Advanced Software Engineering Part 03 - Layered Architecture Style

Dr. Amjad AbuHassan

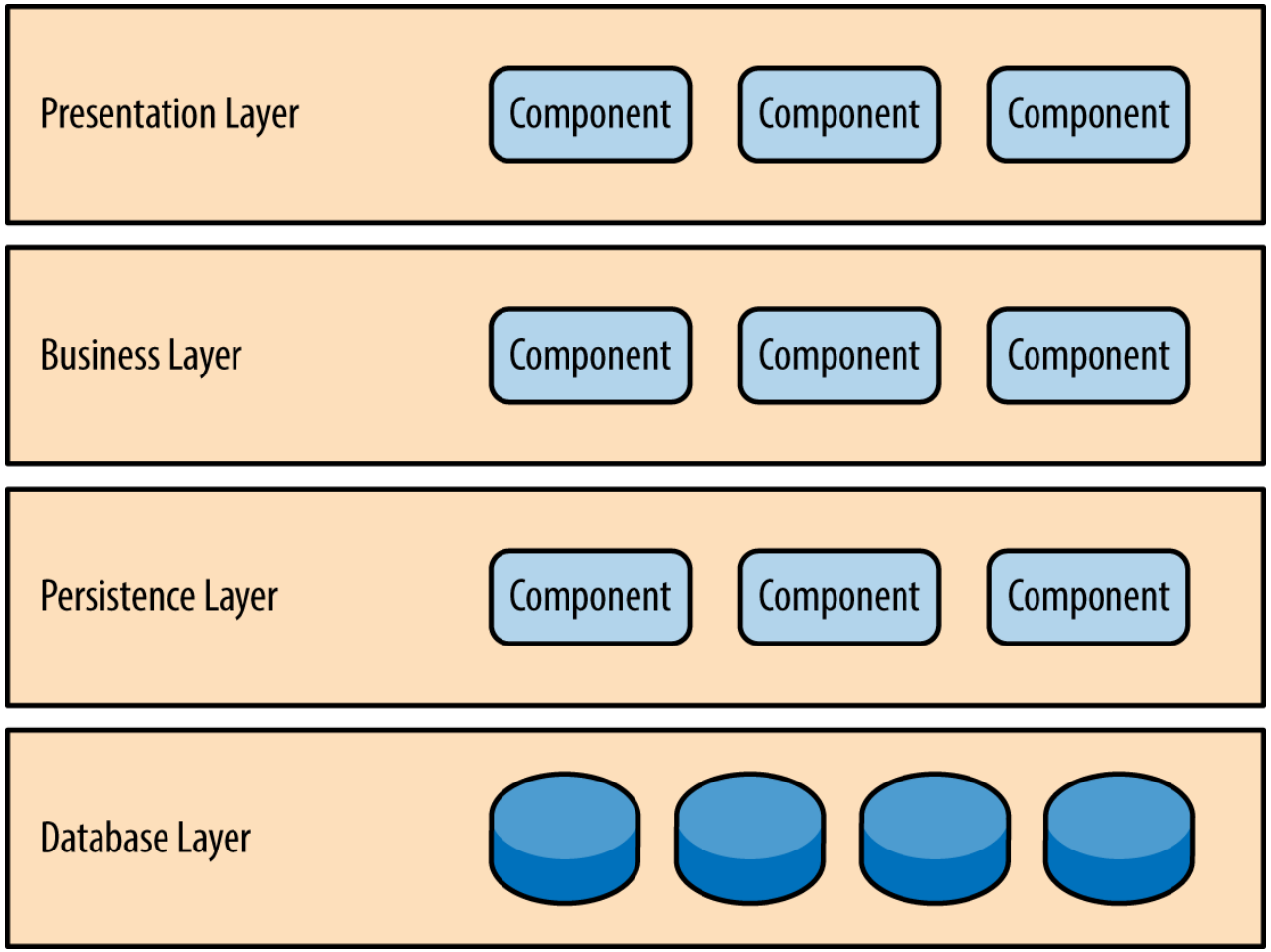
Topics Covered

- Topology
- Layers of Isolation
- Adding Layers
- Why Use This Architecture Style
- Architecture Style Issues

