

Core Java 8 and Development Tools

Lesson 17 : Introduction to
Layered Architecture

Lesson Objectives

- After completing this lesson, participants will be able to
 - Understand the concept of Layered Architecture
 - Implement layers in Java applications



Copyright © Capgemini 2015. All Rights Reserved. 2

This lesson covers Layered architecture and advanced testing concepts.

Lesson outline:

- 17.1: Introduction
- 17.2: Layered Architecture

17.1: Introduction

What is Layered Architecture?

- Layered architecture is one of the architectural pattern based on call-and-return style
- In layered architecture, business rules, behavior, and data are obtained and manipulated, based on activity via the user interface.
- Layered architecture provides a clean separation between the business implementation, presentation and data-access logic.

The diagram illustrates a layered architecture with three main layers:

- Presentation Layer:** Contains the **Application UI**.
- Service/Business Logic Layer:** Contains **Service Interfaces** and **Service Implementation**.
- Data Access Layer:** Contains **DAO Interfaces** and **DAO Implementation**.

Below the Data Access Layer is the **Database**. To the right of the layers is a box labeled **Domain Objects**. Arrows indicate the flow of data and control: from the Application UI to Service Interfaces, then to Service Implementation, then to DAO Interfaces, then to DAO Implementation, and finally to the Database. There are also bidirectional arrows between the Application UI and Domain Objects, and between Domain Objects and both Service Implementation and DAO Implementation.

Capgemini
ENHANCING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved. 3

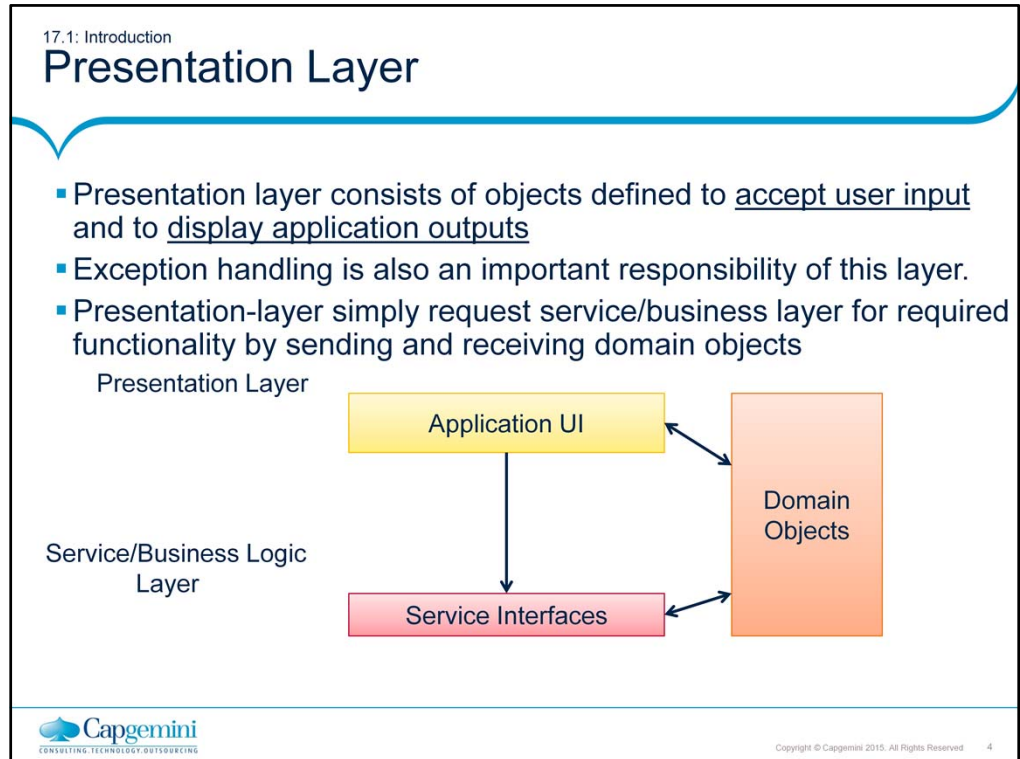
What is Layered Architecture?

