ProTech protechtraining.com *"Charting the Course ... ... to Your Success!"*

Scala Programming for Java Developers

Course Summary

Description

Scala is a relatively new language which integrates seamlessly into the Java platform. It provides an implicitly typed alternative to Java which is not limited by the constraints of backward compatibility. Scala was designed as a hybrid language that combines full object-orientation with the core features of functional programming, making it an ideal choice for tasks such as highly concurrent applications and XML manipulation.

This course provides a high speed introduction to Scala for experienced Java developers. It does not waste time rehashing familiar territory but instead builds on top of students existing knowledge. By the end of the course students will be able to write and test JSE applications using all the features of the Scala language.

Topics

- The Evolution of Scala
- Key Features of the Scala Language
- Basic Programming in Scala
- OO Development in Scala
- Functional Programming in Scala
- Pattern Matching in Depth
- Test Driven Development in Scala
- XML Manipulating in Scala
- Writing Concurrent Apps Using Actors

Prerequisites

Students must have a minimum of three years commercial Java programming experience. Knowledge of the basics of Test Driven Development is helpful, as is prior exposure to functional programming concepts (via Lisp, XSLT, F# etc…).

Duration

Three days

ProTech "Charting the Course ...

protechtraining.com

... to Your Success!"

Scala Programming for Java Developers

Course Outline

I. The Evolution of Scala
- A. A brief history of the Java platform to date
- B. Distinguishing between the Java language and platform
- C. Pain points when using Java for software development
- D. Possible criteria for an improved version of Java
- E. How and why the Scala language was created

II. Key Features of the Scala Language
- A. Everything is an object
- B. Class declarations are simplified
- C. Data typing is strong but mostly inferred
- D. Operators are methods and methods can be operators
- E. There is very powerful support for pattern matching
- F. Functions are values and first class citizens
- G. Anonymous and nested functions are supported
- H. Traits are used instead of interfaces

III. Basic Programming in Scala
- A. Built in types, literals and operators
- B. Testing for equality of state and reference
- C. Conditionals, simple matching and external iteration
- D. Working with lists, arrays, sets and maps
- E. Throwing and catching exceptions
- F. Adding annotations to your code
- G. Using standard Java libraries

IV. OO Development in Scala
- A. A minimal class declaration
- B. Understanding primary constructors
- C. Specifying alternative constructors
- D. Declaring and overriding methods
- E. Creating base classes and class hierarchies
- F. Creating traits and mixing them into classes
- G. How a Scala inheritance tree is linearized

V. Functional Programming in Scala
- A. Advanced uses of for expressions
- B. Understanding function values and closures
- C. Using closures to create internal iterators
- D. Creating and using higher order functions
- E. Practical examples of higher order functions
- F. Currying and partially applied functions
- G. Creating your own Domain Specific Languages (DSL's)

VI. Pattern Matching in Depth
- A. Using the match keyword to return a value
- B. Using case classes for pattern matching
- C. Adding pattern guards to match conditions
- D. Partially specifying matches with wildcards
- E. Deep matching using case constructors
- F. Matching against collections of items
- G. Using extractors instead of case classes

VII. Test Driven Development in Scala
- A. Writing standard JUnit tests in Scala
- B. Conventional TDD using the ScalaTest tool
- C. Behavior Driven Development using ScalaTest
- D. Using functional concepts in TDD

VIII. XML Manipulating in Scala
- A. Working with XML literals in code
- B. Embedding XPath like expressions
- C. Using Pattern Matching to process XML data
- D. Serializing and de-serializing to and from XML
- E. Writing Concurrent Apps Using Actors
- F. Issues with conventional approaches to multi-threading
- G. How an actor-based approach helps you write thread safe code
- H. The Scala architecture for creating actor based systems
- I. Different coding styles supported by the actor model