AMBIENT-IT

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

Updated on 27/06/2025

Sign up

Vert.X training

3 days (21 hours)

Presentation

Master Vert.x to design responsive, asynchronous and high-performance applications on the JVM. This step-by-step course guides you through the development of APIs, microservices and distributed systems, harnessing the full power of Vert.x and its non-blocking event-driven model.

You'll learn how to create and organize your services around Verticles, structure your REST APIs with Vert.x Web, communicate between services via the Event Bus, and secure your exchanges with JWT, OAuth2 or TLS. You'll also discover how to manage access to SQL or NoSQL databases, and structure efficient processing via the reactive model.

You'll be trained to build robust distributed architectures with service discovery, resilience, WebSockets and Kafka, MQTT or RabbitMQ integration, to meet real-time or high-availability needs.

You'll learn how to test, package and deploy your Vert.x services in production with Maven, Docker and Kubernetes, while ensuring their observability via Prometheus and Grafana-compatible monitoring tools.

As with all our training courses, it will be run on my latest version of Vert.X.

Objectives

- Understand Vert.x's reactive architecture and its non-blocking event-driven model for designing applications.
- Install, configure and structure a Vert.x project with Maven or Gradle, mastering verticles, event loops and component lifecycles
- Develop robust REST APIs with Vert.x Web, integrating routing, error handling, endpoint security and essential middleware.

- Interfacing a Vert.x application with SQL and NoSQL databases, managing transactions, and structuring data exchanges with a reactive model
- Build a distributed architecture based on Vert.x microservices, using Event Bus, service discovery WebSockets, Kafka or MQTT
- Supervise, test and deploy a Vert.x application in production, integrating Docker, Kubernetes, Prometheus and resilience strategies such as circuit breakers.

Target audience

- Backend developers
- DevOps developers
- Software architects

Prerequisites

- Basic knowledge of asynchronous programming
- Proficiency in a JVM language

Vert.x training program

Introduction to Vert.x and reactive programming

- Reactive architecture vs. imperative architecture
- Performance, scalability, asynchronism
- Concrete use cases
- Event loop, event bus, back-pressure
- Threads, concurrent, non-blocking
- Core, Web, Data, Auth, MQTT, Kafka, etc.
- Comparison with Spring WebFlux, Akka, Quarkus

Installation, structure and lifecycle

- Via Maven / Gradle
- Using vertx CLI
- Organization into modules
- Verticle types: standard, worker, coroutine
- Start(), stop() methods
- Deployment, scaling, shutdown

Event-driven programming with Verticles

- Creation in Java or Kotlin
- Using Vertx.vertx().deployVerticle
- Local and distributed EventBus
- Addressing, publish/subscribe, RPC via proxy
- Event loops, threading models
- Worker Verticles for blocking tasks

REST API development with Vert.x Web

- Use of Router, Route, RoutingContext
- URL matching, query params, path params
- Headers, bodies, status
- Uploads, downloading files
- HTML, CSS, JS files
- Template engines
- Authentication, CORS, cookies, sessions
- Logger, BodyHandler, ErrorHandler

Data access

- Connection, connection pool
- Query execution, object mapping
- MongoDB Client: insertion, search, aggregation
- Redis Client: cache, pub/sub
- Rollback, propagation, failover
- Use of futures/promises for flow control

Security and authentication

- Users and roles
- Session management, JWT, OAuth2
- TLS configuration
- Good security practices
- Security interceptors
- Customized permissions

Reactive architecture and microservices

- Inter-service communication via EventBus
- Loosely-coupled design
- Consul, Kubernetes, service registry
- Vert.x Circuit Breaker
- Retry, timeout, fallback

Integration with other systems

- Vert.x Kafka Client
- Message production/consumption
- Using vertx-mqtt-client
- IoT protocols
- Real-time bidirectional communication
- Client connection management

Testing, deployment and monitoring

- Units with JUnit + VertxUnit
- Integration tests and mocks
- Fat JAR with vertx-maven-plugin
- Dockerization and orchestration
- Vert.x Dropwizard Metrics
- Integration with Prometheus / Grafana

Companies involved

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.

Positioning on entry to training

Positioning at the start of training complies with Qualiopi quality criteria. As soon as registration is finalized, the learner receives a self-assessment questionnaire which enables us to assess his or her estimated level of proficiency in different types of technology, as well as his or her expectations and personal objectives for the forthcoming course, within the limits imposed by the selected format. This questionnaire also enables us to anticipate any connection or security difficulties within the company (intra-company or virtual classroom) which could be problematic for the follow-up and smooth running of the training session.

Teaching methods

Practical training: 60% hands-on, 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 is used to check that skills have been correctly acquired.

Certification

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