

# Introduction to Layered Architecture

Layered Architecture: Organizing Software for Clarity and Efficiency

DEFINITION

Layered architecture is a design pattern that separates software into distinct layers, each with specific responsibilities. This structure enhances maintainability, scalability, and clarity in software development.

KEY CONCEPTS

Presentation Layer

This layer handles user interface and user interaction, presenting data to users.

Business Logic Layer

It contains the core functionality and rules of the application, processing data between layers.

Data Access Layer

This layer manages data storage and retrieval, interacting with databases or external services.

Separation of Concerns

Each layer focuses on a specific aspect of the application, reducing complexity and improving maintainability.

Inter-layer Communication

Layers communicate with each other through well-defined interfaces, promoting loose coupling.

EXAMPLES

- Web applications using MVC architecture with distinct layers for views, controllers, and models.
- Enterprise applications separating user interface, business logic, and database access for better organization.
- Mobile apps that utilize layered architecture to manage UI, logic, and data efficiently.

MEMORY TIPS

- ★ Think 'UI, Logic, Data' to remember the main layers.
- ★ Use the acronym 'PL-BL-DL' for Presentation, Business Logic, and Data Access layers.
- ★ Visualize a cake with layers to remember the separation of concerns.

COMMON MISTAKES

- ✗ Confusing the roles of layers, leading to tightly coupled components.
- ✗ Neglecting to define clear interfaces for inter-layer communication.
- ✗ Overcomplicating the architecture by adding unnecessary layers.

QUICK RECAP

Layered architecture organizes software into distinct layers for better management and clarity. Each layer has a specific role, facilitating easier maintenance and scalability. Remember the key layers: Presentation, Business Logic, and Data Access.