FORMATION SCALA AVEC TYPESAFE

Durée : 2 jours soit 14h

LA FORMATION

This two-day course, designed by Martin Odersky, the creator of the Scala programming language, and Heiko Seeberger, a recognized Scala expert, will give you an excellent grounding in Scala.

It is intended to enable developers or development managers, who are experienced programmers in Java or other production languages like C++, C# or Ruby, to confidently start programming in Scala. No previous knowledge of Scala is assumed. Although intense, the course ensures you will have a solid understanding of the fundamentals of the language, the tooling and the development process as well as a good appreciation of the more advanced features. If you already have Scala programming experience, then this course could be a useful refresher.

AFTER HAVING PARTICIPATED IN THIS COURSE YOU SHOULD :

- be a competent user of Scala constructs in application code
- know and be able to apply the functional programming style in Scala
- know how to use the fundamental Scala tools
- be confident to start using Scala in production applications
- The presentation will frequently be mixed with hands-on exercises that give you a good opportunity to try what you have learnt and a chance to clarify your understanding. Therefore it is necessary that you bring your notebook with Java 6 installed.

PROGRAM :

Why Scala?

- Short history
- Overview of Scala's core characteristics

Setting up the development environment

- Installing the Scala distribution
- Installing Eclipse and the Scala plugin
- Installing sbt

First steps

- Interactive programming in the REPL
- Variables and methods
- Expressions and type inference
- First glance at functions

Basic OO features

- Composer des activités multiples
- Rechercher et utiliser un service dans le système
- Mise en œuvre
- Testing in Scala
- Composer des activités multiples

Durée : 2 jours soit 14h

- Rechercher et utiliser un service dans le système
- Mise en œuvre

Learning FP by collections

- Collection hierarchy, creating instances
- Type parameters
- Tuples
- Immutability versus mutability
- Some important collection methods
- Higher-order functions and function literals
- Functions values, function types, short notation
- Important higher order functions: map, flatMap and filter

For-expressions and -loops

- Generators
- Filters
- Definitions
- Translation of for-expressions and -loops
- Inheritance and traits
- Extending classes
- Final and sealed classes
- Enumerations
- Overriding members
- Abstract classes
- Implementing abstract members
- Standard type hierarchy
- Traits and mix-in composition

Pattern Matching

- Match-expressions
- Pattern types
- Pattern guards
- Patterns outside of match expressions

XML support

- Built-in literals
- XML library