AMBIENT-IT

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

Updated on 19/03/2024

Sign up

Advanced Rust training

2 days (14 hours)

Presentation

Our Advanced Rust training course is designed for developers who already have basic experience with Rust and are looking to deepen their knowledge to take full advantage of the advanced capabilities of the Rust language.

This course will take you beyond the fundamentals to explore advanced techniques, asynchronous programming concepts, concurrency, interoperability with other languages, and more, enabling you to develop secure, high-performance Rust applications.

You'll learn how to integrate Rust using the Foreign Function Interface (FFI), and apply efficient concurrency models to develop highly secure, parallel applications.

As with all our training courses, this one will introduce you to the latest version of Rust Programming Language, Rust 1.76.

Objectives

- Master advanced Rust concepts
- Optimize your Rust applications for maximum performance
- Developing web applications and services with Rust

Target audience

Developers.

Prerequisites

- A solid understanding of basic Rust concepts
- Practical experience with the Rust language, ideally having completed projects or our Rust training course
- Familiarity with system and asynchronous programming concepts
- Configured Rust development environment, including Cargo and Rust build tools

Software requirements

- Installing Docker and Docker Compose
- Installing gnuplot
- Configured Rust development environment, including Cargo and Rust build tools

Advanced Rust training program

ADVANCED FEATURES AND GENERICS

- Deeper understanding of traits and generic types
- Using strokes as function parameters
- Feature terminals and specialization
- Generics and performance: monomorphization
- Design patterns with features and generics

ASYNCHRONOUS PROGRAMMING IN RUST

- Understanding the asynchronous execution model
- Use `async` and `await` for non-blocking operations
- Asynchronous error handling
- Comparison of `tokio` and `async-std` runtimes
- Design patterns for asynchronous programming

COMPETITION IN RUST

- Competition models in Rust: threads, channels, and `Arc<Mutex>`.
- Use `Rc` and `Arc` for shared memory management
- Strategies for passing messages between threads
- Data security in a competitive environment
- Exploring popular competition craters

ADVANCED USE OF LIFETIMES

- Understanding advanced lifetimes for memory management
- Annotation of lifetimes in complex structures
- Lifetimes in callbacks and closures

- Patterns to avoid lifetime errors
- Advanced reference and borrowing management

MEMORY AND PERFORMANCE

- Manual memory management with `unsafe`.
- Performance optimization in Rust
- Profiling and debugging Rust applications
- Using collections to maximize performance
- Efficient allocation and release techniques

INTEGRATION WITH OTHER LANGUAGES

- FFI (Foreign Function Interface) for integrating C/C++ with Rust
- Creating and using dynamic libraries
- Calling Rust from other languages
- Safety and best practices when using FFI

WEB DEVELOPMENT WITH RUST

- REST API creation with Actix-web and Rocket
- Using Diesel to access databases
- Security and session management in web applications
- WebAssembly to integrate Rust into the browser

RUST FOR SYSTEMS AND EMBEDDED SYSTEMS

- no_std` programming for embedded systems
- Memory management without dynamic allocation
- Interaction with hardware and operating systems
- Crates and libraries for embedded development

VISUALIZATION AND USER INTERFACES

- Creating user interfaces with `egui`, `Druid`, or `Iced`.
- Event management and graphics rendering
- Interface performance and responsiveness
- Integration with windowing systems and graphics tools

PROJECT AND CODE REVIEW

- Applying the concepts learned to a concrete project
- Best practices in Rust development
- Code review and group feedback

- Discuss advanced use cases and share experiences

Companies concerned

This training 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 training to come, 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 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.