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

Updated on 11/04/2025

Register

Hexagonal Architecture and TDD Training

3 days (21 hours)

Overview

Hexagonal Architecture is not an architecture per se, but rather a set of architectural principles put forward by Alistair in 2005 and refined by a number of other authors, including one in particular: Robert C. Martin, aka Uncle Bob, in 2012.

Martin called his version of hexagonal architecture "Clean Architecture" and provided a number of guidelines and precautions.

These architectural principles offer a whole host of benefits:

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

At the end of this Hexagonal Architecture training course, you will learn how 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 spirit of "code design emergence"
- Be aware of crucial global technical architecture decisions
- Know how to use behavior-driven TDD in a Hexagonal Architecture for increased and flawless productivity
- Know how to integrate infrastructure components such as a PostgreSQL database and third-party partner APIs without affecting the core of the application
- Know how to separate the application's business logic from the Spring Boot framework

- Know all the pitfalls to avoid in a Hexagonal/Clean Architecture
- Be fully aware of the major difference between TDD and simply writing tests
- Know how to prioritize the actions to be taken when developing a Hexagonal Architecture
- All questions answered and common misconceptions/misrepresentations about practices revealed

Target audience

- Technical Leaders
- Backend Developers
- Full Stack Developers
- Technical Architects

Prerequisites

- Proficiency in Java or any other object-oriented language
- Understanding of the main concepts of OOP: Interfaces / Abstract classes / Polymorphism
- Knowledge of 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)

Our Training Program Hexagonal Architecture and TDD

Day 1: The Foundations of a Robust Architecture

Architectural Fundamentals

- Why Software Architecture is Essential: Highlighting the challenges of complex systems and the crucial role of well-designed architecture
- The Qualities of Effective Architecture: Presenting the characteristics of good architecture (maintainability, scalability, testability, readability, etc.)
- Concrete Benefits: Highlight the tangible advantages 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 each SOLID principle in detail (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion) with concrete examples
- Modularization, Organizing Code for Clarity: Present modularization strategies and their importance for managing complexity and reuse
- Anatomy of Clean Architecture: Describing each layer (Entities, Use Cases, Interface Adapters, Frameworks & Drivers), their roles, and dependency rules

Practical Introduction to Test-Driven Development (TDD)

- Unit Testing: Your Safety Net: Explain the purpose, characteristics, and best practices of unit testing
- Vertical Slicing: From Functional to Technical: Introduce the idea of slicing features into slices to facilitate development and testing
- The Immutable Cycle of TDD: Red, Green, Refactor: Detail each step of the TDD cycle and its importance in guiding design
- Developing Business Logic Step by Step with TDD: Putting TDD into practice to implement simple business logic, emphasizing emergent design and code quality

Day 2: Mastering Isolation with Hexagonal Architecture

Hexagonal Architecture

- Similarities and differences with Clean Architecture: Establish the similarities (separation of concerns, domain at the center) 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 domain isolation

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 Issues of Integration and End-to-End Testing: Emphasize their role in validating interactions between components and the overall behavior of the system

Managing Data Persistence in a Hexagonal Context

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

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

Up an Isolated Development Environment

- Managing Migrations with Liquibase: Evolving the Schema: Hands-on workshop on using Liquibase to manage database schema changes collaboratively and in a versioned manner
- Simulating External APIs with Wiremock: Hands-on workshop on creating mock external APIs to facilitate integration and end-to-end testing without depending on real systems
- Practical Workshop: Integration Testing: Implementing 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 tests that simulate a complete user flow
  user flow

Day 3: Adopting an Adaptive, Business-Centric Architecture

Screaming Architecture: Code at the Service of the Business

- The Advantages of a Business Tree Structure: Highlight how a code structure that reflects the domain facilitates understanding by non-developers and the evolution of the system
- Pitfalls to Avoid: Warning against excesses and misinterpretations of Screaming Architecture
- Examples of Tree Structures: Present concrete examples of business-oriented project structures

Awareness of Domain-Driven Design (DDD)

- DDD: Taming Business Complexity: Explaining how DDD helps to understand and model a complex domain
- Bounded Contexts: Delineating Responsibilities: Presenting the concept of Bounded Context and its role in isolating domain models and managing business rules in a consistent manner
- Improving Business Rule Management: Discuss the 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 is necessary now
- Let the Architecture Emerge: Discuss how architecture can evolve in response to actual needs and feedback Building Together (Mob

Programming)

- Practical Implementation: Concrete application of all the practices, principles, and tools covered during the training to build a library management API
- Collaboration and Exchange: Encouraging teamwork, knowledge sharing, and collective resolution of challenges

To go further

Target companies

This training is intended for both individuals and companies, large or small, wishing to train their teams in a new advanced IT technology or to acquire specific professional knowledge or modern methods.

Placement at the start of training

The positioning at the start of the training complies with Qualiopi quality criteria. Upon final registration, the learner receives a self-assessment questionnaire that allows us to assess their estimated level of proficiency in different types of technologies, as well as their expectations and personal objectives for the upcoming training, within the limits imposed by the selected format. This questionnaire also allows us to anticipate certain connection or internal security issues within the company (intra-company or virtual classroom) that could be problematic for the monitoring and smooth running of the training session.

Teaching methods

Practical training: 60% practical, 40% theory. Training materials distributed in digital format to all participants.

Organization

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

Assessment

At the end of the session, a multiple-choice questionnaire is used to verify that the skills have been correctly acquired.

Certification

A certificate will be issued to each trainee who has completed the entire training course.