Topology

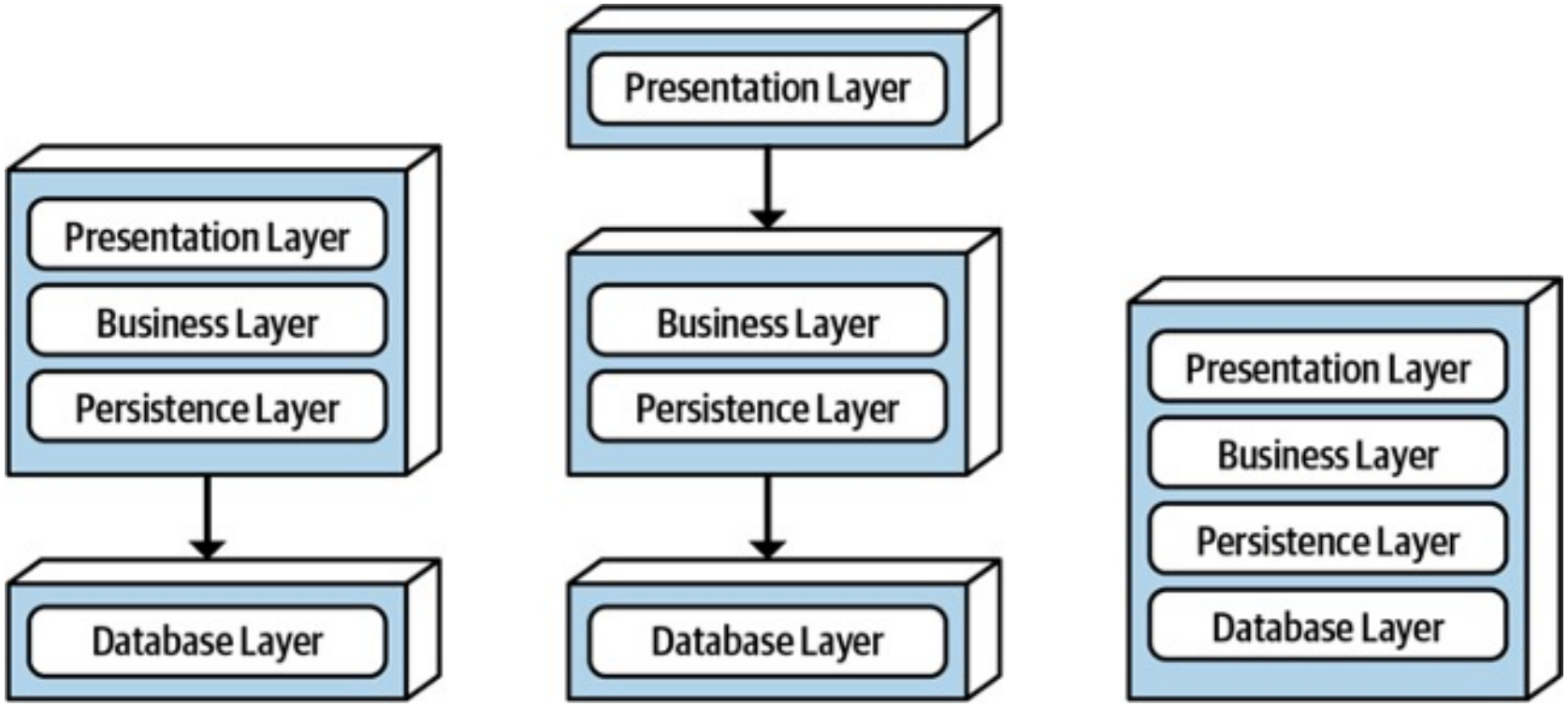
Standard logical layers within the layered architecture style

n-tier architecture pattern



Topology cont.

