+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Updated on 11/10/2023

Sign up

Reactor training: Non-blocking reactive programming

2 days (14 hours)

Presentation

Reactor is a library that implements the reactive programming model. It is based on the Reactive Streams specification, a standard for creating reactive applications.

Reactor is entirely non-blocking and enables efficient demand management. It interacts fully with JAVA's functional API (completableFuture, Stream and Duration).

With this Reactor training course, your team will learn more about Reactor Core functionalities. They will be able to manipulate elements emitted by streams (Mono and Flux) with the aim of grouping, filtering and converting them.

The tools used will enable your company to operate in a simplified and highly readable way, with a syntax that encourages functional programming. As a result, your organization will gain in flexibility, availability and resilience.

For this training, we use: Reactor 3.4 and Java 19.

Objectives

- All about the reactive programming system
- Master Reactor's various functions
- Learn about Reactive Stream and its implementations
- Handle data and control debugging

Target audience

- Developers
- Computer application developers
- Project managers

Prerequisites

Basic knowledge of Java

Our Reactor training program

Introduction to reactive programming

- What is reactive programming?
- Reactive programming history
- Related concepts
- Introduction to reactive flows
- The Reactor ecosystem

Project start Reactor

- The Project Reactor reactive library
- Introduction to the different reagent types
- Reagent type: Flux
- Reagent type: Mono
- [PRACTICE] Writing the first Flux/Mono and testing it with JUnit5

Reactor project configuration

- Setting up the project
- Functional programming
- What are the advantages of different programming styles?
- Imperative style
- Functional styling
- Operator introduction
- introduction of data parallelism
- Parallelism using the parallel() and runOn() operators

Publisher and Subscriber

- Introduction Publisher
- Advanced Publisher
- Introduction Subscriber
- The link with operators
- [PRACTICE] Flux and Mono creation
- Combining reactive flows using different operators

- Adapting the Flow and Reactor APIs

Reactive Stream

- The lambdas
- The Reactor design pattern
- the observer pattern

Reactor advanced reactive programming

- The concept of backpressure in reactive programming
- Introducing BackPressure
- Implementing BackPressure
- [PRACTICE] Writing a JUnit test for BackPressure
- Subscriber: backpressure management with onBackpressure
- Introduction to cold and hot flows
- ConnectableFlux and various options
- Using advanced parallelism
- Debug
- Sink: extending Flux and Mono
- Scheduler: manage multithreaded applications
- Retry: restart processing in the event of an error

Companies concerned

This course is aimed at both individuals and companies, large or small, wishing to train their teams in a new advanced computer technology, or to acquire specific business knowledge or modern methods.

Teaching methods

Practical course: 60% Practical, 40% Theory. Training material distributed in digital format to all participants.

Organization

The course alternates theoretical input from the trainer, supported by examples, with brainstorming sessions and group work.

Validation

At the end of the session, a multiple-choice questionnaire verifies the correct acquisition of skills.

Sanction

A certificate will be issued to each trainee who completes the course.