Cognizant | Customized Scala Hands-On Workshop

Course: Scala + Akka – Practitioner Level

Duration: **Total for all Modules: 40 Hours**

Focus: This course is design for Learner to Practitioner level . This course will help to

- Get a grip on the functional features of the Scala programming language
- Understand and develop optimal applications using object-oriented and functional Scala constructs
- Learn reactive principles with Scala and work with the Akka framework

Training Objectives:

- Get to know the reasons for choosing Scala: its use and the advantages it provides over other languages
- Bring together functional and object-oriented programming constructs to make a manageable application
- Master basic to advanced Scala constructs
- Test your applications using advanced testing methodologies such as TDD
- Select preferred language constructs from the wide variety of constructs provided by Scala
- Make the transition from the object-oriented paradigm to the functional programming paradigm
- Write clean, concise, and powerful code with a functional mindset
- Create concurrent, scalable, and reactive applications utilizing the advantages of Scala

Targeted Audience:

- Participant must have some development experience.
- Experience with any imperative, object oriented language like C/C++/Java/C# would suffice. Experience with pure functional programming langauge is a plus.

Format: This course will have the following delivery components

- Presentation of concepts
- Live demonstrations
- Hands On

Approach: 30 % Theory,70% Demo / LAB

Language / Tools/ Framework*:

- Scala
- Akka
- TTD

* Note: For all modules, it is preferred that participants have product installed on their respective machines to follow through the demos and to do hands-on along with.

Course Structure:

This program is an experiential program to develop knowledge and skills through practical examples

Workshop Topics Covered

Getting Started with Scala Programming
Introduction to Scala
Scala advantages
Working with Scala
Running our first program

Building Blocks of Scala
What is underneath a Scala program?
Vals and vars
Literals
Data types
Type inference
Operators in Scala
Wrapper classes
String Interpolators

Shaping our Scala Program
Looping
The for expressions
Recursion
Conditional statements
Pattern matching

Giving Meaning to Programs with Functions
Function syntax
Calling a function
Function literals
Evaluation strategies
Partial functions

Getting Familiar with Scala Collections
Motivation
Immutable and mutable collections
Hierarchy of collections in Scala
Commonly used collections in Scala
Rich operations performed on collections
Parallel collections in Scala
Converting a Java collection into a Scala collection
Choosing a collection
Collection performance

Object-Oriented Scala Basics
Classes
Abstract classes
Objects as singletons
Companion objects
Case classes

Next Steps in Object-Oriented Scala
Composition and inheritance
Class inheritance
Default and parameterized constructors
Traits
Traits as mix-ins
Linearization
Packaging and importing
Visibility rules
Sealed traits

More on Functions
Function literals
Methods
Functions versus methods
What are closures?
Higher-order functions
Currying
Partially applied functions

Advanced Functional Programming
Why so serious about types?
Here comes type parameterization
Another way around - generic classes and traits
Type parameter names
Container types
Type erasure
Variance under inheritance
Abstract types
Type bounds
Abstract versus parameterized types
Type-classes

Working with Implicits and Exceptions
Exception handling – the old way
Using the Option way
Either left or right
Implicits - what and why
Implicit parameters
The implicitly method
Implicit conversions
Looking for implicits
Type-classes ahead!

Introduction to Akka
Why do we care about Akka?
What's up with the Actor Model?
Understanding the Actor system
Props
Actor references and paths
Selecting existing actorRefs via actorSelection

How the Actor life cycle works
Hello world in Akka
Writing our first Actor
The tell versus ask versus forward method
Stopping Actors
The preStart and postStop hooks
Actor communication via messages and its semantics
Supervising fault in our actors
OneForOne versus AllForOne strategy
Default supervision strategy
Applying the supervision strategy
Testing actors

Concurrent Programming in Scala
Concurrent programming
Building blocks of concurrency
Asynchronous programming
Parallel collections

Programming with Reactive Extensions
Reactive programming
Reactive extensions
React to RxScala

Testing in Scala
The why and what of TDD
ScalaTest
ScalaMock – a native library to mock objects