- Physical topology (deployment) variants



Separation of Concerns

- Components within a specific layer are limited in scope, dealing only with the logic that is related to that layer.
  - For example, components in the presentation layer only handle presentation logic, whereas components residing in the business layer only handle business logic.
- This separation of concerns concept within the layered architecture style *makes it easy to build effective roles and responsibility models within the architecture.*

Separation of Concerns cont.

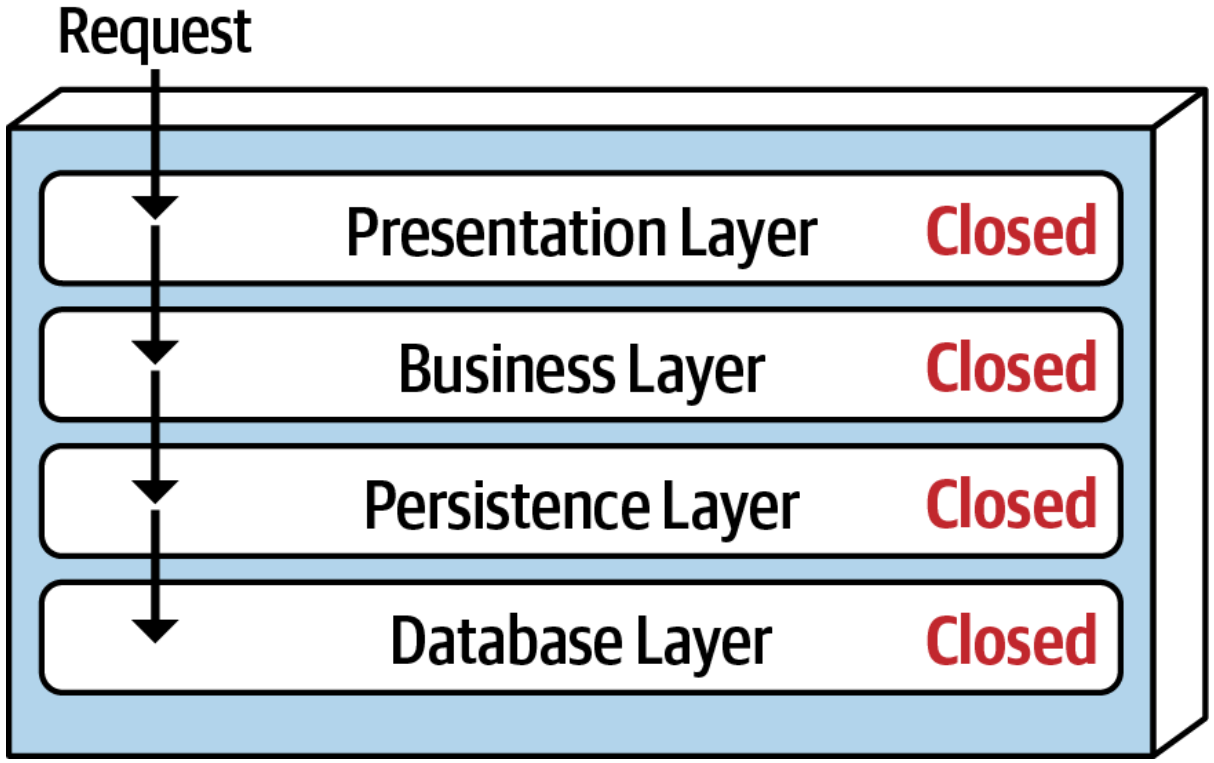
- This allows developers to leverage their particular technical expertise to focus on the technical aspects of the domain (such as presentation logic or persistence logic).
- **The trade-off of this benefit, however, is a lack of overall agility (the ability to respond quickly to change).**

Partition

- The layered architecture is a technically partitioned architecture (as opposed to a domain-partitioned architecture).
- Groups of components, rather than being grouped by domain (such as customer), are grouped by their technical role in the architecture (such as presentation or business).
- As a result, any business domain is spread throughout all the layers of the architecture.