Layering partitions the functionality of an application into separate layers that are stacked vertically. As shown in the figure above, layered architecture enables developer to make changes on one layer without having any side effects on others.

Why Layered Architecture?

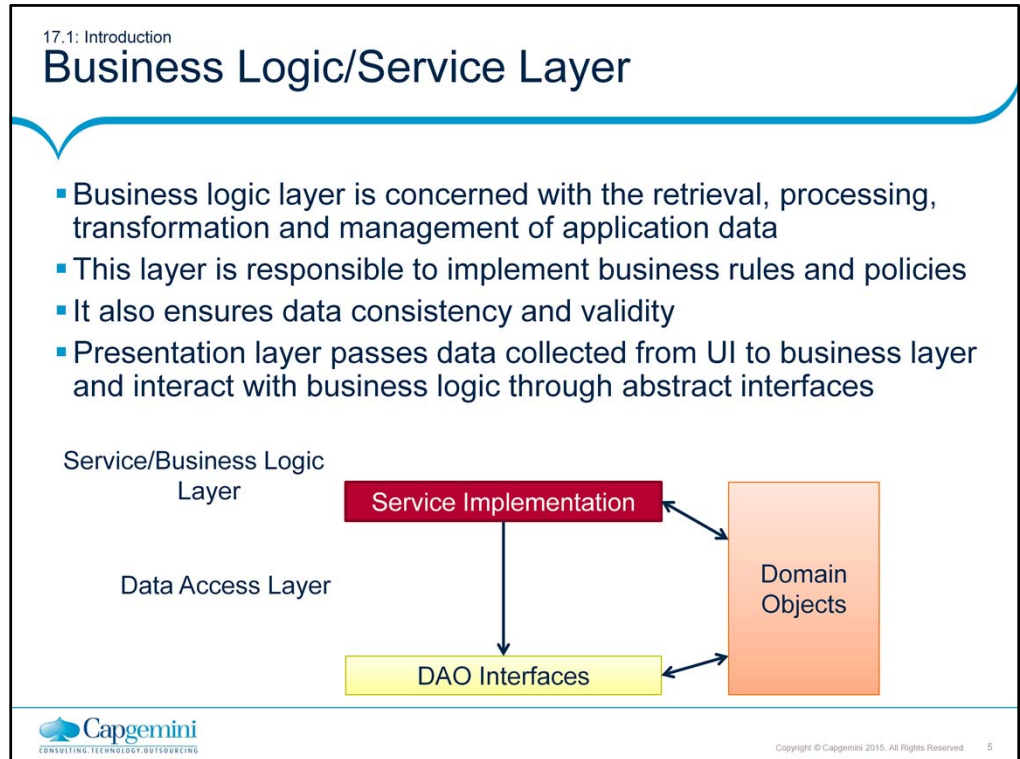
Object technology encouraged the abstraction and reuse of not only presentation logic but also business processes and data. Therefore, decoupling application logic from application presentation is encouraged.

With the explosion of the Internet and related technologies, requirements to scale, rapidly develop, deploy, and react to business changes have made the existence of layered application architecture imperative.



Presentation Layer:

The presentation layer contains the components that implement and display the user interface and manage user interaction. This layer includes controls for user input and display,



Business Logic/Service Layer:

After the presentation layer collect the required data from the user and pass it to the business layer, the application can use this data to perform a business process. Use a business layer to centralize common business logic functions and promote reuse.

17.1: Introduction

Data Access Layer

- This layer abstract the logic required to access the underlying data stores
- It centralize common data access functionality in order to make the application easier to configure and maintain.
- This layer is responsible for managing connections, generating queries, and mapping application domain objects to data source structures
- Business logic layer interacts to data access layer through abstract interfaces using application domain objects

```
graph TD; DAO[DAO Implementation] <--> DO[Domain Objects]; DAO --> DB[(Database)];
```

Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 6

Data Access Layer


The data access layer should hide the details of data source access. It should be responsible for managing connections, generating queries, and mapping application domain objects to data source structures.

Consumers (Business logic layer) of the data access layer interact through abstract interfaces using application domain objects.

