Functional Programming Techniques in Scala Level 1

Description of the training contents

Purely Typed

Duration 8 - 10 hours
Days 1 or 2 as prefered
Price to be negotiated

Trainer

Name Francisco Miguel Arámburo Torres
Website with link to LinkedIn francoara.github.io
Contact francoara@pm.me

Participants

- Programmers who desire a proper introduction to the functional techniques used in Scala and beyond. (A minimal introduction to Scala is required, individuals with different backgrounds are invited to assist)
- Functional programmers who wish to take a step back, and study the mathematical background of the techniques they use.
- Developers who wish to start a path to advanced programming.

Benefits

This trainings provides solid foundations to understand and apply the basic concepts and rationale behind the functional paradigm; it is focused on not just exposing the Scala concepts and techniques, but also in deeper explanation of the mathematical roots with plenty of intuitions to break the barrier of mathematical notation with ease. A link between theory and application is constantly done, to achieve in the student a balance between deep understanding that will help him in current and future assimilation of complex concepts, and a skillful application on the crafting of industrial code.

The trainer will constantly seek didactic interaction with the students, he will make the students think and be actively participating and practicing, with plenty of questions and enough exercises, in order to achieve as much learning experience from the audience as possible.

The training provides to the student with the new abilities and knowledge of:

- Basic techniques of functional programming in Scala that will increase the quality of his code.

- Foundations and mathematical background that will help him expand his understanding of computation and the correct structuring of programs.
- An extra set of mental skills: abstract thinking, equational reasoning, modular design.
- A good notion of how to approach future functional programming concepts, that will help him start a path of self-improvement, to ultimately craft exceptional programs that bring value to people.

About Purely Typed

Purely Typed is a software consultancy company that follows the Vidtecci Ideology: *one of the main objectives of humanity should be to protect and expand life through the cosmos*. In accordance with this, Purely Typed makes a big effort on sharing knowledge, skills, and tools within the software development domain, to empower as many possible people and invite them to discover and craft technology that can bring value to their communities. We believe that knowledge and technology can and should make a positive impact around the globe.

Outline

The training starts with a 10 minute introduction, followed by 5 minutes of explanation of basic nomenclature to be used in the training (like what do we mean by “abstract” or “intuition”) and from there we have 1 hour and 10 minute sections which jump between Theory, Application, Exercices and Social Reinforcement is:

Theory 15 mins -> Exercise 15 mins -> Technical application 15 mins -> Exercise 15 mins -> Rest and discussion 10 mins

This flow allows for constant interaction and mental activity of the students. One level more specific the sections are structured like this:

Introduction to a mathematical object -> Questions and reflection -> Scala code and application to real life tasks -> Coding exercises -> Rest and discussion

Sections

- [0]Introduction - 10 m
- [0] Nomenclature explanation - 5 m: Object, property, abstract, intuition
- [1] Functions 1 - 1h 10 m:

- Definition using tuples
- Function types: Injective, surjective, bijective
- Expression evaluation
- Polymorphic functions
- Immutable data
- Construction of immutable data
- Deconstruction of immutable data
- Some useful functions

- [2] Algebras - 1h 10 m:
  - Objects and operations
  - Equational reasoning
  - Numeric algebras in Scala
  - Non numeric algebras in Scala
  - Function algebras in Scala
  - Code design using algebraic reasoning
- [3] Types - 1h 10 m:
  - Definition
  - Products
  - Coproducts
  - Recursive types
  - Case classes, tuples, hlists = products
  - Either, sealed trait classes = coproducts
  - Examples of recursive types
  - Advantages of expressing programs with types

- Model design using types

- [4] Type classes 1 - 1h 10 m:
    - Ad-hoc polymorphism
    - Encoded properties
    - Implicit resolution
    - Traits as type classes in Scala
    - Scala implicits
    - Code design using type classes
- [5] Algebraic structures - 1h 10 m:
    - Semigroups
    - Monoids
    - Groups
    - Some semigroups in the Scala library
    - Some monoids in the Scala library
    - Some groups in the Scala library
    - Code design using algebraic structures
- [6] Functions 2 - 1h 10 m:
    - Referential transparency
    - Modularity based on referential transparency properties
    - Commutative diagrams and: Identity, Idempotence, isomorphisms
    - Revisiting inheritance
    - Code design using composition (keeping your properties)
- [7] Type classes 2 - 1h 10 m:
    - Higher kinded types

    - Recursive implicit resolution
    - Encoders and decoders as typeclasses
    - Code design using advanced type classes
- [-] Conclusion - 10 m:
    - Everything has mathematical properties
    - Everything is a function
    - Power gained to create exceptional industrial grade code
    - How to approach future concepts in computing and fp
- [-] Extra questions - between 10 m and 2 h (as desired)