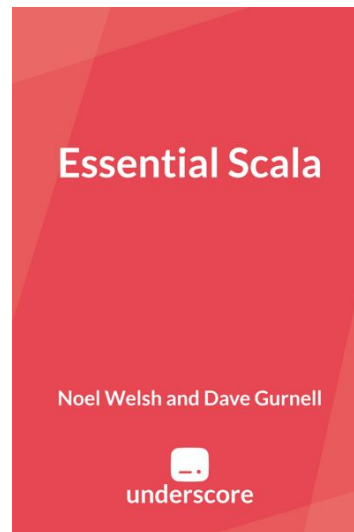
Programming in Scala

M. Odersky, L. Spoons, B. Venners



Essential Scala

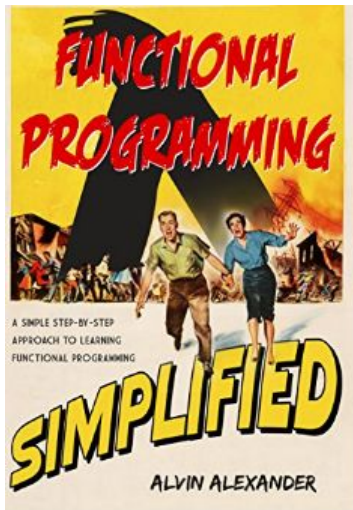
Noel Welsh, Dave Gurnell



Bibliografía

Functional Programming, Simplified

Alvin Alexander



Functional programming in Scala

Chiusano, Bjarnason



[DOCUMENTATION](#)[DOWNLOAD](#)[COMMUNITY](#)[LIBRARIES](#)[CONTRIBUTE](#)[BLOG](#)[docs](#)[API](#)[Learn](#)[Reference](#)[Style Guide](#)[Cheatsheet](#)[Glossary](#)[SIPs](#)

DOCUMENTATION

First Steps...

[Language](#)

GETTING STARTED

Install Scala on your computer and start writing some Scala code!



TOUR OF SCALA

Bite-sized introductions to core language features.



SCALA FOR JAVA PROGRAMMERS

A quick introduction to Scala for those with a Java background.

More Resources:

[Online Courses, Exercises, & Blogs](#) | [Books](#)

<http://www.scala-lang.org/documentation/>