17.1: Introduction

Data Transfer Objects


- Data transfer objects (DTO) or Value Objects (VO) encapsulates business data necessary to represent real world elements, such as Customers or Orders
- These object are POJO's to store data values and expose them through properties
- They contain and manage business data used by the entire application



The diagram consists of an orange rectangular box labeled "Domain Objects" in the center. Below the box, the text "Data Transfer Objects" is written. This visualizes the relationship where DTOs are used to transfer data from domain objects.

Domain Objects

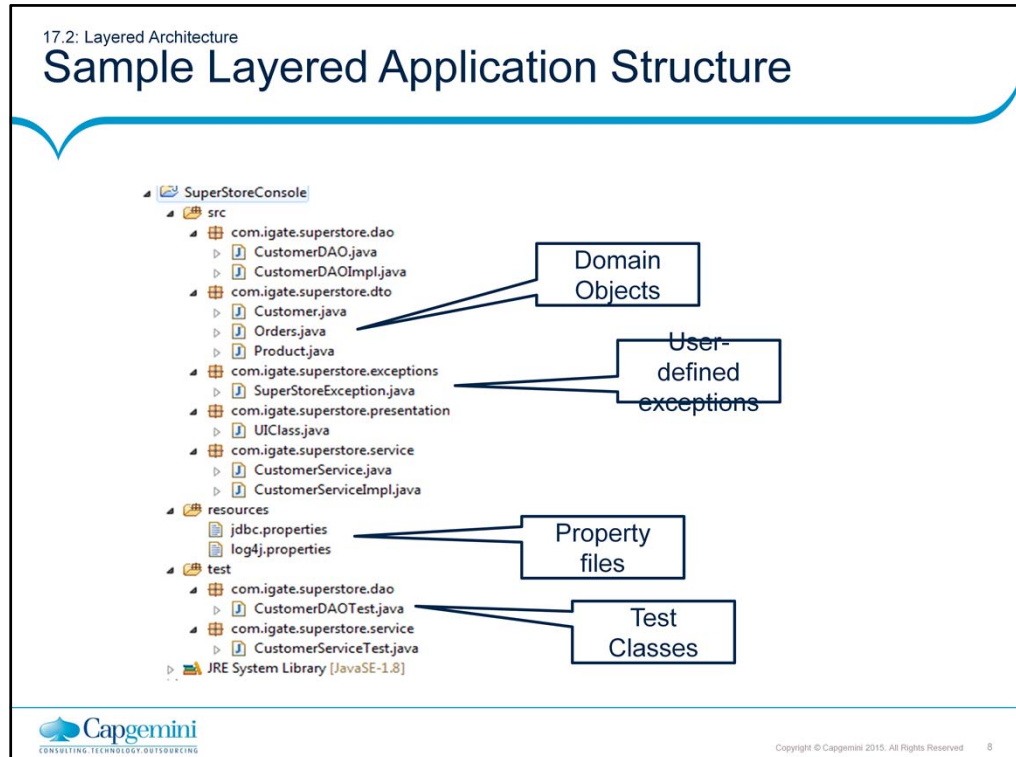
Data Transfer Objects

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved. 7

Data Transfer Objects

DTO's are simply POJO classes that represent application real world entities. These objects are meant to share data between layers of application. These object contains data values and expose them through properties.



Designing Layered Application:

Create separate packages for all the layers including presentation, business logic and data access layer. All POJO based domain object needs to be stored in separate package.

Application specific exceptions must be separated from layered classes as shown in the slide.

Ensure the test classes must be written for DAO layers. Many business processes involve multiple steps that must be performed in the correct order. To ensure correctness of such business process, its better to test business layer classes too.

Database specific properties like username, password, URL etc. must be stored in separate property file. All property files must be store in resources folder of application.

Lab

- Lab 12: Introduction to Layered Architecture



Summary

- In this lesson, you have learnt:
 - Layered architecture for Java applications



Review Question

- Question 1: _____ layer abstract the logic required to access the underlying data stores
 - Option 1: Service
 - Option 2: Data Access
 - Option 3: Presentation
- Question 2: Layered architecture is one of the architectural pattern based on call-wait-process pattern style
 - True / False