Layers of Isolation

- A closed layer means that as a request moves top-down from layer to layer, the request cannot skip any layers, but rather must go through the layer immediately below it to get to the next layer



Layers of Isolation cont.

- The layers of isolation concept means that changes made in one layer of the architecture generally don’t impact or affect components in other layers, providing the contracts between those layers remain unchanged.
- Each layer is independent of the other layers, thereby having little or no knowledge of the inner workings of other layers in the architecture.

Layers of Isolation cont.

- In previous Figure, it would be much faster and easier for the presentation layer to access the database directly for simple retrieval requests, bypassing any unnecessary layers
  - Used to be known in the early 2000s as the fast-lane reader pattern.
- For this to happen, the business and persistence layers would have to be open, allowing requests to bypass other layers.

Layers of Isolation cont.

- If the presentation layer can directly access the persistence layer, then changes made to the persistence layer would impact both the business layer and the presentation layer, producing a very tightly coupled application with layer interdependencies between components.
- This type of architecture then becomes very breakable, as well as difficult and expensive to change.

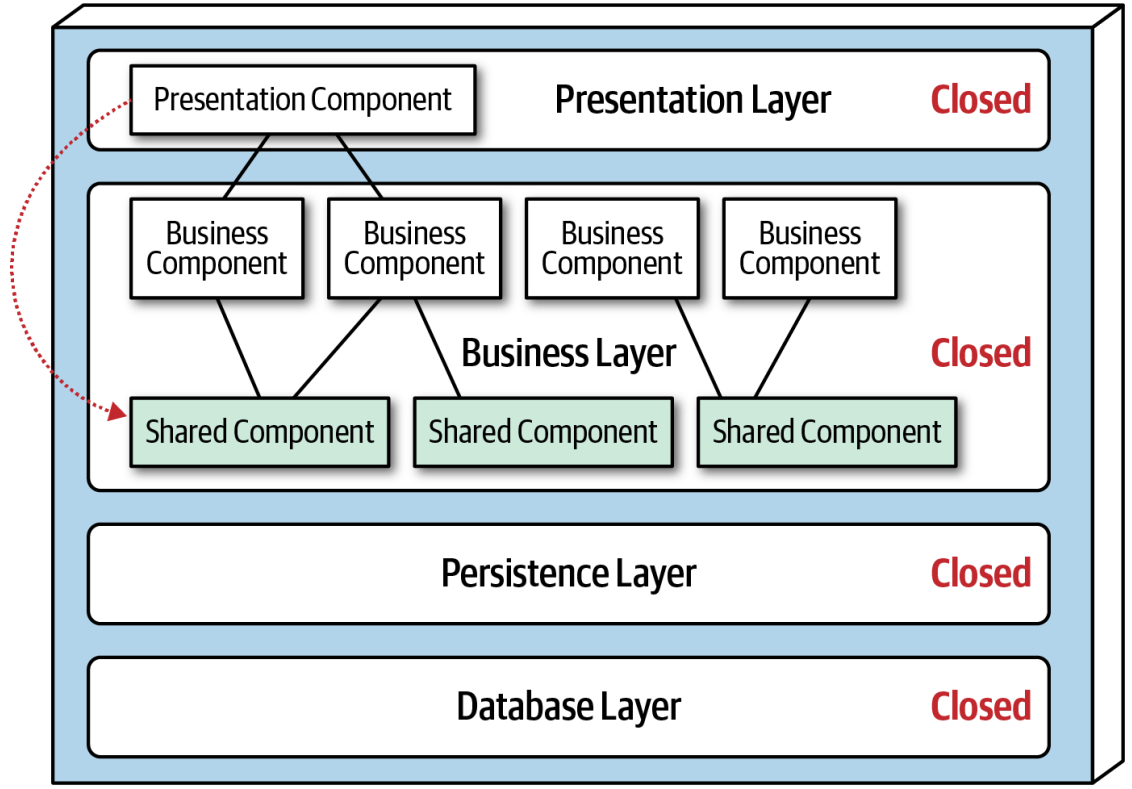
Adding Layers

- Suppose there are shared objects within the business layer that contain common functionality for business components (such as date and string utility classes, auditing classes, logging classes, and so on).

