AMBIENT-IT

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

Updated 04/28/2025

Sign up

Hexagonal Architecture and TDD training

3 days (21 hours)

Presentation

Hexagonal Architecture is not an architecture in the strict sense of the word, but a set of architectural principles put forward by Alistair in 2005 and refined by a whole host of other authors: Robert C. Martin aka Uncle Bob in 2012.

The latter has named its variant of the hexagonal "Clean Architecture", providing a number of guidelines and preventive measures.

These architectural principles deliver a whole host of benefits:

- Strong decoupling between the business part of the application and the infrastructure/technology part, making it much easier to design the brain of the application.
- Business part testability greatly improved; real TDD made possible!
- Ability to postpone infrastructure technology choices.
- Ability to change technologies effortlessly, avoiding time-consuming redesigns

At the end of this Hexagonal Architecture training course, you'll learn to master the essential principles, produce organized software and create high-performance domain models.

Objectives

- Master the principles of Hexa/Clean Architecture, such as dependency inversion
- Know how to start a project from scratch using TDD and Hexa Architecture with a "code design emergence" mindset
- Awareness of crucial global technical architecture decisions
- Know how to handle behavior-oriented TDD in a Hexagonal Architecture increased, seamless productivity
- Know how to integrate infrastructure components such as a PostgreSQL database and Third-party partner APIs without touching the core application
- Know how to dissociate the application's business logic from the Spring-Boot framework

- Knowing all the pitfalls to avoid in Hexa/Clean Architecture
- Be fully aware of the major difference between TDD and simply writing tests
- Know how to order the actions to be carried out when creating a Hexagonal Architecture
- All questions answered and popular misunderstandings about practices revealed

Target audience

- Technical Leaders
- Backend developers
- Full Stack Developers
- Technical architects

Prerequisites

- Mastery of Java or any other object-oriented language
- Notions of the main OOP concepts: Interfaces / Abstract classes / Polymorphism
- Test writing with JUnit 5 and AssertJ

Technologies used

- Java 20
- Maven 3
- Spring-Boot / Rest APIs
- Hibernate/JPA
- PostgreSQL
- JUnit 5 / AssertJ
- TestContainers (Docker)

Hexagonal Architecture and TDD Training Program

Day 1: The Foundations of Robust Architecture

Architectural foundations

- Why Software Architecture is Essential: Highlighting the challenges of complex systems and the crucial role of a well thought-out architecture
- The Qualities of Effective Architecture: Presenting the characteristics of good architecture (maintainability, scalability, testability, readability, etc.)
- Tangible benefits: highlight the tangible benefits for the project and the team (cost reduction, time-to-market, quality, collaboration).

Clean Architecture: A Clear Design Model

- SOLID: The Five Key Principles: Explain in detail each SOLID principle (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Reversal) with concrete examples.
- Modularization, Organizing Code for Clarity: Presenting modularization strategies and their importance for complexity management and reuse
- Anatomy of Clean Architecture: Describe each layer (Entities, Use Cases, Interface Adapters, Frameworks & Drivers), their role and the rules of dependency.

Practical introduction to Test-Driven Development (TDD)

- Unit Testing: Your Security Net: Explain the purpose, characteristics and best practices of unit testing.
- Vertical slicing: from functional to technical: introducing the idea of slicing slice-and-dice functionality for easy development and testing
- The Enduring TDD Cycle: Red, Green, Refactor: Detailing each step of the TDD cycle and its importance in guiding design
- Step-by-step business development with TDD: Putting TDD into practice to implement simple business logic, with an emphasis on emergent design and code quality

Day 2: Mastering Insulation with Hexagonal Architecture

Hexagonal Architecture

- Similarities and differences with Clean Architecture: Identify similarities (separation concerns, domain focus) and differences (ports/adapters vs. layers) between the two architectures.
- Ports and Adapters: Explain in detail the role of Ports (contracts) and Adapters (implementations for interacting with the outside world) Distinguish between primary (Driving) and secondary (Driven) Adapters
- Structure of a Hexagonal Project: Present a typical project structure highlighting estate insulation

Integration and End-to-End Testing

- The Test Pyramid: Present the test pyramid (unit, integration, end-to-end) and the importance of each level.
- The Crucial Stakes of Integration and End-to-End Testing: Highlighting their role in validating interactions between components and overall system behavior Managing

Data Persistence in a Hexagonal Context

- ACID properties: Ensuring Data Integrity: Explaining the concepts Atomicity, Consistency, Isolation and Durability in transaction management

- Spring Transactions: Orchestrating operations : Introducing transaction management with Spring and its integration into a hexagonal architecture
- Understanding the basic operation of Hibernate: Introducing common patterns used with Hibernate (ORM) to interact with the database without coupling the domain

Implementation of an Isolated Development Environment

- Managing Migrations with Liquibase: Evolving the schema: Practical workshop on to use Liquibase to manage database schema changes in a collaborative, versioned way.
- External API simulation with Wiremock: Practical workshop on creating API mocks to facilitate integration and end-to-end testing without relying on real systems
- Practical Workshop: Integration Testing: Implementation of tests that verify the interaction between different parts of the application (e.g. Use Cases and Persistence Adapters).
- Practical Workshop: End-to-End Testing: Introduction and implementation of flow simulation tests complete user

Day 3: Adopting an Adaptive, Business-Centric Architecture

Screaming Architecture: Code at the service of the business

- Benefits of a Business Tree: Highlight how a code structure that reflects the domain facilitates understanding by non-developers and the evolution of the system.
- Pitfalls to avoid: Warn against excesses and misinterpretations of the Screaming Architecture
- Tree structure examples: Present concrete examples of business-oriented project structures

Domain-Driven Design (DDD) awareness

- DDD: Taming Business Complexity: Explaining how DDD helps understand and model a complex domain
- Bounded Contexts: Delineating responsibilities: Introducing the concept of Bounded Context and how it works
  role to isolate domain models and manage business rules consistently
- Improve Business Rules Management: Discuss impact of DDD on the clarity and maintainability of business rules.

Emerging Architectures : Organic Evolution

- The YAGNI Principle: Don't speculate on the future: Explain the importance of building only what's needed now.
- Letting Architecture Emerge: Discussing how architecture can evolve by response to real needs and feedback Construire Ensemble

(Mob Programming)

- Practical Implementation: Practical application of all the practices, principles and tools learned during the course to build a library management API.
- Collaboration and exchange: Encourage teamwork, knowledge sharing and the exchange of ideas. collective resolution of challenges

Further information

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, 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.