- Suppose there is an architecture decision stating that the presentation layer is restricted from using these shared business objects.

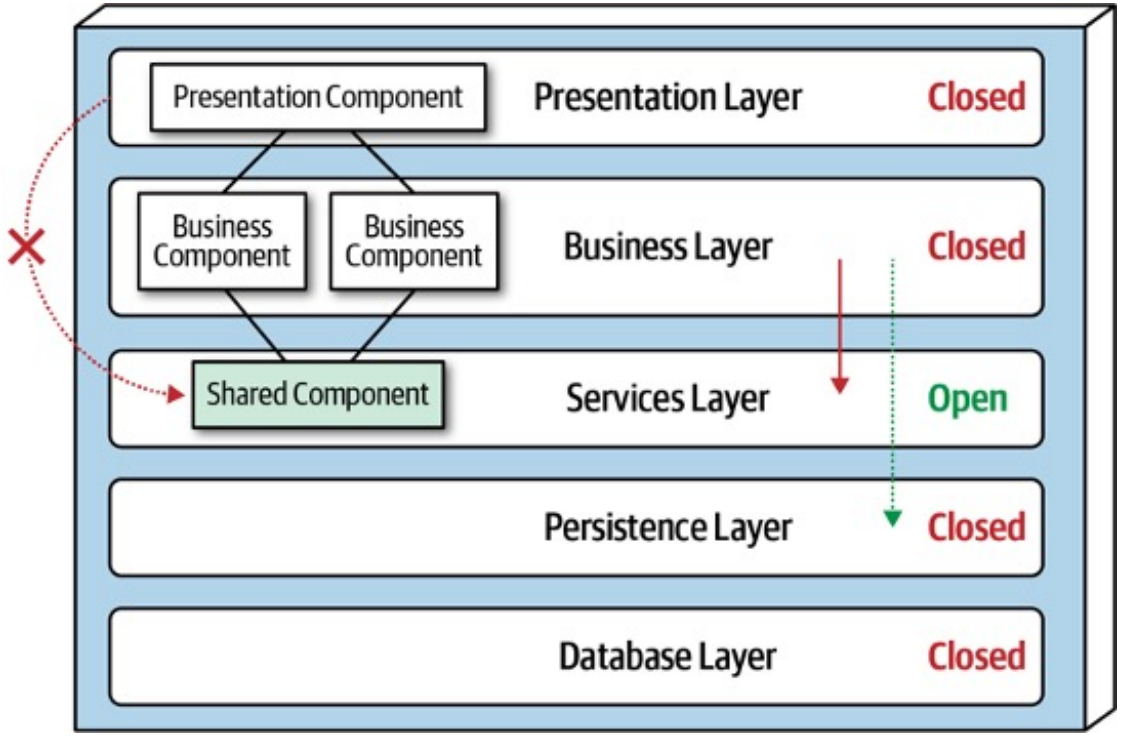
Adding Layers cont.

- Shared objects within the business layer



Adding Layers cont.

- One way to architecturally mandate this restriction is to add to the architecture a new services layer containing all the shared business objects.



Why Use This Architecture Style

- It is a good choice for small, simple applications or websites.

- As applications using the layered architecture style grow, characteristics like maintainability, agility, testability, and deployability are adversely affected.

Architecture Style Issues

- Clean separation between layers is often difficult.
- Performance can be a problem because of multiple layers of processing between call and return.

Sinkhole Problem

- This problem occurs when requests move from layer to layer as simple pass-through processing with no business logic performed within each layer.

Retrieve Customer Information For A Particular Individual